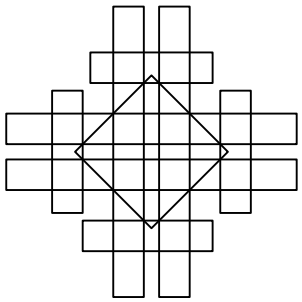
36th International Symposium on Computational Geometry

SoCG 2020, June 23–26, 2020, Zürich, Switzerland (Virtual Conference)

Edited by

Sergio Cabello

Danny Z. Chen



LIPICS

Editors

Sergio Cabello
University of Ljubljana, Ljubljana, Slovenia
sergio.cabello@fmf.uni-lj.si

Danny Z. Chen
University of Notre Dame, Indiana, USA
dchen@nd.edu

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

ISBN 978-3-95977-143-6

Published online and open access by
Schloss Dagstuhl – Leibniz-Zentrum für Informatik GmbH, Dagstuhl Publishing, Saarbrücken/Wadern, Germany. Online available at https://www.dagstuhl.de/dagpub/978-3-95977-143-6.

Publication date
June, 2020

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 https://portal.dnb.de.

Digital Object Identifier: 10.4230/LIPIcs.SoCG.2020.0

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

https://www.dagstuhl.de/lipics

Contents

Regular Papers

36th International Symposium on Computational Geometry (SoCG 2020).
Editors: Sergio Cabello and Danny Z. Chen
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Media Expositions

CG Challenge

Preface

The 36th International Symposium on Computational Geometry (SoCG 2020) was held online, June 23-26, 2020, as part of the Computational Geometry Week (CG Week 2020). The event was planned to take place in Zürich, Switzerland, but eventually it was organized online because of the COVID-19 pandemic. We are very grateful to the organizing team at ETH Zürich to organize the event and adapt to the unusual situation.

Altogether, 205 papers have been submitted to SoCG 2020. After a thorough review process, in which each paper has been evaluated by three or more independent reviewers, the Program Committee accepted 70 papers for presentation at SoCG 2020. These proceedings contain extended abstracts of the accepted papers, limited to 500 lines plus references. If any supporting material (e.g., proofs or experimental details) does not fit in the line limit, the full paper is available at a public repository and referenced in the corresponding extended abstract.

The Best Paper Award of SoCG 2020 went to the paper "*Convex Hulls of Random Order Types*" by Xavier Goaoc and Emo Welzl. The Best Student Presentation Award will be determined and announced at the symposium, based on ballots cast by the attendees. A few selected papers with very positive reviews will be invited to forthcoming special issues of Discrete & Computational Geometry and the Journal of Computational Geometry dedicated to the symposium.

This year, for the first time, there is a SoCG Test of Time Award and a committee was appointed for this task. In this first edition, the winners of the first edition of this new award are the papers "*Epsilon-Nets and Simplex Range Queries*" by David Haussler and Emo Welzl, presented at SoCG 1986, "*Applications of Random Sampling in Computational Geometry, II*" by Kenneth L. Clarkson, presented at SoCG 1988, and "*Algorithms for Diametral Pairs and Convex Hulls That Are Optimal, Randomized, and Incremental*" by Kenneth L. Clarkson and Peter W. Shor, presented at SoCG 1988.

The scientific program of the CG Week 2020 was enriched by two distinguished invited speakers. An invited talk, entitled "Geometric statistics for computational anatomy", was delivered by **Xavier Pennec**, from Université Côte d'Azur and Inria. Another invited talk, entitled "Discrete developable surfaces: Theory and fabrication of 3D shapes from 2D sheets", was delivered by **Olga Sorkine-Hornung**, from ETH Zürich. We thank the invited speakers for their excellent invited talks.

The Media Exposition was significantly changed from previous years, no longer restricted to video submissions. The media portfolio now included animations, software and games, scientific artwork, sculptures, paintings, and more. We only required it to be related to computational geometry, broadly interpreted, as well as being "displayable" during CG Week. All SoCG attendees were also invited to bring media "side pieces" to display during CG Week to complement the Exposition collection. We reviewed 15 submissions (of all media types), of which 12 were accepted. The extended abstracts that describe these lovely submissions are included in this proceedings volume, while their corresponding media content can be found at `http://www.computational-geometry.org`.

A new feature in this year's proceedings is the CG Challenge. The goal was to compute minimum convex decompositions for a benchmark collection of point sets in the plane. 21 teams submitted solutions; these proceedings contain contributions by the three top-placed teams describing their winning approaches.

36th International Symposium on Computational Geometry (SoCG 2020).
Editors: Sergio Cabello and Danny Z. Chen
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

We thank the authors of all submitted works. We are most grateful to the members of the SoCG Program Committee, the Media Exposition Committee and the CG Challenge Committee for their dedication, expertise, and hard work that ensured the high quality of the works in these proceedings. We are grateful to the assistance provided by 331 additional reviewers; without their help it would be nearly impossible to run the selection process.

Finally, we would like to thank Matias Korman, who kindly accepted to be the Proceedings Chair for a second year and did a meticulous work. Many other people contributed to the success of SoCG 2020 and the entire CG Week. We especially thank the local organizers, all the members of the Workshop and YRF Committees, and the Computational Geometry Steering Committee.

Sergio Cabello
SoCG Program Committee, co-chair
University of Ljubljana

Danny Z. Chen
SoCG Program Committee, co-chair
University of Notre Dame

Satyan Devadoss
Media Exposition, chair
University of San Diego

Sándor P. Fekete
CG Challenge, co-chair
TU Braunschweig

Conference Organization

SoCG Program Committee

- Mikkel Abrahamsen, University of Copenhagen
- Therese Biedl, University of Waterloo
- Mickaël Buchet, TU Graz
- Sergio Cabello (co-chair), University of Ljubljana
- Danny Z. Chen (co-chair), University of Notre Dame
- David Eppstein, University of California, Irvine
- Stefan Funke, University Stuttgart
- Marc Glisse, INRIA Saclay
- Dan Halperin, Tel Aviv University
- Iyad Kanj, DePaul University
- Irina Kostitsyna, TU Eindhoven
- Jan Kynčl, Charles University
- Jian Li, Tsinghua University
- Nabil Mustafa, Université Paris-Est, ESIEE Paris
- Eunjin Oh, POSTECH
- Tim Ophelders, Michigan State University
- Florian T. Pokorny, KTH Royal Institute of Technology
- Sharath Raghvendra, Virginia Tech
- Don Sheehy, North Carolina State University
- Primož Škraba, Queen Mary University of London
- Frank Staals, Utrecht University
- Katharine Turner, Australian National University
- Torsten Ueckerdt, Karlsruhe Institute of Technology
- Hubert Wagner, IST Austria
- Bartosz Walczak, Jagiellonian University
- Jinhui Xu, SUNY Buffalo

SoCG Proceedings Chair

- Matias Korman, Tufts University, USA

Media Exposition Committee

- Henry Adams, Colorado State University
- Jit Bose, Carleton University
- Satyan Devadoss (chair), University of San Diego
- David Eppstein, University of California, Irvine
- John McCleary, Vassar College
- Marc van Kreveld, Utrecht University
- Yusu Wang, The Ohio State University
- Lori Ziegelmeier, Macalester College

36th International Symposium on Computational Geometry (SoCG 2020).
Editors: Sergio Cabello and Danny Z. Chen
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

CG Challenge Committee

- Erik Demaine, MIT
- Sándor P. Fekete, TU Braunschweig
- Phillip Keldenich, TU Braunschweig
- Dominik Krupke, TU Braunschweig
- Joseph S. B. Mitchell, Stony Brook University

SoCG Test of Time Award Committee

- Pankaj K. Agarwal, Duke University
- Dan Halperin, Tel Aviv University
- Raimund Seidel, Saarland University

Workshop Committee

- Anne Driemel, University of Bonn
- Elizabeth Munch, Michigan State University
- Jeff M Phillips (chair), University of Utah
- Rodrigo Silveira, Universitat Politècnica de Catalunya
- Jack Snoeyink, University of North Carolina, Chapel Hill

Young Researchers Forum Program Committee

- Erin Chambers, Saint Louis University
- Anne Driemel, University of Bonn
- Tamal Dey, Ohio State University
- Michael Kerber (chair), TU Graz
- Wouter Meulemans, TU Eindhoven
- Evanthia Papadopoulou, Universita della Svizzera italiana
- Zuzana Patáková, IST Austria
- Sharath Raghvendra, Virginia Tech

Local Organizing Committee

- Bernd Gärtner, ETH Zürich
- Michael Hoffmann (chair), ETH Zürich
- Andrea Salow, ETH Zürich
- Patrick Schnider, ETH Zürich
- Emo Welzl, ETH Zürich
- Manuel Wettstein, ETH Zürich

Steering Committee (2018–2020)

- Mark de Berg, TU Eindhoven
- Erin Chambers (Secretary), Saint Louis University
- Michael Hoffmann, ETH Zürich
- Joe Mitchell (Treasurer), State University of New York at Stony Brook
- Bettina Speckmann, TU Eindhoven
- Monique Teillaud (Chair), INRIA Nancy - Grand Est

Additional Reviewers

Anders Aamand
Mohammad Ali Abam
Ahmed Abdelkader
Henry Adams
Thomas Dybdahl Ahle
Hee-Kap Ahn
Taehoon Ahn
Oswin Aichholzer
Noga Alon
Helmut Alt
Alexandr Andoni
Alan Arroyo
Elena Arseneva
Dominique Attali
Sergey Avvakumov
Arturs Backurs
Martin Balko
Djordje Baralic
Imre Barany
Ulrich Bauer
Robin Belton
Sergey Bereg
Ahmad Biniaz
Ranita Biswas
Thomas Bläsius
Omer Bobrowski
Bianca Boeira Dornelas
Jean-Daniel Boissonnat
Prosenjit Bose
Karl Bringmann
Adam Brown
Kevin Buchin
Maike Buchin
Pierre Calka
Jean Cardinal
Paz Carmi
Mathieu Carrière
J. Frederico Carvalho
Wojciech Chachólski
Erin Chambers
Timothy M. Chan
Manoj Changat
Steven Chaplick
Bapi Chatterjee
Bhaskar Ray Chaudhury
Chao Chen
Ke Chen
Xiaoming Chen
Siu-Wing Cheng
Victor Chepoi
Jongmin Choi
Josef Cibulka
Éric Colin de Verdière
Robert Connelly
René Corbet
Mónika Csikós
Guilherme D. Da Fonseca
Daniel Dadush
Ovidiu Daescu
Gautam K Das
Mark de Berg
Alessandro De Gregorio

36th International Symposium on Computational Geometry (SoCG 2020).
Editors: Sergio Cabello and Danny Z. Chen
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Arnaud de Mesmay
Vin de Silva
Olaf Delgado-Friedrichs
Shichuan Deng
Olivier Devillers
Tamal Dey
Emilio Di Giacomo
Hu Ding
Pawel Dlotko
Michael Gene Dobbins
Anne Driemel
Vida Dujmovic
Adrian Dumitrescu
Stephane Durocher
Kunal Dutta
Zdenek Dvorak
Eduard Eiben
Friedrich Eisenbrand
Khaled Elbassioni
Jeff Erickson
Emerson G. Escolar
Esther Ezra
Brittany Terese Fasy
Dan Felmdan
Stefan Felsner
Fabien Feschet
Fabrizio Frati
Bin Fu
Ulderico Fugacci
Radoslav Fulek
Jie Gao
Kirk Gardner
Cyril Gavoille
Arijit Ghosh
Panos Giannopoulos
Emeric Gioan
Xavier Goaoc
Tanja Gologranc
José Carlos Gómez Larrañaga
Lee-Ad Gottlieb
Dejan Govc
Joachim Gudmundsson
Andrea Guidolin
Xiangyu Guo
Mohammadtaghi Hajiaghayi
Thekla Hamm
Sariel Har-Peled
Joel Hass
Teresa Heiss
Darryl Hill
Petr Hliněný
Andreas Holmsen
Ron Holzman
Lingxiao Huang
Ziyun Huang
Alfredo Hubard
Clemens Huemer
Mark Hughes
Dan Ismailescu
Bruno Jartoux
Vít Jelínek
Shaofeng Jiang
Alvin Jin
Mateusz Juda
Taeho Jung
Sara Kalisnik

Haim Kaplan
Matthew Katz
Michael Kerber
Andreas Kerren
Balázs Keszegh
Jisu Kim
Mincheol Kim
Woojin Kim
Philipp Kindermann
Sándor Kisfaludi-Bak
Linda Kleist
Bettina Klinz
Kolja Knauer
Christian Konrad
Matias Korman
Michal Koucký
Laszlo Kozma
Tomasz Krawczyk
Bala Krishnamoorthy
Andrey Kupavskii
Theo Lacombe
Nathaniel Lahn
Kasper Green Larsen
Soeren Laue
Jesus Leanos
Seungjun Lee
Michael Lesnick
Ran Levi
Shi Li
Andre Lieutier
Bingkai Lin
Giuseppe Liotta
Maarten Löffler
Martin Lotz
Ben Lund
Kelly Maggs
Mathieu Mari
Clément Maria
Leonardo Ignacio Martínez Sandoval
Domagoj Matijevic
Andrew McGregor
Tamara Mchedlidze
Killian Meehan
Saeed Mehrabi
Ezra Miller
Till Miltzow
Majid Mirzanezhad
Matthias Mnich
Dylan Molho
Fabrizio Montecchiani
Pat Morin
Sonoko Moriyama
Guillaume Moroz
Dmitriy Morozov
David Mount
Shay Mozes
Sayan Mukherjee
Wolfgang Mulzer
Elizabeth Munch
Meiram Murzabulatov
Anurag Murty Naredla
Marton Naszodi
Abhinandan Nath
Yakov Nekrich
Benjamin Niedermann
Arnur Nigmetov

Aleksandar Nikolov
Gabriel Nivasch
André Nusser
Joseph O'Rourke
Ippei Obayashi
Yoshio Okamoto
Katharina Ölsböck
Krzysztof Onak
Aurélien Ooms
Georg Osang
Nina Otter
Steve Oudot
Arnau Padrol
Eyvindur Ari Palsson
Irene Parada
Ji-won Park
Zuzana Patáková
Amit Patel
Florian Pausinger
Petar Pavešić
Jose Perea
Pablo Pérez-Lantero
Ljubomir Perkovic
William Pettersson
Julian Pfeifle
Thang Pham
Jeff Phillips
Paweł Pilarczyk
Vincent Pilaud
Adam Polak
Vladislav Polianskii
Valentin Polishchuk
Marc Pouget
Ioannis Psarros
Marcel Radermacher
Saladi Rahul
Benjamin Raichel
Rajiv Raman
Saurabh Ray
Vanessa Robins
Oliver Roche-Newton
Marcel Roeloffzen
Edgardo Roldán-Pensado
Alexander Rolle
Jonathan Rollin
Günter Rote
Ignaz Rutter
Paweł Rzążewski
Noushin Saeedi
Robert Samal
Rik Sarkar
Radmila Sazdanovic
Christian Scheffer
Stefan Schirra
Lena Schlipf
Christiane Schmidt
Hannah Schreiber
André Schulz
Jordan Schupbach
Martina Scolamiero
Eric Sedgwick
Michael Segal
Steven Senger
Micha Sharir
Jonathan Shewchuk
Anastasios Sidiropoulos
Rodrigo Silveira
Francesco Silvestri

Meera Sitharam
Michiel Smid
Alexander Soifer
Chinmay Sonar
Willem Sonke
Jonathan Spreer
Thomas Steinke
Noah Stephens-Davidowitz
Martijn Struijs
Peter Stumpf
Andrew Suk
Konrad Swanepoel
Wai Ming Tai
Martin Tancer
Shin-ichi Tanigawa
Erin Taylor
Francesca Tombari
Justin Toth
Csaba Tóth
Chittaranjan Tripathy
Sarah Tymochko
Mees van de Kerkhof
André van Renssen
Kasturi Varadarajan
Anastasiia Varava
Mikael Vejdemo-Johansson
Santosh Vempala
Kevin Verbeek
Stéphane Vialette
Antoine Vigneron
Oliver Vipond
Žiga Virk
Alexander Wagner
Bei Wang
Di Wang
Haitao Wang
Jiayuan Wang
Qingsong Wang
Yusu Wang
Zhewei Wei
Emo Welzl
Carola Wenk
Manuel Wettstein
Andreas Wiese
Sebastian Wild
Mathijs Wintraecken
David R. Wood
Xiaodong Wu
Xuan Wu
Ge Xia
Jiayi Xian
Allen Xiao
Jie Xue
Katsuhisa Yamanaka
Deshi Ye
Emre Alper Yıldırım
Sang Duk Yoon
Huacheng Yu
Yelena Yuditsky
Joshua Zahl
Meirav Zehavi
Wei Zhan
Guochuan Zhang
Hanrui Zhang
Huaming Zhang
Qin Zhang
Binhai Zhu

An Almost Optimal Bound on the Number of Intersections of Two Simple Polygons

Eyal Ackerman
Department of Mathematics, Physics, and Computer Science, University of Haifa at Oranim, Tivon 36006, Israel
ackerman@sci.haifa.ac.il

Balázs Keszegh
Alfréd Rényi Institute of Mathematics, H-1053 Budapest, Hungary
MTA-ELTE Lendület Combinatorial Geometry Research Group, Budapest, Hungary
keszegh@renyi.hu

Günter Rote
Department of Computer Science, Freie Universität Berlin, Takustr. 9, 14195 Berlin, Germany
rote@inf.fu-berlin.de

Abstract

What is the maximum number of intersections of the boundaries of a simple m-gon and a simple n-gon, assuming general position? This is a basic question in combinatorial geometry, and the answer is easy if at least one of m and n is even. If both m and n are odd, the best known construction has $mn-(m+n)+3$ intersections, and it is conjectured that this is the maximum. However, the best known upper bound is only $mn-(m+\lceil\frac{n}{6}\rceil)$, for $m \geq n$. We prove a new upper bound of $mn-(m+n)+C$ for some constant C, which is optimal apart from the value of C.

2012 ACM Subject Classification Theory of computation → Computational geometry; Mathematics of computing → Combinatoric problems

Keywords and phrases Simple polygon, Ramsey theory, combinatorial geometry

Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.1

Related Version A slightly expanded version is available at `http://arxiv.org/abs/2002.05680`.

Funding *Eyal Ackerman*: The main part of this work was performed during a visit to Freie Universität Berlin which was supported by the Freie Universität Alumni Program.
Balázs Keszegh: 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.

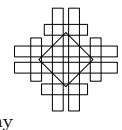
Acknowledgements We thank the reviewers for helpful suggestions.

1 Introduction

To determine the union of two or more geometric objects in the plane is one of the basic computational geometric problems. In strong relation to that, determining the maximum complexity of the union of two or more geometric objects is a basic extremal geometric problem. We study this problem when the two objects are simple polygons.

Let P and Q be two simple polygons with m and n sides, respectively, where $m, n \geq 3$. For simplicity we always assume general position in the sense that no three vertices (of P and Q combined) lie on a line and no two sides (of P and Q combined) are parallel. We are interested in the maximum number of intersections of the boundaries of P and Q.

This problem was first studied in 1993 by Dillencourt, Mount, and Saalfeld [2]. The cases when m or n is even are solved there. If m and n are both even, then every pair of sides may cross and so the answer is mn. Figure 1a shows one of many ways to achieve this number.



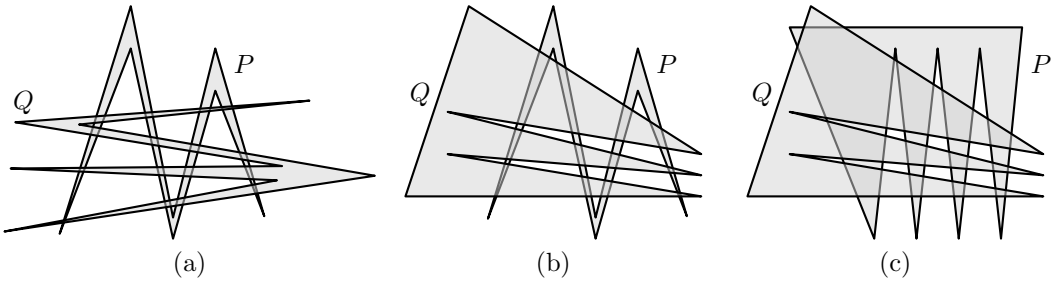



Figure 1 (a) Optimal construction for $m = n = 8$, with $8 \times 8 = 64$ intersections. (b) Optimal construction for $m = 8$, $n = 7$, with $8 \times 6 = 48$ intersections. (c) Lower-bound construction for $m = 9$, $n = 7$. There are $8 \times 6 + 2 = 50$ intersections.

If one polygon, say Q, has an odd number n of sides, no line segment s can be intersected n times by Q, because otherwise each side of Q would have to flip from one side of s to the other side. Thus, each side of the m-gon P is intersected at most $n - 1$ times, for a total of at most $mn - m$ intersections. It is easy to see that this bound is tight when P has an even number of sides, see Figure 1b.

When both m and n are odd, the situation is more difficult; the bound that is obtained by the above argument remains at $mn - \max\{m, n\}$, because the set of m intersections that are necessarily "missing" due to the odd parity of n might conceivably overlap with the n intersections that are "missing" due to the odd parity of m. However, the best known family of examples gives only $mn - (m + n) + 3 = (m - 1)(n - 1) + 2$ intersection points, see Figure 1c. Note that in Figure 1, all vertices of the polygons contribute to the boundary of the union of the polygon areas.

▶ **Conjecture 1.** *Let P and Q be simple polygons with m and n sides, respectively, such that $m, n \geq 3$ are odd numbers. Then there are at most $mn - (m + n) + 3$ intersection points between sides of P and sides of Q.*

In [2] an unrecoverable error appears in a claimed proof of Conjecture 1. Another attempted proof [5] also turned out to have a fault. The only correct improvement over the trivial upper bound is an upper bound of $mn - (m + \lceil \frac{n}{6} \rceil)$ for $m \geq n$, due to Černý, Kára, Král', Podbrdský, Sotáková, and Šámal [1]. We will briefly discuss their proof in Section 2.

We improve the upper bound to $mn - (m + n) + O(1)$, which is optimal apart from an additional constant:

▶ **Theorem 1.** *There is an absolute constant C such that the following holds. Suppose that P and Q are simple polygons with m and n sides, respectively, such that m and n are odd numbers. Then there are at least $m + n - C$ pairs of a side of P and a side of Q that do not intersect. Hence, there are at most $mn - (m + n) + C$ intersections.*

The value of the constant C that we obtain in our proof is around $2^{2^{67}}$. We did not make a large effort to optimize this value, and obviously, there is ample space for improvement.

2 Overview of the proof

First we establish the crucial statement that the odd parity of m and n allows us to *associate* to any two consecutive sides of one polygon a pair of consecutive sides of the other polygon with a restricted intersection pattern among the four involved sides (Lemma 5 and Figure 5). This is the only place where we use the odd parity of the polygons.

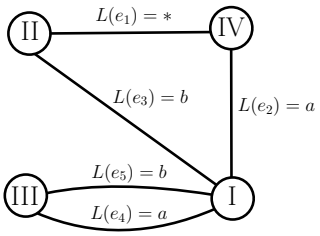


Figure 2 The edge-labeled multigraph G_0 in Proposition 2.

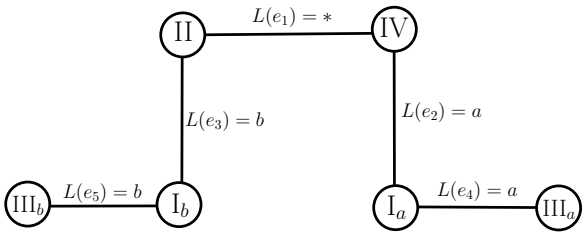


Figure 3 The unfolded graph G_0'.

A simple observation (Observation 3) relates the bound on C in Theorem 1 to the number of connected components of the bipartite "disjointness graph" between the polygon sides of P and Q. Our goal is therefore to show that there are few connected components.

We proceed to consider *two* pairs of associated pairs of sides (4 consecutive pairs with 8 sides in total). Unless they form a special structure, they cannot belong to four different connected components (Lemma 7). (Four is the maximum number of components that they could conceivably have.) The proof involves a case distinction with a moderate amount of cases. This structural statement allows us to reduce the bound on the number of components by a constant factor, and thereby, we can already improve the best previous result on the number of intersections (Proposition 9 in Section 6).

Finally, to get a constant bound on the number of components, our strategy is to use Ramsey-theoretic arguments like the Erdős–Szekeres Theorem on caps and cups or the pigeonhole principle (see Section 7) in order to impose additional structure on the configurations that we have to analyze. This is the place in the argument where we give up control over the constant C in exchange for useful properties that allow us to derive a contradiction. This eventually boils down again to a moderate number of cases (Section 8.2).

By contrast, the proof of the bound $mn-(m+\lceil\frac{n}{6}\rceil)$ for $m \geq n$ proceeds more locally. The core of the argument [1, Lemma 3], which is proved by case distinction, is that it is impossible to have 6 consecutive sides of one polygon together with 6 distinct sides of the other polygon forming a perfect matching in the disjointness graph. This statement is used to bound the number of components of the disjointness graph. (Lemma 8 below uses a similar argument.)

3 An auxiliary lemma on closed odd walks

We begin with the following seemingly unrelated claim concerning a specific small edge-labeled multigraph. Let $G_0 = (V_0, E_0)$ be the undirected multigraph shown in Figure 2. It has four nodes $V_0 = \{\mathrm{I}, \mathrm{II}, \mathrm{III}, \mathrm{IV}\}$ and five edges $E_0 = \{e_1 = \{\mathrm{II}, \mathrm{IV}\}, e_2 = \{\mathrm{I}, \mathrm{IV}\}, e_3 = \{\mathrm{I}, \mathrm{II}\}, e_4 = \{\mathrm{I}, \mathrm{III}\}, e_5 = \{\mathrm{I}, \mathrm{III}\}\}$. Every edge $e_i \in E_0$ has a label $L(e_i) \in \{a, b, *\}$ as follows: $L(e_1) = *$, $L(e_2) = L(e_4) = a$, $L(e_3) = L(e_5) = b$.

▶ **Proposition 2.** *If W is a closed walk in G_0 of odd length, then W contains two cyclically consecutive edges of labels a and b.*

Proof. Suppose for contradiction that W does not contain two consecutive edges of labels a and b. Since W cannot switch between the a-edges and the b-edges in I or III, we can split I (resp., III) into two nodes I_a and I_b (resp., III_a and III_b) such that every a-labeled edge that is incident to I (resp., III) in G_0 becomes incident to I_a (resp., III_a) and every b-labeled edge

that is incident to I (resp., III) in G_0 becomes incident to I_b (resp., III_b). In the resulting graph G_0', which is shown in Figure 3, we can find a closed walk W' that corresponds to W and that uses the edges with the same name as W. Since G_0' is a path, every closed walk has even length. Thus, W cannot have odd length. ◀

4 General assumptions and notations

Let P and Q be two simple polygons with sides $p_0, p_1, \ldots, p_{m-1}$ and $q_0, q_1, \ldots, q_{n-1}$. We assume that $m \geq 3$ and $n \geq 3$ are odd numbers. Addition and subtraction of indices is modulo m or n, respectively. We consider the sides p_i and q_j as closed line segments. The condition that the polygon P is simple means that its edges are pairwise disjoint except for the unavoidable common endpoints between *consecutive* sides p_i and p_{i+1}. Throughout this paper, unless stated otherwise, we regard a polygon as a piecewise linear closed curve, and we disregard the region that it encloses. Thus, by intersections between P and Q, we mean intersection points between the polygon *boundaries.*

As mentioned, we assume that the vertices of P and Q are in general position (no three of them on a line), and so every intersection point between P and Q is an interior point of two polygon sides.

The disjointness graph. As in [1], our basic tool of analysis is the *disjointness graph* of P and Q, which we denote by $G^{\mathrm{D}} = (V^{\mathrm{D}}, E^{\mathrm{D}})$. (Its original name in [1] is *non-intersection graph.*) It is a bipartite graph with node set $V^{\mathrm{D}} = \{p_0, p_1, \ldots, p_{m-1}\} \cup \{q_0, q_1, \ldots, q_{n-1}\}$ and edge set $E^{\mathrm{D}} = \{\, (p_i, q_j) \mid p_i \cap q_j = \emptyset \,\}$. (Since we are interested in the situation where almost all pairs of edges intersect, the disjointness graph is more useful than its more commonly used complement, the intersection graph.)

Our goal is to bound from above the number of connected components of G^{D}:

▶ **Observation 3.** *If G^{D} has at most C connected components, then G^{D} has at least $m+n-C$ edges. Thus, there are at least $m+n-C$ pairs of a side of P and a side of Q that do not intersect, and there are at most $mn-(m+n)+C$ crossings between P and Q.* ◀

Geometric notions. Let s and s' be two line segments. We denote by $\ell(s)$ the line through s and by $I(s, s')$ the intersection of $\ell(s)$ and $\ell(s')$ see Figure 4. We say that s and s' are *avoiding* if neither of them contains $I(s, s')$. (This requirement is stronger than just disjointness.) If s and s' are avoiding or share an endpoint, we denote by $\vec{r}_{s'}(s)$ the ray from $I(s, s')$ to infinity that contains s, and by $\vec{r}_s(s')$ the ray from $I(s, s')$ to infinity that contains s'. Moreover, we denote by $\mathrm{Cone}(s, s')$ the convex cone with apex $I(s, s')$ between these two rays.

▶ **Observation 4.** *If a segment s'' that does not go through $I(s, s')$ has one of its endpoints in the interior of $\mathrm{Cone}(s, s')$, then s'' cannot intersect both $\vec{r}_{s'}(s)$ and $\vec{r}_s(s')$. In particular, it cannot intersect both s and s'.* ◀

For a polygon side s of P or Q, $\mathrm{CC}(s)$ denotes the connected component of the disjointness graph G^{D} to which s belongs.

4.1 Associated pairs of consecutive sides

▶ **Lemma 5.** *Let p_a and p_b be two sides of P that are either consecutive or avoiding such that $\mathrm{CC}(p_a) \neq \mathrm{CC}(p_b)$. Then there are two consecutive sides $q_i, q_{i\pm1}$ of Q such that $(p_a, q_i), (p_b, q_{i\pm1}) \in E^{\mathrm{D}}$ and $(p_a, q_{i\pm1}), (p_b, q_i) \notin E^{\mathrm{D}}$. Furthermore, $I(p_a, p_b) \in \mathrm{Cone}(q_i, q_{i\pm1})$ or $I(q_i, q_{i\pm1}) \in \mathrm{Cone}(p_a, p_b)$.*

The sign "±" is needed since we do not know which of the consecutive sides intersects p_i and is disjoint from p_{i+1}.

Proof. We may assume without loss of generality that $I(p_a, p_b)$ is the origin, p_a lies on the positive x-axis and the interior of p_b is above the x-axis. The lines $\ell(p_a)$ and $\ell(p_b)$ partition the plane into four convex cones ("quadrants"). Denote them in counterclockwise order by $\mathrm{I}, \mathrm{II}, \mathrm{III}, \mathrm{IV}$, starting with $\mathrm{I} = \mathrm{Cone}(p_a, p_b)$, see Figure 4. Every side of Q must intersect p_a

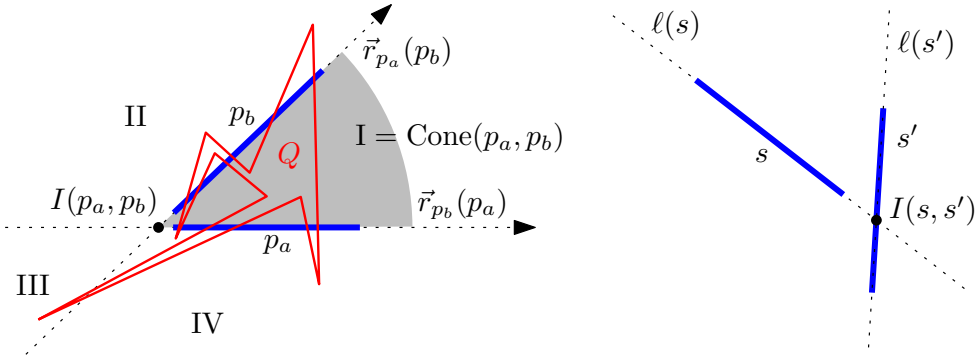


Figure 4 How an odd polygon Q can intersect two segments. The segments p_a and p_b are avoiding, whereas s and s' are disjoint but non-avoiding. In this situation, we say that s *stabs* s'.

or p_b (maybe both), since $\mathrm{CC}(p_a) \neq \mathrm{CC}(p_b)$. One can now check that traversing the sides of Q in order generates a closed walk W in the graph G_0 of Figure 2. For example, a side of Q that we traverse from its endpoint in I to its endpoint in III and that intersects p_a corresponds to traversing the edge $e_4 = \{\mathrm{I}, \mathrm{III}\}$ from I to III, whose label is $L(e_4) = a$. We do not care which of p_a and p_b are crossed when we move between II and IV.

It follows from Proposition 2 that Q has two consecutive sides $q_i, q_{i\pm1}$ such that q_i intersects p_b and does not intersect p_a, while $q_{i\pm1}$ intersects p_a and does not intersect p_b. Hence, $(p_a, q_i), (p_b, q_{i\pm1}) \in E^{\mathrm{D}}$ and $(p_a, q_{i\pm1}), (p_b, q_i) \notin E^{\mathrm{D}}$. Furthermore, $I(q_i, q_{i\pm1})$ must be either in I or III as these are the only nodes in G_0 that are incident both to an edge labeled a and an edge labeled b. In the latter case $I(p_a, p_b) \in \mathrm{Cone}(q_i, q_{i\pm1})$, and in the former case $I(q_i, q_{i\pm1}) \in \mathrm{Cone}(p_a, p_b)$. ◀

Let p_i, p_{i+1} be two sides of P such that $\mathrm{CC}(p_i) \neq \mathrm{CC}(p_{i+1})$. Then by Lemma 5 there are sides $q_j, q_{j\pm1}$ of Q such that $(p_i, q_j), (p_{i+1}, q_{j\pm1}) \in E^{\mathrm{D}}$. We say that the pair $q_j, q_{j\pm1}$ is *associated* to p_i, p_{i+1}. By Lemma 5 we have $I(q_j, q_{j\pm1}) \in \mathrm{Cone}(p_i, p_{i+1})$ or $I(p_i, p_{i+1}) \in \mathrm{Cone}(q_j, q_{j\pm1})$. If the first condition holds we say that p_i, p_{i+1} is *hooking* and $q_j, q_{j\pm1}$ is *hooked*, see Figure 5. In the second case we say that p_i, p_{i+1} is *hooked* and $q_j, q_{j\pm1}$ is *hooking*. Note that it is possible that a pair of consecutive sides is both hooking and hooked (with respect to two different pairs from the other polygon or even with respect to a single pair, as in Figure 5c).

▶ **Observation 6** (The Axis Property). *If the pair p_i, p_{i+1} and the pair $q_j, q_{j\pm1}$ are associated such that $(p_i, q_j), (p_{i+1}, q_{j\pm1}) \in E^{\mathrm{D}}$, then the line through $I(p_i, p_{i+1})$ and $I(q_j, q_{j\pm1})$ separates p_i and $q_{j\pm1}$ on the one side from p_{i+1} and q_j on the other side.* ◀

We call this line the *axis* of the associated pairs. In our figures it appears as a dotted line when it is shown.

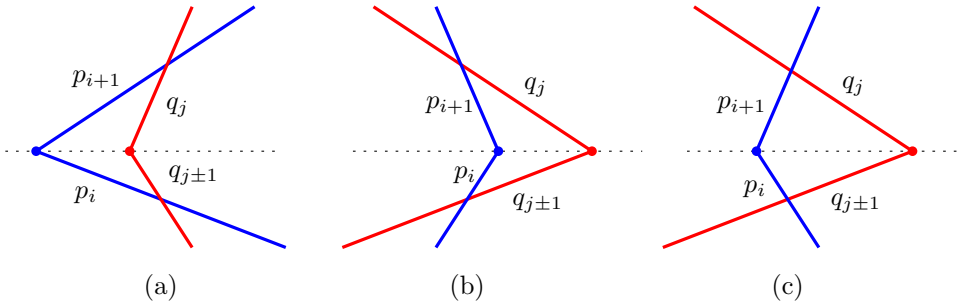


Figure 5 Hooking and hooked pairs of consecutive sides. (a) The pair p_i, p_{i+1} is hooking and the associated pair $q_j, q_{j\pm1}$ is hooked. (b) vice versa. (c) Both pairs are both hooking and hooked.

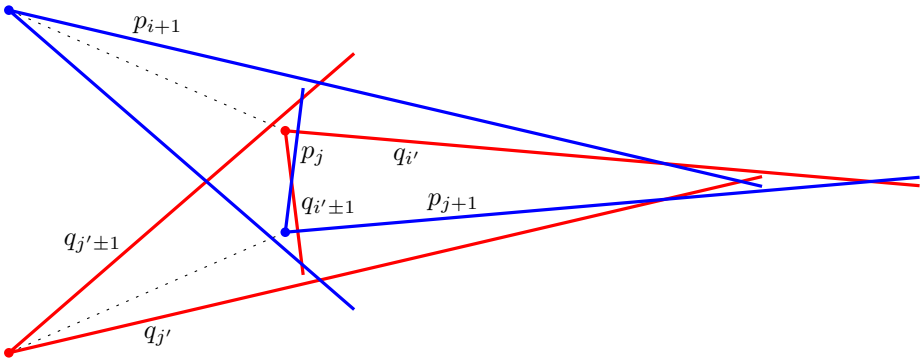


Figure 6 The pair p_i, p_{i+1} is hooking with respect to the pair $q_{i'}, q_{i'\pm1}$, and p_j, p_{j+1} is hooked with respect to $q_{j'}, q_{j'\pm1}$.

5 The principal structure lemma about pairs of associated pairs

▶ **Lemma 7.** *Let $p_i, p_{i+1}, p_j, p_{j+1}$ be two pairs of consecutive sides of P that belong to four different connected components of G^{D}. Then it is impossible that both p_i, p_{i+1} and p_j, p_{j+1} are hooked or that both pairs are hooking.*

Figure 6 shows a scenario with four different components, together with the associated pairs of Q. The combinatorial structure of such a configuration is unique up to relabeling.

Proof. Suppose first that both pairs p_i, p_{i+1} and p_j, p_{j+1}, are hooking and let $q_{i'}, q_{i'\pm1}$ and $q_{j'}, q_{j'\pm1}$ be their associated (hooked) pairs such that: $(p_i, q_{i'}), (p_{i+1}, q_{i'\pm1}) \in E^{\mathrm{D}}$, $(p_j, q_{j'}), (p_{j+1}, q_{j'\pm1}) \in E^{\mathrm{D}}$, $I(q_{i'}, q_{i'\pm1}) \in \mathrm{Cone}(p_i, p_{i+1})$ and $I(q_{j'}, q_{j'\pm1}) \in \mathrm{Cone}(p_j, p_{j+1})$.

For better readability, we rename p_i, p_{i+1} and $q_{i'}, q_{i'\pm1}$ as a, b and A, B, and we rename p_j, p_{j+1} and $q_{j'}, q_{j'\pm1}$ as a', b' and A', B'. The small letters denote sides of P and the capital letters denote sides of Q. In the new notation, a, b are consecutive sides of P with an associated pair A, B of consecutive sides of Q, and a', b' are two other consecutive sides of P with an associated pair A', B' of consecutive sides of Q. The disjointness graph G^{D} contains the edges $(a, A), (b, B), (a', A'), (b', B')$. Since a, b, a', b' belong to different connected components of G^{D}, it follows that the nodes A, B, A', B', to which they are connected, belong to the same four different connected components. There can be no more edges among these eight nodes, and they induce a matching in G^{D}. One can remember as a rule that every side of P intersects every side of Q among the eight involved sides, except when their names

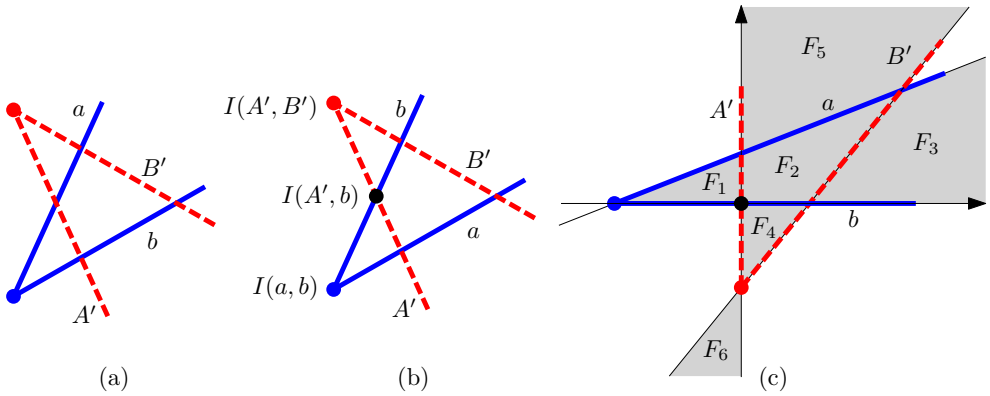


Figure 7 Normalizing the position of a, b, A', B'.

differ only in their capitalization. In particular, each of A' and B' intersects each of a and b and hence they must lie as in Figure 7a. To facilitate the future discussion, we will now normalize the positions of these four sides.

We first ensure that the intersection $I(A', b)$ is directly adjacent to the two polygon vertices $I(a, b)$ and $I(A', B')$ in the arrangement of the four sides, as shown in Figure 7b. This can be achieved by swapping the labels a, A with the labels b, B if necessary, and by independently swapping the labels a', A' with b', B' if necessary. Our assumptions are invariant under these swaps.

By an affine transformation we may finally assume that $I(A', b)$ is the origin; b lies on the x-axis and is directed to the right; and A' lies on the y-axis and is directed upwards. Then a has a positive slope and its interior is in the upper half-plane, and B' has a positive slope and its interior is to the right of the y-axis, see Figure 7c.

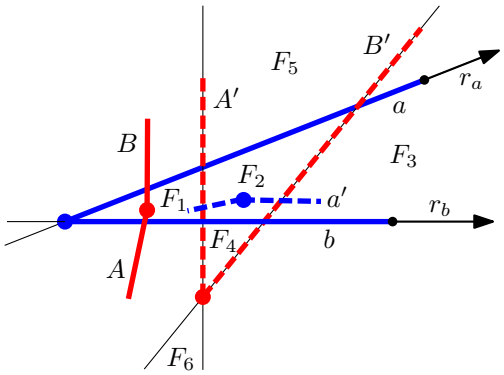


Figure 8 Case 1: $I(A, B) \in F_1$, $I(a', b') \in F_2$.

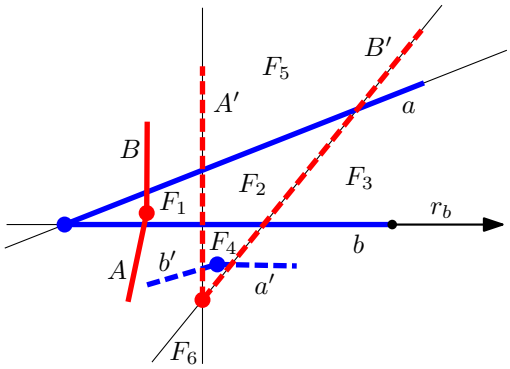


Figure 9 Case 2: $I(A, B) \in F_1$, $I(a', b') \in F_4$.

The arrangement of the lines through a, b, A', B' has 11 faces, some of which are marked as $F_1, \ldots, F_6$ in Figure 7. Our current assumption is that both a, b and a', b' are hooking: The hooking of a, b means that $I(A, B) \in \text{Cone}(a, b) = F_1 \cup F_2 \cup F_3$. By the Axis Property (Observation 6), the line through $I(A', B')$ and $I(a', b')$ must separate A' from B'. Therefore, the vertex $I(a', b')$ can lie only in $F_2 \cup F_4 \cup F_5 \cup F_6$. Thus, based on the faces that contain $I(A, B)$ and $I(a', b')$, there are 12 cases to consider. Some of these cases are symmetric, and all can be easily dismissed, as follows.

In the figures, the four sides a', b', A', B', which are associated to the second associated pair are dashed. All dashed sides of one polygon must intersect all solid sides of the other polygon.

1. $I(A,B) \in F_1$ and $I(a',b') \in F_2$, see Figure 8 (symmetric to $I(A,B) \in F_2$ and $I(a',b') \in F_4$). Let r_a (resp., r_b) be the ray on $\ell(a)$ (resp., $\ell(b)$) that goes from the right endpoint of a (resp., b) to the right. Since a' is not allowed to cross b, the only way for a' to intersect A is by crossing r_b. Similarly, in order to intersect B, a' has to cross r_a. However, it cannot intersect both r_a and r_b, by Observation 4.
   Since we did not use the assumption that A, B are hooked, the analysis holds for the symmetric Case 6, $I(A,B) \in F_2$ and $I(a',b') \in F_4$, as well.
2. $I(A,B) \in F_1$ and $I(a',b') \in F_4$, see Figure 9. Since a' is not allowed to cross b, the only way for a' to intersect B is by crossing r_b. However, in this case a' cannot intersect A.

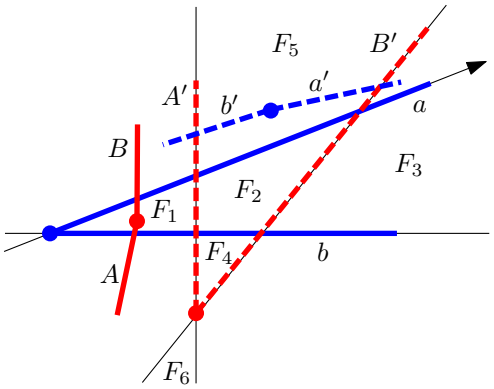
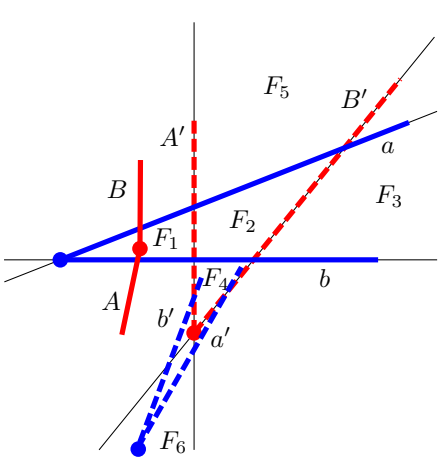
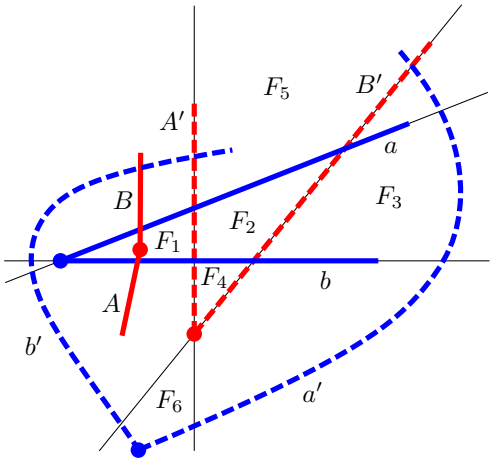


Figure 10 Case 3: $I(A,B) \in F_1$ and $I(a',b') \in F_5$.

3. $I(A,B) \in F_1$ and $I(a',b') \in F_5$, see Figure 10 (symmetric to $I(A,B) \in F_3$ and $I(a',b') \in F_4$). Both a' and b' must intersect A, and they have to go below the line $\ell(b)$ to do so. However, a' can only cross $\ell(b)$ to the right of b, and b' can only cross $\ell(b)$ to the left of b, and therefore they cross A from different sides. This is impossible, because a' and b' start from the same point.
4. $I(A,B) \in F_1$ and $I(a',b') \in F_6$. If one of the polygon sides a' and b' has an endpoint in F_4 (see Figure 11a), then this side cannot intersect B. So assume otherwise, see Figure 11b. The side a' intersects B' and is disjoint from A', while b' is disjoint from B' and intersects A'. (Due to space limitation some line segments are drawn schematically as curves.) Thus, each of a' and b' has an endpoint in $F_2 \cup F_5$. But then $I(A,B) \in \text{Cone}(a',b')$ and it follows from Observation 4 that neither A nor B can intersect both a' and b'.
5. $I(A,B) \in F_2$ and $I(a',b') \in F_2$, see Figure 12. Since a', b' is hooking, $I(A',B') \in \text{Cone}(a',b')$, and the line segments a', b', A', b, B' enclose a convex pentagon. The polygon side A must intersect b, a' and b', but it is restricted to $F_2 \cup F_4$. It follows that A must intersect three sides of the pentagon, which is impossible. (This is in fact the only place where we need the assumption that a', b' is hooking.)
6. $I(A,B) \in F_2$ and $I(a',b') \in F_4$. This is symmetric to Case 1.
7. $I(A,B) \in F_2$ and $I(a',b') \in F_5$, see Figure 13 (symmetric to $I(A,B) \in F_3$ and $I(a',b') \in F_2$). Then A is restricted to $F_2 \cup F_4$, while a' and b' do not intersect F_2 and F_4. Therefore A can intersect neither a' nor b'.
8. $I(A,B) \in F_2$ and $I(a',b') \in F_6$. This case is very similar to Case 4, where $I(A,B) \in F_1$ and $I(a',b') \in F_6$, see Figure 11. If one of the polygon sides a' and b' has an endpoint in F_4, then it cannot intersect B. Otherwise, $I(A,B) \in \text{Cone}(a',b')$ and therefore, neither A nor B can intersect both a' and b'.
9. $I(A,B) \in F_3$ and $I(a',b') \in F_2$. This is symmetric to Case 7.



(a) At least one of the sides a' and b' has an endpoint in F_4.



(b) None of the sides a' and b' has an endpoint in F_4.

Figure 11 Case 4: $I(A,B) \in F_1$ (or $I(A,B) \in F_2$, which is similar) and $I(a',b') \in F_6$.

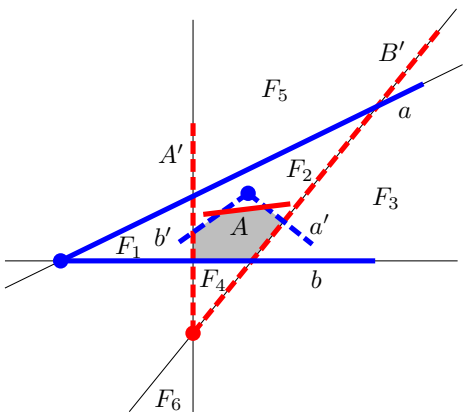


Figure 12 Case 5: $I(A,B) \in F_2$, $I(a',b') \in F_2$.

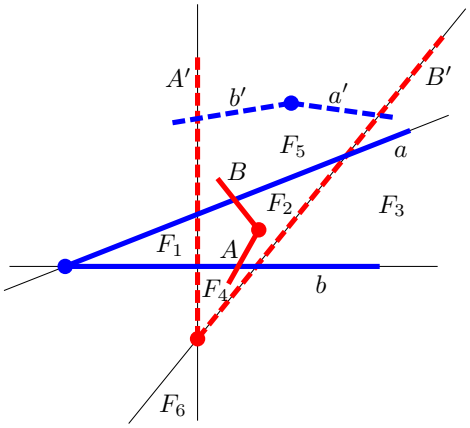


Figure 13 Case 7: $I(A,B) \in F_2$, $I(a',b') \in F_5$.

10. $I(A,B) \in F_3$ and $I(a',b') \in F_4$. This is symmetric to Case 3.
11. $I(A,B) \in F_3$ and $I(a',b') \in F_5$, see Figure 14. Then the intersection of b' and A can lie only in the lower left quadrant. It follows that the triangle whose vertices are $I(a',b')$, $I(a',A)$ and $I(A,b')$ contains a and does not contain $I(A,B)$. This in turn implies that B cannot intersect both b' and a, without intersecting B'.
12. $I(A,B) \in F_3$ and $I(a',b') \in F_6$, see Figure 15. As in Case 4, we may assume that neither a' nor b' has an endpoint in F_4, since then this side could not intersect B. We may also assume that $I(A,B) \notin \mathrm{Cone}(a',b')$ for otherwise neither A nor B intersects both of a' and b', according to Observation 4. If a' has an endpoint in F_2, then it cannot intersect B (see Figure 15a). Otherwise, if a' has an endpoint in F_5, then B cannot intersect b' (Figure 15b).

We have finished the case that a, b and a', b' are hooking. Suppose now that a, b and a', b' are hooked, with respect to some pairs A, B and A', B'. Then A, B is hooking with respect to a, b and A', B' is hooking with respect to a', b'. Recall that A, B, A' and B' belong to four different connected components. Hence, this case can be handled as above, after exchanging the capital letters with the small letters (i.e., exchanging P and Q). ◀

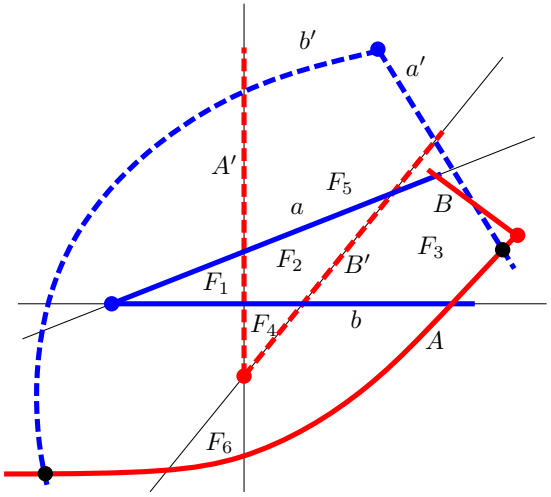
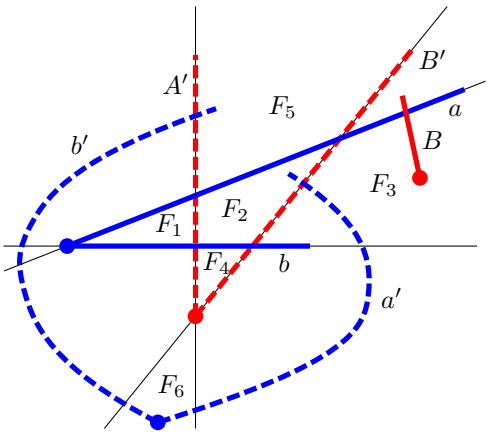
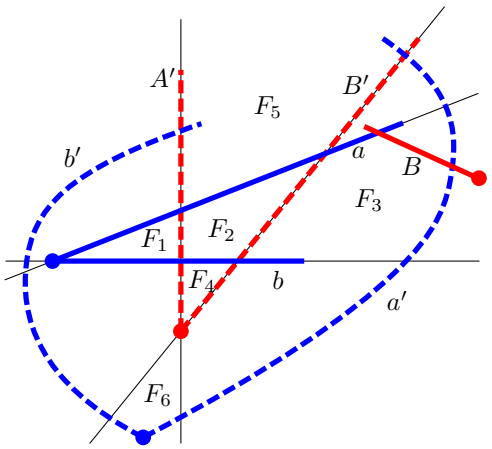


Figure 14 Case 11: $I(A,B) \in F_3$ and $I(a',b') \in F_5$.



(a) If a' has an endpoint in F_2, then it cannot intersect B.



(b) If a' has an endpoint in F_5, then B cannot intersect b'.

Figure 15 Case 12: $I(A,B) \in F_3$ and $I(a',b') \in F_6$.

6 A weaker bound

The principal structure lemma is already powerful enough to get an improvement over the previous best bound:

▶ **Lemma 8.** *G^{D} has at most $(n+5)/2$ connected components.*

Proof. Partition the sides $q_0, q_1, \ldots, q_{n-1}$ of Q into $(n-1)/2$ disjoint pairs q_{2i}, q_{2i+1}, discarding the last side q_{n-1}. Let H_+ denote the subset of these pairs that are hooked. Suppose first that this set contains some pair q_{2i_0}, q_{2i_0+1} of sides that are in two different connected components. Combining q_{2i_0}, q_{2i_0+1} with any of the remaining pairs q_{2i}, q_{2i+1} of H_+, Lemma 7 tells us that the sides q_{2i} and q_{2i+1} must either belong to the same connected component, or one of them must belong to $\mathrm{CC}(q_{2i_0})$ or $\mathrm{CC}(q_{2i_0+1})$. In other words, each remaining pair contributes at most one "new" connected component, and it follows that the sides in H_+ belong to at most $|H_+|+1$ connected components. This conclusion holds also in the case that H_+ contains no pair q_{2i_0}, q_{2i_0+1} of sides that are in different connected components.

The same argument works for the complementary subset H_- of pairs that are not hooked, but hooking. Along with $\mathrm{CC}(q_{n-1})$ there are at most $(|H_+|+1)+(|H_-|+1)+1 = (n-1)/2+3 = (n+5)/2$ components. ◀

Together with Observation 3, this already improves the previous bound $mn-(m+\lceil\frac{n}{6}\rceil)$ for a large range of parameters, namely when $m \geq n \geq 11$:

▶ **Proposition 9.** *Let P and Q be simple polygons with m and n sides, respectively, such that m and n are odd and $m \geq n \geq 3$. Then there are at most $mn-(m+\frac{n-5}{2})$ intersection points between P and Q.* ◀

7 Ramsey-theoretic tools

We recall some classic results. A tournament is a directed graph that contains between every pair of nodes x, y either the arc (x, y) or the arc (y, x) but not both. A tournament is *transitive* if for every three nodes x, y, z the existence of the arcs (x, y) and (y, z) implies the existence of the arc (x, z). Equivalently, the nodes can be ordered on a line such that all arcs are in the same direction. The following is easy to prove by induction.

▶ **Lemma 10** (Erdős and Moser [3]). *Every tournament on a node set V contains a transitive sub-tournament on $1+\lfloor\log_2|V|\rfloor$ nodes.*

Proof. Choose $v \in V$ arbitrarily, and let $N \subseteq V-\{v\}$ with $|N| \geq (|V|-1)/2$ be the set of in-neighbors of v or the set of out-neighbors of v, whichever is larger. Then v together with a transitive sub-tournament of N gives a transitive sub-tournament of size one larger. ◀

A set of points $p_1, p_2, \ldots, p_r$ in the plane sorted by x-coordinates (and with distinct x-coordinates) forms an *r-cup* (resp., *r-cap*) if p_i is below (resp., above) the line through p_{i-1} and p_{i+1} for every i with $1 < i < r$.

▶ **Theorem 11** (Erdős–Szekeres Theorem for caps and cups in point sets [4]). *For any two integers $r \geq 2$ and $s \geq 2$, the value $ES(r,s) := \binom{r+s-4}{r-2}$ fulfills the following statement:*

Suppose that P is a set of $ES(r,s)+1$ points in the plane with distinct x-coordinates such that no three points of P lie on a line. Then P contains an r-cup or an s-cap.

Moreover, $ES(r,s)$ is the smallest value that fulfills the statement. ◀

A similar statement holds for lines by the standard point-line duality. A set of lines $\ell_1, \ell_2, \ldots, \ell_r$ sorted by slope forms an *r-cup* (resp., *r-cap*) if ℓ_{i-1} and ℓ_{i+1} intersect below (resp., above) ℓ_i for every $1 < i < r$.

▶ **Theorem 12** (Erdős–Szekeres Theorem for lines). *For the numbers $ES(r,s)$ from Theorem 11, the following statement holds for any two integers $r \geq 2$ and $s \geq 2$:*

If L is a set of $ES(r,s)+1$ non-vertical lines in the plane no two of which are parallel and no three of which intersect at a common point, then L contains an r-cup or an s-cap. ◀

▶ **Theorem 13** (Erdős–Szekeres Theorem for monotone subsequences [4]). *For any integer $r \geq 0$, a sequence of r^2+1 distinct numbers contains either an increasing subsequence of length $r+1$ or a decreasing subsequence of length $r+1$.* ◀

8 Proof of Theorem 1

8.1 Imposing more structure on the examples

Going back to the proof of Theorem 1, recall that in light of Observation 3 it is enough to prove that G^{D}, the disjointness graph of P and Q, has at most constantly many connected components. We will use the following constants: $C_6 := 6$; $C_5 := (C_6)^2 + 1 = 37$; $C_4 := ES(C_5, C_5) + 1 = \binom{70}{35} + 1 = 112{,}186{,}277{,}816{,}662{,}845{,}433 < 2^{70}$; $C_3 := 2^{C_4 - 1}$; $C_2 := C_3 + 5$; $C_1 := 8C_2$; $C := C_1 - 1 < 2^{2^{70}}$.

We claim that G^{D} has at most C connected components. Suppose that G^{D} has at least $C_1 = C + 1$ connected components, numbered as $1, 2, \ldots, C_1$. For each connected component j, we find two consecutive sides q_{i_j}, q_{i_j+1} of Q such that $\mathrm{CC}(q_{i_j}) = j$ and $\mathrm{CC}(q_{i_j+1}) \neq j$. We call q_{i_j} the *primary* side and q_{i_j+1} the *companion* side of the pair. We take these C_1 consecutive pairs in their cyclic order along Q and remove every second pair. This ensures that the remaining $C_1/2$ pairs are disjoint in the sense that no side of Q belongs to two different pairs.

We apply Lemma 5 to each of the remaining $C_1/2$ pairs q_{i_j}, q_{i_j+1} and find an associated pair $p_{k_j}, p_{k_j \pm 1}$ such that $(q_{i_j}, p_{k_j}), (q_{i_j+1}, p_{k_j \pm 1}) \in E^{\mathrm{D}}$. Therefore, $\mathrm{CC}(q_{i_j}) = \mathrm{CC}(p_{k_j})$ and $\mathrm{CC}(q_{i_j+1}) = \mathrm{CC}(p_{k_j \pm 1}) \neq \mathrm{CC}(q_{i_j})$. Again, we call p_{k_j} the primary side and $p_{k_j \pm 1}$ the companion side. We delete half of the pairs $p_{k_j}, p_{k_j \pm 1}$ in cyclic order along P, along with their associated pairs from Q, and thus we ensure that the remaining $C_1/4$ pairs are disjoint also on P.

At least $C_1/8$ of the remaining pairs q_{i_j}, q_{i_j+1} are hooking or at least $C_1/8$ of them are hooked. We may assume that at least $C_2 = C_1/8$ of the pairs q_{i_j}, q_{i_j+1} are hooking with respect to their associated pair, $p_{k_j}, p_{k_j \pm 1}$, for otherwise, $p_{k_j}, p_{k_j \pm 1}$ is hooking with respect to q_{i_j}, q_{i_j+1} and we may switch the roles of P and Q. Let us denote by Q_2 the set of C_2 hooking consecutive pairs $(q_{i_j}, q_{i_j \pm 1})$ at which we have arrived. (Because of the potential switch, we have to denote the companion side by $q_{i_j \pm 1}$ instead of q_{i_j+1} from now on.)

By construction, all C_2 primary sides q_{i_j} of these pairs belong to distinct components. We now argue that all C_2 adjacent companion sides $q_{i_j \pm 1}$ with at most one exception lie in the same connected component, provided that $C_2 \geq 4$.

We model the problem by a graph whose nodes are the connected components of G^{D}. For each pair $q_{i_j}, q_{i_j \pm 1}$, we insert an edge between $\mathrm{CC}(q_{i_j})$ and $\mathrm{CC}(q_{i_j \pm 1})$. The result is a multigraph with C_2 edges and without loops. Two disjoint edges would represent two consecutive pairs of the form $(q_{i_j}, q_{i_j \pm 1})$ whose four sides are in four distinct connected components, but this is a contradiction to Lemma 7. Thus, the graph has no two disjoint edges, and such graphs are easily classified: they are the triangle (cycle on three vertices) and the star graphs K_{1t}, possibly with multiple edges. Overall, the graph involves at least $C_2 \geq 4$ distinct connected components $\mathrm{CC}(q_{i_j})$, and therefore the triangle graph is excluded. Let v be the central vertex of the star. There can be at most one j with $\mathrm{CC}(q_{i_j}) = v$, and we discard it. All other sides q_{i_j} have $\mathrm{CC}(q_{i_j}) \neq v$, and therefore $\mathrm{CC}(q_{i_j \pm 1})$ must be the other endpoint of the edge, that is, v.

In summary, we have found $C_2 - 1$ adjacent pairs $q_{i_j}, q_{i_j \pm 1}$ with the following properties.

- The primary sides q_{i_j} belong to $C_2 - 1$ distinct components.
- All companion sides $q_{i_j \pm 1}$ belong to the same component, distinct from the other $C_2 - 1$ components.
- All $2C_2 - 2$ sides of the pairs $q_{i_j}, q_{i_j \pm 1}$ are distinct.
- Each $q_{i_j}, q_{i_j \pm 1}$ is hooking with respect to an associated pair $p_{k_j}, p_{k_j \pm 1}$.
- All $2C_2 - 2$ sides of the pairs $p_{k_j}, p_{k_j \pm 1}$ are distinct.

Let us denote by Q_2' the set of $C_2 - 1$ sides q_{i_j}.

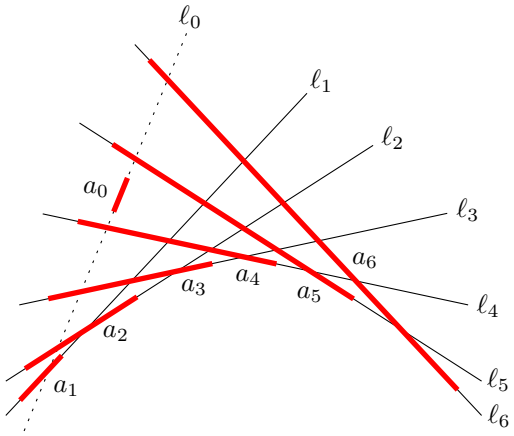


Figure 16 The seven sides $a_0, a_1, \ldots, a_6$. The lines $\ell_0, \ldots, \ell_6$ form a 7-cup.

▶ **Proposition 14.** *There are no six distinct sides $q_a, q_b, q_c, q_d, q_e, q_f$ among the $C_2 - 1$ sides $q_{i_j} \in Q_2'$ such that q_a, q_b are avoiding or consecutive, q_c, q_d are avoiding or consecutive, and q_e, q_f are avoiding or consecutive.*

Proof. Suppose for contradiction that six such sides exist. It follows from Lemma 5 that there are two consecutive sides $p_{a'}$ and $p_{b'}$ of P such that $\mathrm{CC}(p_{a'}) = \mathrm{CC}(q_a)$ and $\mathrm{CC}(p_{b'}) = \mathrm{CC}(q_b)$.

Similarly, we find a pair of consecutive sides $p_{c'}$ and $p_{d'}$ of P such that $\mathrm{CC}(p_{c'}) = \mathrm{CC}(q_c)$ and $\mathrm{CC}(p_{d'}) = \mathrm{CC}(q_d)$, and the same story for e and f. By the pigeonhole principle, two of the three consecutive pairs $(p_{a'}, p_{b'})$, $(p_{c'}, p_{d'})$, $(p_{e'}, p_{f'})$ are hooking or two of them are hooked. This contradicts Lemma 7. ◀

Define a complete graph whose nodes are the $C_2 - 1$ sides $q_{i_j} \in Q_2'$, and color an edge (q_{i_j}, q_{i_k}) red if q_{i_j} and q_{i_k} are avoiding or consecutive and blue otherwise. Proposition 14 says that this graph contains no red matching of size three. This means that we can get rid of all red edges by removing at most 4 nodes. To see this, pick any red edge and remove its two nodes from the graph. If any red edge remains, remove its two nodes. Then all red edges are gone, because otherwise we would find a matching with three red edges.

We conclude that there is a blue clique of size $C_3 = C_2 - 5$, i.e., there is a set $Q_3 \subset Q_2'$ of C_3 polygon sides among the $C_2 - 1$ sides $q_{i_j} \in Q_2'$ that are pairwise non-avoiding and disjoint, i.e., they do not share a common endpoint.

Our next goal is to find a subset of 7 segments in Q_3 that are arranged as in Figure 16. To define this precisely, we say for two segments s and s' that s *stabs* s' if $I(s, s') \in s'$, see Figure 4. Among any two non-avoiding and non-consecutive sides s and s', either s stabs s' or s' stabs s, but not both. Define a tournament T whose nodes are the C_3 sides $q_{i_j} \in Q_3$, and the arc between each pair of nodes is oriented towards the stabbed side. It follows from Lemma 10 that T has a transitive sub-tournament of size $1 + \lfloor \log_2 C_3 \rfloor = C_4$.

Furthermore, since $C_4 = ES(C_5, C_5) + 1$, it follows from Theorem 12 that there is a subset of C_5 sides such that the lines through them form a C_5-cup or a C_5-cap. By a vertical reflection if needed, we may assume that they form a C_5-cup.

We now reorder these C_5 sides q_{i_j} of Q in stabbing order, according to the transitive sub-tournament mentioned above. By the Erdős–Szekeres Theorem on monotone subsequences (Theorem 13), there is a subsequence of size $C_6 + 1 = \sqrt{C_5 - 1} + 1 = 7$ such that their slopes form a monotone sequence. By a horizontal reflection if needed, we may assume that they have decreasing slopes.

We rename these 7 segments to $a_0, a_1, \ldots, a_6$, and we denote the line $\ell(a_i)$ by ℓ_i, see Figure 16. We have achieved the following properties:

- The lines $\ell_0, \ldots, \ell_6$ form a 7-cup, with decreasing slopes in this order.
- The segments a_i are pairwise disjoint and non-avoiding.
- a_i stabs a_j for every $i < j$.

These properties allow a_0 to lie between any two consecutive intersections on ℓ_0. There is no such flexibility for the other sides: Every side a_j is stabbed by every preceding side a_i. For $1 \leq i < j$, a_i cannot stab a_j from the right, because then a_0 would not be able to stab a_i. Hence, the arrangement of the sides $a_1, \ldots, a_6$ must be exactly as shown in Figure 16, in the sense that the order of endpoints and intersection points along each line ℓ_i is fixed. We will ignore a_0 from now on.

8.2 Finalizing the analysis

Recall that every a_i is the primary side of two consecutive sides a_i, b_i of Q that are hooking with respect to an associated pair A_i, B_i of consecutive sides of P. The sides a_i and A_i are the primary sides and b_i and B_i are the companion sides. All these 4×6 sides are distinct, and they intersect as follows: a_i intersects B_i and is disjoint from A_i; b_i intersects A_i and is disjoint from B_i; and $I(A_i, B_i) \in \mathrm{Cone}(a_i, b_i)$.

Figure 17 summarizes the intersection pattern among these sides. A side A_i must intersect every side a_j with $j \neq i$ and every side b_j since $\mathrm{CC}(A_i) = \mathrm{CC}(a_i) \neq \mathrm{CC}(a_j)$ and $\mathrm{CC}(A_i) = \mathrm{CC}(a_i) \neq \mathrm{CC}(b_i) = \mathrm{CC}(b_j)$. (Recall that all companion sides b_i belong to the same component.) Similarly, every side B_i must intersect every side a_j. We have no information about the intersection between B_i and b_j, as these sides belong to the same connected component.

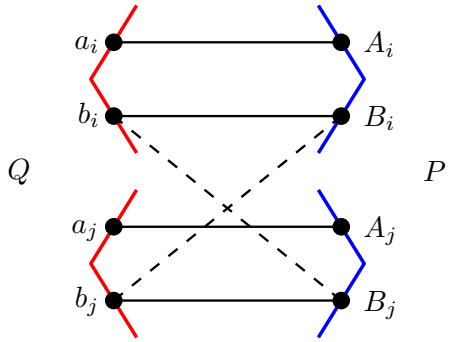


Figure 17 The subgraph of G^{D} induced on two pairs of consecutive sides a_i, b_i and a_j, b_j of P and their associated partner pairs A_i, B_i and A_j, B_j of Q. Parts of P and Q are shown to indicate consecutive sides. The dashed edges may or may not be present.

We will now derive a contradiction through a series of case distinctions.

Case 1: There are three segments A_i with the property that A_i crosses ℓ_i to the left of a_i. Without loss of generality, assume that these segments are A_1, A_2, A_3, see Figure 18. The segments A_1, A_2, A_3 must not cross because P is a simple polygon. Therefore A_1 intersects a_2 to the right of $I(a_1, a_2)$ because otherwise A_1 would cross A_2 on the way between its intersections with ℓ_2 and with a_1. A_3 must cross ℓ_3, a_2, a_1 in this order, as shown. But then A_1 and A_3 (and a_2) block A_2 from intersecting a_3.

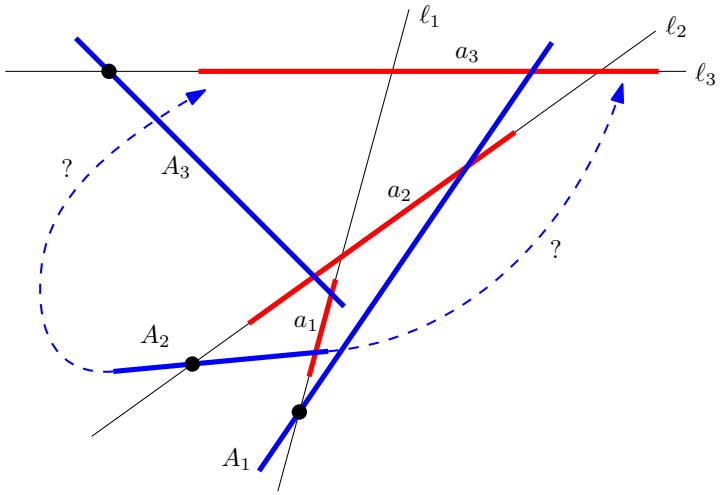


Figure 18 The assumed intersection points between A_i and ℓ_i are marked.

Case 2: There at most two segments A_i with the property that A_i crosses ℓ_i to the left of a_i. In this case, we simply discard these segments. We select four of the remaining segments and renumber them from 1 to 4.

From now on, we can make the following assumption:

> **General Assumption:** For every $1 \leq i \leq 4$, the segment A_i does not cross ℓ_i at all, or it crosses ℓ_i to the right of a_i.

This implies that A_3 must intersect the sides a_2, a_1, a_4 in this order, and it is determined in which cell of the arrangement of the lines $\ell_1, \ell_2, \ell_3, \ell_4$ the left endpoint of A_3 lies (see Figures 16 and 19). For the right endpoint, we have a choice of two cells, depending on whether A_3 intersects ℓ_3 or not.

We denote by left(s) and right(s) the left and right endpoints of a segment s. We distinguish four cases, based on whether the common endpoint of A_3 and B_3 lies at left(A_3) or right(A_3), and whether the common endpoint of a_3 and b_3 lies at left(a_3) or right(a_3).

Case 2.1: $I(A_3, B_3) = \text{left}(A_3)$ and $I(a_3, b_3) = \text{right}(a_3)$, see Figure 19.

As indicated in the figure, we leave it open whether and where A_3 intersects ℓ_3. We know that b_3 must lie below ℓ_3 because $I(A_3, B_3) \in \text{Cone}(a_3, b_3)$.

We claim that A_2 cannot have the required intersections with a_1, a_3, and b_3. Let us first consider a_1: It is cut into three pieces by A_3 and B_3.

If A_2 intersects the middle piece of a_1 in the wedge between A_3 and B_3, then A_2 intersects exactly one of a_3 and b_3 inside the wedge, as these parts together with a_1 are three sides of a convex pentagon. If A_2 intersects a_3, then it has crossed ℓ_3 and it cannot cross b_3 thereafter. If A_2 intersects b_3, it must cross ℓ_4 before leaving the wedge, and then it cannot cross a_3 thereafter.

Suppose now that A_2 crosses the bottom piece of a_1. Then it cannot go around A_3, B_3 to the right in order to reach a_3 because it would have to intersect ℓ_4 twice. A_2 also cannot pass to the left of A_3, B_3 because it cannot cross ℓ_2 through a_2 or, by the general assumption, to the left of a_2.

Suppose finally that A_2 crosses the top piece of a_1. Then it would have to cross ℓ_3 twice before reaching b_3.

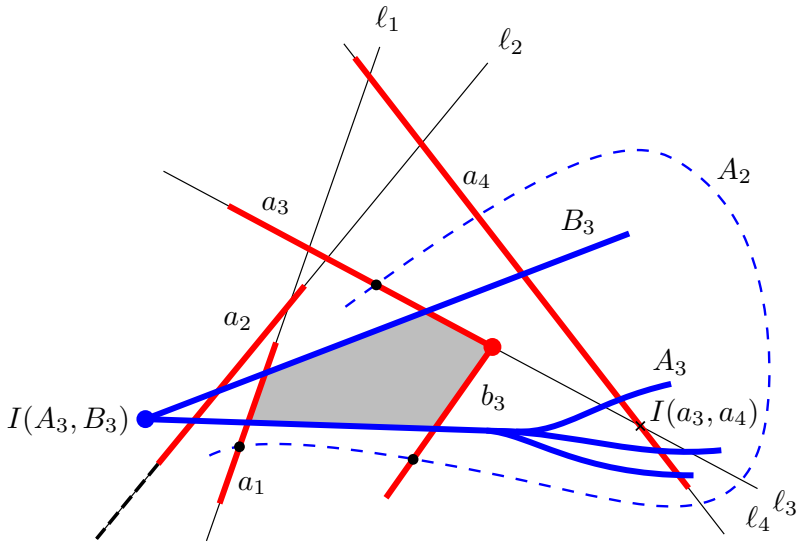


Figure 19 Case 2.1, $I(A_3, B_3) = \text{left}(A_3)$ and $I(a_3, b_3) = \text{right}(a_3)$. A hypothetical segment A_2 is shown as a dashed curve. The side a_2 and the part of ℓ_2 to the left of a_2 is blocked for A_2.

Case 2.2: $I(A_3, B_3) = \text{left}(A_3)$ and $I(a_3, b_3) = \text{left}(a_3)$.

If $\ell(A_3)$ does not intersect a_3, we derive a contradiction as follows, see Figure 20. We know that the sides a_2, a_3, a_4 must be arranged as shown. The segment A_3 crosses a_2 but not a_3. Now, the parts of a_3 and A_3 to the left of ℓ_2 form two opposite sides of a quadrilateral, as shown in the figure. If this quadrilateral were not convex, then either $\ell(A_3)$ would intersect a_3, which we have excluded by assumption, or ℓ_3 would intersect A_3 left of a_3, contradicting the General Assumption. Thus, the sides a_3 and A_3 violate the Axis Property (Observation 6), which requires a_3 and A_3 to lie on different sides of the line through $I(A_3, B_3)$ and $I(a_3, b_3)$.

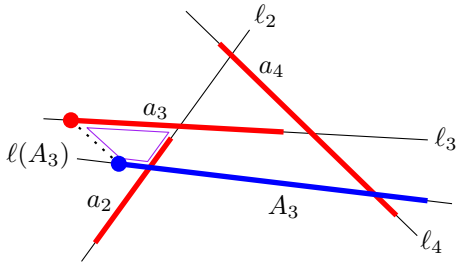


Figure 20 Case 2.2. $I(A_3, B_3) = \text{left}(A_3)$, $I(a_3, b_3) = \text{left}(a_3)$, $\ell(A_3)$ does not intersect a_3.

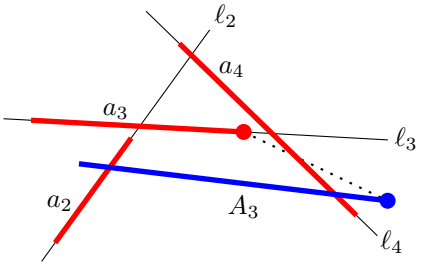


Figure 21 Case 2.3. $I(A_3, B_3) = \text{right}(A_3)$, and $I(a_3, b_3) = \text{right}(a_3)$, A_3 lies below ℓ_3.

If $\ell(A_3)$ intersects a_3, the situation must be as shown in Figure 22: the pair A_3, B_3 is hooked by a_3 and b_3. The analysis of Case 2.1 (Figure 19) applies verbatim, except that the word "pentagon" must be replaced by "hexagon".

Case 2.3: $I(A_3, B_3) = \text{right}(A_3)$, and $I(a_3, b_3) = \text{right}(a_3)$.

If A_3 lies entirely below ℓ_3, then A_3 together with a_3 violates the Axis Property (Observation 6), see Figure 21. Let us therefore assume that A_3 intersects ℓ_3 (to the right of a_3), and thus $\text{right}(A_3) = I(A_3, B_3)$ lies above ℓ_3, see Figure 23a. Then b_3 must also lie above ℓ_3, because a_3, b_3 is supposed to be hooking, that is, $I(A_3, B_3) \in \text{Cone}(a_3, b_3)$.

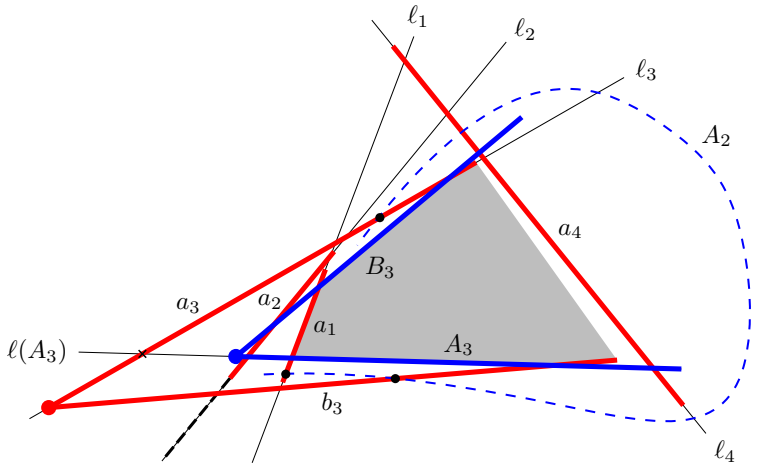


Figure 22 Case 2.2, $I(A_3, B_3) = \text{left}(A_3)$, $I(a_3, b_3) = \text{left}(a_3)$, and $\ell(A_3)$ intersects A_3. A hypothetical segment A_2 is shown as a dashed curve.

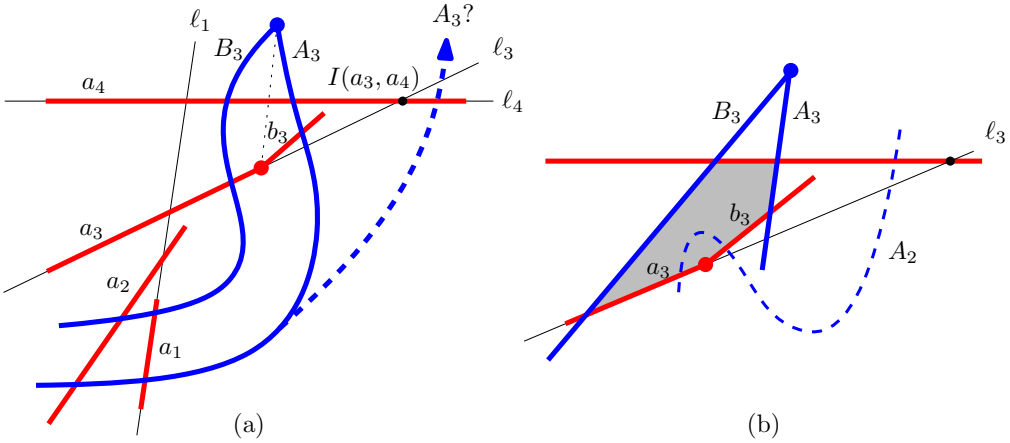


Figure 23 Case 2.3. A_3 intersects ℓ_3.

It follows that A_3 cannot intersect ℓ_3 to the right of $I(a_3, a_4)$ (the option shown as a dashed curve), because otherwise it would miss b_3: b_3 is blocked by a_4. Thus, the situation looks like in Figure 23a. Figure 23b shows the position of the relevant pieces. The segments a_4, B_3, a_3, b_3, A_3 enclose a convex pentagon. Now, the segment A_2 should intersect a_3, b_3, and a_4 without crossing A_3 and B_3, like the dashed curve in the figure. This is impossible.

Case 2.4: $I(A_3, B_3) = \text{right}(A_3)$ and $I(a_3, b_3) = \text{left}(a_3)$.

If A_3 intersects ℓ_3 (to the right of a_3), then A_3 together with a_3 violates the Axis Property (Observation 6), see Figure 24. We thus assume that A_3 lies entirely below ℓ_3.

If $\ell(A_3)$ passes above $I(a_3, b_3) = \text{left}(a_3)$, the sides a_3 and A_3 violate the Axis Property see Figure 25a. On the other hand, if $\ell(A_3)$ passes below $I(a_3, b_3) = \text{left}(a_3)$, as shown in Figure 25b, then b_3 must cross ℓ_1 to the right of a_1 in order to reach A_2. Again by the Axis Property, B_3 must remain above the dotted axis line through $I(A_3, B_3) = \text{right}(A_3)$ and $I(a_3, b_3) = \text{left}(a_3)$. On ℓ_1, b_3 separates a_1 from the axis line, and hence a_1 lies below the axis line. Therefore B_3 and a_1 cannot intersect.

This concludes the proof of Theorem 1. ◀

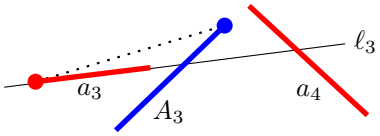


Figure 24 Case 2.4. A_3 intersects ℓ_3.

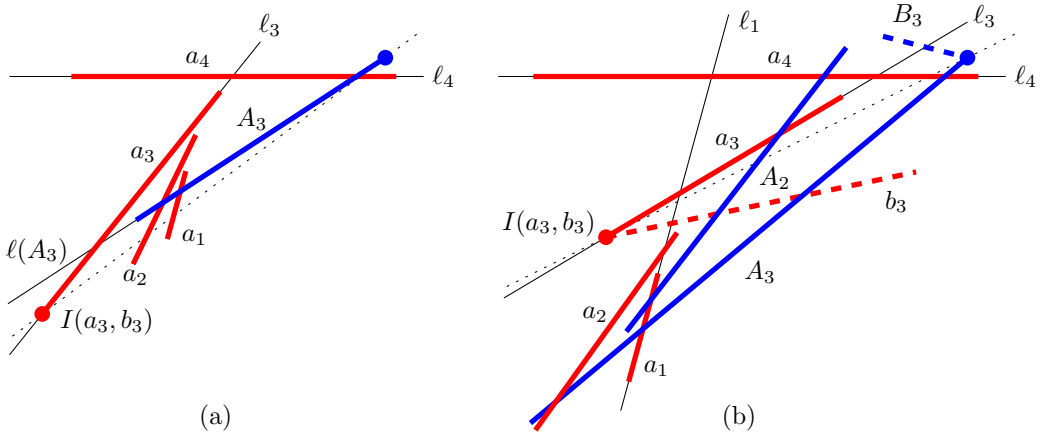


Figure 25 Case 2.4. A_3 lies below ℓ_3.

References

1 J. Černý, J. Kára, D. Král', P. Podbrdský, M. Sotáková, and R. Šámal. On the number of intersections of two polygons. *Comment. Math. Univ. Carolinae*, 44(2):217–228, 2003. URL: `https://cmuc.karlin.mff.cuni.cz/cmuc0302/cmuc0302.htm`.

2 Michael B. Dillencourt, David M. Mount, and Alan Saalfeld. On the maximum number of intersections of two polyhedra in 2 and 3 dimensions. In *Proceedings of the 5th Canadian Conference on Computational Geometry, Waterloo, Ontario, Canada, August 1993*, pages 49–54. University of Waterloo, 1993.

3 P. Erdős and L. Moser. On the representation of directed graphs as unions of orderings. *Magyar Tud. Akad. Mat. Kutató Int. Közl.*, 9:125–132, 1964.

4 P. Erdős and G. Szekeres. A combinatorial problem in geometry. *Compositio Mathematica*, 2:463–470, 1935.

5 Felix Günther. The maximum number of intersections of two polygons, July 2012. withdrawn by the author. `arXiv:1207.0996`.

Dynamic Geometric Set Cover and Hitting Set

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

Hsien-Chih Chang
Department of Computer Science, Duke University, Durham, NC, USA
hsienchih.chang@duke.edu

Subhash Suri
Department of Computer Science, University of California at Santa Barbara, CA, USA
suri@cs.ucsb.edu

Allen Xiao
Department of Computer Science, Duke University, Durham, NC, USA
axiao@cs.duke.edu

Jie Xue
Department of Computer Science, University of California at Santa Barbara, CA, USA
jiexue@ucsb.edu

Abstract

We investigate dynamic versions of geometric set cover and hitting set where points and ranges may be inserted or deleted, and we want to efficiently maintain an (approximately) optimal solution for the current problem instance. While their static versions have been extensively studied in the past, surprisingly little is known about dynamic geometric set cover and hitting set. For instance, even for the most basic case of one-dimensional interval set cover and hitting set, no nontrivial results were known. The main contribution of our paper are two frameworks that lead to efficient data structures for dynamically maintaining set covers and hitting sets in $\mathbb{R}^1$ and $\mathbb{R}^2$. The first framework uses bootstrapping and gives a $(1+\varepsilon)$-approximate data structure for dynamic interval set cover in $\mathbb{R}^1$ with $O(n^\alpha/\varepsilon)$ amortized update time for any constant $\alpha > 0$; in $\mathbb{R}^2$, this method gives $O(1)$-approximate data structures for unit-square (and quadrant) set cover and hitting set with $O(n^{1/2+\alpha})$ amortized update time. The second framework uses local modification, and leads to a $(1+\varepsilon)$-approximate data structure for dynamic interval hitting set in $\mathbb{R}^1$ with $\widetilde{O}(1/\varepsilon)$ amortized update time; in $\mathbb{R}^2$, it gives $O(1)$-approximate data structures for unit-square (and quadrant) set cover and hitting set in the *partially* dynamic settings with $\widetilde{O}(1)$ amortized update time.

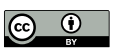
2012 ACM Subject Classification Theory of computation → Computational geometry

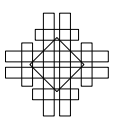
Keywords and phrases Geometric set cover, Geometric hitting set, Dynamic data structures

Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.2

Related Version A full version of this paper is available at `https://arxiv.org/abs/2003.00202`.

Funding *Pankaj K. Agarwal*: NSF grants CCF-15-13816, CCF-15-46392, and IIS-14-08846; ARO grant W911NF-15-1-0408; BSF Grant 2012/229 from the U.S.-Israel Binational Science Foundation.
Hsien-Chih Chang: NSF grants CCF-15-13816, CCF-15-46392, and IIS-14-08846; ARO grant W911NF-15-1-0408; BSF Grant 2012/229 from the U.S.-Israel Binational Science Foundation.
Subhash Suri: NSF grant CCF-18-14172.
Allen Xiao: NSF grants CCF-15-13816, CCF-15-46392, and IIS-14-08846; ARO grant W911NF-15-1-0408; BSF Grant 2012/229 from the U.S.-Israel Binational Science Foundation.
Jie Xue: NSF grant CCF-18-14172.





1 Introduction

A *range space* $(X, \mathcal{R})$ consists of a set X of objects and a family $\mathcal{R}$ of subsets of X called *ranges*. A subset $H \subseteq X$ is called a *hitting set* of $(X, \mathcal{R})$ if it intersects every (nonempty) range in $\mathcal{R}$, and a subset $\mathcal{C} \subseteq \mathcal{R}$ is called a *set cover* of $(X, \mathcal{R})$ if $\bigcup_{C \in \mathcal{C}} C = X$. In many applications X is a set of points in $\mathbb{R}^d$ and $\mathcal{R}$ is induced by a set of geometric regions (rectangles, balls, simplices, etc.), i.e., each is the subset of points lying inside one of the regions. With a slight abuse of notation, we will use $\mathcal{R}$ to denote the set of ranges as well as the set of regions that define these ranges. Given a geometric range space $(S, \mathcal{R})$, the *geometric set-cover* (resp., *hitting-set*) problem is to find the smallest number of ranges in $\mathcal{R}$ (resp., points in S) that cover all points in S (resp., hit all ranges in $\mathcal{R}$). Geometric set cover and hitting set are classical geometric optimization problems, with numerous applications in databases, sensor networks, VLSI design, etc.

In many applications, the problem instance can change over time and re-computing a new solution after each change is too costly. In these situations, a dynamic data structure that can update the solution after a change more efficiently than constructing the entire new solution from scratch is highly desirable. This motivates the main problem studied in our paper: dynamically maintaining geometric set covers and hitting sets under insertion and deletion of points and ranges. In this paper, we formulate the problem as follows: after each update, our data structure should (implicitly) store an approximate set-cover solution $\mathcal{R}'$ (resp., hitting-set solution S') for the current instance such that the following queries can be supported efficiently.

- *Size query*: report the size of $\mathcal{R}'$ (resp., S').
- *Membership query*: for a given range $R \in \mathcal{R}$ (resp., point $a \in S$), report whether R (resp., a) is contained in $\mathcal{R}'$ (resp., S').
- *Reporting query*: report all elements in $\mathcal{R}'$ (resp., S').

We require the size query to be answered in $O(1)$ time, a membership query to be answered in $O(\log |\mathcal{R}'|)$ time (resp., $O(\log |S'|)$ time), and the reporting query to be answered in $O(|\mathcal{R}'|)$ time (resp., $O(|S'|)$ time); this is the best one can expect in the pointer machine model.

We say that a set-cover (resp., hitting-set) instance is *fully dynamic* if insertions and deletions on both points and ranges are allowed, and *partially dynamic* if only the points (resp., ranges) can be inserted and deleted. In this paper, unless explicitly mentioned otherwise, problems are always considered in the fully dynamic setting.

Related work

The set-cover and hitting-set problems for general range spaces are well-known to be NP-complete [12]. A simple greedy algorithm achieves an $O(\log n)$-approximation [7, 13, 15], which is tight under appropriate complexity-theoretic assumptions [8, 14]. The problems remain NP-hard or even hard to approximate in many geometric settings [5, 16, 17]. However, by exploiting the geometric nature of the problems, efficient algorithms with $o(\log n)$ approximation factors can be obtained. For example, Mustafa and Ray [18] showed the existence of polynomial-time approximation schemes (PTAS) for halfspace hitting set in $\mathbb{R}^3$ and disk hitting set in $\mathbb{R}^2$. There is also a PTAS for unit-square set cover given by Erlebach and van Leeuwen [9]. Agarwal and Pan [3] proposed approximation algorithms with near-linear running time to the set-cover and hitting-set problems for halfspaces in $\mathbb{R}^3$, disks in $\mathbb{R}^2$, and orthogonal rectangles.

Despite extensive work on the static versions of hitting set and set cover, very little is known about these problems in the dynamic setting. There is some recent work on set cover in the partially dynamic setting. Gupta et al. [11] showed that an $O(\log n)$-approximation

can be maintained with $O(f \log n)$ amortized update time and an $O(f^3)$-approximation can be maintained with $O(f^2)$ amortized update time, where f is the maximum number of ranges that a point belongs to. These bounds were subsequently improved by Bhattacharya et al. [6] to $O(f^2)$-approximation factor and $O(f \log n)$ amortized update time, and by Abboud et al. [1] to $(1+\varepsilon)f$-approximation factor and $O(f^2 \log n/\varepsilon^5)$ amortized update time.

In geometric settings, there has been some work on the dynamic hitting-set problem. Agarwal et al. [4] described a dynamic data structure for maintaining an $(1+\varepsilon)$-approximation of the optimal hitting set when the set of points S is $\mathbb{R}^1$ and $\mathcal{R}$ is a set of intervals. Ganjugunte [10] studied the dynamic hitting-set problem for the case where S is a set of points in $\mathbb{R}^2$ and $\mathcal{R}$ is a set of squares or discs, under two different dynamic settings: (a) only the range set $\mathcal{R}$ is dynamic and (b) $\mathcal{R}$ is dynamic and S is semi-dynamic (i.e., insertion-only). We are not aware of any non-trivial results for geometric set cover or hitting set even in 1D, except that the greedy algorithm can be implemented in an output-sensitive manner in some special cases (see below).

Our results

The main contribution of this paper are two frameworks for designing fully dynamic geometric set-cover and hitting-set data structures, leading to efficient data structures in $\mathbb{R}^1$ and $\mathbb{R}^2$ (see Table 1). The first framework is based on bootstrapping, which results in efficient (approximate) dynamic data structures for interval set cover and quadrant/unit-square set cover and hitting set (the first three rows of Table 1). The second framework is based on local modification, which results in efficient (approximate) dynamic data structures for interval hitting set and quadrant/unit-square set cover and hitting set in the *partially* dynamic setting (the last three rows of Table 1).

Table 1 Summary of our results for dynamic geometric set cover and hitting set (SC = set cover and HS = hitting set). All update times are amortized. The notation $\widetilde{O}(\cdot)$ hides logarithmic factors; n is the size of the current instance, and $\alpha > 0$ is any small constant. All data structures can be constructed in $\widetilde{O}(n_0)$ time where n_0 is the size of the initial instance.

Framework	Problem	Range	Approx.	Update time	Setting
Bootstrapping	SC	Interval	$1+\varepsilon$	$\widetilde{O}(n^\alpha/\varepsilon)$	Fully dynamic
	SC & HS	Quadrant	$O(1)$	$\widetilde{O}(n^{1/2+\alpha})$	Fully dynamic
	SC & HS	Unit square	$O(1)$	$\widetilde{O}(n^{1/2+\alpha})$	Fully dynamic
Local modification	HS	Interval	$1+\varepsilon$	$\widetilde{O}(1/\varepsilon)$	Fully dynamic
	SC & HS	Quadrant	$O(1)$	$\widetilde{O}(1)$	Part. dynamic
	SC & HS	Unit square	$O(1)$	$\widetilde{O}(1)$	Part. dynamic

For technical reasons, our algorithms maintain a *multiset* solution, as opposed to a regular subset of $\mathcal{R}$ (resp., S). That is, we allow the solution to be a multiset of elements in $\mathcal{R}$ (resp., S) that cover all points in S (resp., hit all ranges in $\mathcal{R}$), and the quality of the solution is also evaluated in terms of the multiset cardinality. Unless explicitly mentioned otherwise, solutions for set cover and hitting set always refer to multiset solutions hereafter.

Overview of the techniques

The basic idea of our *bootstrapping* framework is as follows: We begin from a simple dynamic set-cover or hitting-set data structure (e.g., a data structure that re-computes a solution after each update), and repeatedly use the current data structure to obtain an improved

one. The main challenge here is to design the bootstrapping procedure: how to use a given data structure to construct a new data structure with improved update time. We achieve this by using output-sensitive algorithms and carefully partitioning the problem instances to sub-instances. We say an algorithm is *output-sensitive* if it computes an (approximate) optimal solution in time proportional to the size of the output, using only *basic data structures*. A data structure built on a dataset of size n is *basic* if it can be constructed in $\widetilde{O}(n)$ time and made dynamic with $\widetilde{O}(1)$ update time. One of our technical contributions is in designing an $O(1)$-approximate output-sensitive algorithm for 2D quadrant set cover, which is new to the best of our knowledge.

Our second framework is much simpler, which is based on *local modification*. Namely, we construct a new solution by slightly modifying the previous one after each update, and re-compute a new solution periodically using an output-sensitive algorithm. This framework applies to the problems which are *stable*, i.e., the optimum of a dynamic instance does not change significantly. The discussion of this framework can be found in the full version [2].

Organization

The rest of the paper is organized as follows. Due to limited space, only the results of our first framework (bootstrapping) are discussed in this conference version: Section 2 presents the 1D results and Section 3 presents the 2D results. The results of our second framework (local modification), as well as all the omitted proofs and details, can be found in the full version [2].

2 Warm-up: 1D set cover for intervals

As a warm up for our bootstrapping framework, we first study the 1D problem. Let S be a set of points in $\mathbb{R}^1$ and $\mathcal{I}$ a set of intervals in $\mathbb{R}^1$; set $n = |S| + |\mathcal{I}|$. Our goal is to maintain a small-size set cover of the range space $(S, \mathcal{I})$ as S and $\mathcal{I}$ are updated dynamically; we refer to this instance as the dynamic *interval cover* problem. We note that (static) interval set cover can be solved using the greedy algorithm that repeatedly picks the leftmost uncovered point and covers it using the interval with the rightmost right endpoint. By storing S and $\mathcal{I}$ in a height-balanced tree, the greedy algorithm can be made output sensitive, i.e., it reports an optimal set cover of size opt in $O(\mathsf{opt} \log n)$ time.

▶ **Lemma 1.** *Interval set cover admits an exact output-sensitive algorithm.*

Using the output-sensitive algorithm, now we sketch how to design a fully dynamic data structure to solve the interval-set-cover problem. Set a threshold n^α for some $\alpha \in (0, 1)$. The main bootstrapping step is as follows: Assume that we have a dynamic data structure solving the interval-set-cover problem with $O(n^{\alpha/(1-\alpha)})$ update time (modulo the dependencies on the approximation parameter ε). Using this data structure, we construct a dynamic data structure with $\widetilde{O}(n^\alpha)$ update time as follows: Run the output-sensitive algorithm for $O(n^\alpha)$ steps; if the optimal set cover has size opt at most n^α then the algorithm correctly computes an optimal set cover. Otherwise, we know that opt is at least n^α. We partition the points and the intervals on the real line into roughly εn^α portions in a balanced way. The number of points and endpoints of the intervals per portion is $O(n^{1-\alpha}/\varepsilon)$, and the number of portions is at most $O(\varepsilon n^\alpha)$. On each portion, we build a sub-instance using the points contained in the portion, with all intervals that have an endpoint in the portion. We use the slower data structure to maintain a set cover of this sub-instance. If one of these intervals completely covers the portion (the portion is *coverable*), we solve its sub-instance with a single covering interval. If the portion is not coverable, we rely on the slower data structure

to provide an approximate solution. Because opt is so large, an $(\varepsilon/2)$-approximation on each uncoverable portion combined with the single intervals of each coverable portion still gives an $(1+\varepsilon)$-approximate multiset solution to the entire instance.

The size of each portion is $O(n^{1-\alpha})$ (omitting ε for now), so each update to the dynamic data structure on the portions takes about $O((n^{1-\alpha})^{\alpha/(1-\alpha)}) = O(n^\alpha)$ time, which only happens at most two times per insertion (an interval can partially intersects at most two portions). The data structure periodically reconstructs itself after processing about $n^{1-\alpha}$ operations; which means in amortization each operation costs an extra $O(n^\alpha)$ time. Overall, this gives $O(n^\alpha)$ update time for the new data structure.

We now describe the data structure in detail and analyze its performance.

2.1 Bootstrapping

We begin by stating the bootstrapping theorem, which is the technical heart of our result.

▶ **Theorem 2.** *Let $(S,\mathcal{I})$ be an instance of interval set cover, with $n = |S| + |\mathcal{I}|$. Let $\alpha, \varepsilon \in (0,1)$ be parameters. If there exists a $(1+\varepsilon)$-approximate dynamic set-cover structure $\mathcal{D}_{\text{old}}$ for $(S,\mathcal{I})$ with $\widetilde{O}(n^\alpha/\varepsilon^{1-\alpha})$ amortized update time and $\widetilde{O}(n)$ construction time for any $\varepsilon > 0$, then there exists a $(1+\varepsilon)$-approximate dynamic interval-set-cover data structure $\mathcal{D}_{\text{new}}$ with $\widetilde{O}(n^{\alpha'}/\varepsilon^{1-\alpha'})$ amortized update time and $\widetilde{O}(n)$ construction time for any $\varepsilon > 0$, where $\alpha' = \alpha/(1+\alpha)$. Here n denotes the size of the current problem instance.*

Constructing $\mathcal{D}_{\text{new}}$

The data structure $\mathcal{D}_{\text{new}}$ consists of two parts. The first part is the basic data structure $\mathcal{A}$ required for the output-sensitive algorithm of Lemma 1. The second part is a family of $\mathcal{D}_{\text{old}}$ data structures. Let $f(n,\varepsilon) = \min\{n^{\frac{1}{1+\alpha}}/\varepsilon^{\frac{\alpha}{1+\alpha}}, n/2\}$. $\mathcal{D}_{\text{new}}$ is reconstructed periodically. Let $n_0 = |S| + |\mathcal{I}|$ when the data structure is being constructed. Set $r = \lceil n_0/f(n_0,\varepsilon)\rceil$. We partition the real line $\mathbb{R}$ into r intervals $J_1,\ldots,J_r$ such that each interval J_i contains at most $2f(n_0,\varepsilon)$ points in S plus the endpoints of the intervals in $\mathcal{I}$. For $i \le r$, define $S_i = S \cap J_i$ and $\mathcal{I}_i \subseteq \mathcal{I}$ the subsets of intervals that intersect J_i but do not cover it, i.e., $\mathcal{I}_i = \{I \in \mathcal{I} : J_i \cap I \neq \emptyset \text{ and } J_i \nsubseteq I\}$. When $\mathcal{D}_{\text{new}}$ is updated, the partition $J_1,\ldots,J_r$ remains unchanged, but the S_i's and $\mathcal{I}_i$'s change as S and $\mathcal{I}$ are updated. We view each $(S_i,\mathcal{I}_i)$ as a dynamic interval-set-cover instance, and build the data structure $\mathcal{D}^{(i)}_{\text{old}}$ on $(S_i,\mathcal{I}_i)$ using $\mathcal{D}_{\text{old}}$, with the approximation parameter $\tilde{\varepsilon} = \varepsilon/2$. Thus, $\mathcal{D}^{(i)}_{\text{old}}$ maintains a $(1+\tilde{\varepsilon})$-approximate set cover for $(S_i,\mathcal{I}_i)$. The second part of $\mathcal{D}_{\text{new}}$ consists of the data structures $\mathcal{D}^{(1)}_{\text{old}},\ldots,\mathcal{D}^{(r)}_{\text{old}}$.

Maintaining a set cover

We now describe the algorithm for maintaining a set cover $\mathcal{I}_{\text{appx}}$ for $(S,\mathcal{I})$, if there exists one. Set $\delta = \min\{(6+2\varepsilon)\cdot r/\varepsilon, n\}$. We simulate the output-sensitive greedy algorithm for at most δ steps. If the algorithm successfully computes a set cover, we use it as our $\mathcal{I}_{\text{appx}}$. Otherwise, we construct $\mathcal{I}_{\text{appx}}$ as follows. For $i \in \{1,\ldots,r\}$, we say J_i is *coverable* if there exists $I \in \mathcal{I}$ such that $J_i \subseteq I$ and *uncoverable* otherwise. Let $P = \{i : J_i \text{ is coverable}\}$ and $\overline{P} = \{i : J_i \text{ is uncoverable}\}$. For each $i \in P$, we choose an interval in $\mathcal{I}$ that contains J_i, and denote by $\mathcal{I}^*$ the collection of these intervals. If for some $i \in \overline{P}$, the data structure $\mathcal{D}^{(i)}_{\text{old}}$ tells us that the current $(S_i,\mathcal{I}_i)$ does not have a set cover, then we immediately conclude that the current $(S,\mathcal{I})$ has no feasible set cover, record as such. Otherwise, for every $i \in \overline{P}$, $\mathcal{D}^{(i)}_{\text{old}}$ maintains a $(1+\tilde{\varepsilon})$-approximate optimal set cover $\mathcal{I}^*_i$ for $(S_i,\mathcal{I}_i)$. Set $\mathcal{I}_{\text{appx}} = \mathcal{I}^* \sqcup \left(\bigsqcup_{J_i \in \mathcal{P}'} \mathcal{I}^*_i\right)$.

It can be shown that if $(S, \mathcal{I})$ has a feasible set cover, our algorithm maintains one, namely, $\mathcal{I}_{\text{appx}}$. Later we prove that $\mathcal{I}_{\text{appx}}$ is always a $(1+\varepsilon)$-approximate optimal set cover for $(S, \mathcal{I})$. We now describe how to store $\mathcal{I}_{\text{appx}}$ properly to support the size, membership, and reporting queries in the required query times. If $\mathcal{I}_{\text{appx}}$ is computed by the output-sensitive algorithm, then the size of $\mathcal{I}_{\text{appx}}$ is at most δ, and we have all the elements of $\mathcal{I}_{\text{appx}}$ in hand. In this case, it is not difficult to build a data structure on $\mathcal{I}_{\text{appx}}$ to support the desired queries. On the other hand, if $\mathcal{I}_{\text{appx}}$ is defined as the disjoint union of $\mathcal{I}^*$ and $\mathcal{I}_i^*$'s, the size of $\mathcal{I}_{\text{appx}}$ might be very large and we do not explicitly store all elements of $\mathcal{I}_{\text{appx}}$. Fortunately, in this case, each $\mathcal{I}_i^*$ is already maintained in the data structure $\mathcal{D}_{\text{old}}^{(i)}$. Therefore, we only compute P, $\overline{P}$, and $\mathcal{I}^*$; with these in hand, we easily build a data structure to support the desired queries for $\mathcal{I}_{\text{appx}}$. A detailed discussion is presented in the full version [2].

Updating $\mathcal{D}_{\text{new}}$

Let n_0 be the size of the data structure when $\mathcal{D}_{\text{new}}$ was previously constructed. We reconstruct $\mathcal{D}_{\text{new}}$ after $f(n_0, \varepsilon)$ update operations and reset n_0 to the current value of $|S|+|\mathcal{I}|$. If $\mathcal{D}_{\text{new}}$ is not being reconstructed after an update operation, we first update the basic data structure $\mathcal{A}$. Then, we update the data structure $\mathcal{D}_{\text{old}}^{(i)}$ if the instance $(S_i, \mathcal{I}_i)$ changes due to the operation. Note that an update on S changes exactly one S_i and an update on $\mathcal{I}$ changes at most two $\mathcal{I}_i$'s (because an interval can belong to at most two $\mathcal{I}_i$'s). Thus, we in fact only need to update at most two $\mathcal{D}_{\text{old}}^{(i)}$'s.

Correctness

Now we show that the set cover $\mathcal{I}_{\text{appx}}$ maintained by $\mathcal{D}_{\text{new}}$ is a $(1+\varepsilon)$-approximate optimal set cover for $(S, \mathcal{I})$. Let opt be the size of the optimal set cover of $(S, \mathcal{I})$. If $\mathcal{I}_{\text{appx}}$ is computed by the output-sensitive algorithm, then it is an optimal set cover for $(S, \mathcal{I})$. Otherwise, $\mathsf{opt} > \delta = \min\{(6+2\varepsilon)\cdot r/\varepsilon, n\}$. If $\mathsf{opt} > n$, then the current $(S, \mathcal{I})$ has no set cover (i.e., $\mathsf{opt} = \infty$) and thus $\mathcal{D}_{\text{new}}$ makes a no-solution decision. So assume $\mathsf{opt} > (6+2\varepsilon)\cdot r/\varepsilon$. In this case, $\mathcal{I}_{\text{appx}} = \mathcal{I}^* \sqcup (\bigsqcup_{i\in\overline{P}} \mathcal{I}_i^*)$. For each $i \in \overline{P}$, let opt_i be the optimum of the instance $(S_i, \mathcal{I}_i)$. Then we have $|\mathcal{I}_i^*| \le (1+\tilde{\varepsilon})\cdot \mathsf{opt}_i$ for all $i \in \overline{P}$ where $\tilde{\varepsilon} = \varepsilon/2$. Since $|\mathcal{I}^*| \le r$, we have

$$|\mathcal{I}_{\text{appx}}| = |\mathcal{I}^*| + \sum_{i\in\overline{P}} |\mathcal{I}_i^*| \le r + \left(1+\frac{\varepsilon}{2}\right) \sum_{i\in\overline{P}} \mathsf{opt}_i. \tag{1}$$

Let $\mathcal{I}_{\text{opt}}$ be an optimal set cover for $(S, \mathcal{I})$. We observe that for $i \in \overline{P}$, $\mathcal{I}_{\text{opt}} \cap \mathcal{I}_i$ is a set cover for $(S_i, \mathcal{I}_i)$, because J_i is uncoverable (so the points in S_i cannot be covered by any interval in $\mathcal{I}\backslash\mathcal{I}_i$). It immediately follows that $\mathsf{opt}_i \le |\mathcal{I}_{\text{opt}} \cap \mathcal{I}_i|$ for all $i \in \overline{P}$. Therefore, we have

$$\sum_{i\in\overline{P}} \mathsf{opt}_i \le \sum_{i\in\overline{P}} |\mathcal{I}_{\text{opt}} \cap \mathcal{I}_i|. \tag{2}$$

The right-hand side of the above inequality can be larger than $|\mathcal{I}_{\text{opt}}|$ as some intervals in $\mathcal{I}_{\text{opt}}$ can belong to two $\mathcal{I}_i$'s. The following lemma bounds the number of such intervals.

▶ **Lemma 3.** *There are at most $2r$ intervals in $\mathcal{I}_{opt}$ that belong to exactly two $\mathcal{I}_i$'s.*

Proof. Suppose the portions $J_1, \ldots, J_r$ are sorted from left to right. Let s_i be the separation point of J_i and J_{i+1}. Observe that an interval $I \in \mathcal{I}_{\text{opt}}$ belongs to exactly two $\mathcal{I}_i$'s only if I contains one of the separation points $s_1, \ldots, s_{r-1}$. We claim that for each s_i, at most two intervals in $\mathcal{I}_{\text{opt}}$ contain s_i. Assume there are three intervals I^-, I, I^+ that contain s_i.

Without loss of generality, assume that I^- (resp., I^+) has the leftmost left endpoint (resp., the rightmost right endpoint) among I^-, I, I^+. Then one can easily see that $I \subseteq I^- \cup I^+$. Therefore, $\mathcal{I}_{\text{opt}} \backslash \{I\}$ is also a set cover for $(S, \mathcal{I})$, contradicting the optimality of $\mathcal{I}_{\text{opt}}$. Thus, at most two intervals in $\mathcal{I}_{\text{opt}}$ contain s_i. It follows that there are at most $2(r-1)$ intervals in $\mathcal{I}_{\text{opt}}$ that contain some separation point, and only these intervals can belong to exactly two $\mathcal{I}_i$'s, which proves the lemma. ◀

The above lemma immediately implies

$$\sum_{i \in \overline{P}} |\mathcal{I}_{\text{opt}} \cap \mathcal{I}_i| \leq |\mathcal{I}_{\text{opt}}| + 2r = \mathsf{opt} + 2r. \tag{3}$$

Combining Inequalities 1, 2, and 3, we deduce that

$$\begin{aligned}
|\mathcal{I}_{\text{appx}}| &\leq r + \left(1 + \frac{\varepsilon}{2}\right) \sum_{i \in \overline{P}} \mathsf{opt}_i \\
&\leq r + \left(1 + \frac{\varepsilon}{2}\right) \sum_{i \in \overline{P}} |\mathcal{I}_{\text{opt}} \cap \mathcal{I}_i| \\
&\leq r + \left(1 + \frac{\varepsilon}{2}\right) \cdot (\mathsf{opt} + 2r) = (3 + \varepsilon) \cdot r + \left(1 + \frac{\varepsilon}{2}\right) \cdot \mathsf{opt} \\
&< \frac{\varepsilon}{2} \cdot \mathsf{opt} + \left(1 + \frac{\varepsilon}{2}\right) \cdot \mathsf{opt} = (1 + \varepsilon) \cdot \mathsf{opt},
\end{aligned}$$

where the last inequality follows from the assumption $\mathsf{opt} > (6 + 2\varepsilon) \cdot r/\varepsilon$.

Time complexity analysis

We briefly discuss the amortized update time of $\mathcal{D}_{\text{new}}$; a detailed analysis can be found in the full version [2]. Recall that $f(n, \varepsilon) = \min\{n^{\frac{1}{1+\alpha}}/\varepsilon^{\frac{\alpha}{1+\alpha}}, n/2\}$ and that $\mathcal{D}_{\text{new}}$ is reconstructed after $f(n_0, \varepsilon)$ update operations, where n_0 is the size of $(S, \mathcal{I})$ when $\mathcal{D}_{\text{new}}$ was last constructed, $|n - n_0| \leq n_0/2$ and the size of each $(S_i, \mathcal{I}_i)$ is $O(f(n_0, \varepsilon))$. The construction of $\mathcal{D}_{\text{new}}$ can be easily done in $\widetilde{O}(n_0)$ time.

The update time of $\mathcal{D}_{\text{new}}$ consists of the time for updating the data structures $\mathcal{A}$ and $\mathcal{D}^{(1)}_{\text{old}}, \ldots, \mathcal{D}^{(r)}_{\text{old}}$, the time for maintaining the solution, and the (amortized) time for reconstruction. As argued before, we only need to update at most two $\mathcal{D}^{(i)}_{\text{old}}$'s after each operation. Thus, updating the $\mathcal{D}_{\text{old}}$ data structures takes $\widetilde{O}(f(n_0, \varepsilon)^\alpha/\varepsilon^{1-\alpha})$ amortized time. Maintaining $\mathcal{I}_{\text{appx}}$ takes $\widetilde{O}(\delta + r) = O(n_0/(f(n_0, \varepsilon) \cdot \varepsilon))$ time, with a careful implementation. The reconstruction time is $\widetilde{O}(n) = \widetilde{O}(n_0 + f(n_0, \varepsilon))$, which we pay for by charging $\widetilde{O}(n_0/f(n_0, \varepsilon)) = \widetilde{O}(r)$ to each update operation since the previous reconstruction. In total, the amortized update time of $\mathcal{D}_{\text{new}}$ is $\widetilde{O}(f(n_0, \varepsilon)^\alpha/\varepsilon^{1-\alpha} + n_0/(f(n_0, \varepsilon) \cdot \varepsilon))$. By substituting the value of $f(n_0, \varepsilon)$ (which balances the two terms in the update time) and using the inequality $|n - n_0| \leq n_0/2$, the amortized update time is $\widetilde{O}(n^{\frac{\alpha}{1+\alpha}}/\varepsilon^{1-\frac{\alpha}{1+\alpha}}) = \widetilde{O}(n^{\alpha'}/\varepsilon^{1-\alpha'})$ (recall that $\alpha' = \alpha/(1+\alpha)$).

2.2 Putting everything together

With the bootstrapping theorem in hand, we are now able to design our dynamic interval-set-cover data structure. The starting point is a "trivial" data structure, which simply uses the output-sensitive algorithm of Lemma 1 to recompute an optimal interval set cover after each update. Clearly, the update time of this data structure is $\widetilde{O}(n)$ and the construction time is $\widetilde{O}(n_0)$. Thus, there exists a $(1+\varepsilon)$-approximate dynamic interval-set-cover data structure with $\widetilde{O}(n^{\alpha_0}/\varepsilon^{1-\alpha_0})$ amortized update time for $\alpha_0 = 1$ and $\widetilde{O}(n_0)$ construction time. Define

$\alpha_i = \alpha_{i-1}/(1+\alpha_{i-1})$ for $i \geq 1$. By applying Theorem 2 i times for a constant $i \geq 1$, we see the existence of a $(1+\varepsilon)$-approximate dynamic interval-set-cover data structure with $\widetilde{O}(n^{\alpha_i}/\varepsilon^{1-\alpha_i})$ amortized update time and $\widetilde{O}(n_0)$ construction time. One can easily verify that $\alpha_i = 1/(i+1)$ for all $i \geq 0$. Therefore, for any constant $\alpha > 0$, we have an index $i \geq 0$ satisfying $\alpha_i < \alpha$ and hence $\widetilde{O}(n^{\alpha_i}/\varepsilon^{1-\alpha_i}) = O(n^{\alpha}/\varepsilon)$. We finally conclude the following.

▶ **Theorem 4.** *Let $(S, \mathcal{I})$ be an instance of interval set cover, with $n = |S| + |\mathcal{I}|$. Let $\alpha, \varepsilon \in (0,1)$ be two constants. There exists a $(1+\varepsilon)$-approximate dynamic interval-set-cover data structure with $O(n^{\alpha}/\varepsilon)$ amortized update time.*

3 2D set cover for quadrants and unit squares

In this section, we present dynamic set-cover data structures for quadrants and unit squares using the bootstrapping framework. Most of the section focuses on dynamic quadrant set cover. At the end of the section, we reduce dynamic unit-square set cover to dynamic quadrant set cover.

Let $(S, \mathcal{Q})$ be a range space where S is a set of points in $\mathbb{R}^2$ and $\mathcal{Q}$ is a set of quadrants in $\mathbb{R}^2$. We wish to maintain a set cover of $(S, \mathcal{Q})$ as both S and $\mathcal{Q}$ are updated dynamically. In order to apply the bootstrapping framework, we need an output-sensitive algorithm for quadrant set cover, analog to the one in Lemma 1 for intervals. Designing such an algorithm is considerably more difficult compared to the 1D case, and we defer it to Section 3.2. We first discuss the bootstrapping procedure, assuming the existence of a μ-approximate output-sensitive algorithm for quadrant set cover.

3.1 Bootstrapping

We prove the following bootstrapping theorem, which is the technical heart of our result.

▶ **Theorem 5.** *Let $(S, \mathcal{Q})$ be an instance of quadrant set cover, with $n = |S| + |\mathcal{Q}|$. Let $\alpha, \varepsilon \in (0,1)$ be parameters and $\mu > 0$. If there exist a $(\mu+\varepsilon)$-approximate output-sensitive algorithm for quadrant set cover and a $(\mu+\varepsilon)$-approximate dynamic set-cover structure $\mathcal{D}_{\text{old}}$ for $(S, \mathcal{Q})$ with $\widetilde{O}(n^{\alpha}/\varepsilon^{1-\alpha})$ amortized update time and $\widetilde{O}(n)$ construction time, then there exists a $(\mu+\varepsilon)$-approximate dynamic quadrant-set-cover data structure $\mathcal{D}_{\text{new}}$ with $\widetilde{O}(n^{\alpha'}/\varepsilon^{1-\alpha'})$ amortized update time and $\widetilde{O}(n)$ construction time, where $\alpha' = 2\alpha/(1+2\alpha)$.*

Constructing $\mathcal{D}_{\text{new}}$

As in the 1D case, the data structure $\mathcal{D}_{\text{new}}$ consists of two parts. The first part is the data structure $\mathcal{A}$ required for the μ-approximate output-sensitive algorithm. The second part is a family of $\mathcal{D}_{\text{old}}$ data structures defined as follows. Let $f(n,\varepsilon) = \min\{n^{\frac{1+\alpha}{1+2\alpha}}/\varepsilon^{\frac{\alpha}{1+2\alpha}}, n/2\}$. $\mathcal{D}_{\text{new}}$ is reconstructed periodically. Let $n_0 = |S| + |\mathcal{I}|$ when the data structure is being constructed. Set $r = \lceil n_0/f(n_0,\varepsilon)\rceil$. We use an orthogonal grid to partition the plane $\mathbb{R}^2$ into $r \times r$ cells such that each row (resp., column) of the grid contains $f(n_0,\varepsilon)$ points in S plus vertices of the quadrants in $\mathcal{Q}$ (see the left of Figure 1 for an illustration). Denote by $\square_{i,j}$ the cell in the i-th row and j-th column. Define $S_{i,j} = S \cap \square_{i,j}$. We also define a sub-collection $\mathcal{Q}_{i,j} \subseteq \mathcal{Q}$, as follows: We include in $\mathcal{Q}_{i,j}$ all the quadrants in $\mathcal{Q}$ whose vertices lie in $\square_{i,j}$. Besides, we also include in $\mathcal{Q}_{i,j}$ the following (at most) four *special* quadrants. We say a quadrant Q *left intersects* $\square_{i,j}$ if Q partially intersects $\square_{i,j}$ and contains the left edge of $\square_{i,j}$ (see the right of Figure 1 for an illustration); similarly, we define "right intersects", "top intersects", and "bottom intersects". Among a collection of quadrants, the

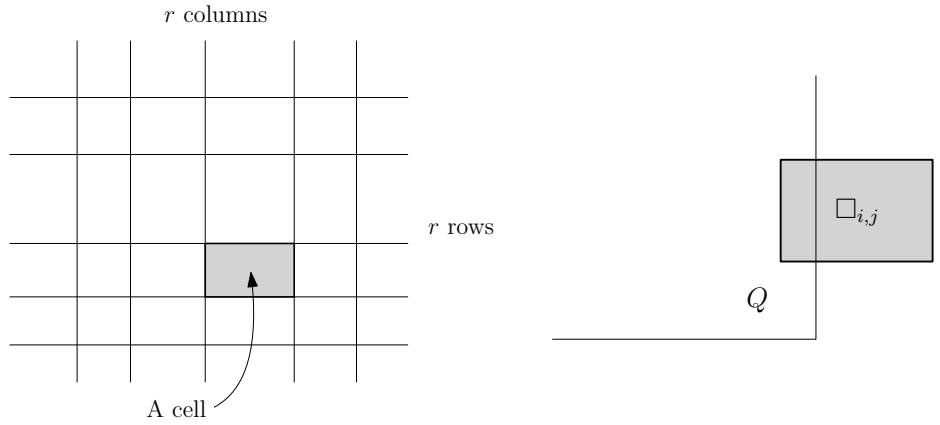


Figure 1 Left: The $r \times r$ grid. Note that the cells may have different sizes. Right: A quadrant Q that left intersects $\square_{i,j}$.

leftmost/rightmost/topmost/bottommost quadrant refers to the quadrant whose vertex is the leftmost/rightmost/topmost/bottommost. We include in $\mathcal{Q}_{i,j}$ the rightmost quadrant in $\mathcal{Q}$ that left intersects $\square_{i,j}$, the leftmost quadrant in $\mathcal{Q}$ that right intersects $\square_{i,j}$, the bottommost quadrant in $\mathcal{Q}$ that top intersects $\square_{i,j}$, and the topmost quadrant in $\mathcal{Q}$ that bottom intersects $\square_{i,j}$ (if these quadrants exist).[1] When the instance $(S, \mathcal{Q})$ is updated, the grid remains unchanged, but the $S_{i,j}$'s and $\mathcal{Q}_{i,j}$'s change as S and $\mathcal{Q}$ are updated. We view each $(S_{i,j}, \mathcal{Q}_{i,j})$ as a dynamic quadrant-set-cover instance, and build the data structure $\mathcal{D}_{\text{old}}^{(i,j)}$ on $(S_{i,j}, \mathcal{Q}_{i,j})$ using $\mathcal{D}_{\text{old}}$, with approximation factor $\tilde{\varepsilon} = \varepsilon/2$. The second part of $\mathcal{D}_{\text{new}}$ consists of the data structures $\mathcal{D}_{\text{old}}^{(i,j)}$ for $i, j \in \{1, \ldots, r\}$.

Maintaining a set cover

We now describe the algorithm for maintaining a set cover $\mathcal{Q}_{\text{appx}}$ for $(S, \mathcal{Q})$, if one exists. Set $\delta = \min\{(8\mu + 4\varepsilon + 2) \cdot r^2/\varepsilon, n\}$. We simulate the output-sensitive algorithm for at most δ steps. If the algorithm successfully computes a set cover, we use it as our $\mathcal{Q}_{\text{appx}}$. Otherwise, we construct $\mathcal{Q}_{\text{appx}}$ as follows. We say the cell $\square_{i,j}$ is *coverable* if there exists $Q \in \mathcal{Q}$ that contains $\square_{i,j}$ and *uncoverable* otherwise. Let $P = \{(i,j) : \square_{i,j} \text{ is coverable}\}$ and $\overline{P} = \{(i,j) : \square_{i,j} \text{ is uncoverable}\}$. For each $(i,j) \in P$, we choose a quadrant in $\mathcal{Q}$ that contains $\square_{i,j}$, and denote by $\mathcal{Q}^*$ the set of these quadrants. If for some $(i,j) \in \overline{P}$, $\mathcal{D}_{\text{old}}^{(i,j)}$ tells us that the instance $(S_{i,j}, \mathcal{Q}_{i,j})$ has no set cover, then we immediately conclude that the current $(S, \mathcal{Q})$ has no feasible set cover, and record as such. Otherwise, for each $(i,j) \in \overline{P}$, $\mathcal{D}_{\text{old}}^{(i,j)}$ maintains a $(\mu + \tilde{\varepsilon})$-approximate optimal set cover $\mathcal{Q}_{i,j}^*$ for $(S_{i,j}, \mathcal{Q}_{i,j})$. Set $\mathcal{Q}_{\text{appx}} = \mathcal{Q}^* \sqcup \left(\bigsqcup_{(i,j)\in\overline{P}} \mathcal{Q}_{i,j}^*\right)$. $\mathcal{Q}_{\text{appx}}$ is stored in roughly the same way as $\mathcal{I}_{\text{appx}}$ is in the 1D case, and we omit the details from here.

[1] Recall that in the 1D case, we define $\mathcal{I}_i$ as the sub-collection of intervals in $\mathcal{I}$ that partially intersect the portion J_i. However, we cannot simply define $\mathcal{Q}_{i,j}$ as the sub-collection of quadrants in $\mathcal{Q}$ that partially intersect $\square_{i,j}$ because a quadrant partially intersects too many cells.

Updating $\mathcal{D}_{\text{new}}$

Let n_0 be the size of the data structure when $\mathcal{D}_{\text{new}}$ was previously constructed. We reconstruct $\mathcal{D}_{\text{new}}$ after $f(n_0, \varepsilon)$ update operations and reset n_0 to the current value of $|S| + |\mathcal{I}|$. If $\mathcal{D}_{\text{new}}$ is not being reconstructed after an update operation, we first update the basic data structure $\mathcal{A}$. Then, we update those data structures $\mathcal{D}_{\text{old}}^{(i,j)}$ for which $(S_{i,j}, \mathcal{Q}_{i,j})$ change. Note that an update on S changes exactly one $S_{i,j}$, and an update on $\mathcal{Q}$ may only change the $\mathcal{Q}_{i,j}$'s in one row and one column (specifically, if the vertex of the inserted/deleted quadrant lies in $\square_{i,j}$, then only $\mathcal{Q}_{i,1}, \ldots, \mathcal{Q}_{i,r}, \mathcal{Q}_{1,j}, \ldots, \mathcal{Q}_{r,j}$ may change). Thus, we in fact only need to update the $\mathcal{D}_{\text{old}}^{(i,j)}$'s in one row and one column.

Correctness

We show that the set cover $\mathcal{Q}_{\text{appx}}$ maintained by $\mathcal{D}_{\text{new}}$ is a $(\mu + \varepsilon)$-approximate optimal set cover for $(S, \mathcal{Q})$. If $\mathcal{Q}_{\text{appx}}$ is computed by the output-sensitive algorithm, then it is a μ-approximate optimal set cover for $(S, \mathcal{Q})$. Otherwise, $\mathsf{opt} > \delta = \min\{(8\mu + 4\varepsilon + 2) \cdot r^2/\varepsilon, n\}$, i.e., either $\mathsf{opt} > (8\mu + 4\varepsilon + 2) \cdot r^2/\varepsilon$ or $\mathsf{opt} > n$. If $\mathsf{opt} > n$, then $(S, \mathcal{Q})$ has no set cover (i.e., $\mathsf{opt} = \infty$) and $\mathcal{D}_{\text{new}}$ makes a no-solution decision. So assume $(8\mu + 4\varepsilon + 2)r^2/\varepsilon < \mathsf{opt} < n$. In this case, $\mathcal{Q}_{\text{appx}} = \mathcal{Q}^* \sqcup (\bigsqcup_{(i,j) \in \overline{P}} \mathcal{Q}^*_{i,j})$. For each $(i,j) \in \overline{P}$, let $\mathsf{opt}_{i,j}$ be the optimum of $(S_{i,j}, \mathcal{Q}_{i,j})$. Then we have $|\mathcal{Q}^*_{i,j}| \leq (\mu + \tilde{\varepsilon}) \cdot \mathsf{opt}_{i,j}$ for all $(i,j) \in \overline{P}$ where $\tilde{\varepsilon} = \varepsilon/2$. Since $|\mathcal{Q}^*| \leq r^2$, we have

$$\mathcal{Q}_{\text{appx}} = |\mathcal{Q}^*| + \sum_{(i,j) \in \overline{P}} |\mathcal{Q}^*_{i,j}| \leq r^2 + \left(\mu + \frac{\varepsilon}{2}\right) \sum_{(i,j) \in \overline{P}} \mathsf{opt}_{i,j}. \tag{4}$$

Let $\mathcal{Q}'_{i,j} \subseteq \mathcal{Q}_{i,j}$ consist of the non-special quadrants, i.e., those whose vertices are in $\square_{i,j}$.

▶ **Lemma 6.** *We have* $\mathsf{opt}_{i,j} \leq |\mathcal{Q}_{\text{opt}} \cap \mathcal{Q}'_{i,j}| + 4$ *for all* $(i,j) \in \overline{P}$*, and in particular,*

$$\sum_{(i,j) \in \overline{P}} \mathsf{opt}_{i,j} \leq \mathsf{opt} + 4r^2. \tag{5}$$

Using Equations 4 and 5, we deduce that

$$\begin{aligned}
|\mathcal{Q}_{\text{appx}}| &\leq r^2 + \left(\mu + \frac{\varepsilon}{2}\right) \sum_{(i,j) \in \overline{P}} \mathsf{opt}_{i,j} \\
&\leq r^2 + \left(\mu + \frac{\varepsilon}{2}\right) (\mathsf{opt} + 4r^2) \\
&\leq (4\mu + 2\varepsilon + 1) \cdot r^2 + \left(\mu + \frac{\varepsilon}{2}\right) \cdot \mathsf{opt} \\
&< \frac{\varepsilon}{2} \cdot \mathsf{opt} + \left(\mu + \frac{\varepsilon}{2}\right) \cdot \mathsf{opt} = (\mu + \varepsilon) \cdot \mathsf{opt},
\end{aligned}$$

where the last inequality follows from the fact that $\mathsf{opt} > (8\mu + 4\varepsilon + 2) \cdot r^2/\varepsilon$.

Time complexity analysis

We briefly discuss the amortized update time of $\mathcal{D}_{\text{new}}$; a detailed analysis can be found in the full version [2]. Recall that $f(n, \varepsilon) = \min\{n^{1-\alpha'/2}/(\sqrt{\varepsilon})^{\alpha'}, n/2\}$ and that $\mathcal{D}_{\text{new}}$ is reconstructed after $f(n_0, \varepsilon)$ update operations, where n_0 is the size of $(S, \mathcal{Q})$ when $\mathcal{D}_{\text{new}}$ was last constructed, $|n - n_0| \leq n_0/2$. We first observe the following fact.

▶ **Lemma 7.** *At any time between reconstructions, we have* $\sum_{k=1}^{r}(|S_{i,k}| + |\mathcal{Q}_{i,k}|) = O(f(n_0,\varepsilon)+r)$ *for all* $i \in \{1,\ldots,r\}$ *and* $\sum_{k=1}^{r}(|S_{k,j}| + |\mathcal{Q}_{k,j}|) = O(f(n_0,\varepsilon)+r)$ *for all* $j \in \{1,\ldots,r\}$.

The above lemma implies that the sum of the sizes of all $(S_{i,j}, \mathcal{Q}_{i,j})$ is $O(n_0+r^2)$ at any time in the first period. Therefore, constructing $\mathcal{D}_{\text{new}}$ can be done in $\widetilde{O}(n_0+r^2) = \widetilde{O}(n_0)$ time.

The update time of $\mathcal{D}_{\text{new}}$ consists of the (amortized) time for reconstruction, the time for updating $\mathcal{A}$ and $\mathcal{D}_{\text{old}}^{(i,j)}$'s, and the time for maintaining the solution. Using almost the same analysis as in the 1D problem, we can show that the reconstruction takes $\widetilde{O}(r + r^2/f(n_0,\varepsilon))$ amortized time and maintaining $\mathcal{Q}_{\text{appx}}$ takes $\widetilde{O}(\delta + r^2) = \widetilde{O}(r^2/\varepsilon)$ time, with a careful implementation. The time for updating the $\mathcal{D}_{\text{old}}^{(i,j)}$ requires a different analysis. Let $m_{i,j}$ denote the current size of $(S_{i,j}, \mathcal{Q}_{i,j})$. As argued before, we in fact only need to update the $\mathcal{D}_{\text{old}}^{(i,j)}$ in one row and one column (say the i-th row and j-th column). Hence, updating the $\mathcal{D}_{\text{old}}^{(i,j)}$ takes $\widetilde{O}(\sum_{k=1}^{r} m_{i,k}^{\alpha}/\varepsilon^{1-\alpha} + \sum_{k=1}^{r} m_{k,j}^{\alpha}/\varepsilon^{1-\alpha})$ amortized time. Lemma 7 implies that $\sum_{k=1}^{r} m_{i,k} = O(f(n_0,\varepsilon)+r)$ and $\sum_{k=1}^{r} m_{k,j} = O(f(n_0,\varepsilon)+r)$. Since $\alpha \le 1$, by Hölder's inequality and Lemma 7,

$$\sum_{k=1}^{r} m_{i,k}^{\alpha} \le \left(\frac{\sum_{k=1}^{r} m_{i,k}}{r}\right)^{\alpha} \cdot r = O(r^{1-\alpha}\cdot(f(n_0,\varepsilon)+r)^{\alpha}) = O(r + r^{1-\alpha} f^{\alpha}(n_0/\varepsilon))$$

and similarly $\sum_{k=1}^{r} m_{k,j}^{\alpha} = O(r + r^{1-\alpha} f^{\alpha}(n_0/\varepsilon))$. It follows that updating the $\mathcal{D}_{\text{old}}$ data structures takes $\widetilde{O}((r + r^{1-\alpha} f^{\alpha}(n_0/\varepsilon))/\varepsilon^{1-\alpha})$ amortized time. In total, the amortized update time of $\mathcal{D}_{\text{new}}$ (during the first period) is $\widetilde{O}((r + r^{1-\alpha} f^{\alpha}(n_0/\varepsilon))/\varepsilon^{1-\alpha} + r^2/\varepsilon)$. By substituting the value of $f(n_0,\varepsilon) = n^{\frac{1+\alpha}{1+2\alpha}}/\varepsilon^{\frac{\alpha}{1+2\alpha}}$ (which balances the two main terms in the update time) and using the fact that $|n - n_0| \le n_0/2$, we obtain that the amortized update time is $\widetilde{O}(n^{\frac{2\alpha}{1+2\alpha}}/\varepsilon^{1-\frac{2\alpha}{1+2\alpha}}) = \widetilde{O}(n^{\alpha'}/\varepsilon^{1-\alpha'})$.

3.2 An output-sensitive cover algorithm

The key to Theorem 5 is an output-sensitive algorithm for the quadrant-set-cover problem. In this section, we develop such an algorithm, which computes an $O(1)$-approximation of the set cover in $\widetilde{O}(\mathsf{opt})$ time, using basic data structures.

For simplicity, let us assume that $(S, \mathcal{Q})$ has a set cover; how the no-solution case is handled is discussed in the full version [2]. There are four types of quadrants in $\mathcal{Q}$, southeast, southwest, northeast, northwest; we denote by $\mathcal{Q}^{\text{SE}}, \mathcal{Q}^{\text{SW}}, \mathcal{Q}^{\text{NE}}, \mathcal{Q}^{\text{NW}} \subseteq \mathcal{Q}$ the sub-collections of these types of quadrants, respectively. Let U^{SE} denote the union of the quadrants in $\mathcal{Q}^{\text{SE}}$, and define $U^{\text{SW}}, U^{\text{NE}}, U^{\text{NW}}$ similarly. Since $(S, \mathcal{Q})$ has a set cover, we have $S = (S \cap U^{\text{SE}}) \cup (S \cap U^{\text{SW}}) \cup (S \cap U^{\text{NE}}) \cup (S \cap U^{\text{NW}})$. Therefore, if we can compute $O(1)$-approximate optimal set covers for each of $(S \cap U^{\text{SE}}, \mathcal{Q})$, $(S \cap U^{\text{SW}}, \mathcal{Q})$, $(S \cap U^{\text{NE}}, \mathcal{Q})$, and $(S \cap U^{\text{NW}}, \mathcal{Q})$, then the union of these four set covers is an $O(1)$-approximate optimal set cover for $(S, \mathcal{Q})$.

With this observation, it now suffices to show how to compute an $O(1)$-approximate optimal set cover for $(S \cap U^{\text{SE}}, \mathcal{Q})$ in $\widetilde{O}(\mathsf{opt}^{\text{SE}})$ time, where opt^{SE} is the optimum of $(S \cap U^{\text{SE}}, \mathcal{Q})$. The main challenge is to guarantee the running time and approximation ratio simultaneously. We begin by introducing some notation. Let γ denote the boundary of U^{SE}, which is an orthogonal staircase curve from bottom-left to top-right. If $\gamma \cap U^{\text{SW}} \neq \emptyset$, then $\gamma \cap U^{\text{SW}}$ is a connected portion of γ that contains the bottom-left end of γ. Define σ as the "endpoint" of $\gamma \cap U^{\text{SW}}$, i.e., the point on $\gamma \cap U^{\text{SW}}$ that is closest the top-right end of γ. See Figure 2 for an illustration. If $\gamma \cap U^{\text{SW}} = \emptyset$, we define σ as the bottom-left end of γ (which

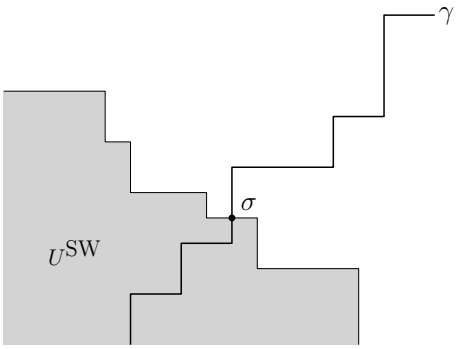


Figure 2 Illustrating the curve γ and the point σ.

is a point whose y-coordinate equals to $-\infty$). For a number $\tilde{y} \in \mathbb{R}$, we define $\phi(\tilde{y})$ as the *leftmost* point in $S \cap U^{\mathrm{SE}}$ whose y-coordinate is greater than $\tilde{y}$; we say $\phi(\tilde{y})$ does not exist if no point in $S \cap U^{\mathrm{SE}}$ has y-coordinate greater than $\tilde{y}$. For a point $a \in \mathbb{R}^2$ and a collection $\mathcal{P}$ of quadrants, we define $\Phi_{\rightarrow}(a, \mathcal{P})$ and $\Phi_{\uparrow}(a, \mathcal{P})$ as the rightmost and topmost quadrants in $\mathcal{P}$ that contains a, respectively. For a quadrant Q, we denote by $x(Q)$ and $y(Q)$ the x- and y-coordinates of the vertex of Q, respectively.

To get some intuition, let us consider a very simple case, where $\mathcal{Q}$ only consists of southeast quadrants. In this case, one can compute an optimal set cover for $(S \cap U^{\mathrm{SE}}, \mathcal{Q})$ using a greedy algorithm similar to the 1D interval-set-cover algorithm: repeatedly pick the leftmost uncovered point in $S \cap U^{\mathrm{SE}}$ and cover it using the topmost (southeast) quadrant in $\mathcal{Q}$. Using the notations defined above, we can describe this algorithm as follows. Set $\mathcal{Q}_{\mathrm{ans}} \leftarrow \emptyset$ and $\tilde{y} \leftarrow -\infty$ initially, and repeatedly do $a \leftarrow \phi(\tilde{y})$, $Q \leftarrow \Phi_{\uparrow}(\sigma, \mathcal{Q}^{\mathrm{SE}})$, $\mathcal{Q}_{\mathrm{ans}} \leftarrow \mathcal{Q}_{\mathrm{ans}} \cup \{Q\}$, $\tilde{y} \leftarrow y(Q)$ until $\phi(\tilde{y})$ does not exist. Eventually, $\mathcal{Q}_{\mathrm{ans}}$ is the set cover we want.

Now we try to extend this algorithm to the general case. However, the situation here becomes much more complicated, since we may have three other types of quadrants in $\mathcal{Q}$, which have to be carefully dealt with in order to guarantee the correctness. But the intuition remains the same: we still construct the solution in a greedy manner. The following procedure describes our algorithm, see also Figure 3.

1. $\mathcal{Q}_{\mathrm{ans}} \leftarrow \emptyset$. $\tilde{y} \leftarrow -\infty$. If $\phi(\tilde{y})$ does not exist, then go to Step 6.
2. $\mathcal{Q}_{\mathrm{ans}} \leftarrow \{\Phi_{\rightarrow}(\sigma, \mathcal{Q}^{\mathrm{SW}}), \Phi_{\uparrow}(\sigma, \mathcal{Q}^{\mathrm{SE}})\}$. $\tilde{y} \leftarrow y(\Phi_{\uparrow}(\sigma, \mathcal{Q}^{\mathrm{SE}}))$. If $\phi(\tilde{y})$ exists, then $a \leftarrow \phi(\tilde{y})$, else go to Step 6.
3. If $a \in U^{\mathrm{NE}}$, then $\mathcal{Q}_{\mathrm{ans}} \leftarrow \mathcal{Q}_{\mathrm{ans}} \cup \{\Phi_{\uparrow}(a, \mathcal{Q}^{\mathrm{NE}}), \Phi_{\uparrow}(a, \mathcal{Q}^{\mathrm{SE}})\}$ and go to Step 6.
4. If $a \in U^{\mathrm{NW}}$, then $\mathcal{Q}_{\mathrm{ans}} \leftarrow \mathcal{Q}_{\mathrm{ans}} \cup \{\Phi_{\rightarrow}(a, \mathcal{Q}^{\mathrm{NW}}), \Phi_{\uparrow}(a, \mathcal{Q}^{\mathrm{SE}})\}$ and $Q \leftarrow \Phi_{\uparrow}(v, \mathcal{Q}^{\mathrm{SE}})$ where v is the vertex of $\Phi_{\rightarrow}(a, \mathcal{Q}^{\mathrm{NW}})$, otherwise $Q \leftarrow \Phi_{\uparrow}(a, \mathcal{Q}^{\mathrm{SE}})$.
5. $\mathcal{Q}_{\mathrm{ans}} \leftarrow \mathcal{Q}_{\mathrm{ans}} \cup \{Q\}$. $\tilde{y} \leftarrow y(Q)$. If $\phi(\tilde{y})$ exists, then $a \leftarrow \phi(\tilde{y})$ and go to Step 3.
6. Output $\mathcal{Q}_{\mathrm{ans}}$.

The following lemma proves the correctness of our algorithm.

▶ **Lemma 8.** *$\mathcal{Q}_{\mathrm{ans}}$ covers all points in $S \cap U^{\mathrm{SE}}$, and $|\mathcal{Q}_{\mathrm{ans}}| = O(\mathsf{opt}^{\mathrm{SE}})$.*

The remaining task is to show how to perform our algorithm in $\widetilde{O}(\mathsf{opt}^{\mathrm{SE}})$ time using basic data structures. It is clear that our algorithm terminates in $O(\mathsf{opt}^{\mathrm{SE}})$ steps, since we include at least one quadrant to $\mathcal{Q}_{\mathrm{ans}}$ in each iteration of the loop Step 3–5 and eventually $|\mathcal{Q}_{\mathrm{ans}}| = O(\mathsf{opt}^{\mathrm{SE}})$ by Lemma 8. Thus, it suffices to show that each step can be done in $\widetilde{O}(1)$ time. In every step of our algorithm, all work can be done in constant time except the tasks of computing the point σ, testing whether $a \in U^{\mathrm{NE}}$ and $a \in U^{\mathrm{NW}}$ for a given point

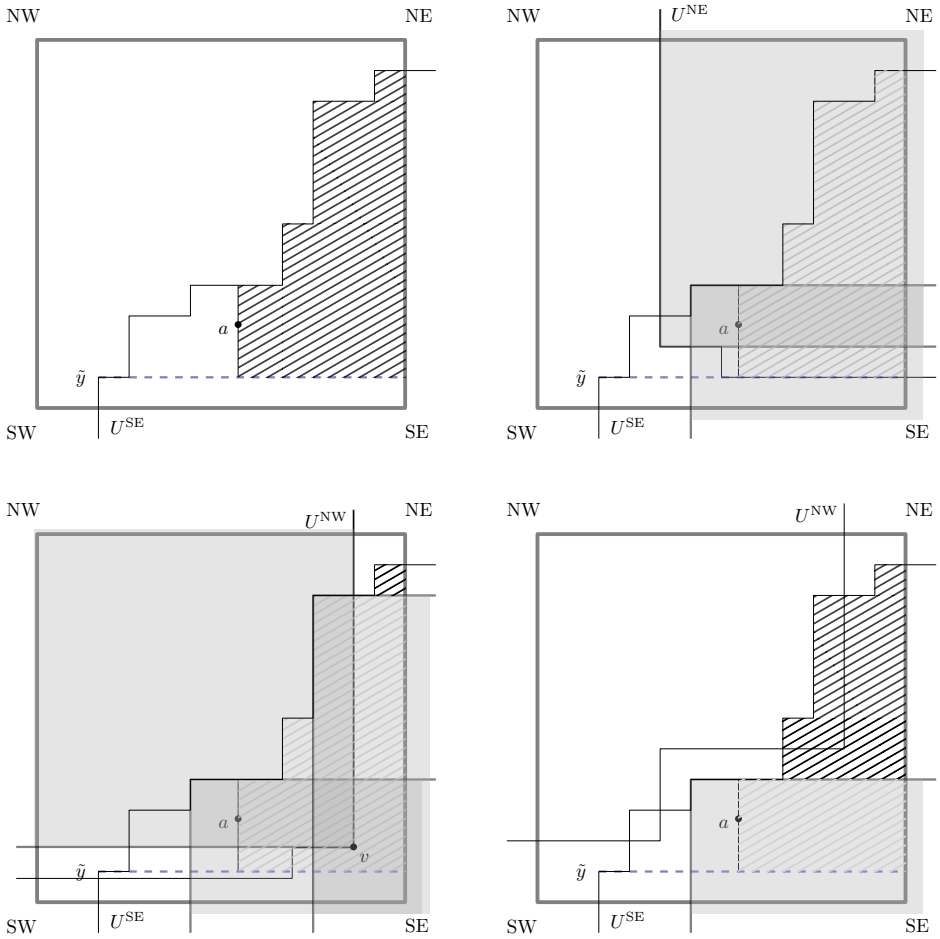


Figure 3 Yet uncovered points lie in the hatch-shaded region (top left). In Step 3, the entire region can be covered by two quadrants if $a \in U^{\mathrm{NE}}$ (top right). In Step 4, the hatch-shaded region can be reduced using one or three quadrants, depending on whether $a \in U^{\mathrm{NW}}$ (bottom). After Step 4, any quadrant intersecting the remaining hatch-shaded region will not cover a.

a, computing the quadrants $\Phi_{\rightarrow}(a, \mathcal{Q}^{\mathrm{SW}}), \Phi_{\rightarrow}(a, \mathcal{Q}^{\mathrm{NW}}), \Phi_{\uparrow}(a, \mathcal{Q}^{\mathrm{SE}}), \Phi_{\uparrow}(a, \mathcal{Q}^{\mathrm{NE}})$ for a given point a, and computing $\phi(\tilde{y})$ for a given number $\tilde{y}$. All these tasks except the computation of $\phi(\cdot)$ can be easily done in $\widetilde{O}(1)$ time by storing the quadrants in binary search trees. To compute $\phi(\cdot)$ in $\widetilde{O}(1)$ time is more difficult, and we achieve this by properly using range trees built on both S and $\mathcal{Q}^{\mathrm{SE}}$. The details are presented in the full version [2].

Using the above algorithm, we can compute $O(1)$-approximate optimal set covers for $(S \cap U^{\mathrm{SE}}, \mathcal{Q})$, $(S \cap U^{\mathrm{SW}}, \mathcal{Q})$, $(S \cap U^{\mathrm{NE}}, \mathcal{Q})$, and $(S \cap U^{\mathrm{NW}}, \mathcal{Q})$. As argued before, the union of these four set covers, denoted by $\mathcal{Q}^*$, is an $O(1)$-approximate optimal set covers for $(S, \mathcal{Q})$.

▶ **Theorem 9.** *Quadrant set cover admits an $O(1)$-approximate output-sensitive algorithm.*

3.3 Putting everything together

With the bootstrapping theorem in hand, we are now able to design our dynamic quadrant-set-cover data structure. Again, the starting point is a "trivial" data structure which uses the output-sensitive algorithm of Theorem 9 to re-compute an optimal quadrant set cover after

each update. Clearly, the update time of this data structure is $\widetilde{O}(n)$ and the construction time is $\widetilde{O}(n_0)$. Let $\mu = O(1)$ be the approximation ratio of the output-sensitive algorithm. The trivial data structure implies the existence of a $(\mu+\varepsilon)$-approximate dynamic quadrant-set-cover data structure with $\widetilde{O}(n^{\alpha_0}/\varepsilon^{1-\alpha_0})$ amortized update time for $\alpha_0 = 1$ and $\widetilde{O}(n_0)$ construction time. Define $\alpha_i = 2\alpha_{i-1}/(1+2\alpha_{i-1})$ for $i \geq 1$. By applying Theorem 5 i times for a constant $i \geq 1$, we see the existence of a $(\mu+\varepsilon)$-approximate dynamic quadrant-set-cover data structure with $\widetilde{O}(n^{\alpha_i}/\varepsilon^{1-\alpha_i})$ amortized update time and $\widetilde{O}(n_0)$ construction time. One can easily verify that $\alpha_i = 2^i/(2^{i+1}-1)$ for all $i \geq 0$. Therefore, for any constant $\alpha > 0$, we have a constant $i \geq 0$ satisfying $\alpha_i < 1/2 + \alpha$ and hence $\widetilde{O}(n^{\alpha_i}/\varepsilon^{1-\alpha_i}) = O(n^{1/2+\alpha}/\varepsilon)$. Setting ε to be any constant, we finally conclude the following.

▶ **Theorem 10.** *Let $(S, \mathcal{Q})$ be an instance of quadrant set cover, with $n = |S| + |\mathcal{Q}|$. Let $\alpha > 0$ be an arbitrarily small constant. There exists an $O(1)$-approximate dynamic quadrant-set-cover data structure with $O(n^{1/2+\alpha})$ amortized update time.*

As mentioned earlier, set cover for unit squares can be reduced to instances of quadrant set cover. In particular, we prove the following lemma in the full version [2]:

▶ **Lemma 11.** *Suppose there exists a c-approximate dynamic quadrant-set-cover data structure with $f(n)$ amortized update time, where f is an increasing function. Then there exist $O(c)$-approximate dynamic unit-square-set-cover, dynamic unit-square-hitting-set, and dynamic quadrant-hitting-set data structures with $\widetilde{O}(f(n))$ amortized update time.*

Finally, it can be verified that the hitting-set problem for quadrants (resp., unit squares) is the same as the set-cover problem for quadrants (resp., unit squares). We thus conclude the following.

▶ **Theorem 12.** *Let $(S, \mathcal{R})$ be an instance of unit-square set cover, unit-square hitting set, or quadrant hitting set, with $n = |S| + |\mathcal{R}|$. Let $\alpha > 0$ be an arbitrarily small constant. There exists an $O(1)$-approximate dynamic data structure for $(S, \mathcal{R})$ with $O(n^{1/2+\alpha})$ amortized update time.*

References

1. Amir Abboud, Raghavendra Addanki, Fabrizio Grandoni, Debmalya Panigrahi, and Barna Saha. Dynamic set cover: improved algorithms and lower bounds. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 114–125. ACM, 2019.
2. Pankaj K. Agarwal, Hsien-Chih Chang, Subhash Suri, Allen Xiao, and Jie Xue. Dynamic geometric set cover and hitting set. *arXiv preprint*, 2020. `arXiv:2003.00202`.
3. Pankaj K. Agarwal and Jiangwei Pan. Near-linear algorithms for geometric hitting sets and set covers. In *Proceedings of the thirtieth annual symposium on Computational geometry*, page 271. ACM, 2014.
4. Pankaj K. Agarwal, Junyi Xie, Jun Yang, and Hai Yu. Monitoring continuous band-join queries over dynamic data. In *16th International Symposium on Algorithms and Computation (ISAAC)*, pages 349–359. Springer, 2005.
5. Piotr Berman and Bhaskar DasGupta. Complexities of efficient solutions of rectilinear polygon cover problems. *Algorithmica*, 17(4):331–356, 1997.
6. Sayan Bhattacharya, Monika Henzinger, and Giuseppe F Italiano. Design of dynamic algorithms via primal-dual method. In *International Colloquium on Automata, Languages, and Programming*, pages 206–218. Springer, 2015.
7. Vasek Chvatal. A greedy heuristic for the set-covering problem. *Mathematics of operations research*, 4(3):233–235, 1979.

8 Irit Dinur and David Steurer. Analytical approach to parallel repetition. In *Proceedings of the forty-sixth annual ACM Symposium on Theory of Computing*, pages 624–633. ACM, 2014.

9 Thomas Erlebach and Erik Jan Van Leeuwen. Ptas for weighted set cover on unit squares. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 166–177. Springer, 2010.

10 Shashidhara K. Ganjugunte. *Geometric hitting sets and their variants*. PhD thesis, Duke University, 2011.

11 Anupam Gupta, Ravishankar Krishnaswamy, Amit Kumar, and Debmalya Panigrahi. Online and dynamic algorithms for set cover. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 537–550. ACM, 2017.

12 Juris Hartmanis. Computers and intractability: a guide to the theory of np-completeness. *Siam Review*, 24(1):90, 1982.

13 David S. Johnson. Approximation algorithms for combinatorial problems. *Journal of computer and system sciences*, 9(3):256–278, 1974.

14 Subhash Khot and Oded Regev. Vertex cover might be hard to approximate to within 2-ε. *Journal of Computer and System Sciences*, 74(3):335–349, 2008.

15 László Lovász. On the ratio of optimal integral and fractional covers. *Discrete mathematics*, 13(4):383–390, 1975.

16 Nimrod Megiddo and Kenneth J. Supowit. On the complexity of some common geometric location problems. *SIAM journal on computing*, 13(1):182–196, 1984.

17 Nimrod Megiddo and Arie Tamir. On the complexity of locating linear facilities in the plane. *Operations research letters*, 1(5):194–197, 1982.

18 Nabil H. Mustafa and Saurabh Ray. Improved results on geometric hitting set problems. *Discrete & Computational Geometry*, 44(4):883–895, 2010.

The Parameterized Complexity of Guarding Almost Convex Polygons

Akanksha Agrawal
Ben-Gurion University, Beer-Sheba, Israel
agrawal@post.bgu.ac.il

Kristine V. K. Knudsen
University of Bergen, Bergen, Norway
kristine.knudsen@ii.uib.no

Daniel Lokshtanov
University of California, Santa Barbara, CA, USA
daniello@ucsb.edu

Saket Saurabh
The Institute of Mathematical Sciences, HBNI, Chennai, India
saket@imsc.res.in

Meirav Zehavi
Ben-Gurion University, Beer-Sheba, Israel
meiravze@bgu.ac.il

Abstract

The Art Gallery problem is a fundamental visibility problem in Computational Geometry. The input consists of a simple polygon P, (possibly infinite) sets G and C of points within P, and an integer k; the task is to decide if at most k guards can be placed on points in G so that every point in C is visible to at least one guard. In the classic formulation of Art Gallery, G and C consist of all the points within P. Other well-known variants restrict G and C to consist either of all the points on the boundary of P or of all the vertices of P. Recently, three new important discoveries were made: the above mentioned variants of Art Gallery are all W[1]-hard with respect to k [Bonnet and Miltzow, ESA'16], the classic variant has an $\mathcal{O}(\log k)$-approximation algorithm [Bonnet and Miltzow, SoCG'17], and it may require irrational guards [Abrahamsen et al., SoCG'17]. Building upon the third result, the classic variant and the case where G consists only of all the points on the boundary of P were both shown to be $\exists\mathbb{R}$-complete [Abrahamsen et al., STOC'18]. Even when both G and C consist only of all the points on the boundary of P, the problem is not known to be in NP.

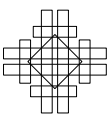
Given the first discovery, the following question was posed by Giannopoulos [Lorentz Center Workshop, 2016]: Is Art Gallery FPT with respect to r, the number of reflex vertices? In light of the developments above, we focus on the variant where G and C consist of all the vertices of P, called Vertex-Vertex Art Gallery. Apart from being a variant of Art Gallery, this case can also be viewed as the classic Dominating Set problem in the visibility graph of a polygon. In this article, we show that the answer to the question by Giannopoulos is *positive*: Vertex-Vertex Art Gallery is solvable in time $r^{\mathcal{O}(r^2)}n^{\mathcal{O}(1)}$. Furthermore, our approach extends to assert that Vertex-Boundary Art Gallery and Boundary-Vertex Art Gallery are both FPT as well. To this end, we utilize structural properties of "almost convex polygons" to present a two-stage reduction from Vertex-Vertex Art Gallery to a new constraint satisfaction problem (whose solution is also provided in this paper) where constraints have arity 2 and involve monotone functions.

2012 ACM Subject Classification Theory of computation → Fixed parameter tractability; Theory of computation → Computational geometry

Keywords and phrases Art Gallery, Reflex vertices, Monotone 2-CSP, Parameterized Complexity, Fixed Parameter Tractability

Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.3

Related Version A full version of this paper is available at `https://arxiv.org/abs/2003.07793`.



36th International Symposium on Computational Geometry (SoCG 2020).
Editors: Sergio Cabello and Danny Z. Chen; Article No. 3; pp. 3:1–3:16

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

Funding *Akanksha Agrawal*: the PBC Fellowship Program for Outstanding Post-Doctoral Researchers from China and India.
Daniel Lokshtanov: European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (no. 715744), and United States – Israel Binational Science Foundation (no. 2018302).

Saket Saurabh: European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (no. 819416), and Swarnajayanti Fellowship (no. DST/SJF/MSA01/2017-18).
Meirav Zehavi: Israel Science Foundation grant no. 1176/18, and United States – Israel Binational Science Foundation (no. 2018302).

Acknowledgements We thank anonymous reviewers for helpful comments that improved and simplified the paper.

1 Introduction

Given a *simple* polygon P on n vertices, two points x and y within P are *visible* to each other if the line segment between x and y is contained in P. Accordingly, a set S of points within P is said to *guard* another set Q of points within P if, for every point $q \in Q$, there is some point $s \in S$ such that q and s are visible to each other. The computational problem that arises from this notion is loosely termed the Art Gallery problem. In its general formulation, the input consists of a simple polygon P, possibly infinite sets G and C of points within P, and a non-negative integer k. The task is to decide whether at most k guards can be placed on points in G so that every point in C is visible to at least one guard. The most well-known cases of Art Gallery are identified as follows: the X-Y Art Gallery problem is the Art Gallery problem where G is the set of all points within P (if X=Point), all boundary points of P (if X=Boundary), or all vertices of P (if X=Vertex), and C is defined analogously with respect to Y. The classic variant of Art Gallery is the Point-Point Art Gallery problem. Nevertheless, all variants where X=Vertex or Y=Point received attention in the literature.[1] In particular, Vertex-Vertex Art Gallery is equivalent to the classic Dominating Set problem in the visibility graph of a polygon.

Art Gallery is a fundamental visibility problem in Discrete and Computational Geometry, which was extensively studied from both combinatorial and algorithmic viewpoints. The problem was first proposed by Victor Klee in 1973, which prompted a flurry of results [15, page 1]. The main combinatorial question posed by Klee was *how many guards are sufficient to see every point of the interior of an n-vertex simple polygon?* Chvátal [6] showed in 1975 that $\lfloor \frac{n}{3} \rfloor$ guards are always sufficient and sometimes necessary for any n-vertex simple polygon (see [8] for a simpler proof by Fisk). After this, many variants of Art Gallery, based on different definitions of visibility, restricted classes of polygons, different shapes of guards, and mobility of guards, have been defined and analyzed. A book [15] and several extensive surveys and book chapters were dedicated to Art Gallery and its variants (see, e.g., [7, 18, 19]). In this article, our main proof states that Vertex-Vertex Art Gallery is *fixed-parameter tractable (*FPT*)* parameterized by r, the number of reflex vertices of P. Additionally, we show that both Vertex-Boundary Art Gallery and Boundary-Vertex Art Gallery are FPT with respect to the number of reflex vertices as well.

[1] The X-Y Art Gallery problem, for any X,Y $\in$ {Point, Boundary, Vertex}, is often loosely termed the Art Gallery problem. For example, in the survey of open problems by Ghosh and Goswami [9], the term Art Gallery problem refers to the Vertex-Vertex Art Gallery problem.

1.1 Background: Related Algorithmic Works

We focus only on algorithmic works on X-Y Art Gallery for X,Y∈ {Point,Boundary, Vertex}. (The discussions regarding known approximation and exact algorithms can be found in the full version [4] of the paper.)

Hardness. In 1983, O'Rourke and Supowit [16] proved that Point-Point Art Gallery is NP-hard if the polygon can contain holes. The requirement to allow holes was lifted shortly afterwards [3]. In 1986, Lee and Lin [12] showed that Vertex-Point Art Gallery is NP-hard. This result extends to Vertex-Vertex Art Gallery and Vertex-Boundary Art Gallery. Later, numerous other restricted cases were shown to be NP-hard as well. For example, NP-hardness was established for orthogonal polygons by Katz and Roisman [11] and Schuchardt and Hecker [17]. We remark that the reductions that show that X-Y Art Gallery (for X,Y ∈ {Point, Boundary, Vertex}) is NP-hard also imply that these cases cannot be solved in time $2^{o(n)}$ under the Exponential-Time Hypothesis (ETH).

While it has long been known that even very restricted cases of Art Gallery are NP-hard, the inclusion of X-Y Art Gallery, for X,Y ∈ {Point, Boundary}, in NP remained open. (When X=Vertex, the problem is clearly in NP.) In 2017, Abrahamsen et al. [1] began to reveal the reasons behind this discrepancy for the Point-Point Art Gallery problem: they showed that *exact* solutions to this problem sometimes require placement of guards on points with *irrational* coordinates. Shortly afterwards, they extended this discovery to prove that Point-Point Art Gallery and Boundary-Point Art Gallery are $\exists\mathbb{R}$-complete [2]. Roughly speaking, this result means that *(i)* any system of polynomial equations over the real numbers can be encoded as an instance of Point/Boundary-Point Art Gallery, and *(ii)* these problems are not in the complexity class NP unless NP = $\exists\mathbb{R}$.

Parameterized Complexity. Two years ago, Bonnet and Miltzow [5] showed that Vertex-Point Art Gallery and Point-Point Art Gallery are W[1]-hard with respect to the *solution size*, k. With straightforward adaptations, their results extend to most of the known variants of the problem, including Vertex-Vertex Art Gallery. Thus, *the classic parameterization by solution size leads to a dead-end.* However, this does not rule out the existence of FPT algorithms for non-trivial structural parametrizations. We refer to the nice surveys by Niedermeier on the art of parameterizations [13, 14].

1.2 Giannopoulos's Parameterization and Our Contribution

In light of the W[1]-hardness result by Bonnet and Miltzow [5], Giannopoulos [10] proposed to parameterize the Art Gallery problem by the number r of reflex vertices of the input polygon P. Specifically, Giannopoulos [10] posed the following open problem: *"Guarding simple polygons has been recently shown to be* W[1]*-hard w.r.t. the number of (vertex or edge) guards. Is the problem* FPT *w.r.t. the number of reflex vertices of the polygon?"* The motivation behind this proposal is encapsulated by the following well-known proposition, see [15, Sections 2.5-2.6].

▶ **Proposition 1** (Folklore). *For any polygon P, the set of reflex vertices of P guards the set of all points within P.*

That is, the minimum number k of guards needed (for any of the cases of Art Gallery) is upper bounded by the number of reflex vertices r. Clearly, k can be arbitrarily smaller than r (see Fig. 1). Our main result is that the Vertex-Vertex Art Gallery problem is FPT parameterized by r. This implies that guarding the vertex set of "almost convex polygons" is easy. In particular, whenever $r^2 \log r = \mathcal{O}(\log n)$, the problem is solvable in polynomial time.

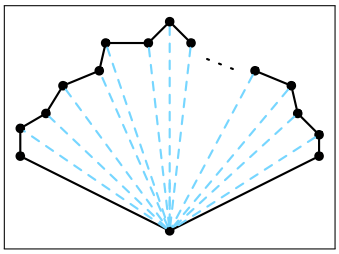


Figure 1 The solution size $k = 1$, yet the number of reflex vertices r is arbitrarily large.

▶ **Theorem 2.** *Vertex-Vertex Art Gallery is* FPT *parameterized by r, the number of reflex vertices. In particular, it admits an algorithm with running time $r^{\mathcal{O}(r^2)}n^{\mathcal{O}(1)}$.*

A few remarks are in place. First, our result extends (with straightforward adaptation) to the most general discrete annotated case of Art Gallery where G and C are each a subset of the vertex set of the polygon, which can include points where the interior angle is of 180 degrees. Consequently, a simple discretization procedure shows that Vertex-Boundary Art Gallery and Boundary-Vertex Art Gallery are both FPT parameterized by r as well. However, we do not know how to handle Vertex-Point Art Gallery and Point-Vertex Art Gallery; determining whether these variants are FPT with respect to r remains open. Second, for variants where both X $\neq$ Vertex and Y $\neq$ Vertex, the design of *exact* algorithms poses extremely difficult challenges. As discussed earlier, these cases are not even known to be in NP; in particular, Point-Point Art gallery is $\exists\mathbb{R}$-hard [2]. Moreover, there is only one known exact algorithm that resolves these cases and it employs extremely powerful machinery (as a black box), not known to be avoidable. Third, note that our result is among very few *positive* results that concern *optimal* solutions to (any case of) Art Gallery.

Along the way to establish our main result, we prove that a constraint satisfaction problem called Monotone 2-CSP is solvable in polynomial time. This result might be of independent interest. Informally, in Monotone 2-CSP, we are given k variables and m constraints. Each constraint is of the form $[x \,\mathsf{sign}\, f(x')]$ where x and x' are variables, $\mathsf{sign} \in \{\leq, \geq\}$, and f is a *monotone* function. The objective is to assign an integer from $\{0, 1, \ldots, N\}$ to each variable so that all of the constraints will be satisfied. For this problem, we develop a surprisingly simple algorithm based on a reduction to 2-CNF-SAT.

▶ **Theorem 3 (♠[2]).** *Monotone 2-CSP is solvable in polynomial time.*

The main technical component of our work is an exponential-time reduction that creates an exponential (in r) number of instances of Monotone 2-CSP so that the original instance is a Yes-instance if and only if at least one of the instances of Monotone 2-CSP is a Yes-instance. Our reduction is done in two stages due to its structural complexity. In the first stage of the reduction, we aim to make "guesses" that determine the relations between the "elements" of the problem (that are the "critical" visibility relations in our case) and thereby elucidate and further binarize them (which, in our case, is required to impose order on guards). This part requires exponential time (given that there are exponentially many guesses) and captures the "NP-hardness" of the problem. Then, the second stage of the reduction is to translate each guess into an instance of Monotone 2-CSP. This part, while

[2] Details of the results marked with ♠ can be found in the full version of the paper [4].

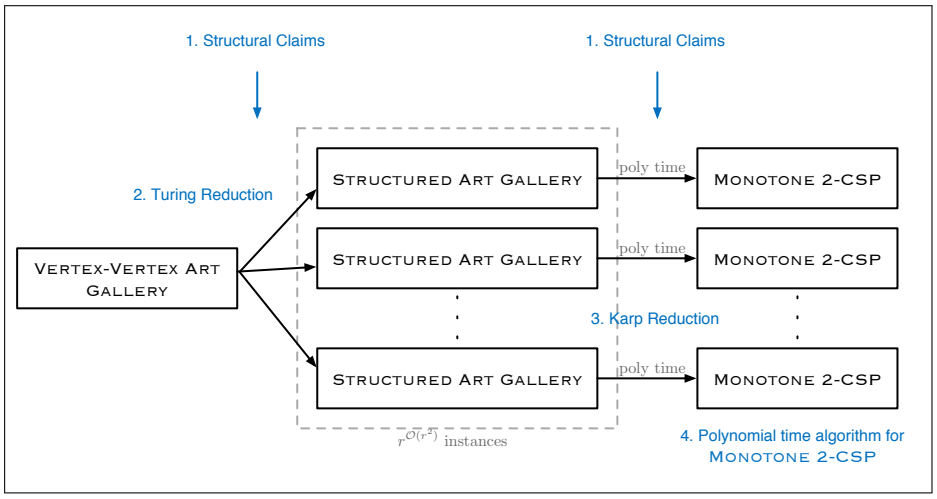


Figure 2 The four components of our proof.

requiring polynomial time, relies on a highly non-trivial problem-specific insight – specifically, here we need to assert that the relations considered earlier can be encoded by constraints that are not only binary, but that the functions they involve are *monotone*. We strongly believe that our approach can be proven fruitful to resolve the parameterized complexity of other problems of discrete geometric flavour.

1.3 Our Methods and Preliminaries

Our Methods. The proof of Theorem 2 consists of four components (see Fig. 2). The first component (in Section 2.1) establishes several structural claims regarding monotone properties of visibility in polygons. Informally, we order the vertices of the polygon according to their appearance on the boundary, and consider each sequence between two reflex vertices to be a "convex region". Then, we argue that for every pair of convex regions, as we "move along" one of them, the (index of the) first vertex in the other region that we see either never becomes smaller or never becomes larger. Symmetrically, this claim also holds for the last visible vertices that we encounter. In addition, we argue that if a vertex sees some two vertices in a convex region, then it also sees all vertices in between these two vertices.

Our second component (in Section 2.2) is a Turing reduction to an intermediate problem that we term Structured Art Gallery. Roughly speaking, in this problem, each convex region "announces" how many guards it will contain, and how many guards are necessary to see it completely. In addition, it announces that a prefix of the sequence that forms this region will be guarded by, say, "the i^{th} guard to be placed on region C", then the following subsequence will be guarded by, say, "the j^{th} guard to be placed on region C'", and so on, until it announces how a suffix of it is to be guarded. We stress that the identity of what is "the i^{th} guard to be placed on region C", or what is "the j^{th} guard to be placed on region C'", are of course not known, and should be discovered. Moreover, even the division into subsequences is not known. In Structured Art Gallery, we only focus on solutions that are of the above form. We utilize our second component not only to impose these additional conditions, but also to begin the transition from the usage of visibility-based conditions to function-based constraints. Specifically, functions called first and last will encode, for any vertex v and convex region C, the first and last vertices in C visible to v. To argue that such simple functions encode all necessary information concerning visibility, we make use of the structural claims established earlier.

Our third component (in Section 2.3) is a Karp reduction from Structured Art Gallery to the constraint satisfaction problem, Monotone 2-CSP, discussed in Section 1.2. This is the part of the proof that most critically relies on all of the structural claims established earlier. Here, we need to translate the constraints imposed by Structured Art Gallery into constraints that comply with the very restricted form of an instance of Monotone 2-CSP, namely, being monotone and involving only two variables. We remark that if one removes the requirement of monotonicity, or allows each constraint to consist of more variables, then the problem can be easily shown to encode Clique and hence become W[1]-hard (see Section 2.3). The translation entails a non-trivial analysis to ensure that all functions are indeed monotone. Specifically, each convex region requires its own set of tailored functions to enforce some relationships between the (unknown) guards it announced to contain and the (unknown) subsequences that these guards are supposed to see. In a sense, our first three components extract the algebraic essence of the Vertex-Vertex Art Gallery problem: by identifying monotone properties and making guesses to ensure binary dependencies between solution elements, the problem is encoded by a restricted constraint satisfaction problem.

Lastly, our fourth component is a relatively simple polynomial-time algorithm for Monotone 2-CSP (see Theorem 3), based on a reduction to 2-CNF-SAT. The crux is *not* to encode every pair of a variable of Monotone 2-CSP and a potential value for it as a variable of 2-CNF-SAT that signifies equality, because then, although the functions become easily encodable in the language of 2-CNF-SAT, it is unclear how to ensure that each variable of Monotone 2-CSP will be in exactly one pair that corresponds to a variable assigned true when satisfying the 2-CNF-SAT formula. Indeed, the naive approach seems futile, because it does not exploit the monotonicity of the input functions. Instead, for each pair of a variable of Monotone 2-CSP and a potential value for it with the exception of 0, we introduce a variable of 2-CNF-SAT signifying that the variable is assigned *at least* the value in the pair. The assignment of value 0 is implicitly encoded by the negation of pairs with the value 1. Then, we can ensure that each variable is assigned exactly one value (when translating a truth assignment for the 2-CNF-SAT instance we created back into an assignment for the Monotone 2-CSP input instance), and by relying on the monotonicty of the input functions, we are able to encode them correctly in the language of 2-CNF-SAT.

For notational clarity, we describe our proof for Vertex-Vertex Art Gallery. However, all arguments extend in a straightforward manner to solve the annotated generalization of Vertex-Vertex Art Gallery where G and C are each a subset of the vertex set of the polygon. Then, simple discretization procedures yield the positive resolution of the parameterized complexity also of Vertex-Boundary Art Gallery and Boundary-Vertex Art Gallery (see Section 5 of the full version [4]).

Preliminaries. We use the abbreviation Art Gallery to refer to Vertex-Vertex Art Gallery. We model a polygon by a graph $P = (V, E)$ with $V = \{1, 2, \ldots, n\}$ and $E = \{\{i, i+1\} : i \in \{1, \ldots, n-1\}\} \cup \{\{n, 1\}\}$. For a simple polygon P, we consider the boundary of P as part of its interior. We slightly abuse notation and refer to vertices $i \in V$ where the interior angle of P at i is 180 degrees as convex vertices. We denote the set of reflex vertices of P by $\mathsf{reflex}(P)$, and the set of convex vertices of P by $\mathsf{convex}(P)$. Given a non-convex polygon $P = (V, E)$, we suppose w.l.o.g. that $1 \in V$ is a reflex vertex. We say that a point p *sees* (or is *visible* to) a point q if every point of the line segment $\overline{pq}$ belongs to the interior of P. More generally, a set of points S *sees* a set of points Q if every point in Q is seen by at least one point in S. The definition of a convex polygon asserts the following.

▶ **Observation 4.** *Any point within a convex polygon P sees all points within P.*

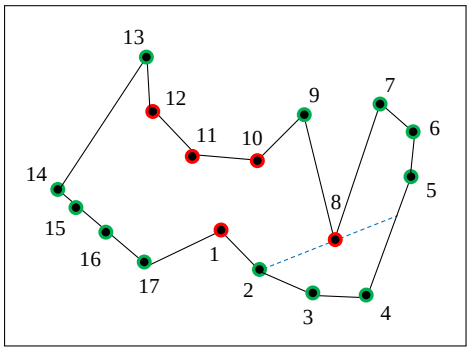


Figure 3 A simple polygon with three maximal convex regions: $[2, 7]$, $[9]$ and $[13, 17]$. Although $2, 5 \in [2, 7]$ belong to the same convex region, they do not see each other.

2 Algorithm for Art Gallery

In this section, we prove that Art Gallery is FPT with respect to r, the number of reflex vertices, by developing an algorithm with running time $2^{\mathcal{O}(r^2 \log r)} n^{\mathcal{O}(1)}$. We first present structural claims that exhibit the monotone way in which vertices in a so called "convex region" see vertices in another such region (Section 2.1). Then, we present a Turing reduction from Art Gallery to a problem called Structured Art Gallery (Section 2.2). Next, based on the claims in Section 2.1, we present our main reduction, which translates Structured Art Gallery to Monotone 2-CSP (Section 2.3). By developing an algorithm for Monotone 2-CSP, we conclude the proof.

2.1 Simple Structural Claims

We begin our analysis with the definition of a subsequence of vertices termed a convex region, illustrated in Fig. 3. Below, $j + 1$ for $j = n$ refers to 1. Because we assumed that vertex 1 of any non-convex polygon is a reflex vertex, any convex region $[i, j]$ satisfies $i \neq 1$.

▶ **Definition 5.** *Let $P = (V, E)$ be a simple polygon. A non-empty set of vertices $[i, j] = \{i, i + 1, \ldots, j\}$ is a* convex region *of P if all the vertices in $[i, j]$ are convex. In addition, if $i - 1 \geq 1$ and $j + 1$ are reflex vertices, then $[i, j]$ is a* maximal *convex region.*

In what follows, we would like to argue that for every two (not necessarily distinct) convex regions, one convex region sees the other in a manner that is "monotone" for each "orientation" in which we traverse these regions. To formalize this, we make use of the following notation, illustrated in Fig. 4. For a polygon $P = (V, E)$, a convex region $[i, j]$ of P and a vertex $v \in V$, denote the smallest and largest vertices in $[i, j]$ that are seen by v by $\mathsf{first}(v, [i, j])$ and $\mathsf{last}(v, [i, j])$, respectively. If v sees no vertex in $[i, j]$, define $\mathsf{first}(v, [i, j]) = \mathsf{last}(v, [i, j]) = \mathsf{nil}$. Accordingly, we define two types of monotone views. First, we address the orientation corresponding to first (see Fig. 4). Roughly speaking, we say that the way a convex region $[i, j]$ views a convex region $[i', j']$ is, say, non-decreasing with respect to first, if when we traverse $[i, j]$ from i to j and consider the first vertices in $[i', j']$ that vertices in $[i, j]$ see, then the sequence of these first vertices (viewed as integers) is a monotonically non-decreasing sequence once we omit all occurrences of nil from it.[3] We further demand that, between two equal vertices in this sequence, no nil occurs. Formally,

[3] A non-decreasing function (or sequence) is one that *never* decreases but can sometimes *not* increase.

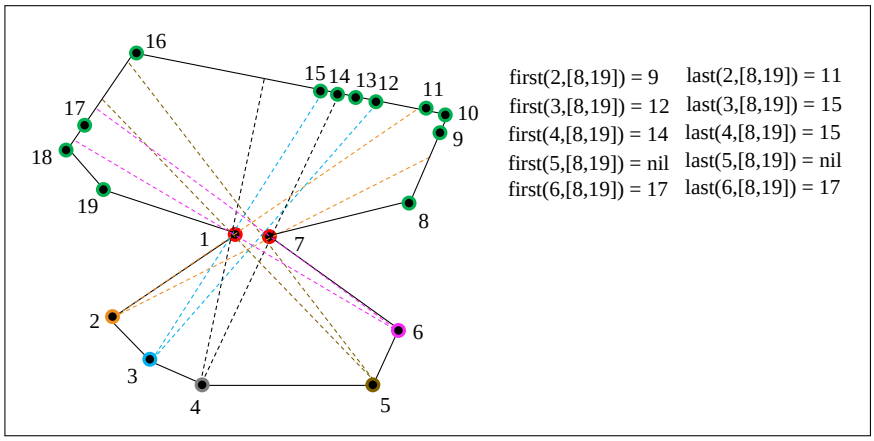


Figure 4 The way $[2, 6]$ views $[8, 19]$ is non-decreasing with respect to both first and last.

▶ **Definition 6.** *Let $P = (V, E)$ be a simple polygon. We say that the way a convex region $[i, j]$ of P views a (not necessarily distinct) convex region $[i', j']$ of P is* non-decreasing *(resp.* non-increasing*)* with respect to first *if for all $t, \widehat{t} \in \{i, i+1, \ldots, j\}$ such that $t \leq \widehat{t}$, $\mathsf{first}(t, [i', j']) \neq \mathsf{nil}$ and $\mathsf{first}(\widehat{t}, [i', j']) \neq \mathsf{nil}$, we have that*

- *$\mathsf{first}(t, [i', j']) \leq \mathsf{first}(\widehat{t}, [i', j'])$ (resp. $\mathsf{first}(t, [i', j']) \geq \mathsf{first}(\widehat{t}, [i', j'])$), and*
- *if $\mathsf{first}(t, [i', j']) = \mathsf{first}(\widehat{t}, [i', j'])$, then for all $p \in \{t, \ldots, \widehat{t}\}$, $\mathsf{first}(p, [i', j']) = \mathsf{first}(t, [i', j'])$.*[4]

Symmetrically, we address the orientation corresponding to the notation last.

▶ **Definition 7.** *Let $P = (V, E)$ be a simple polygon. We say that the way a convex region $[i, j]$ of P views a (not necessarily distinct) convex region $[i', j']$ of P is* non-decreasing *(resp.* non-increasing*)* with respect to last *if for all $t, \widehat{t} \in \{i, i+1, \ldots, j\}$ such that $t \leq \widehat{t}$, $\mathsf{last}(t, [i', j']) \neq \mathsf{nil}$ and $\mathsf{last}(\widehat{t}, [i', j']) \neq \mathsf{nil}$, we have that*

- *$\mathsf{last}(t, [i', j']) \leq \mathsf{last}(\widehat{t}, [i', j'])$ (resp. $\mathsf{last}(t, [i', j']) \geq \mathsf{last}(\widehat{t}, [i', j'])$), and*
- *if $\mathsf{last}(t, [i', j']) = \mathsf{last}(\widehat{t}, [i', j'])$, then for all $p \in \{t, \ldots, \widehat{t}\}$, $\mathsf{last}(p, [i', j']) = \mathsf{last}(t, [i', j'])$.*

The main purpose of this section is to prove the following two lemmas. We believe that some arguments required to establish their proofs might be folklore. The first lemma asserts that the subsequence seen by a vertex within a convex region does not contain "gaps".

▶ **Lemma 8 (♠).** *Let $P = (V, E)$ be a simple polygon, $v \in V$, and $[i, j]$ be a convex region of P. Then, v sees every vertex $t \in [i, j]$ such that $\mathsf{first}(v, [i, j]) \leq t \leq \mathsf{last}(v, [i, j])$.*[5]

The second lemma asserts that views are monotone. Intuitively, whenever we move along a convex region $[i, j]$ while viewing a convex region $[i', j']$ as described earlier, the first vertices (and last vertices) seen form a non-increasing or non-decreasing sequence.[6]

▶ **Lemma 9 (♠).** *Let $P = (V, E)$ be a simple polygon, and let $[i, j]$ and $[i', j']$ be two (not necessarily distinct) maximal convex regions of P. Then,* (i) *the way in which $[i, j]$ views $[i', j']$ with respect to* first *is either non-decreasing or non-increasing, and* (ii) *the way in which $[i, j]$ views $[i', j']$ with respect to* last *is either non-decreasing or non-increasing.*

---

[4] We remark that this condition cannot be replaced by "for all $p \in \{t, \ldots, \widehat{t}\}$, $\mathsf{first}(p, [i', j']) \neq \mathsf{nil}$". For example, in Fig. 4, neither $\mathsf{first}(4, [8, 19])$ nor $\mathsf{first}(6, [8, 19])$ is nil, but $\mathsf{first}(5, [8, 19]) = \mathsf{nil}$.

[5] If v does not see any vertex in $[i, j]$, the claim holds vacuously.

[6] We remark that we do not know whether it is possible that the first vertices would form a non-increasing (or non-decreasing) sequence and the last vertices would not. Our weaker claim suffices for our purposes.

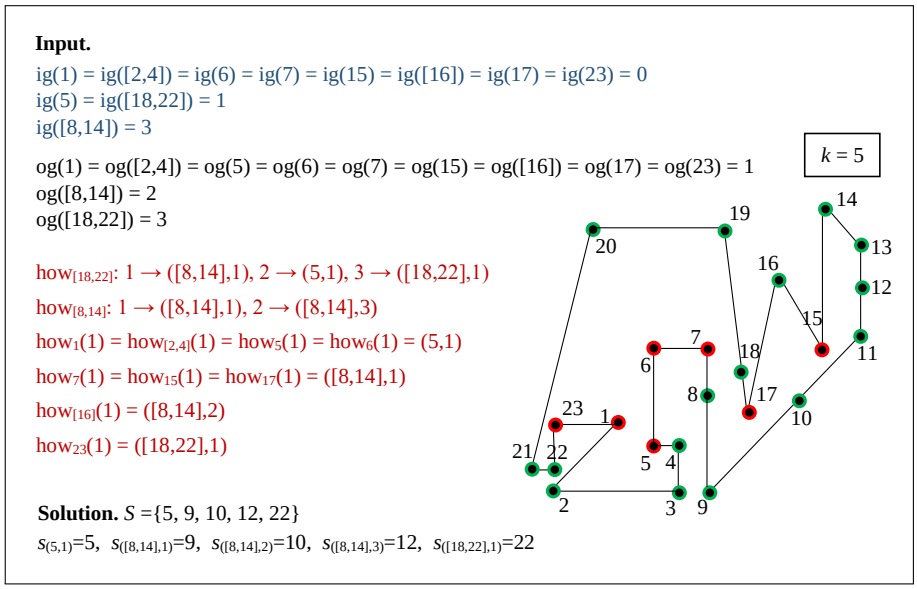


Figure 5 An input and a solution for the Structured Art Gallery problem.

2.2 Turing Reduction to Structured Art Gallery

An intermediate step in our reduction from Art Gallery to Monotone 2-CSP addresses an annotated version of Art Gallery, called Structured Art Gallery. Intuitively, in Structured Art Gallery each convex region "announces" how many guards it should contain, and how many guards are to be used to see it completely. In addition, each convex region announces by which *unknown guard* (identified as "the i^{th} guard to be placed on region C" for some i and C) its prefix should be guarded, by which unknown guard a region after this prefix should be guarded, and so on. In what follows, we formally define the Structured Art Gallery problem; then, we present our reduction from Art Gallery to Structured Art Gallery, and afterwards argue that this reduction is correct. For a polygon P, let $\mathcal{C}(P)$ be the set of maximal convex regions of P. Note that $|\mathcal{C}(P)| \leq r$.

Problem Definition. The input of Structured Art Gallery consists of a simple polygon $P = (V, E)$, a non-negative integer $k < r$, and the following functions (see Fig. 5).

- $\mathsf{ig} : \mathcal{C}(P) \cup \mathsf{reflex}(P) \to \{0, \ldots, k\}$, where $\sum_{x \in \mathcal{C}(P) \cup \mathsf{reflex}(P)} \mathsf{ig}(x) \leq k$. Intuitively, for a convex region or reflex vertex x, ig assigns the number of guards to be placed in x.
- $\mathsf{og} : \mathcal{C}(P) \cup \mathsf{reflex}(P) \to \{1, \ldots, k\}$, where for all $x \in \mathsf{reflex}(P)$, $\mathsf{og}(x) = 1$. Intuitively, for a convex region or reflex vertex x, og assigns the number of guards required to see x.
- For each $x \in \mathcal{C}(P) \cup \mathsf{reflex}(P)$, $\mathsf{how}_x : \{1, \ldots, \mathsf{og}(x)\} \to (\mathcal{C}(P) \cup \mathsf{reflex}(P)) \times \{1, \ldots, k\}$, where for each (y, i) in the image of how_x, $i \leq \mathsf{ig}(y)$. Intuitively, for any $j \in \{1, \ldots, \mathsf{og}(x)\}$, $\mathsf{how}_x(j) = (y, i)$ indicates that the j^{th} guard required to see x is the i^{th} guard placed in y.

The objective of Structured Art Gallery is to determine whether there exists a set $S \subseteq V$ of size at most k such that the following conditions hold:

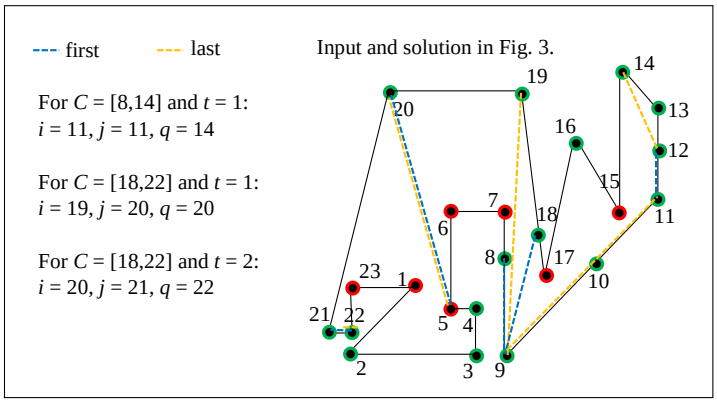


Figure 6 Condition 3b satisfied by a solution for Structured Art Gallery.

1. For each $x \in \mathcal{C}(P) \cup \mathsf{reflex}(P)$, $|S \cap x| = \mathsf{ig}(x)$.[7] Accordingly, for each $x \in \mathcal{C}(P) \cup \mathsf{reflex}(P)$ and $i \in \{1, \ldots, \mathsf{ig}(x)\}$, let $s_{(x,i)}$ denote the i^{th} largest vertex in $S \cap x$ (see Fig. 5).
2. For each $x \in \mathsf{reflex}(P)$, $s_{\mathsf{how}_x(1)}$ sees x.
3. For each $C \in \mathcal{C}(P)$, the following conditions hold:
   a. $\mathsf{first}(s_{\mathsf{how}_C(1)}, C)$ is the smallest vertex in C.
   b. For every $t \in \{1, \ldots, \mathsf{og}(C) - 1\}$, denote $i = \mathsf{last}(s_{\mathsf{how}_C(t)}, C)$, $j = \mathsf{first}(s_{\mathsf{how}_C(t+1)}, C)$ and $q = \mathsf{last}(s_{\mathsf{how}_C(t+1)}, C)$. Then, *(i)* $i \geq j - 1$, and *(ii)* $i \leq q - 1$. (See Fig. 6.)
   c. $\mathsf{last}(s_{\mathsf{how}_C(\mathsf{og}(C))}, C)$ is the largest vertex in C.

Informally, Condition 3b states that *(i)* the last vertex in C seen by its t^{th} guard should be at least as large as the predecessor of the first vertex in C seen by its $(t+1)^{th}$ guard, and *(ii)* the last vertex in C seen by its t^{th} guard should be smaller than the last vertex in C seen by its $(t+1)^{th}$ guard. The first condition ensures that no unseen "gaps" are created within C, while the second condition ensures that as the index t grows larger, the last vertex seen by the t^{th} guard grows larger as well. (The second condition will be part of our transition towards the interpretation of the objective of Art Gallery by *binary* constraints.)

Turing Reduction. Given an instance (P, k) of Art Gallery, in case $r \leq k$, output Yes.[8] Otherwise, the output of the reduction, $\mathsf{reduction}(P, k)$, is the set of all instances $(P, k, \mathsf{ig}, \mathsf{og}, \{\mathsf{how}_x\}|_{x \in \mathcal{C}(P) \cup \mathsf{reflex}(P)})$ of Structured Art Gallery.

Observe that $|\mathcal{C}(P) \cup \mathsf{reflex}(P)| \leq 2r$, and therefore the number of possible functions ig is upper bounded by $(k+1)^{2r}$, the number of possible functions og is upper bounded by k^{2r}, and for each $x \in \mathcal{C}(P) \cup \mathsf{reflex}(P)$, the number of possible functions how_x is upper bounded by $(2rk)^k$. Hence, the number of instances produced is upper bounded by $(k+1)^{2r} \cdot k^{2r} \cdot ((2rk)^k)^{2r}$. When $k \leq r$, this number is upper bounded by $r^{\mathcal{O}(r^2)}$. Moreover, the instances in $\mathsf{reduction}(P, k)$ can be enumerated with polynomial delay. Thus,

▶ **Observation 10.** *Let (P, k) be an instance of Art Gallery. Then, $|\mathsf{reduction}(P, k)| = r^{\mathcal{O}(r^2)}$, and $\mathsf{reduction}(P, k)$ is computable in time $r^{\mathcal{O}(r^2)} n^{\mathcal{O}(1)}$.*

[7] If $x \in \mathsf{reflex}(P)$, by $S \cap x$ we mean $S \cap \{x\}$.
[8] To comply with the formal definition of a Turing reduction, by Yes we mean a set with a single trivial Yes-instance of Structured Art Gallery.

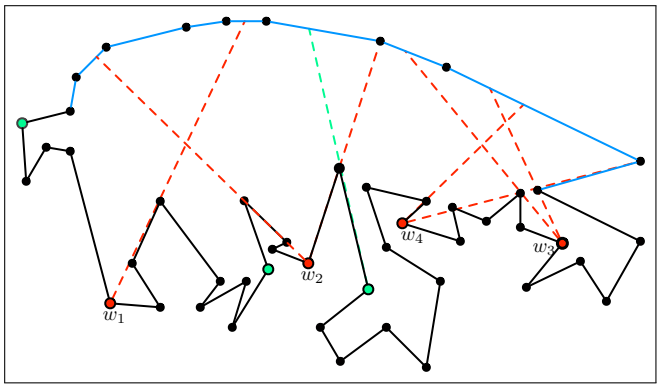


Figure 7 Example of a possible selection of $w_1, w_2, \ldots, w_p$. Solution vertices are colored green and red, and C is colored blue.

Correctness. Our proof of correctness crucially relies on Lemma 8 and Proposition 1.

▶ **Lemma 11.** *An instance (P, k) is a* Yes*-instance of* Art Gallery *if and only if there is a* Yes*-instance of* Structured Art Gallery *in* reduction(P, k).

Proof.

Forward Direction. Suppose that (P, k) is a Yes-instance of Art Gallery and that $r > k$. Accordingly, let $S \subseteq V$ be a solution to (P, k). We first define the function $\mathsf{ig} : \mathcal{C}(P) \cup \mathsf{reflex}(P) \to \{0, \ldots, k\}$ as follows. For each $x \in \mathcal{C}(P) \cup \mathsf{reflex}(P)$, let $\mathsf{ig}(x) = |S \cap x|$. Because $|S| \leq k$ (since S is a solution to (P, k)), we have that $\sum_{x \in \mathcal{C}(P) \cup \mathsf{reflex}(P)} \mathsf{ig}(x) \leq k$. For each $x \in \mathcal{C}(P) \cup \mathsf{reflex}(P)$, we order the vertices in $S \cap x$ from smallest to largest, and denote them accordingly by $s_{(x,1)}, s_{(x,2)}, \ldots, s_{(x,\mathsf{ig}(x))}$.

We define the functions $\mathsf{og} : \mathcal{C}(P) \cup \mathsf{reflex}(P) \to \{1, \ldots, k\}$ and $\mathsf{how}_x : \{1, \ldots, \mathsf{og}(x)\} \to (\mathcal{C}(P) \cup \mathsf{reflex}(P)) \times \{1, \ldots, k\}$ for all $x \in \mathcal{C}(P) \cup \mathsf{reflex}(P)$. For each reflex vertex $x \in \mathsf{reflex}(P)$, define $\mathsf{og}(x) = 1$, and $\mathsf{how}_x(1) = (y, i)$ for some vertex $s_{(y,i)} \in S$ that sees x. The existence of such a vertex $s_{(y,i)}$ follows from the assertion that S is a solution to (P, k). For each convex region $C \in \mathcal{C}(P)$, define $\mathsf{og}(C)$ and how_C as follows. Let W denote the set of vertices in S that see at least one vertex in C. Since W sees C, there exists a vertex in W that sees the smallest vertex in C. Pick such a vertex arbitrarily and denote it by w_1. Now, if w_1 does not see the largest vertex in C, then there exists a vertex in W that sees the smallest vertex in C that is larger than the largest vertex seen by w_1. We pick such a vertex arbitrarily, and denote it by w_2. Next, if w_2 does not see the largest vertex in C, then there exists a vertex in W that sees the smallest vertex in C that is larger than the largest vertex seen by w_2. We pick such a vertex arbitrarily, and denote it by w_3. Similarly, we define $w_4, w_5, \ldots, w_p$, for the appropriate $p \in \{1, \ldots, k\}$ (see Fig. 7). Here, the supposition that $p \leq k$ follows from Lemma 8, which implies that $w_i \neq w_j$ for all distinct $i, j \in \{1, \ldots, p\}$. We define $\mathsf{og}(C) = p$, and for all $t \in \{1, \ldots, \mathsf{og}(C)\}$, we define $\mathsf{how}_C(t) = (y, i)$ for the pair $(y, i) \in (\mathcal{C}(P) \cup \mathsf{reflex}(P)) \times \{1, \ldots, k\}$ that satisfies $w_t = s_{(y,i)}$.

Our definitions directly ensure that for each $C \in \mathcal{C}(P)$, the following conditions hold:

1. $\mathsf{first}(s_{\mathsf{how}_C(1)}, C)$ is the smallest vertex in C.
2. For every $t \in \{1, \ldots, \mathsf{og}(C) - 1\}$, denote $i = \mathsf{last}(s_{\mathsf{how}_C(t)}, C)$, $j = \mathsf{first}(s_{\mathsf{how}_C(t+1)}, C)$ and $q = \mathsf{last}(s_{\mathsf{how}_C(t+1)}, C)$. Then, *(i)* $i \geq j - 1$, and *(ii)* $i \leq q - 1$.
3. $\mathsf{last}(s_{\mathsf{how}_C(\mathsf{og}(C))}, C)$ is the largest vertex in C.

By the arguments above, $I = (P, k, \mathsf{ig}, \mathsf{og}, \{\mathsf{how}_x\}|_{x\in\mathcal{C}(P)\cup\mathsf{reflex}(P)})$ is an instance of Structured Art Gallery, and S is a solution to I. Since $I \in \mathsf{reduction}(P, k)$, the proof of the forward direction is complete.

Reverse Direction. If $k \geq r$, then we output Yes (or rather a trivial Yes-instance), and by Proposition 1, indeed the input is a Yes-instance as well. Next, suppose that $k < r$, and there is a Yes-instance $I = (P, k, \mathsf{ig}, \mathsf{og}, \{\mathsf{how}_x\}|_{x\in\mathcal{C}(P)\cup\mathsf{reflex}(P)})$ in $\mathsf{reduction}(P, k)$. Accordingly, let $S \subseteq V$ be a solution to I. Then, $|S| \leq k$. Thus, to prove that (P, k) is a Yes-instance of Art Gallery, it suffices to show that S sees V. For each $x \in \mathsf{reflex}(P)$, $s_{\mathsf{how}_x(1)}$ sees x, and therefore S sees $\mathsf{reflex}(P)$.

Now, we show that S sees $\mathsf{convex}(P)$. To this end, we choose a convex region $[i, j] \in \mathcal{C}(P)$, and show that S sees $[i, j]$. Specifically, for each $p \in \{i, \ldots, j\}$, we prove that there is $t \in \{1, \ldots, \mathsf{og}([i, j])\}$ such that $s_{\mathsf{how}_{[i,j]}(t)}$ (which is a vertex in S) sees p. The proof is by induction on p. In the basis, where $p = i$, correctness follows from the assertion that $\mathsf{first}(s_{\mathsf{how}_{[i,j]}(1)}, [i, j])$ is the smallest vertex in $[i, j]$. Now, we suppose that the claim is correct for p, and prove it for $p + 1$. By the inductive hypothesis, there is $t \in \{1, \ldots, \mathsf{og}([i, j])\}$ such that $s_{\mathsf{how}_{[i,j]}(t)}$ sees p. If $s_{\mathsf{how}_{[i,j]}(t)}$ sees $p + 1$, then we are done. Thus, we now suppose that $s_{\mathsf{how}_{[i,j]}(t)}$ does not see $p + 1$. Then, $\mathsf{last}(s_{\mathsf{how}_{[i,j]}(t)}, [i, j]) = p$. We have two cases:

- First, consider the case where $t < \mathsf{og}([i, j])$. Then, because S is a solution to I, the vertex $p = \mathsf{last}(s_{\mathsf{how}_{[i,j]}(t)}, [i, j])$ is larger or equal to $d-1$ for $d = \mathsf{first}(s_{\mathsf{how}_{[i,j]}(t+1)}, [i, j])$. This means that $\mathsf{first}(s_{\mathsf{how}_{[i,j]}(t+1)}, [i, j]) \leq p + 1$. Moreover, p is smaller than the vertex $\mathsf{last}(s_{\mathsf{how}_{[i,j]}(t+1)}, [i, j])$. Thus, $p + 1 \leq \mathsf{last}(s_{\mathsf{how}_{[i,j]}(t+1)}, [i, j])$. Then, $\mathsf{first}(s_{\mathsf{how}_{[i,j]}(t+1)}, [i, j]) \leq p + 1 \leq \mathsf{last}(s_{\mathsf{how}_{[i,j]}(t+1)}, [i, j])$. By Lemma 8, $s_{\mathsf{how}_{[i,j]}(t+1)}$ sees $p + 1$.
- Second, consider the case where $t = \mathsf{og}([i, j])$. In this case, because S is a solution to I, we have that $\mathsf{last}(s_{\mathsf{how}_{[i,j]}(\mathsf{og}([i,j]))}, [i, j])$ is the largest vertex in $[i, j]$. Thus, $p + 1 \leq \mathsf{last}(s_{\mathsf{how}_{[i,j]}(\mathsf{og}([i,j]))}, [i, j])$, which is a contradiction. ◀

2.3 Karp Reduction to Monotone 2-CSP

We proceed to the second part of our proof, a reduction from Structured Art Gallery to Monotone 2-CSP.[9] (The analysis of this can be found in the full version of the paper [4]).

Problem Definition. The input of Monotone 2-CSP consists of a set X of *variables*, denoted by $X = \{x_1, x_2, \ldots, x_{|X|}\}$, a set C of *constraints*, and $N \in \mathbb{N}$ given in unary. Each constraint $c \in C$ has the form $[x_i \,\mathsf{sign}\, f(x_j)]$ where $i, j \in \{1, \ldots, |X|\}$, $\mathsf{sign} \in \{\geq, \leq\}$ and $f : \{0, \ldots, N\} \to \{0, \ldots, N\}$ is a monotone function. An assignment $\alpha : X \to \{0, \ldots, N\}$ *satisfies* a constraint $c = [x_i \,\mathsf{sign}\, f(x_j)] \in C$ if $[\alpha(x_i) \,\mathsf{sign}\, f(\alpha(x_j))]$ is true. The objective of Monotone 2-CSP is to decide if there exists an assignment $\alpha : X \to \{0, \ldots, N\}$ that satisfies all the constraints in C (see Fig. 8).

If the function f of a constraint $c = [x_i \,\mathsf{sign}\, f(x_j)]$ is constantly β (that is, for every $t \in \{0, \ldots, N\}$, $f(t) = \beta$), then we use the shorthand $c = [x_i \,\mathsf{sign}\, \beta]$. Moreover, we suppose that every constraint represented by a quadruple is associated with two distinct variables.

Karp Reduction. Given an instance $I = (P, k, \mathsf{ig}, \mathsf{og}, \{\mathsf{how}_x\}|_{x\in\mathcal{C}(P)\cup\mathsf{reflex}(P)})$ of Structured Art Gallery, define an instance $\mathsf{reduction}(I) = (X, C, N)$ of Monotone 2-CSP as follows. Let $k^\star = \sum_{e\in\mathcal{C}(P)\cup\mathsf{reflex}(P)} \mathsf{ig}(e)$, $X = \{x_1, x_2, \ldots, x_{k^\star}\}$ and $N = n + 1$. (Here,

[9] CSP is an abbreviation of Constraint Satisfaction Problem, and 2 is the maximum arity of a constraint.

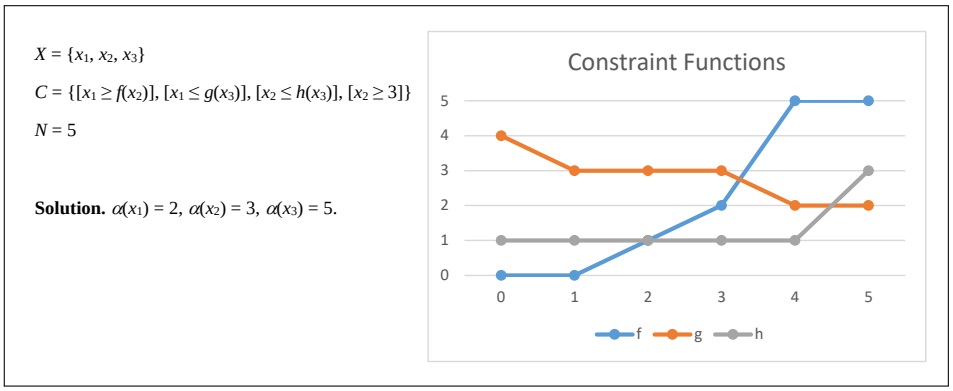


Figure 8 An input for MONOTONE 2-CSP that has a unique solution.

$n = |V|$.) Additionally, let bij be an arbitrary bijective function from X to $\{(e, i) : e \in \mathcal{C}(P) \cup \mathsf{reflex}(P), i \in \{1, \ldots, \mathsf{ig}(e)\}\}$. Intuitively, for any variable $x \in X$ with $\mathsf{bij}(x) = (e, i)$, we think of x as the i^{th} guard to be placed in region e. In particular, the value to be assigned to x is the identity of this guard. The values 0 and $n + 1$ are not identities of vertices in V, and we will ensure that no solution assignment assigns them; we note that these two values are useful because they will allow us to exclude assignments that should not be solutions. Next, we define our constraints and show that their functions are monotone.

Association. For each $x \in X$ with $\mathsf{bij}(x) = (e, i)$, we need to ensure that the vertex assigned to x is within the region e. To this end, we introduce the following constraints.

- If $e \in \mathsf{reflex}(P)$, then insert the constraint $[x = e]$. (That is, insert $[x \leq e]$ and $[x \geq e]$.)
- Else, $\mathsf{bij}(x) = (e, j)$ for $e \in \mathcal{C}(P)$. Let ℓ and h be the smallest and largest vertices in e, respectively, and insert the constraints $[x \geq \ell]$ and $[x \leq h]$.

Let A denote this set of constraints.

Order in a convex region. For all $x, x' \in X$ where $\mathsf{bij}(x) = (C, i)$ and $\mathsf{bij}(x') = (C, j)$ for the same convex region $C \in \mathcal{C}(P)$ and $i < j$, we need to ensure that the vertex assigned to x' is larger than the one assigned to x. To this end, we introduce the constraint $[x' \geq f(x)]$ where f is defined as follows. For all $q \in \{0, \ldots, N-1\}$, $f(q) = q+1$, and $f(N) = N$. Let O denote this set of constraints. We note that the constraints in $A \cup O$ together enforce each variable $x \in X$ with $\mathsf{bij}(x) = (C, i)$ for $C \in \mathcal{C}(P)$ to be assigned the i^{th} guard placed in C.

Guarding reflex vertices. For every reflex vertex $y \in \mathsf{reflex}(P)$ with $\mathsf{how}_y(1) = (e, i)$, we need to ensure that the vertex assigned to $x = \mathsf{bij}^{-1}(e, i)$ sees y. To this end, consider two cases. First, suppose that $e \in \mathsf{reflex}(P)$. Then, *(i)* if e does not see y, output NO, and *(ii)* else, no constraint is introduced. Second, suppose that $e \in \mathcal{C}(P)$. Denote $\ell = \mathsf{first}(y, e)$ and $h = \mathsf{last}(y, e)$. Then, *(i)* if ℓ (and thus also h) is nil, then output NO, and *(ii)* else, introduce the constraints $c^1_y = [x \geq \ell]$ and $c^2_y = [x \leq h]$.

Guarding first vertices in convex regions. For every convex region $C = [q, q'] \in \mathcal{C}(P)$ with $\mathsf{how}_C(1) = (e, i)$, we need to ensure that the vertex assigned to $x = \mathsf{bij}^{-1}(e, i)$ sees q, the first vertex of C. To this end, consider two cases. First, suppose that $e \in \mathsf{reflex}(P)$. Then, *(i)* if e does not see q, output NO, and *(ii)* else, no constraint is introduced. Second, suppose that $e \in \mathcal{C}(P)$. Denote $\ell = \mathsf{first}(q, e)$ and $h = \mathsf{last}(q, e)$. Then, *(i)* if ℓ is nil, then output NO, and *(ii)* else, insert the constraints $c^1_{(C,1)} = [x \geq \ell]$ and $c^2_{(C,1)} = [x \leq h]$.

Guarding last vertices in convex regions. For every convex region $C = [q, q'] \in \mathcal{C}(P)$ with $\mathsf{how}_C(\mathsf{og}(C)) = (e, i)$, we need to ensure that the vertex assigned to $x = \mathsf{bij}^{-1}(e, i)$ sees q', the last vertex of C. To this end, consider two cases. First, suppose that $e \in \mathsf{reflex}(P)$. Then, *(i)* if e does not see q', output No, and *(ii)* else, no constraint is introduced. Second, suppose that $e \in \mathcal{C}(P)$. Denote $\ell = \mathsf{first}(q', e)$ and $h = \mathsf{last}(q', e)$. Then, *(i)* if ℓ is nil, then output No, and *(ii)* else, insert the constraints $c^1_{(C,\mathsf{og}(C))} = [x \geq \ell]$ and $c^2_{(C,\mathsf{og}(C))} = [x \leq h]$.

Guarding middle vertices in convex regions. For every convex region $C \in \mathcal{C}(P)$ and $t \in \{2, \ldots, \mathsf{og}(C)\}$, we introduce four constraints based on the following notation.

- $(e, \gamma) = \mathsf{how}_C(t)$ and $x = \mathsf{bij}^{-1}(e, \gamma)$. Intuitively, the t^{th} vertex to guard C should be the γ^{th} guard to be placed in e, and its precise identity should be assigned to x. If no vertex in e sees at least one vertex in C, then return No.[10] Let ℓ and h be the smallest and largest vertices in e that see at least one vertex in C, respectively.
- $(e', \gamma') = \mathsf{how}_C(t-1)$ and $x' = \mathsf{bij}^{-1}(e', \gamma')$. Intuitively, the $(t-1)^{\text{th}}$ vertex to guard C should be the γ'^{th} guard to be placed in e', and its precise identity should be assigned to x'. If no vertex in e' sees at least one vertex in C, then return No. Let ℓ' and h' be the smallest and largest vertices in e' that see at least one vertex in C, respectively.

Now, insert the constraints $\widetilde{c}^1_{(C,t)} = [x \geq \ell]$ and $\widetilde{c}^2_{(C,t)} = [x \leq h]$. Intuitively, these two constraints *help* to ensure that x will be assigned a vertex that sees at least one vertex in C. However, these constraints alone are insufficient for this task – ensuring that we pick a guard between two vertices that see vertices in C does not ensure that this guard sees vertices in C.[11] Nevertheless, combined with our final constraints, this task is achieved.

Lastly, we consider two sets of four cases. The first set introduces a constraint to ensure that x, which stands for the t^{th} vertex to guard C, should satisfy that the first vertex in C seen by x is smaller or equal than the vertex larger by 1 than the last vertex in C seen by x', which stands for the $(t-1)^{\text{th}}$ vertex to guard C. On the other hand, the second set introduces a constraint to ensure that the last vertex in C seen by x is larger than the last vertex in C seen by x'. Together, because views have no "gaps", this would imply that x sees the vertex in C that is larger by 1 than the last vertex in C seen by x'. Due to lack of space, we only present the first case of each set. (Omitted details can be found in the full version [4]). To unify notation, if e (or e') is a reflex vertex, we say that the way e (or e') views C is non-decreasing with respect to both first and last.

First, consider the case where the way e' views C is non-decreasing with respect to last, and the way e views C is non-decreasing with respect to first. We insert a constraint $[x \leq f(x')]$, where f (having domain and range $\{0, \ldots, N\}$) is defined as follows.

- For all $i < \ell'$: $f(i) = 0$. Intuitively, we forbid x to be assigned a vertex smaller than the first vertex in e that can see C.
- For $i = \ell', \ell'+1, \ldots, h'$: Denote $a = \mathsf{last}(i, C)$. We have two subcases.
  - If **(i)** $a = \mathsf{nil}$, **(ii)** $a + 1 \notin C$, or **(iii)** $\mathsf{first}(j, C) \leq a + 1$ for no $j \in e$, let $f(i) = f(i-1)$. Roughly speaking, given that x' sees C, $a \neq \mathsf{nil}$ (in cases we will care about). Moreover, $a + 1 \in C$ will be ensured by the second set of cases and the way we guard the last vertex of a convex region. Lastly, $\mathsf{first}(j, C) \leq a + 1$ for some $j \in e$ will be ensured using that $f(i-1)$ (unless $f(i-1) = 0$) is a vertex that sees $a + 1$.

[10] In case $e \in \mathsf{reflex}(P)$, we mean that e itself does not see any vertex in C.

[11] For example, in Fig. 4, neither $\mathsf{first}(4, [8, 19])$ nor $\mathsf{first}(6, [8, 19])$ is nil, but $\mathsf{first}(5, [8, 19]) = \mathsf{nil}$.

- Else, let j be the largest vertex in e such that $\mathsf{first}(j, C) \leq a + 1$. Define $f(i) = j$. Intuitively, by enforcing x to be smaller or equal than j – the largest vertex in e that might see $a + 1$ – we ensure that the following condition holds: the first vertex x sees in C, under the assumption that it is not nil,[12] is smaller or equal to $a + 1$ (because the way e views C is non-decreasing with respect to first).

- For all $i > h'$: $f(i) = N$.

Second, consider the case where the ways e' and e view C are both non-decreasing with respect to last. We insert a constraint $[x \geq f(x')]$, where f is defined as follows.

- For all $i > h'$: $f(i) = N$.
- For $i = h', h' - 1, \ldots, \ell'$: Denote $a = \mathsf{last}(i, C)$. We have two subcases.
  - If **(i)** $a = \mathsf{nil}$, **(ii)** $a + 1 \notin C$, or **(iii)** $\mathsf{last}(j, C) \geq a + 1$ for no $j \in e$, let $f(i) = f(i + 1)$.
  - Else, let j be the smallest vertex in e such that $\mathsf{last}(j, C) \geq a + 1$. Define $f(i) = j$.
- For all $i < \ell'$: $f(i) = 0$.

Here, as the sign is $\geq$ and f is monotonically non-decreasing, f must be defined first for N, then for $N - 1$, and so on. Then, as long as i is such that $\mathsf{last}(j, C) \geq a + 1$ for no $j \in e$ (a case that we want to avoid), $f(i) = N$ and hence $[x \geq f(i)]$ cannot be satisfied.

References

1. Mikkel Abrahamsen, Anna Adamaszek, and Tillmann Miltzow. Irrational guards are sometimes needed. In *Proceedings of the 33rd International Symposium on Computational Geometry (SoCG)*, pages 3:1–3:15, 2017. doi:10.4230/LIPIcs.SoCG.2017.3.
2. Mikkel Abrahamsen, Anna Adamaszek, and Tillmann Miltzow. The art gallery problem is $\exists\mathbb{R}$-complete. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 65–73, 2018.
3. Alok Aggarwal. *The art gallery theorem: its variations, applications and algorithmic aspects*. PhD thesis, The Johns Hopkins University, Baltimore, Maryland, 1986.
4. Akanksha Agrawal, Kristine V. K. Knudsen, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. The parameterized complexity of guarding almost convex polygons, 2020. arXiv:2003.07793.
5. Édouard Bonnet and Tillmann Miltzow. Parameterized hardness of art gallery problems. In *Proceedings of the 24th Annual European Symposium on Algorithms (ESA)*, pages 19:1–19:17, 2016.
6. Vasek Chvátal. A combinatorial theorem in plane geometry. *Journal of Combinatorial Theory, Series B*, 18(1):39–741, 1975.
7. Pedro J. de Rezende, Cid C. de Souza, Stephan Friedrichs, Michael Hemmer, Alexander Kröller, and Davi C. Tozoni. Engineering art galleries. In *Algorithm Engineering - Selected Results and Surveys*, pages 379–417. Springer, 2016.
8. Steve Fisk. A short proof of Chvátal's watchman theorem. *Journal of Combinatorial Theory, Series B*, 24(3):374, 1978.
9. Subir Kumar Ghosh and Partha P. Goswami. Unsolved problems in visibility graphs of points, segments, and polygons. *ACM Computing Surveys*, 46(2):22:1–22:29, 2013.
10. Panos Giannopoulos. Open problems: Guarding problems. *Lorentz Workshop on Fixed-Parameter Computational Geometry, Leiden, the Netherlands*, page 12, 2016.
11. Matthew J Katz and Gabriel S Roisman. On guarding the vertices of rectilinear domains. *Computational Geometry*, 39(3):219–228, 2008.

[12] In the proof, to ensure that this vertex is indeed not nil, we will utilize both sets of cases, together with $\widetilde{c}^1_{(C,t)}$ and $\widetilde{c}^2_{(C,t)}$, to argue that x is between two vertices seen by $a + 1$ and hence must see $a + 1$ itself.

12 D. T. Lee and Arthur K. Lin. Computational complexity of art gallery problems. *IEEE Transactions on Information Theory*, 32(2):276–282, 1986.

13 Rolf Niedermeier. Ubiquitous parameterization - invitation to fixed-parameter algorithms. In *Proceedings of the 29th International Symposium on Mathematical Foundations of Computer Science (MFCS)*, pages 84–103, 2004.

14 Rolf Niedermeier. Reflections on multivariate algorithmics and problem parameterization. In *Proceedings of the 27th International Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 17–32, 2010.

15 Joseph O'rourke. *Art gallery theorems and algorithms*, volume 57. Oxford University Press Oxford, 1987.

16 Joseph O'Rourke and Kenneth J. Supowit. Some NP-hard polygon decomposition problems. *IEEE Transactions on Information Theory*, 29(2):181–189, 1983.

17 Dietmar Schuchardt and Hans-Dietrich Hecker. Two NP-hard art-gallery problems for ortho-polygons. *Mathematical Logic Quarterly*, 41(2):261–267, 1995.

18 Thomas C Shermer. Recent results in art galleries (geometry). *Proceedings of the IEEE*, 80(9):1384–1399, 1992.

19 Jorge Urrutia. Art gallery and illumination problems. *Handbook of computational geometry*, 1(1):973–1027, 2000.

Euclidean TSP in Narrow Strips

Henk Alkema
Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands
h.y.alkema@tue.nl

Mark de Berg
Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands
m.t.d.berg@tue.nl

Sándor Kisfaludi-Bak
Max Planck Institute for Informatics, Saarland Informatics Campus, Saarbrücken, Germany
sandor.kisfaludi-bak@mpi-inf.mpg.de

Abstract

We investigate how the complexity of Euclidean TSP for point sets P inside the strip $(-\infty, +\infty) \times [0, \delta]$ depends on the strip width δ. We obtain two main results.

- For the case where the points have distinct integer x-coordinates, we prove that a shortest bitonic tour (which can be computed in $O(n \log^2 n)$ time using an existing algorithm) is guaranteed to be a shortest tour overall when $\delta \leqslant 2\sqrt{2}$, a bound which is best possible.
- We present an algorithm that is fixed-parameter tractable with respect to δ. More precisely, our algorithm has running time $2^{O(\sqrt{\delta})}n^2$ for sparse point sets, where each $1 \times \delta$ rectangle inside the strip contains $O(1)$ points. For random point sets, where the points are chosen uniformly at random from the rectangle $[0, n] \times [0, \delta]$, it has an expected running time of $2^{O(\sqrt{\delta})}n^2 + O(n^3)$.

2012 ACM Subject Classification Theory of computation → Computational geometry; Theory of computation → Design and analysis of algorithms

Keywords and phrases Computational geometry, Euclidean TSP, bitonic TSP, fixed-parameter tractable algorithms

Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.4

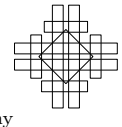
Related Version A full version of the paper is available at `https://arxiv.org/abs/2003.09948`.

Funding The work in this paper is supported by the Netherlands Organisation for Scientific Research (NWO) through Gravitation-grant NETWORKS-024.002.003.

Acknowledgements We thank Remco van der Hofstad for discussions about the probabilistic analysis of an earlier version of the algorithm.

1 Introduction

In the Traveling Salesman Problem one is given an edge-weighted complete graph and the goal is to compute a tour – a simple cycle visiting all nodes – of minimum total weight. Due to its practical as well as theoretical importance, the Traveling Salesman Problem and its many variants are among the most famous problems in computer science and combinatorial optimization. In this paper we study the Euclidean version of the problem. In Euclidean TSP the input is a set P of n points in $\mathbb{R}^d$, and the goal is to compute a minimum-length tour visiting each point. Euclidean TSP in the plane was proven to be NP-hard in the 1970s [16, 21]. Around the same time, Christofides [4] gave an elegant (3/2)-approximation algorithm, which works in any metric space. For a long time it was unknown if Euclidean TSP is APX-hard, until Arora [2], and independently Mitchell [20], presented a PTAS. Mitchell's algorithm works for the planar case, while Arora's algorithm also works in higher dimensions. Rao and Smith [22] later improved the running time of Arora's PTAS, obtaining a running time of $2^{(1/\varepsilon)^{O(d)}}n + (1/\varepsilon)^{O(d)}n \log n$ in $\mathbb{R}^d$.




We are interested in exact algorithms for Euclidean TSP. As mentioned, the problem is already NP-hard in the plane. Unlike the general (metric) version, however, it can be solved in *subexponential* time, that is, in time $2^{o(n)}$. In particular, Kann [19] and Hwang et al. [17] presented algorithms with $n^{O(\sqrt{n})}$ running time. Smith and Wormald [26] gave a subexponential algorithm that works in any (fixed) dimension; its running time in $\mathbb{R}^d$ is $n^{O(n^{1-1/d})}$. Very recently De Berg et al. [8] improved this to $2^{O(n^{1-1/d})}$, which is tight up to constant factors in the exponent, under the Exponential-Time Hypothesis (ETH) [18].

There has also been considerable research on special cases of Euclidean TSP that are polynomial-time solvable. One example is Bitonic TSP, where the goal is to find a shortest *bitonic* tour. (A tour is bitonic if any vertical line crosses it at most twice; here the points from the input set P are assumed to have distinct x-coordinates.) It is a classic exercise [5] to prove that Bitonic TSP can be solved in $O(n^2)$ time by dynamic programming. De Berg et al. [9] showed how to speed up the algorithm to $O(n \log^2 n)$. When P is in convex position, then the convex hull of P is a shortest tour and so one can solve Euclidean TSP in $O(n \log n)$ time [10]. Deineko et al. [12] studied the case where the points need not all be on the convex hull; the points inside the convex hull, however, are required to be collinear. Their algorithm runs in $O(n^2)$ time. Deineko and Woeginger [13] extended this to the case where the points in the interior of the convex hull lie on k parallel lines, obtaining an $O(n^{k+2})$ algorithm. These results generalize earlier work by Cutler [6] and Rote [24] who consider point sets lying on three, respectively k, parallel lines. Deineko et al. [11] gave a fixed-parameter tractable algorithm for Euclidean TSP where the parameter k is the number of points inside the convex hull, with running time $O(2^k k^2 n)$. Finally, Reinhold [23] and Sanders [25] proved that when there exists a collection of disks centered at the points in P whose intersection graph is a single cycle – this is called the necklace condition – then the tour following the cycle is optimal. Edelsbrunner et al. [15] gave an $O(n^2 \log n)$ algorithm to verify if such a collection of disks exists (and, if so, find one).

Our contribution. The computational complexity of Euclidean TSP in $\mathbb{R}^d$ is $2^{\Theta(n^{1-1/d})}$ (for $d \geqslant 2$), assuming ETH. Thus the complexity depends heavily on the dimension d. This is most pronounced when we compare the complexity for $d = 2$ with the trivial case $d = 1$: in the plane Euclidean TSP takes $2^{\Theta(\sqrt{n})}$ time in the worst case, while the 1-dimensional case is trivially solved in $O(n \log n)$ time by just sorting the points. We study the complexity of Euclidean TSP for planar point sets that are "almost 1-dimensional". In particular, we assume that the point set P is contained in the strip $(-\infty, \infty) \times [0, \delta]$ for some relatively small δ and investigate how the complexity of Euclidean TSP depends on the parameter δ. As any instance of Euclidean TSP can be scaled to fit inside a strip, we need to make some additional restriction on the input. We consider three scenarios.

- *Integer x-coordinates.* Bitonic TSP can be solved in $O(n \log^2 n)$ time [9]. It is natural to conjecture that for points with distinct integer x-coordinates inside a sufficiently narrow strip, an optimal bitonic tour is a shortest tour overall. We give a (partially computer-assisted) proof that this is indeed the case: we prove that when $\delta \leqslant 2\sqrt{2}$ an optimal bitonic tour is optimal overall, and we show that the bound $2\sqrt{2}$ is best possible.
- *Sparse point sets.* We generalize the case of integer x-coordinate to the case where each rectangle $[x, x+1] \times [0, \delta]$ contains $O(1)$ points, and we investigate how the complexity of Euclidean TSP grows with δ. We show in the full version [1] that for sparse point sets an optimal tour must be k-tonic – a tour is k-tonic if it intersects any vertical line at most k-times – for $k = O(\sqrt{\delta})$. This suggests that one might be able to use a dynamic-programming algorithm similar to the ones for for points on k parallel lines [13, 24].

The latter algorithms run in $O(n^k)$ time, suggesting that a running time of $n^{O(\sqrt{\delta})}$ is achievable in our case. We give a much more efficient algorithm, which is fixed-parameter tractable (and subexponential) with respect to the parameter δ. Its running time is $2^{O(\sqrt{\delta})}n^2$.

- *Random point sets.* In the third scenario the points in P are drawn independently and uniformly at random from the rectangle $R := [0, n] \times [0, \delta]$. For this case we prove that the same algorithm as for sparse point sets has a (now expected) running time of $2^{O(\sqrt{\delta})}n^2 + O(n^3)$.

Notation and terminology. Let $P := \{p_1, \ldots, p_n\}$ be a set of points in a horizontal strip of width δ – we call such a strip a *δ-strip* – which we assume without loss of generality to be the strip $(-\infty, \infty) \times [0, \delta]$. We denote the x-coordinate of a point $p \in \mathbb{R}^2$ by $x(p)$, and its y-coordinate by $y(p)$. To simplify the notation, we also write x_i for $x(p_i)$, and y_i for $y(p_i)$. We sort the points in P such that $0 \leqslant x_i \leqslant x_{i+1}$ for all $1 \leqslant i < n$.

For two points $p, q \in \mathbb{R}^2$, we write pq to denote the *directed* edge from p to q. Paths are written as lists of points, so $(q_1, q_2, ..., q_m)$ denotes the path consisting of the edges $q_1q_2, \ldots, q_{m-1}q_m$. All points in a path must be distinct, except possibly $q_1 = q_m$ in which case the path is a tour. The length of an edge pq is denoted by $|pq|$, and the total length of a set E of edges is denoted by $\|E\|$.

A *separator* is a vertical line not containing any of the points in P that separates P into two non-empty subsets.

2 Bitonicity for points with integer x-coordinates

In this section we consider the case where the points in P have distinct integer x-coordinates. For our purposes, two separators s, s' that induce the same partitioning of P are equivalent. Therefore, we can define $\mathcal{S} := \{s_1, \ldots, s_{n-1}\}$ as the set of all combinatorially distinct separators, obtained by taking one separator between any two points p_i, p_{i+1}. Let E be a set of edges with endpoints in P. The *tonicity of E at a separator s*, written as $\mathrm{ton}(E, s)$, is the number of edges in E crossing s. We say that a set E has *lower tonicity* than a set F of edges, denoted by $E \preccurlyeq F$, if $\mathrm{ton}(E, s_i) \leqslant \mathrm{ton}(F, s_i)$ for all $s_i \in \mathcal{S}$. The set E has *strictly lower tonicity*, denoted by $E \prec F$, if there also exists at least one i for which $\mathrm{ton}(E, s_i) < \mathrm{ton}(F, s_i)$. Finally, we call a set E of edges *k-tonic* – or *monotonic* when $k = 1$, and *bitonic* when $k = 2$ – if $\mathrm{ton}(E, s_i) \leqslant k$ for all $s_i \in \mathcal{S}$.

The goal of this section is to prove the following theorem.

▶ **Theorem 1.** *Let P be a set of points with distinct and integer x-coordinates in a δ-strip. When $\delta \leqslant 2\sqrt{2}$, a shortest bitonic tour on P is a shortest tour overall. Moreover, for any $\delta > 2\sqrt{2}$ there is a point set P in a δ-strip such that a shortest bitonic tour on P is not a shortest tour overall.*

The construction for the case $\delta > 2\sqrt{2}$ is shown in Fig. 1. It is easily verified that, up to symmetrical solutions, the tours T_1 and T_2 are the only candidates for the shortest tour. Observe that $\|T_2\| - \|T_1\| = |p_1p_4| - |p_4p_5| = 3 - \sqrt{1 + \delta^2}$. Hence, for $\delta > 2\sqrt{2}$ we have $\|T_2\| < \|T_1\|$, which proves the lower bound of Theorem 1. The remainder of the section is devoted to proving the first statement.

Let P be a point set in a δ-strip for $\delta = 2\sqrt{2}$, where all points in P have distinct integer x-coordinates. Among all shortest tours on P, let T_{opt} be one that is minimal with respect to the $\preccurlyeq$-relation; T_{opt} exists since the number of different tours on P is finite. We claim that T_{opt} is bitonic, proving the upper bound of Theorem 1.

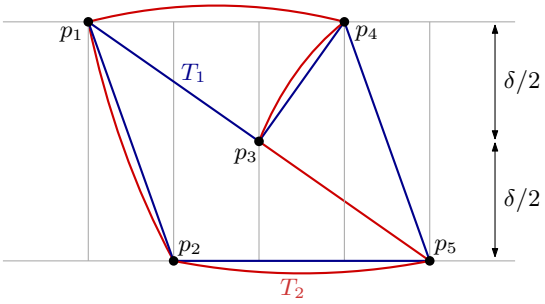


Figure 1 Construction for $\delta > 2\sqrt{2}$ for Theorem 1. The grey vertical segments are at distance 1 from each other. If $\delta > 2\sqrt{2}$ then T_1, the shortest bitonic tour (in blue), is longer than T_2, the shortest non-bitonic tour (in red).

Suppose for a contradiction that T_{opt} is not bitonic. Let $s^* \in \mathcal{S}$ be the rightmost separator for which $\text{ton}(T_{\text{opt}}, s^*) > 2$. We must have $\text{ton}(T_{\text{opt}}, s^*) = 4$ because otherwise $\text{ton}(T_{\text{opt}}, s) > 2$ for the separator $s \in \mathcal{S}$ immediately to the right of s^*, since there is only one point from P between s^* and s. Let F be the four edges of T_{opt} crossing s^*, and let E be the remaining set of edges of T_{opt}. Let Q be the set of endpoints of the edges in F. We will argue that there exists a set F' of edges with endpoints in Q such that $E \cup F'$ is a tour and (i) $\|F'\| < \|F\|$, or (ii) $\|F'\| = \|F\|$ and $F' \prec F$. We will call such an F' *superior to* F. Option (i) contradicts that T_{opt} is a shortest tour, and (ii) contradicts that T_{opt} is a shortest tour that is minimal with respect to $\preccurlyeq$ (since $E \cup F' \prec E \cup F$ if and only if $F' \prec F$). Hence, proving that such a set F' exists finishes the proof.

The remainder of the proof proceeds in two steps. In the first step we move the points in Q to obtain a set $\overline{Q}$ with consecutive integer coordinates, in such a way that there exists an edge set $\overline{F}$ on $\overline{Q}$ such that if an $\overline{F}'$ superior to $\overline{F}$ exists, then there also exists an F' superior to F. In the second step we then give a computer-assisted proof that the desired set $\overline{F}'$ exists.

Step 1: Finding a suitable $\overline{Q}$ with consecutive x-coordinates. Let T_{opt}, s^*, E, F and Q be defined as above. We assume without loss of generality that the x-coordinate of s^* is equal to $x^* + \frac{1}{2}$, where x^* is the largest integer such that the line $x = x^* + \frac{1}{2}$ intersects all four edges in F. Since the actual edges in E are not important for our arguments, we replace them by abstract "connections" specifying which pairs of endpoints of the edges in F are connected by paths of edges in E. It will be convenient to duplicate the points in Q that are shared endpoints of two edges in F, and add a connection between the two copies; see Fig. 2. We denote the set of connections obtained in this way by $\widetilde{E}$, and we call $\widetilde{E}$ the *connectivity pattern* of F (in $E \cup F$).

Next we show how to move the points in Q such that the modified set $\overline{Q}$ uses consecutive x-coordinates. Recall that $s^* : x = x^* + \frac{1}{2}$ is a separator that intersects all edges in F. Let Q_{left} and Q_{right} be the subsets of points from Q lying to the left and right of s^*, respectively. We will move the points in Q_{left} such that they will get consecutive x-coordinates with the largest one being equal to x^*, while the points in Q_{right} will get consecutive x-coordinates with the smallest one being $x^* + 1$.

We move the points in Q_{left} as follows. Let $z \leqslant x^*$ be the largest x-coordinate currently not in use by any of the points in Q_{left}. If Q_{left} lies completely to the right of the line $\ell(z) : x = z$, then we are done: the set of x-coordinates used by points in Q_{left} is $\{z+1, \ldots, x^*\}$. Otherwise we take an arbitrary edge $e \in F$ that crosses $\ell(z)$, and we move its left endpoint to the point $e \cap \ell(z)$; see Fig. 3(i). This process is repeated until Q_{left} uses consecutive x-coordinates.

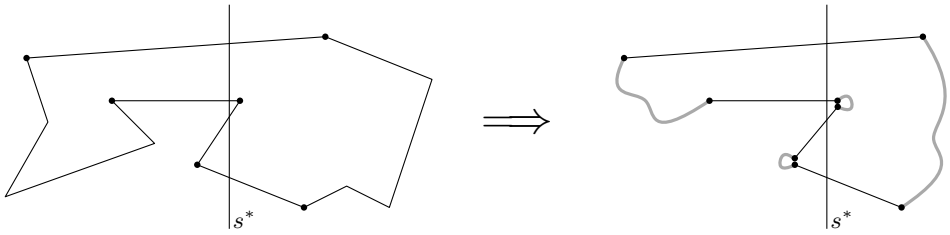


Figure 2 Replacing the paths connecting endpoints of edges in F by abstract connections. The copies of duplicated shared endpoints are slightly displaced in the figure to be able to distinguish them, but they are actually coinciding.

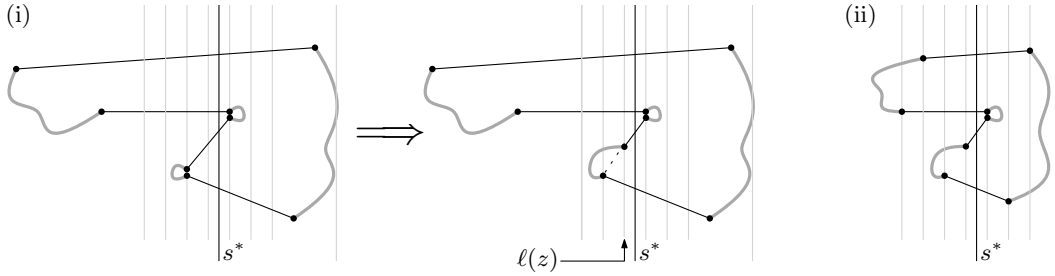


Figure 3 The process of moving the points in Q. Grey vertical lines have integer x-coordinates. (i) Moving a point in Q_{left} so that it gets x-coordinate z. (ii) A possible configuration after Q_{left} and Q_{right} have been treated.

After moving the points in Q_{left} we treat Q_{right} in a similar manner; the only difference is that now we define $z > x^*$ to be the smallest x-coordinate currently not in use by any of the points in Q_{right}. Fig. 3(ii) shows the final result for the example in part (i) of the figure.

Before we prove that this procedure preserves the desired properties, two remarks are in order about the process described above. First, in each iteration we may have different choices for the edge e crossing $\ell(z)$, and the final result depends on these choices. Second, when we move a point in Q to a new location, then the new x-coordinate is not used by Q but it may already be used by points in $P \setminus Q$. Neither of these facts cause any problems for the coming arguments.

Let $\overline{Q}$ be the set of points from Q after they have been moved to their new locations, and let $\overline{F}$ be the set of edges from F after the move. With a slight abuse of notation we still use $\widetilde{E}$ to specify the connectivity pattern on $\overline{F}$, which is simply carried over from F. The following lemma shows that we can use $\overline{F}$, $\overline{Q}$ and $\widetilde{E}$ in Step 2 of the proof.

▶ **Lemma 2.** *Let $E, F, Q, \widetilde{E}, \overline{F}, \overline{Q}$ be defined as above. Let $\overline{F}'$ be any set of edges (with endpoints in $\overline{Q}$) superior to $\overline{F}$, such that $\widetilde{E} \cup \overline{F}'$ is a tour. Then there is a set of edges F' (with endpoints in Q) superior to F such that $E \cup F'$ is a tour.*

Proof. We define F' in the obvious way, by simply taking $\overline{F}'$ and replacing each endpoint (which is a point in $\overline{Q}$) by the corresponding point in Q. Clearly $E \cup F'$ forms a tour if $\widetilde{E} \cup \overline{F}'$ forms a tour.

Suppose $\overline{F}'$ is superior to $\overline{F}$. We will now show that F' is superior to F. We will show this by first proving that $\|F\| - \|F'\| \geqslant \|\overline{F}\| - \|\overline{F}'\|$, and then proving that for any separator we have $s \in \mathcal{S}$, $\text{ton}(F, s) - \text{ton}(F', s) = \text{ton}(\overline{F}, s) - \text{ton}(\overline{F}', s)$.

Recall that each edge $\overline{e} \in \overline{F}$ is obtained from the corresponding edge $e \in F$ by moving one or both endpoints along the edge e itself. Also recall that we duplicated shared endpoints of edges in F, so if we move a point in Q we move the endpoint of a single edge in F. Hence,

$$\|F\| - \|\overline{F}\| = \text{total distance over which the points in } Q \text{ are moved.}$$

Since we added a connection to $\widetilde{E}$ between the two copies of a shared endpoint, each point in Q is incident to exactly one connection in $\widetilde{E}$ and, hence, to exactly one edge in F'. This means that if we move a point in Q we move the endpoint of a single edge in F', so

$$\|F'\| - \|\overline{F}'\| \leqslant \text{total distance over which the points in } Q \text{ are moved.}$$

We conclude that $\|F\| - \|F'\| \geqslant \|\overline{F}\| - \|\overline{F}'\|$.

Next, consider any separator $s \in \mathcal{S}$ to the left of s^*. Since points in Q_{left} are only moved towards s^* we know that

$$\text{ton}(F, s) - \text{ton}(\overline{F}, s) = (\text{number of points in } Q_{\text{left}} \text{ moved across } s) = \text{ton}(F', s) - \text{ton}(\overline{F}', s).$$

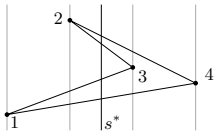
A similar argument shows that $\text{ton}(F, s) - \text{ton}(F', s) = \text{ton}(\overline{F}, s) - \text{ton}(\overline{F}', s)$ for any separator s to the right of s^*.

Now, since we have proven that $\|F\| - \|F'\| \geqslant \|\overline{F}\| - \|\overline{F}'\|$ and that for any separator we have $s \in \mathcal{S}$, $\text{ton}(F, s) - \text{ton}(F', s) = \text{ton}(\overline{F}, s) - \text{ton}(\overline{F}', s)$, we can conclude that if $\overline{F}'$ is superior to $\overline{F}$, then F' is superior to F. ◀

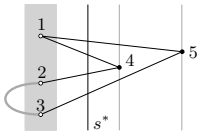
Step 2: Finding the set F'. The goal of Step 2 of the proof is the following: given the tour $T_{\text{opt}} = E \cup F$ inside a δ-strip of width $\delta = 2\sqrt{2}$, show that there exists a set F' of edges such that $E \cup F'$ is a tour and F' is superior to F. Lemma 2 implies that we may work with $\widetilde{E}$ and $\overline{F}$ instead of E and F (and then find $\overline{F}'$ instead of F').

In Step 1 we duplicated shared endpoints of edges in E. We now merge these two copies again if they are still at the same location. This will always be the case for the shared endpoint immediately to the right of the separator s^*, since we picked $s^* : x = x^* + \frac{1}{2}$ such that there is a shared endpoint at $x = x^* + 1$ and the copies of this endpoint will not be moved. So if n_{left} and n_{right} denote the number of distinct endpoints to the right and left of s^*, respectively, then $n_{\text{right}} \in \{2, 3\}$ and $n_{\text{left}} \in \{2, 3, 4\}$. We thus have six cases in total for the pair $(n_{\text{left}}, n_{\text{right}})$, as depicted in Fig. 4. Each of the six cases has several subcases, depending on the left-to-right order of the vertices inside the gray rectangles in the figure. Once we fixed the ordering, we can still vary the y-coordinates in the range $[0, \delta]$, which may lead to scenarios where different sets $\overline{F}'$ are required. We handle this potentially huge amount of cases in a computer-assisted manner, using an automated prover *FindShorterTour*$(n_{\text{left}}, n_{\text{right}}, \overline{F}, \widetilde{E}, X, \delta, \varepsilon)$. The input parameter X is an array where $X[i]$ specifies the set from which the x-coordinate of the i-th point in the given scenario may be chosen, where we assume w.l.o.g. that $x(s^*) = -1/2$; see Fig. 4. The role of the parameter ε will be explained below.

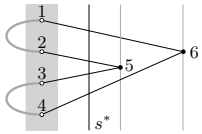
The output of *FindShorterTour* is a list of *scenarios* and an *outcome* for each scenario. A scenario contains for each point q an x-coordinate $x(q)$ from the set of allowed x-coordinates for q, and a range y-range$(q) \subseteq [0, 2\sqrt{2}]$ for its y-coordinate, where the y-range is an interval of length at most ε. The outcome is either Success or Fail. Success means that a set $\overline{F}'$ has been found with the desired properties: $\widetilde{E} \cup \overline{F}'$ is a tour, and for all possible instantiations of the scenario – that is, all choices of y-coordinates from the y-ranges in the scenario – we have $\|\overline{F}'\| < \|\overline{F}\|$. Fail means that such an $\overline{F'}$ has not been found, but it does not guarantee that such an $\overline{F'}$ does not exist for this scenario. The list of scenarios is complete in the sense that for any instantiation of the input case there is a scenario that covers it.



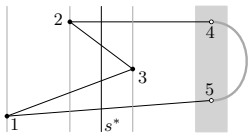
- $(n_{\text{left}}, n_{\text{right}}) = (2, 2)$
- $\overline{F} = \{(1,3), (1,4), (2,3), (2,4)\}$
- $\widetilde{E} = \emptyset$
- $X = [\{-2\}, \{-1\}, \{0\}, \{1\}]$



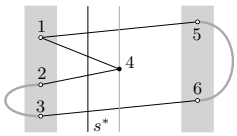
- $(n_{\text{left}}, n_{\text{right}}) = (3, 2)$
- $\overline{F} = \{(1,4), (1,5), (2,4), (3,5)\}$
- $\widetilde{E} = \{(2,3)\}$
- $X = [\{-3,-2,-1\}, \{-3,-2,-1\}, \{-3,-2,-1\}, \{0\}, \{1\}]$



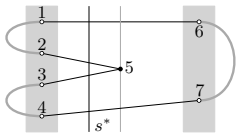
- $(n_{\text{left}}, n_{\text{right}}) = (4, 2)$
- $\overline{F} = \{(1,6), (2,5), (3.5), (4,6)\}$
- $\widetilde{E} = \{(1,2), (3,4)\}$
- $X = [\{-3,-2,-1\}, \{-3,-2,-1\}, \{-3,-2,-1\}, \{-3,-2,-1\}, \{0\}, \{2\}]$



- $(n_{\text{left}}, n_{\text{right}}) = (2, 3)$
- $\overline{F} = \{(1,3), (1,5), (2,3), (2,4)\}$
- $\widetilde{E} = \{(4,5)\}$
- $X = [\{-2\}, \{-1\}, \{0\}, \{1,2\}, \{1,2\}]$



- $(n_{\text{left}}, n_{\text{right}}) = (3, 3)$
- $\overline{F} = \{(1,4), (1,5), (2,4), (3,6)\}$
- $\widetilde{E} = \{(2,3), (5,6)\}$
- $X = [\{-3,-2,-1\}, \{-3,-2,-1\}, \{-3,-2,-1\}, \{0\}, \{1,2\}, \{1,2\}]$



- $(n_{\text{left}}, n_{\text{right}}) = (4, 3)$
- $\overline{F} = \{(1,6), (2,5), (3.5), (4,7)\}$
- $\widetilde{E} = \{(1,2), (3,4), (6,7)\}$
- $X = [\{-3,-2,-1\}, \{-3,-2,-1\}, \{-3,-2,-1\}, \{-3,-2,-1\}, \{0\}, \{1,2\}, \{1,2\}]$

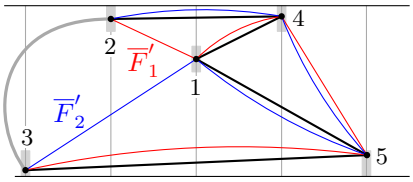
Figure 4 The six different cases that result after applying Step 1 of the proof. Points indicated by filled disks have a fixed x-coordinate. The left-to-right order of points drawn inside a grey rectangle, on the other hand, is not known yet. The vertical order of the edges is also not fixed, as the points can have any y-coordinate in the range $[0, 2\sqrt{2}]$.

FindShorterTour works brute-force, by checking all possible combinations of x-coordinates and subdividing the y-coordinate ranges until a suitable $\overline{F}'$ can be found or until the y-ranges have length at most ε. The implementation details of the procedure are the full version [1].

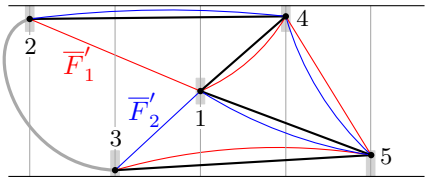
Note that case $(n_{\text{left}}, n_{\text{right}}) = (2, 3)$ in Fig. 4 is a subcase of case $(n_{\text{left}}, n_{\text{right}}) = (3, 2)$, if we exchange the roles of the points lying to the left and to the right of s^*. Hence, we ignore this subcase and run our automated prover on the remaining five cases, where we set $\varepsilon := 0.001$. It successfully proves the existence of a suitable set $\overline{F}'$ in four cases; the case where the prover fails is the case $(n_{\text{left}}, n_{\text{right}}) = (3, 2)$. For this case it fails for the two scenarios depicted in Fig. 5; all other scenarios for these cases are handled successfully (up to symmetries). For both scenarios we consider two alternatives for the set $\overline{F}'$: the set $\overline{F}'_1$ shown in red in Fig. 5, and the set $\overline{F}'_2$ shown in blue in Fig. 5. We will show that in any instantiation of both scenarios, either $\overline{F}'_1$ or $\overline{F}'_2$ is at least as short as $\overline{F}$; since both alternatives are bitonic this finishes the proof.

For $1 \leqslant i \leqslant 5$, let q_i be the point labeled i in Fig. 5. We first argue that (for both scenarios) we can assume without loss of generality that $y(q_2) = y(q_4) = 2\sqrt{2}$ and $y(q_3) = y(q_5) = 0$. To this end, consider arbitrary instantiations of these scenarios, and imagine moving q_2 and q_4 up to the line $y = 2\sqrt{2}$, and moving q_3 and q_5 down to the line $y = 0$. It suffices to show, for $i \in \{1, 2\}$, that if we have $\|\overline{F}'_i\| \leqslant \|\overline{F}\|$ after the move, then we also have $\|\overline{F}_i\| \leqslant \|\overline{F}\|$ before the move. This can easily be proven by repeatedly applying the following observation.

▶ **Observation 3.** *Let a, b, c be three points. Let ℓ be the vertical line through c, and let us move c downwards along ℓ. Let α be the smaller angle between ac and ℓ if $y(c) < y(a)$, and the larger angle otherwise, and let β be the smaller angle between bc and ℓ if $y(c) < y(b)$, and the larger angle otherwise, and suppose $\alpha < \beta$ throughout the move. Then the move increases $|ac|$ more than it increases $|bc|$.*



- $X = [\{-1\}, \{-2\}, \{-3, \}, \{0\}, \{1\}]$
- $Y = [[1.61, 1.62], [2.82, 2\sqrt{2}],$
  $[0, 0.01], [2.82, 2\sqrt{2}], [0, 0.01]]$



- $X = [\{-1\}, \{-3\}, \{-2, \}, \{0\}, \{1\}]$
- $Y = [[1.41, 1.42], [2.82, 2\sqrt{2}],$
  $[0, 0.01], [2.82, 2\sqrt{2}], [0, 0.01]]$

Figure 5 Two scenarios covering all subscenarios where the automated prover fails. Each point has a fixed x-coordinate and a y-range specified by the array Y; the resulting possible locations are shown as small grey rectangles (drawn larger than they actually are for visibility). For all subscenarios, at least one of $\overline{F}'_1$ (in red) and $\overline{F}'_2$ (in blue) is at most as long as $\overline{F}$ (in black).

So now assume $y(q_2) = y(q_4) = 2\sqrt{2}$ and $y(q_3) = y(q_5) = 0$. Consider the left scenario in Fig. 5, and let $y := y(q_3)$. If $y \geqslant (8\sqrt{2})/7$ then

$$|q_2q_1| + |q_4q_5| = \sqrt{1 + (2\sqrt{2} - y)^2} + 3 \leqslant 2 + \sqrt{4 + y^2} = |q_2q_4| + |q_1q_5|,$$

so $\|\overline{F}'_1\| \leqslant \|\overline{F}\|$. On the other hand, If $y \leqslant (8\sqrt{2})/7$ then

$$|q_3q_1| + |q_4q_5| = \sqrt{4 + y^2} + 3 \leqslant \sqrt{1 + (2\sqrt{2} - y)^2} + 4 = |q_1q_4| + |q_3q_5|,$$

so $\|\overline{F}'_2\| \leqslant \|\overline{F}\|$. So either $\overline{F}'_1$ or $\overline{F}'_2$ is at least as short as $\overline{F}$, finishing the proof for the left scenario in Fig. 5. The proof for the right scenario in Fig. 5 is analogous, with cases $y \geqslant \sqrt{2}$ and $y \leqslant \sqrt{2}$. This finishes the proof for the right scenario and, hence, for Theorem 1.

3 An algorithm for narrow strips

In this section we investigate how the complexity of Euclidean TSP depends on the width δ of the strip containing the point set P. Recall that a point set P inside a δ-strip is *sparse* if for every $x \in \mathbb{R}$ the rectangle $[x, x + 1] \times [0, \delta]$ contains $O(1)$ points.

▶ **Theorem 4.** *Let P be a set of n points in δ-strip.*

(i) *If for any $i \in \mathbb{Z}$ the square $[(i-1)\delta, i\delta] \times [0, \delta]$ contains at most k points, then we can solve Euclidean TSP on P in $2^{O(\sqrt{k})}n^2$ time.*

(ii) *If P is sparse then we can solve Euclidean TSP in $2^{O(\sqrt{\delta})}n^2$ time.*

Part (ii) of the theorem is a trivial consequence of part (i), so the rest of the section focuses on proving part (i). Our proof uses and modifies some techniques of [8]. For $i \in \mathbb{Z}$, let σ_i be the square $[(i-1)\delta, i\delta] \times [0, \delta]$. Define $n_i := |\sigma_i \cap P|$ – we assume without loss of generality that all points from P lie in the interior of a square σ_i – and let $k := \max_i n_i$. We say that a square σ_i is *empty* if $n_i = 0$.

We will regularly use that any subset E of edges from an optimal tour of a planar point set P has the *Packing Property* [8]: for any $t > 0$ and any square σ of side length t, the number of edges from E of length at least $t/4$ that intersect σ is $O(1)$. The Packing Property is at the heart of several subexponential algorithms [19, 26]. We also need the following lemma, which is essentially a special case of a recent result by De Berg et al. [8, Theorem 5]; see the full version [1] for a proof.

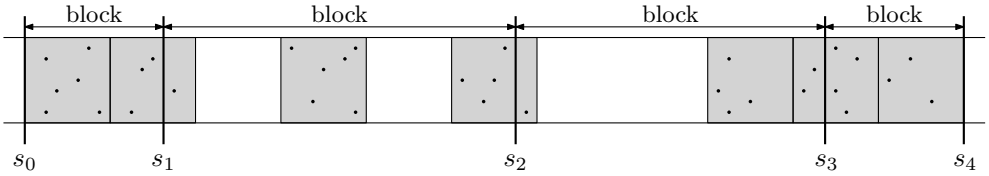


Figure 6 The separators $s_0, \ldots, s_{t+1}$ and the blocks they define.

▶ **Lemma 5.** *Let σ_i be a square as defined above. Then we can compute in $O(k^3)$ time a separator s intersecting σ_i such that the following holds. Let T_{opt} be an optimal TSP tour on P. For a separator s, let $T(s, \sigma_i)$ denote the set of edges from T_{opt} with both endpoints in $\sigma_{i-1} \cup \sigma_i \cup \sigma_{i+1}$ and crossing s. Then $|T(s, \sigma_i)| = O(\sqrt{k})$. Furthermore, there is a family $\mathcal{C}$ of $2^{O(\sqrt{k})}$ sets, which we call* candidate sets*, such that $T(s, \sigma_i) \in \mathcal{C}$, and this family can be computed in $2^{O(\sqrt{k})}$ time.*

Separators and blocks. Consider the sequence of non-empty squares σ_i, ordered from left to right. We use Lemma 5 to place a separator in every second block of this sequence. Let $\mathcal{S} := \{s_1, \ldots, s_t\}$ be the resulting (ordered) set of separators, and let s_0 and s_{t+1} denote separators coinciding with the left side of the leftmost non-empty square and the right side of the rightmost non-empty square, respectively; see Fig. 6. We call the region of the δ-strip between two consecutive separators s_{j-1} and s_j a *block*. Let $P_j \subseteq P$ denote the set of points in this block. Note that $|P_j| \leqslant 3k$.

For an edge set E and a separator s, let $E(s) \subseteq E$ denote the subset of edges intersecting s, and define $P_{\text{right}}(E, s)$ to be the set of endpoints of the edges in $E(s)$ that lie on or to the right of s. We call $P_{\text{right}}(E, s)$ the *endpoint configuration of E at s*. The next two lemmas rule out endpoint configurations with two "distant" points from the separator.

▶ **Lemma 6.** *Let $s_{\text{left}} : x = x_{\text{left}}$ and $s_{\text{right}} : x = x_{\text{right}}$ be two separators such that $x_{\text{right}} - x_{\text{left}} > 3\delta$, and suppose there is a point $z \in P$ with $x_{\text{left}} + 3\delta/2 < x(z) < x_{\text{right}} - 3\delta/2$. Then an optimal tour on P cannot have two edges that both cross s_{left} and s_{right}.*

Proof. Suppose for a contradiction that an optimal tour T has two directed edges, q_1q_2 and r_1r_2, that both cross s_{left} and s_{right}. (The direction of q_1q_2 and r_1r_2 is according to a fixed traversal of the tour.) If both edges cross s_{left} and s_{right} from left to right (or both cross from right to left) then replacing q_1q_2 and r_1r_2 by q_1r_1 and r_2q_2 gives a shorter tour – see Observation 14 in the full version [1] – leading to the desired contradiction.

Now suppose that q_1q_2 and r_1r_2 cross s_{left} and s_{right} in opposite directions. Assume w.l.o.g. that $x(q_1) < x_{\text{left}}$ and $x(r_2) < x_{\text{left}}$, and that z lies on the path from r_2 to q_1. Let $u_1, \ldots, u_k$, $v_1, \ldots, v_l$, and $w_1, \ldots, w_m$ be such that

$$T = (q_1, q_2, u_1, \ldots, u_k, r_1, r_2, v_1, \ldots, v_l, z, z_2, w_1, \ldots, w_m, q_1).$$

We claim that the tour T' defined as

$$T' = (q_1, r_2, v_1, \ldots, v_l, z, r_1, u_k, \ldots, u_1, q_2, z_2, w_1, \ldots, w_m, q_1)$$

is a strictly shorter tour. To show this, we will first change our point set P into a point set P' such that if $\|T'\| < \|T\|$ on P', then $\|T'\| < \|T\|$ also on P. To this end we replace q_1 by $q_1' := q_1q_2 \cap s_{\text{left}}$ and q_2 by $q_2' := q_1q_2 \cap s_{\text{right}}$, and we replace r_1 by $r_1' := r_1r_2 \cap s_{\text{left}}$ and r_2 by $r_2' := r_1r_2 \cap s_{\text{right}}$; see Fig. 7.

Finally, we replace z_2 by a point z_2' coinciding with z (note that if $z_2 = q_1$, we can split it before moving the resulting two points, analogous to the proof of Theorem 1). Using a similar reasoning as in the proof of Lemma 2, one can argue that the point set $P' := (P \setminus \{q_1, q_2, r_1, r_2, z_2\}) \cup \{q_1', q_2', r_1', r_2', z_2'\}$ has the required property. To get the desired contradiction it thus suffices to show that $\|T\| - \|T'\| > 0$ on P'. This is true because

$$\begin{aligned} \|T\| - \|T'\| &= |q_1'q_2'| + |r_1'r_2'| + |zz_2'| - |q_1'r_2'| - |zr_1'| - |q_2'z_2'| \\ &\geqslant |x_{\text{left}} - x_{\text{right}}| + |x_{\text{right}} - x_{\text{left}}| + 0 \\ &\qquad -\delta - (|x(z) - x_{\text{right}}| + \delta) - (|x_{\text{right}} - x(z)| + \delta) \\ &= 2(x(t) - x_{\text{left}}) - 3\delta > 0, \end{aligned}$$

where the last line uses that $x_{\text{left}} + 3\delta/2 < x(z)$. ◀

▶ **Lemma 7.** *Let $s_j \in \mathcal{S}$ be a separator, and let $\sigma_{j^*} = [(j^*-1)\delta), j^*\delta] \times [0, \delta]$ denote the square in which it is placed. Let T_{opt} be an optimal tour on P and let $V := P_{\text{right}}(T_{\text{opt}}, s_j)$ be its endpoint configuration at s_j. Let P' denote the set of input points with x-coordinates between $(j^*+1)\delta$ and $x(s_{j+3})$, and let P'' be the set of input points with x-coordinate larger than $x(s_{j+3})$. Then (i) $|P' \cap V| \leqslant c^*$ for some absolute constant c^*, and (ii) $|P'' \cap V| \leqslant 1$.*

Proof. Let uv be a tour edge crossing s_j. By definition of s_j, the number of edges crossing s_j with both endpoints in $\sigma_{j^*-1} \cup \sigma_{j^*} \cup \sigma_{j^*+1}$ is $O(\sqrt{k})$. Any other edge crossing s_j must fully cross σ_{j^*-1} or σ_{j^*+1} (or both), see Fig. 8. Therefore such edges have length at least δ. By the Packing Property, there can be at most $c^* = O(1)$ such edges. This proves (i).

To prove (ii), note there are five non-empty squares between s_j and s_{j+3}. Hence, there is a non-empty square between s_j and s_{j+3} with distance at least 2δ from s_j and s_{j+3}. Lemma 6 thus implies that T_{opt} has at most one edge crossing both s_j and s_{j+3}, proving (ii). ◀

Putting Lemma 5 and Lemma 7 together, we get the following corollary.

▶ **Corollary 8.** *Let T_{opt} be an optimal tour, let $s_j \in \mathcal{S}$ be a separator, and let $V \subset P$ be the endpoint configuration of T_{opt} at s_j. Then we can enumerate in $2^{O(\sqrt{k})} \cdot n$ time a family $\mathcal{B}_j$ of candidate endpoint sets such that $V \in \mathcal{B}_j$.*

In addition to the sets $\mathcal{B}_j$ $(j = 1, \ldots, t)$, we define $\mathcal{B}_0 = \mathcal{B}_{t+1} := \emptyset$.

Matchings, the rank-based approach, and representative sets. When we cut a tour using a vertical separator line, the tour falls apart into several paths. As in other TSP algorithms, we need to make sure that the paths on each side of the separator can be patched up into a

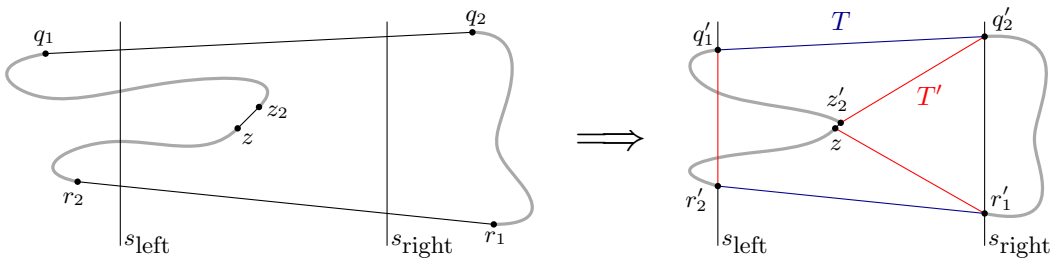


Figure 7 Illustration for the proof of Lemma 6. The point z_2' coincides with z but is slightly displaced for visibility. The sum of the length of the edges unique to T (displayed in blue) is strictly larger than the sum of the length of the edges unique to T' (displayed in red).

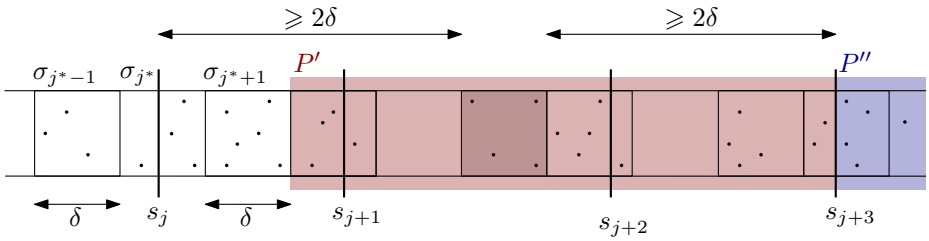


Figure 8 Tour edges crossing s_j and $x = (j^* + 1)\delta$ satisfy the Packing Property. Tour edges crossing s_j and s_{j+3} obey Lemma 6. The points in the red area form P'. The points in the blue area form P''.

Hamiltonian cycle. Following the terminology of [8], let P be our input point set, and let M be a perfect matching on a set $B \subseteq P$, where the points of B are called *boundary points*. A collection $\mathcal{P} = \{\pi_1, \ldots, \pi_{|B|/2}\}$ of paths *realizes* M *on* P if (i) for each edge $(p, q) \in M$ there is a path $\pi_i \in \mathcal{P}$ with p and q as endpoints, and (ii) the paths together visit each point $p \in P$ exactly once. We define the total length of $\mathcal{P}$ as the length of the edges in its paths. In general, the type of problem that needs to be solved on one side of a separator is called Euclidean Path Cover. The input to such a problem is a point set $P \subset \mathbb{R}^2$, a set of boundary points $B \subseteq P$, and a perfect matching M on B. The task is to find a collection of paths of minimum total length that realizes M on P.

To get the claimed running time, we need to avoid iterating over all matchings. We can do this with the so-called *rank-based approach* [3, 7]. As our scenario is very similar to the general Euclidean TSP, we can reuse most definitions and some proof ideas from [8].

Let $\mathcal{M}(B)$ denote the set of all perfect matchings on B, and consider a matching $M \in \mathcal{M}(B)$. We can turn M into a weighted matching by assigning to it the minimum total length of any collection of paths realizing M. In other words, weight(M) is the length of the solution of Euclidean Path Cover for input (P, B, M). We use $\mathcal{M}(P, B)$ to denote the set of all such weighted matchings on B. Note that $|\mathcal{M}(P, B)| = |\mathcal{M}(B)| = 2^{O(|B| \log |B|)}$.

We say that two matchings $M, M' \in \mathcal{M}(B)$ *fit* if their union is a Hamiltonian cycle. Consider a pair P, B. Let $\mathcal{R}$ be a set of weighted matchings on B and let M be another matching on B. We define $\text{opt}(M, \mathcal{R}) := \min\{\text{weight}(M') : M' \in \mathcal{R}, M' \text{ fits } M\}$, that is, $\text{opt}(M, \mathcal{R})$ is the minimum total length of any collection of paths on P that together with the matching M forms a cycle. A set $\mathcal{R} \subseteq \mathcal{M}(P, B)$ of weighted matchings is defined to be *representative* of another set $\mathcal{R}' \subseteq \mathcal{M}(P, B)$ if for any matching $M \in \mathcal{M}(B)$ we have $\text{opt}(M, \mathcal{R}) = \text{opt}(M, \mathcal{R}')$. Note that our algorithm is not able to compute a representative set of $\mathcal{M}(P, B)$, because it is also restricted by the Packing Property and Lemma 6, while a solution of Euclidean Path Cover for a generic P, B, M may not satisfy them. Let $\mathcal{M}^*(P, B)$ denote the set of weighted matchings in $\mathcal{M}(P, B)$ that have a corresponding Euclidean Path Cover solution satisfying the Packing Property and Lemma 6.

The basis of the rank-based method is the following result.

▶ **Lemma 9** (Bodlaender et al. [3], Theorem 3.7). *There exists a set $\overline{\mathcal{R}}$ consisting of $2^{|B|-1}$ weighted matchings that is representative of the set $\mathcal{M}(P, B)$. Moreover, there is an algorithm* Reduce *that, given a representative set $\mathcal{R}$ of $\mathcal{M}(P, B)$, computes such a set $\overline{\mathcal{R}}$ in $|\mathcal{R}| \cdot 2^{O(|B|)}$ time.*

Lemma 9 can also be applied for our case, where $\mathcal{R}$ is representative of $\mathcal{M}^*(P, B) \subseteq \mathcal{M}(P, B)$, the set of weighted matchings in $\mathcal{M}(P, B)$ that have a corresponding Euclidean Path Cover solution satisfying the Packing Property and Lemma 6.

We say that perfect matchings M on B and M' on B' are *compatible* if their union on $B \cup B'$ is either a single cycle or a collection of paths. The *join* of these matchings, denoted by $\text{Join}(M, M')$ is a perfect matching on the symmetric difference $B \triangle B'$ obtained by iteratively contracting edges with an incident vertex of degree 2 in the graph $(B \cup B', M \cup M')$.

The algorithm. Our algorithm is a dynamic program, where we define a subproblem for each separator index j, and each set of endpoints $B \in \mathcal{B}_j$. The value of $A[j, B]$ will be a representative set containing pairs (M, x), where M is a perfect matching on B and x is a real number equal to the total length of the path cover of $P_1 \cup \cdots \cup P_j \cup B$ realizing the matching M. The length of the entire tour will be the value corresponding to the empty matching at index $t + 1$, that is, it will be the value x such that $A[t + 1, \emptyset] = \{(\emptyset, x)\}$.

Our dynamic-programming algorithm works on a block-by-block basis (which explains the parameter j) and it solves subproblems inside a block using the algorithm *TSP-repr* by De Berg et al. [8] for Euclidean Path Cover on arbitrary planar point sets. Algorithm 1 gives our algorithm in a pseudocode, which is further explained below.

Algorithm 1 NarrowRectTSP-DP(P, δ).

Input: A set P of points in $[0, |P|] \times [0, \delta]$ chosen independently, uniformly at random
Output: The length of the shortest tour through all points in P

1: Compute the separators $s_1, \ldots, s_t$ using Lemma 5, as explained above.
2: $A[0, \emptyset] := \{(\emptyset, 0)\}$
3: **for** $j = 1$ to $t + 1$ **do**
4: **for all** $B \in \mathcal{B}_j$ **do**
5: $A[j, B] := \emptyset$
6: **for** $B' \in \mathcal{B}_{j-1}$ where $B' \cap \text{distant}(s_j) \subseteq B$ **do**
7: **for all** $(M, x) \in$ *TSP-repr*$(P_j \cup B' \cup B,\ B' \triangle B))$ **do**
8: **for all** $(M', x') \in A[j - 1, B']$ **do**
9: **if** M' and M are compatible **then**
10: Insert $(\text{Join}(M, M'), x + x')$ into $A[j, B]$
11: *Reduce*$(A[j, B])$
12: **return** length$(A[t + 1, \emptyset])$

The goal of Lines 4–11 is to compute a representative set $A[j, B]$ of $\mathcal{M}^*(P_1 \cup \cdots \cup P_j \cup B,\ B)$ of size $2^{O(\sqrt{k})}$. We say that a point $p \in P$ is *distant* (with respect to a separator s_j) if it is more than five non-empty squares after s_j, and denote the set of distant input points from s_j by $\text{distant}(s_j)$. First, we iterate over all sets $B \in \mathcal{B}_j$ in Line 4. Next, we consider certain boundary sets $B' \in \mathcal{B}_{j-1}$. Notice that if there is a distant point $p \in B' \cap \text{distant}(s_j)$, then a tour edge crossing s_{j-1} ending at p also crosses s_j, and thus p is also a (distant) point of B.

In Line 7 we call the algorithm of De Berg et al. [8] for Euclidean Path Cover within the block P_j and the boundary points $B' \cup B$. This gives us a representative set $\mathcal{R}$ of $\mathcal{M}^*(P_j \cup B' \cup B,\ B' \triangle B)$. For each weighted matching $(M, x) \in \mathcal{R}$, and for each weighted matching from the representative set $(M', x') \in A[j - 1, B']$, we check if M and M' are compatible. If so, then taking the union of the corresponding path covers gives a path cover of $P_1 \cup \cdots \cup P_j \cup B$ of total length $x + x'$, which realizes the matching $\text{Join}(M, M')$ on B; we then add $(\text{Join}(M, M^*), x + x^*)$ to $A[j, B]$ in Line 10.

After iterating over all boundary sets B', the entry $A[j, B]$ stores a set of weighted matchings, which we reduce to size $2^{O(\sqrt{k})}$ using the *Reduce* algorithm [3] in Line 11.

Note that in the final iteration, when $j = t + 1$, we take $B = \emptyset$. Now M and M' are compatible if and only if the union of the corresponding path covers is a Hamiltonian cycle. Line 11 then gives to a single entry the smallest weight. Therefore, the length of the only entry in $A[t + 1, \emptyset]$ after the loops have ended, is the length of the optimal TSP tour. Hence, the correctness of NarrowRectTSP-DP follows from the next lemma, proved in the full version [1].

▶ **Lemma 10.** *After Step 11, the set $A[j, B]$ is a representative set of $\mathcal{M}^*(P_1 \cup \cdots \cup P_j \cup B,\ B)$.*

Analysis of the running time. The loop of Lines 3–11 has $t + 2 = O(n)$ iterations. Each set $\mathcal{B}_j$ contains $2^{O(\sqrt{k})}n$ sets. For each choice of $B \in \mathcal{B}_j$, we have $2^{O(\sqrt{k})}$ options for B', since B can have at most one point distant from s_j by Lemma 6. The running time of *TSP-repr* is $T(|P|, |B|) = 2^{O(\sqrt{|P|+|B|})}$ [8, Lemma 8]. By Lemma 5 we have $|B| = O(\sqrt{k})$, so the running time of each call to *TSP-repr* in Algorithm 1 is $2^{O(\sqrt{3k+|B|}+\sqrt{|B|})} = 2^{O(\sqrt{k})}$. The representative set returned by *TSP-repr* has $2^{O(\sqrt{k})}$ weighted matchings, and the representative set of $A[j - 1, B']$ also has $2^{O(\sqrt{k})}$ matchings. Checking compatibility, joining and insertion in Lines 9 and 10 takes $\text{poly}(|M|, |M'|) = \text{poly}(k)$ time. Consequently, before executing the reduction in Line 11, the set $A[j, B]$ contains at most $2^{O(\sqrt{k})} \cdot 2^{O(\sqrt{k})} = 2^{O(\sqrt{k})}$ entries. The application of the *Reduce* algorithm ensures that the constant in the exponent in the $2^{O(\sqrt{k})}$ is kept under control; see Lemma 9. Hence, the total running time is $n \cdot 2^{O(\sqrt{k})}n \cdot 2^{O(\sqrt{k})} = 2^{O(\sqrt{k})}n^2$.

4 Random point sets inside a narrow rectangle

The algorithm from Theorem 4 also works efficiently on random point sets inside a narrow rectangle, as stated in the following theorem.

▶ **Theorem 11.** *Let P be a set of n points chosen independently and uniformly at random from $[0, n] \times [0, \delta]$. Then a shortest tour on P can be computed in $2^{O(\sqrt{\delta})}n^2 + O(n^3)$ expected time.*

Proof. To prove that the expected running time of our algorithm is as claimed, we need a good bound on k, the expected maximum number of points falling in any square $\sigma_i := [(i - 1)\delta, i\delta) \times [0, \delta]$. Note that $n_i := |P \cap \sigma_i|$ is a random variable with binomial distribution with parameters n and δ/n, so

$$\Pr[n_i = \ell] \;=\; \binom{n}{\ell}\left(\frac{\delta}{n}\right)^{\ell}\left(\frac{n-\delta}{n}\right)^{n-\ell}.$$

As above, let $k := \max_i n_i$. We need a strong upper bound on k. We have that

$$\Pr[k \geqslant \ell] \;=\; \Pr[\text{there is an } i \text{ such that } n_i \geqslant \ell] \;\leqslant\; \sum_{i=1}^{\lceil n/\delta \rceil} \Pr[n_i \geqslant \ell] \;=\; n \Pr[n_1 \geqslant \ell].$$

We use the Chernoff-Hoeffding theorem [14]: for a binomially distributed random variable x with parameters n, p and for $\ell > np$, we have that $\Pr(x \geqslant \ell) \leqslant \exp\left(-n \cdot D\left(\frac{\ell}{n}||p\right)\right)$, where $D(\frac{\ell}{n}||p) = \frac{\ell}{n}\ln(\frac{\ell/n}{p}) + \frac{n-\ell}{n}\ln(\frac{(n-\ell)/n}{1-p})$. Consequently,

$$\begin{aligned} n \Pr[n_1 \geqslant \ell] &\leqslant n \cdot \exp\left(-n\left(\tfrac{\ell}{n}\ln\tfrac{\ell/n}{\delta/n} + \tfrac{n-\ell}{n}\ln\tfrac{(n-\ell)/n}{(n-\delta)/n}\right)\right) \\ &= n \cdot \exp\left(-\ell\ln\tfrac{\ell}{\delta} - (n-\ell)\ln\tfrac{n-\ell}{n-\delta}\right). \end{aligned}$$

Assuming $e^2\delta < \ell$, we get

$$\begin{aligned}
\Pr[k \geqslant \ell] &< n \cdot \exp\left(-\ell \ln \tfrac{\ell}{\delta} + (n-\ell)\ln \tfrac{n-\delta}{n-\ell}\right) \\
&< n \cdot \exp\left(-\ell \ln \tfrac{\ell}{\delta} + (n-\ell)\ln \tfrac{n}{n-\ell}\right) \\
&< n \cdot \exp\left(-\ell \ln \tfrac{\ell}{\delta} + (n-\ell)\tfrac{\ell}{n-\ell}\right) \\
&= n \cdot \exp(-\ell(\ln \tfrac{\ell}{\delta} - 1)) \\
&< ne^{-\ell},
\end{aligned}$$

where the third inequality uses that $\ln(x) < x - 1$ for $x > 1$. The running time of the algorithm can now be bounded the following way.

$$\begin{aligned}
\mathbb{E}[\text{running time}] &\leqslant \Pr[k \leqslant e^2\delta] \cdot 2^{O(\sqrt{\delta})}n^2 + \sum_{\ell=\lfloor e^2\delta+1\rfloor}^{n} \Pr[k=\ell] \cdot 2^{O(\sqrt{\ell})}n^2 \\
&\leqslant 2^{O(\sqrt{\delta})}n^2 + n^2 \sum_{\ell=\lfloor e^2\delta+1\rfloor}^{n} \Pr[k \geqslant \ell] \cdot 2^{O(\sqrt{\ell})} \\
&\leqslant 2^{O(\sqrt{\delta})}n^2 + n^3 \sum_{\ell=\lfloor e^2\delta+1\rfloor}^{n} e^{-\ell}2^{O(\sqrt{\ell})} \\
&\leqslant 2^{O(\sqrt{\delta})}n^2 + n^3 \sum_{\ell=0}^{\infty} e^{-\ell+O(\sqrt{\ell})} \\
&\leqslant 2^{O(\sqrt{\delta})}n^2 + O(n^3)
\end{aligned}$$

◀

5 Concluding remarks

Our paper contains two main results on Euclidean TSP. First, we proved that for points with integer x-coordinates in a strip of width δ, an optimal bitonic tour is optimal overall when $\delta \leqslant 2\sqrt{2}$. The proof of this bound, which is tight in the worst case, is partially automated to reduce the potentially very large number of cases to two worst-case scenarios. It would be interesting to see if a direct proof can be given for this fundamental result. Furthermore, we note that the proof of Theorem 1 can easily be adapted to point sets of which the x-coordinates of the points need not be integer, as long as the difference between x-coordinates of any two consecutive points is at least 1.

Second, we gave a $2^{O(\sqrt{\delta})}n^2$ algorithm for sparse point sets, which also works in $2^{O(\sqrt{\delta})}n^2 + O(n^3)$ expected time for random point sets. For $\delta = \Theta(n)$ the running time becomes $2^{O(\sqrt{n})}$, which is optimal under ETH. For small δ it would be interesting to improve the dependency on n in the running time. Another direction for future research is to study the problem in higher dimensions. We believe that our algorithmic results may carry over to $\mathbb{R}^d$ to points that are almost collinear, that is, that lie in a narrow cylinder. Generalizing the results to, say, points lying in a narrow slab will most likely be more challenging.

References

1 H Alkema, M. de Berg, and S. Kisfaludi-Bak. Euclidean TSP in narrow strips. *arXiv*, 2020. arXiv:2003.09948.

2 S. Arora. Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems. *J. ACM*, 45(5):753–782, 1998. doi:10.1145/290179.290180.

3 H. L. Bodlaender, M. Cygan, S. Kratsch, and J. Nederlof. Deterministic single exponential time algorithms for connectivity problems parameterized by treewidth. *Inf. Comput.*, 243:86–111, 2015. doi:10.1016/j.ic.2014.12.008.

4 N. Christofides. Worst-case analysis of a new heuristic for the travelling salesman problem. Technical report, Graduate School of Industrial Administration, Carnegie Mellon University, 1976.

5 T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. *Introduction to Algorithms (3rd edition)*. MIT Press, 2009.

6 M. Cutler. Efficient special case algorithms for the n-line planar traveling salesman problem. *Networks*, 10:183–195, 1980. doi:10.1002/net.3230100302.

7 M. Cygan, S. Kratsch, and J. Nederlof. Fast hamiltonicity checking via bases of perfect matchings. *J. ACM*, 65(3):12:1–12:46, 2018. doi:10.1145/3148227.

8 M. de Berg, H.L. Bodlaender, S. Kisfaludi-Bak, and S. Kolay. An ETH-tight exact algorithm for Euclidean TSP. In *Proc. 59th IEEE Symp. Found. Comput. Sci. (FOCS)*, pages 450–461, 2018. doi:10.1109/FOCS.2018.00050.

9 M. de Berg, K. Buchin, B.M.P. Jansen, and G. Woeginger. Fine-grained complexity analysis of two classic TSP variants. In *Proc. 43rd Int. Conf. Automata Lang. Prog. (ICALP)*, pages 5:1–5:14, 2016. doi:10.4230/LIPIcs.ICALP.2016.5.

10 M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars. *Computational Geometry: Algorithms and Applications*. Springer, 2008. doi:10.1007/978-3-540-77974-2.

11 V.G. Deineko, M. Hoffmann, Y. Okamoto, and G.J. Woeginger. The traveling salesman problem with few inner points. *Oper. Res. Lett.*, 34(1):106–110, 2006. doi:10.1016/j.orl.2005.01.002.

12 V.G. Deineko, R. van Dal, and G. Rote. The convex-hull-and-line traveling salesman problem: a solvable case. *Inf. Proc. Lett.*, 51:141–148, 1994. doi:10.1016/0020-0190(94)00071-9.

13 V.G. Deineko and G. Woeginger. The convex-hull-and-k-line traveling salesman problem. *Inf. Proc. Lett.*, 59(3):295–301, 1996. doi:10.1016/0020-0190(96)00125-1.

14 J. Doe. Probability inequalities for sums of bounded random variables. *The Collected Works of Wassily Hoeffding*, pages 409–426, 1994. doi:10.1007/978-1-4612-0865-5_26.

15 H. Edelsbrunner, G. Rote, and E. Welzl. Testing the necklace condition for shortest tours and optimal factors in the plane. *Theoret. Comput. Sci.*, 66:157–180, 1989. doi:10.1016/0304-3975(89)90133-3.

16 M.R. Garey, R.L. Graham, and D.S. Johnson. Some NP-complete geometric problems. In *Proc. 8th ACM Symp. Theory Comp. (STOC)*, pages 10–22, 1976. doi:10.1145/800113.803626.

17 R.Z. Hwang, R.C. Chang, and R.C.T. Lee. The searching over separators strategy to solve some NP-hard problems in subexponential time. *Algorithmica*, 9(4):398–423, 1993. doi:10.1007/BF01228511.

18 R. Impagliazzo and R. Paturi. On the complexity of k-SAT. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001. doi:10.1006/jcss.2000.1727.

19 V. Kann. *On the approximability of NP-complete optimization problems*. PhD thesis, Royal Institute of Technology, Stockholm, 1992.

20 J.S.B. Mitchell. Guillotine subdivisions approximate polygonal subdivisions: A simple polynomial-time approximation scheme for geometric TSP, k-MST, and related problems. *SIAM J. Comput.*, 28(4):1298–1309, 1999. doi:10.1137/S0097539796309764.

21 C.H. Papadimitriou. The Euclidean traveling salesman problem is NP-complete. *Theoret. Comput. Sci.*, 4(3):237–244, 1977. doi:10.1016/0304-3975(77)90012-3.

22 S. Rao and W. D. Smith. Approximating geometrical graphs via 'spanners' and 'banyans'. In *Proc. 30th ACM Symp. Theory Comp. (STOC)*, pages 540–550, 1998. doi:10.1145/276698.276868.

23 A.G. Reinhold. Some results on minimal covertex polygons. *Manuscript, City College of New York*, 1965.

24 G. Rote. The n-line traveling salesman problem. Networks, 22:91–108, 1992. doi:10.1002/net.3230220106.

25 D. Sanders. *On extreme circuits*. PhD thesis, City University of New York, 1968.

26 W.D. Smith and N.C. Wormald. Geometric separator theorems and applications. In *Proc. 39th IEEE Symp. Found. Comput. Sci. (FOCS)*, pages 232–243, 1998. doi:10.1109/SFCS.1998.743449.

The ϵ-t-Net Problem

Noga Alon
Princeton University, NJ, USA
Tel Aviv University, Israel
nogaa@tau.ac.il

Bruno Jartoux
Ben-Gurion University of the Negev, Be'er-Sheva, Israel
jartoux@post.bgu.ac.il

Chaya Keller
Ariel University, Ariel, Israel
chayak@ariel.ac.il

Shakhar Smorodinsky
Ben-Gurion University of the Negev, Be'er-Sheva, Israel
shakhar@math.bgu.ac.il

Yelena Yuditsky
Ben-Gurion University of the Negev, Be'er-Sheva, Israel
yuditskyL@gmail.com

Abstract

We study a natural generalization of the classical ϵ-net problem (Haussler–Welzl 1987), which we call *the ϵ-t-net problem*: Given a hypergraph on n vertices and parameters t and $\epsilon \geq \frac{t}{n}$, find a minimum-sized family S of t-element subsets of vertices such that each hyperedge of size at least ϵn contains a set in S. When $t = 1$, this corresponds to the ϵ-net problem.

We prove that any sufficiently large hypergraph with VC-dimension d admits an ϵ-t-net of size $O(\frac{(1+\log t)d}{\epsilon} \log \frac{1}{\epsilon})$. For some families of geometrically-defined hypergraphs (such as the dual hypergraph of regions with linear union complexity), we prove the existence of $O(\frac{1}{\epsilon})$-sized ϵ-t-nets.

We also present an explicit construction of ϵ-t-nets (including ϵ-nets) for hypergraphs with bounded VC-dimension. In comparison to previous constructions for the special case of ϵ-nets (i.e., for $t = 1$), it does not rely on advanced derandomization techniques. To this end we introduce a variant of the notion of VC-dimension which is of independent interest.

2012 ACM Subject Classification Mathematics of computing → Hypergraphs; Theory of computation → Computational geometry

Keywords and phrases epsilon-nets, geometric hypergraphs, VC-dimension, linear union complexity

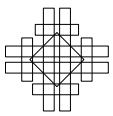
Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.5

Related Version A full version of this paper is available at `https://arxiv.org/abs/2003.07061`.

Funding *Noga Alon*: Research supported in part by NSF grant DMS-1855464, ISF grant 281/17, GIF grant G-1347-304.6/2016, and the Simons Foundation.
Bruno Jartoux: Research supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (Grant agreement No. 678765) and by Grant 635/16 from the Israel Science Foundation.
Chaya Keller: Part of the research was done when the author was at the Technion, Israel, and was supported by Grant 409/16 from the Israel Science Foundation.
Shakhar Smorodinsky: Research partially supported by Grant 635/16 from the Israel Science Foundation.
Yelena Yuditsky: Research supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (Grant agreement No. 678765) and by Grant 635/16 from the Israel Science Foundation.

Acknowledgements The authors are grateful to Adi Shamir for fruitful suggestions regarding the application of ϵ-t-nets to secret sharing.

36th International Symposium on Computational Geometry (SoCG 2020).
Editors: Sergio Cabello and Danny Z. Chen; Article No. 5; pp. 5:1–5:15

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


1 Introduction

1.1 Preliminaries

Hypergraphs and VC-dimension

A *hypergraph* is a pair $H = (V, \mathcal{E})$ where V is a set of *vertices* and $\mathcal{E} \subseteq 2^V$ is the set of *hyperedges* of H. When V is finite, H is a *finite* hypergraph.

A subset $V' \subseteq V$ is *shattered* if all its subsets are realized by $\mathcal{E}$, meaning $\{V' \cap e \colon e \in \mathcal{E}\} = 2^{V'}$. The *VC-dimension* of H, denoted by $\dim H$, is the cardinality of a largest shattered subset of V or $+\infty$ if arbitrarily large subsets are shattered (which does not happen in finite hypergraphs). This parameter plays a central role in statistical learning, computational geometry, and other areas of computer science and combinatorics [36, 26, 28].

ϵ-nets, Mnets

Let $\epsilon \in (0, 1)$. An *ϵ-net* for a finite hypergraph $(V, \mathcal{E})$ is a subset of vertices $S \subseteq V$ such that $S \cap e \neq \emptyset$ for every hyperedge $e \in \mathcal{E}$ such that $|e| \geq \epsilon|V|$.

Haussler and Welzl [18] proved that finite hypergraphs with VC-dimension d admit ϵ-nets of size $O(\frac{d}{\epsilon} \log \frac{d}{\epsilon})$, later improved to $O(\frac{d}{\epsilon} \log \frac{1}{\epsilon})$ [22]. In the last three decades, ϵ-nets have found applications in diverse areas of computer science, including machine learning [9], algorithms [12], computational geometry [6] and social choice [2].

Mustafa and Ray introduced the notion of *Mnets* [27]. For a hypergraph $(V, \mathcal{E})$ and for a fixed $\epsilon \in (0, 1)$, an *ϵ-Mnet* is a family $\{V_1, V_2, \ldots, V_\ell\}$ such that each $V_i \subseteq V$, each V_i is of size $\Theta(\epsilon|V|)$, and, for each $e \in \mathcal{E}$ such that $|e| \geq \epsilon|V|$, $V_i \subseteq e$ for some V_i. They constructed small ϵ-Mnets (i.e., such families with small ℓ) for several classes of geometric hypergraphs. These results were extended by Dutta et al. [16] using polynomial partitioning.

Explicit constructions

Although Hausssler and Welzl's proof of the ϵ-net theorem is probabilistic, several deterministic constructions of ϵ-nets for hypergraphs with finite VC-dimension have been devised [10, 24, 13]. The best result of this kind is Brönniman, Chazelle and Matoušek's $O(\epsilon^{-d} \log^d \frac{1}{\epsilon}|V|)$-time algorithm for computing an ϵ-net of size $O(\frac{d}{\epsilon} \log \frac{d}{\epsilon})$ [10]. These constructions are used to derandomize applications of ϵ-nets, such as low-dimensional linear programming [12].

In scenarios where the VC-dimension is $\Omega(\log|V|)$, the running time of these constructions becomes exponential in $|V|$. For one such scenario – the hypergraph induced by half-spaces on the discrete cube $V = \{-1, 1\}^d$ – Rabani and Shpilka [31] presented an efficient explicit construction of an ϵ-net, alas of sub-optimal size: $O(\epsilon^{-b}|V|^a)$ for some universal constants $a, b > 0$, whereas $O(|V|/\epsilon)$ can be obtained by random sampling. Like the aforementioned explicit constructions, the construction of [31] is based on derandomization.

1.2 Our problem

We denote by $\binom{X}{k}$ the set of all subsets of cardinality k (or "k-subsets") of the set X.

▶ **Definition 1.** *Let $H = (V, \mathcal{E})$ be a finite hypergraph, t a positive integer and $\epsilon \in (t/|V|, 1)$. A family $S \subseteq \binom{V}{t}$ of t-subsets of V is an* ϵ-t-net *for H if for every $e \in \mathcal{E}$ with $|e| \geq \epsilon|V|$ there is an $s \in S$ such that $s \subseteq e$.*

As mentioned already, for $t = 1$ this is equivalent to the ϵ-net notion, and for $t = \Theta(\epsilon|V|)$ this corresponds to the notion of ϵ-Mnets. In this paper we study the following problem.

▶ **Problem.** *How small are the smallest ϵ-t-nets for H? Can we compute them efficiently?*

Motivation

Instances of the ϵ-t-net problem appear naturally in various contexts in computer science and combinatorics. For example, the following is a basic motivating example for *secret sharing* [23, 34]: "Eleven scientists are working on a secret project. They wish to lock up the documents in a cabinet so that the cabinet can be opened if and only if six or more of the scientists are present. What is the smallest number of locks needed?". Consider a variant of this question in which the number of scientists is large. We still insist on the basic security condition – that no less than six scientists can open the cabinet. On the other hand, due to the large number of scientists, we do not require that any six should be able to do so, but rather any sufficiently large group of a certain kind, e.g., at least one tenth of all scientists including a representative of each university involved.

The classical secret sharing methods (see, e.g., [8]) distribute "keys" to subsets of 6 scientists so that any six scientists will be able to open the cabinet but no five will be able to do that. But as we require only certain groups of scientists to be able to open it, it is possible to distribute shared keys to only some of the 6-subsets. The questions: "What is the minimal number of 6-subsets we can achieve? and how can we choose the 6-subsets of scientists we distribute keys to?" are an instance of the ϵ-t-net problem – with $t = 6$, $\epsilon = 1/10$, and the hyperedges of the hypergraph being all groups of scientists that are required to be able to open the cabinet.

Other contexts in which the ϵ-t-net problem appears (described in the full version of this paper [3]) include the Turán numbers of hypergraphs, χ-boundedness of graphs, edge-coloring of hypergraphs and more.

Related work: ϵ-Nets and Mnets

For any t, the minimum size of an ϵ-t-net is sandwiched between the corresponding minimum sizes of ϵ-nets and of Mnets. Indeed, given an Mnet, one obtains an ϵ-t-net by picking one t-subset from each subset, and given an ϵ-t-net, one obtains an ϵ-net by taking one vertex from each t-subset. The survey [28] has most known bounds on these objects.

1.3 Results

Notation: we write $O_{x,y}(\cdot)$ when the implicit constants depend on parameters x and y.

Hypergraphs of finite VC-dimension have small ϵ-t-nets

Our main result is an existence result for small ϵ-t-nets.

▶ **Theorem 2.** *For every $\epsilon \in (0,1)$ and $t \in \mathbb{N} \setminus \{0\}$, every hypergraph on $\geq C_1 \left(\frac{t-1}{\epsilon}\right)^{d^*}$ vertices with VC-dimension d and dual shatter function $\pi^*_H(m) \leq Cm^{d^*}$ admits an ϵ-t-net of size $O(\frac{d(1+\log t)}{\epsilon} \log \frac{1}{\epsilon})$, all elements of which are pairwise disjoint. Here $C_1 = C_1(d^*, C)$.*

(The dual shatter function, described in Section 2, is a property of the hypergraph such that we may always take $d^* < 2^{d+1}$.)

This bound is asymptotically tight when $t = O(1)$, in the sense that there exist hypergraphs for which any ϵ-net, and consequently also any ϵ-t-net, is of size $\Omega(\frac{1}{\epsilon}\log\frac{1}{\epsilon})$ [22]. The proof of Theorem 2 involves a surprising relation between the ϵ-t-net problem and the existence of *spanning trees with a low crossing number*, proved by Welzl in 1988 [37].

Hypergraphs with VC-dimension 1 admit $O(\frac{1}{\epsilon})$-sized ϵ-nets [22] and ϵ-Mnets [16]. The latter fact yields the following result, albeit with worse constants. We offer a simple proof.

▶ **Theorem 3.** *For every positive integer t and $\epsilon \le \frac{1}{2}$, every finite hypergraph on $\ge t\lceil\frac{1}{\epsilon}\rceil$ vertices with VC-dimension 1 admits an ϵ-t-net of size at most $t\lceil\frac{1}{\epsilon}\rceil + 1$.*

An efficient explicit construction of ϵ-t-nets

Our second result is a new explicit construction of ϵ-t-nets, for all $t \ge 1$. The case of $t = 1$ (i.e., ϵ-nets) is of independent interest, as in this case our construction does not follow the proof strategy of Haussler and Welzl and does not use derandomization (unlike all previously known explicit constructions of ϵ-nets). On the other hand, it has a sub-optimal size of $O_d(\frac{1}{\epsilon^d})$, where d is the VC-dimension of the underlying hypergraph.

For a higher t, we introduce a new parameter of the hypergraph, which we call the *t-VC-dimension*. For hypergraphs of t-VC-dimension d, we construct ϵ-t-nets of size $O_d(\frac{1}{\epsilon^{d+t-1}})$. We give some first results on the relation between this new parameter and the standard VC-dimension.

Small ϵ-2-nets for geometric hypergraphs

In view of Theorem 2, which shows that for hypergraphs with a constant VC dimension one can obtain an ϵ-t-net of roughly the same size as the smallest ϵ-net, it is natural to ask whether a similar result can be achieved for geometrically-defined hypergraphs that admit an ϵ-net of size $O(\frac{1}{\epsilon})$. We obtain such results for several geometrically-defined hypergraphs in $\mathbb{R}^2$, including the intersection hypergraph of two families of pseudo-disks and the dual hypergraph of a family of regions with linear union complexity. Namely, we show that these hypergraphs have $O(\frac{1}{\epsilon})$-sized ϵ-2-nets provided they have $\Omega(\frac{1}{\epsilon})$ vertices. Interestingly, in some scenarios the minimum size of an ϵ-2-net is sensitive to the exact multiplicative constant: there are subhypergraphs (of the same hypergraph which is described in the appendix) on $\Theta(\frac{1}{\epsilon})$ vertices for which any ϵ-2-net is of size $\Omega(\frac{1}{\epsilon^2})$.

2 Construction of auxiliary hypergraphs

2.1 Some preparatory results

Sauer's lemma

Given a hypergraph $H = (V, \mathcal{E})$ the *trace (also known as projection or restriction) of H on $A \subseteq V$* is $\Pi_H(A) = \{A \cap e \colon e \in \mathcal{E}\}$; shattered subsets are those for which $\Pi_H(A) = 2^A$. The shatter function of H is

$$\pi_H \colon n \in \mathbb{N} \mapsto \max\{|\Pi_H(A)| : A \subseteq V,\ |A| \le n\}.$$

It is bounded by the *Sauer–Shelah lemma*:

▶ **Lemma 4** ([36, 33, 35])**.** *If $\dim H = d$ then $\pi_H(n) \le \binom{n}{0} + \binom{n}{1} + \cdots + \binom{n}{d}$. In particular, for $1 \le d \le n$ one has $\pi_H(n) \le (\frac{e}{d})^d \cdot n^d$, where e is Euler's number.*

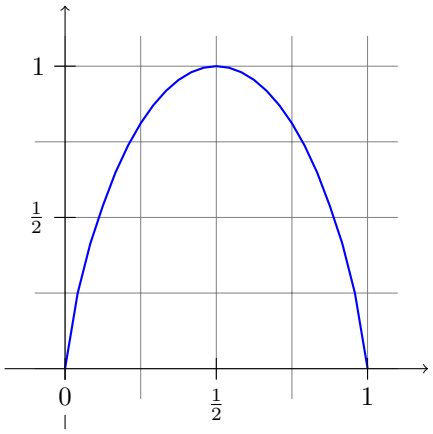


Figure 1 The binary entropy function.

Binary entropy function

This is $h\colon x \in (0,1) \mapsto -x\log x - (1-x)\log(1-x)$. (All logarithms are binary. See Figure 1.) We will use the following inequality:

$$\forall \alpha \in \left(0, \frac{1}{2}\right], \ \forall n \in \mathbb{N}, \quad \log \sum_{i=0}^{\lfloor \alpha n \rfloor} \binom{n}{i} \leq nh(\alpha). \tag{1}$$

The binary entropy function restricted to $(0, \frac{1}{2}]$ is invertible, and [11, Th. 2.2]:

$$\forall x \in (0,1), \quad \frac{x}{2\log\frac{6}{x}} \leq h^{-1}(x) \leq \frac{x}{\log\frac{1}{x}}. \tag{2}$$

2.2 A first hypergraph on t-subsets

▶ **Definition 5.** *Given a hypergraph $H = (V, \mathcal{E})$ and a positive integer t, let H^t be the hypergraph $(V^t, \mathcal{E}^t)$ where $V^t = \binom{V}{t}$ and $\mathcal{E}^t = \left\{\binom{e}{t} \colon e \in \mathcal{E}\right\}$. That is, its vertices are all t-element subsets of V and each hyperedge of H^t consists of all such subsets contained in a given hyperedge of H.*

For $t \in \mathbb{N} \setminus \{0,1\}$, let $\gamma_t = (th^{-1}(1/t))^{-1}$. Note that $\log t \leq \gamma_t \leq 2\log 6t$.

▶ **Proposition 6.** *If H is a hypergraph with $\dim H = d$ then $d - t + 1 \leq \dim H^t \leq \gamma_t d$.*

Proof. We assume that $t \geq 2$, as for $t = 1$, $\dim H^t = \dim H^1 = \dim H = d$.

To prove the left inequality, let $\{v_1, \ldots, v_d\}$ be a shattered subset of vertices in H, with $d \geq t-1$. There are $d - t + 1$ sets containing all vertices in $\{v_1, v_2, \ldots, v_{t-1}\}$ and exactly one in $\{v_t, v_{t+1}, \ldots v_d\}$. It is easy to see that they form a shattered subset in H^t.

For the right inequality, suppose to the contrary that P is a shattered set in H^t with $d' = |P| > \gamma_t d$. Let $S = \cup_{p\in P} p$; clearly $|S| \leq td'$. Observe also that $d' + t - 1 \leq |S|$. If this were not the case there would exist some $p_1 \in P$ such that $p_1 \subseteq \cup_{p\in P\setminus\{p_1\}} p$, which would contradict the fact that P is shattered.

We denote $|S| = \beta d'$; we have $1 < \beta \leq t$.

Since P is shattered in H^t, each $P_1 \subseteq P$ is of the form $P \cap \binom{e}{t} = \{p \in P \colon p \subseteq (S \cap e)\}$ for some $e \in \mathcal{E}$. Thus $|\Pi_H(S)| \geq |2^P| = 2^{d'}$.

On the other hand, $\dim H = d$, and so by Lemma 4, $|\Pi_H(S)| \le \binom{\beta d'}{0} + \binom{\beta d'}{1} + \cdots + \binom{\beta d'}{d}$. It follows from Equation (1) (with $\beta d' \ge \beta\gamma_t d > 2d$) that $d' \le \log|\Pi_H(S)| \le \beta d' h(\frac{d}{\beta d'})$. We show that $1 > \beta h(\frac{d}{\beta d'})$, a contradiction.

Note that $\frac{1}{t\gamma_t} \le \frac{1}{\gamma_t \beta} < \frac{1}{2}$. Since $t \mapsto \frac{h(t)}{t}$ is monotone decreasing in the range $(0,1)$, we have $\gamma_t\beta \cdot h(\frac{1}{\gamma_t\beta}) \le t\gamma_t \cdot h(\frac{1}{t\gamma_t}) = \gamma_t$. As h is increasing on $(0, \frac{1}{2})$, it follows that $\beta h(\frac{d}{\beta d'}) < \beta h(\frac{1}{\beta\gamma_t}) \le 1$. ◀

Proposition 6 allows us to slightly improve the "trivial" upper bound of $O(\frac{d^t}{\epsilon^t}(\log\frac{1}{\epsilon})^t)$ on the minimum size of an ϵ-t-net for any hypergraph with constant VC-dimension.

▶ **Corollary 7.** *Let H be a hypergraph on n vertices with VC-dimension d. For any t, ϵ such that $n \ge \frac{t}{\epsilon}$, H admits an ϵ-t-net of size $O(\frac{dt(1+\log t)}{\epsilon^t}\log\frac{1}{\epsilon})$.*

Indeed, observe that an ϵ^t-net for H^t is an ϵ-t-net for H, and apply the classical ϵ-net theorem to H^t.

2.3 A smaller, well-behaved hypergraph on t-subsets

A *spanning cycle* P for $H = (V, \mathcal{E})$ is a cycle graph on V that visits all vertices (exactly once). For $e \in \mathcal{E}$, let $\mathrm{cr}(P, e)$ be the number of edges of P with one endpoint in e and the other in $V \setminus e$. The *crossing number* of P with respect to H is $\sup\{\mathrm{cr}(P, e) \colon e \in \mathcal{E}\}$.

The *dual hypergraph* of H is $H^* = (\mathcal{E}, \mathcal{E}^*)$, where $\mathcal{E}^*$ consists of all hyperedges $v^* = \{e \in \mathcal{E} \colon v \in e\}$ for $v \in V$. Its shatter function is the *dual shatter function* of H, and is denoted by π^*_H.

If $\dim H = d$ then $\dim H^* \le 2^{d+1}$ [7], and hence $\pi^*_H(m) \le C_d m^{2^{d+1}}$ for every positive m, where C_d is a constant depending on d. In particular, any hypergraph with finite VC-dimension satisfies the hypotheses of the following theorem.

▶ **Theorem 8** ([37, Lemma 3.3 and Theorem 4.2])**.** *Let H be a hypergraph on n vertices such that $\pi^*_H(m) \le Cm^d$ for some constants $C > 0$ and $d > 1$. Then there exists another constant C_1 (depending on C and d) and a spanning cycle for H with crossing number $\le C_1 n^{1-\frac{1}{d}}$.*

(An additional $\log n$ factor in Welzl's original result was later removed [25, Sec. 5.4]. Up to constant factors, this theorem is equivalent to the same result for paths or trees.)

▶ **Definition 9.** *Let $H = (V, \mathcal{E})$ be a finite hypergraph with $\pi^*_H(m) \le Cm^d$. Let P be a spanning cycle for H whose crossing number is minimal (and thus $\le C_1|V|^{1-\frac{1}{d}}$). Fix an arbitrary starting point $v_0 \in P$ and orientation of P. For $0 \le i < |V|$, let $v_i \in V$ be the i-th vertex along P. Let $V^t_{lc} = \{\{v_{kt}, v_{kt+1}, \ldots, v_{kt+t-1}\} \colon 0 \le k < \lfloor\frac{|V|}{t}\rfloor\} \subsetneq \binom{V}{t}$ (where the subscript lc stands for low crossing). Observe that its elements are pairwise disjoint. Let H^t_{lc} be the hypergraph on V^t_{lc} whose hyperedges are of the form $\{v \in V^t_{lc} \colon v \subseteq e\}$ for each $e \in \mathcal{E}$.*

▶ Remark 10. In order to make H^t_{lc} uniquely defined, P is chosen arbitrarily from all suitable spanning cycles. As H^t_{lc} is a subhypergraph of H^t, $\dim H^t_{lc} \le \dim H^t$, and thus we also have $\dim H^t_{lc} \le \gamma_t \dim H$.

3 Existence of small ϵ-t-nets

▶ **Theorem 2.** *For every $\epsilon \in (0,1)$ and $t \in \mathbb{N} \setminus \{0\}$, every hypergraph on $\ge C_1\left(\frac{t-1}{\epsilon}\right)^{d^*}$ vertices with VC-dimension d and dual shatter function $\pi^*_H(m) \le Cm^{d^*}$ admits an ϵ-t-net of size $O(\frac{d(1+\log t)}{\epsilon}\log\frac{1}{\epsilon})$, all elements of which are pairwise disjoint. Here $C_1 = C_1(d^*, C)$.*

Proof. For $t = 1$, this is simply the ϵ-net theorem. For higher t, let $H = (V, \mathcal{E})$ be such a hypergraph and $n = |V|$. Consider the hypergraph H^t_{lc} defined in Section 2. It has $\lfloor \frac{n}{t} \rfloor$ vertices and VC-dimension $\leq \gamma_t d$ (by Remark 10), and thus admits an $\frac{\epsilon}{2}$-net of size $O(\frac{\gamma_t d}{\epsilon} \log \frac{1}{\epsilon})$. We claim that any such $\frac{\epsilon}{2}$-net $N \subseteq \binom{V}{t}$ is also an ϵ-t-net for H.

Indeed, the crossing number of the associated spanning cycle is $O_{C,d^*}(n^{1-1/d^*})$. Every hyperedge e of H with $|e| \geq \epsilon n$ fully contains at least $\lfloor \frac{\epsilon n}{t} \rfloor - O_{C,d^*}(n^{1-1/d^*})$ elements of V^t_{lc}, which is $\geq \frac{\epsilon n}{2t}$ as soon as $n = \Omega_{C,d^*}(\frac{t}{\epsilon} n^{1-1/d^*})$, or equivalenty (noting also that $2(t-1) \geq t$ for $t \geq 2$) when $n = \Omega_{C,d^*}(\frac{t-1}{\epsilon}^{d^*})$. One of these t-subsets is in N. ◀

▶ Remark 11. In general, some fast growth of $n = |V|$ as a function of $\frac{1}{\epsilon}$ is necessary. For example, given any ϵ such that $\frac{t}{\epsilon} \in \mathbb{N}$, the complete t-uniform hypergraph on $\frac{t}{\epsilon}$ vertices does not have any ϵ-t-net with fewer than $\binom{t/\epsilon}{t}$ elements. Moreover, there exist geometrically-defined hypergraphs that do not admit ϵ-2-nets of size $o(\frac{1}{\epsilon^2})$ (see the full version of the paper [3]). On the other hand, in Section 5 we show that certain classes of geometrically-defined hypergraphs have "small" ϵ-t-nets even for "small" values of n.

Small ϵ-nets, small ϵ-2-nets

A natural question arising from Theorem 2 is whether any hypergraph that admits small ϵ-nets must also admit ϵ-t-nets of approximately same size. In general, the answer is negative. Take for example a hypergraph whose smallest ϵ-net is of size $\Omega(\frac{1}{\epsilon} \log \frac{1}{\epsilon})$ (see [22], [29]), and augment it by adding a vertex that belongs to all hyperedges. Clearly, this second hypergraph has the same VC-dimension and a one-element ϵ-net, but any ϵ-2-net is of size $\Omega(\frac{1}{\epsilon} \log \frac{1}{\epsilon})$.

However, this example is quite artificial. In "natural" scenarios (and for sufficiently large vertex sets) the smallest ϵ-nets and ϵ-2-nets might still have approximately same size. In Section 5 we show that this is the case for some geometrically-defined hypergraphs.

Another scenario in which there exist both an ϵ-net and an ϵ-2-net of size $O(\frac{1}{\epsilon})$ is when the VC-dimension of the hypergraph is 1. In this case, the existence of an ϵ-net of size $O(\frac{1}{\epsilon})$ was proved in [22]. The next theorem could be derived from results on Mnets [16], at the cost of poor multiplicative constants. Here we give a simpler proof for it.

▶ **Theorem 3.** *For every positive integer t and $\epsilon \leq \frac{1}{2}$, every finite hypergraph on $\geq t\lceil \frac{1}{\epsilon} \rceil$ vertices with VC-dimension 1 admits an ϵ-t-net of size at most $t\lceil \frac{1}{\epsilon} \rceil + 1$.*

Proof. Let $(V, \mathcal{E})$ be such a hypergraph and $n = |V|$. Without loss of generality, $\min\{|e| : e \in \mathcal{E}\} \geq \epsilon n$. For $1 \leq i \leq t$, there exists an ϵ-net N_i that hits each $e \in \mathcal{E}$ at least i times, and $|N_i| = i\lceil \frac{1}{\epsilon} \rceil$. To see this let N_1 be an ϵ-net of size $\lceil \frac{1}{\epsilon} \rceil$ [22]. In the hypergraph induced on $V \setminus N_i$ the hyperedges hit only i times by N_i have cardinality $\geq \epsilon n - i$, while the number of vertices is $n - i\lceil \frac{1}{\epsilon} \rceil$, for a ratio $\frac{\epsilon n - i}{n - i\lceil \frac{1}{\epsilon} \rceil} \geq \epsilon$. Take an ϵ-net N of size $\lceil \frac{1}{\epsilon} \rceil$ for this hypergraph and let $N_{i+1} = N_i \cup N$. Finally, let the desired ϵ-t-net consist of one t-subset from each element of $\Pi_H(N_t)$ with $\geq t$ vertices, of which there are at most $|N_t| + 1$ by Lemma 4. ◀

4 Deterministic construction of ϵ-t-nets

Let $H = (V, \mathcal{E})$ be a finite hypergraph with VC-dimension d, and fix $\epsilon \in (0, 1)$. In this section we provide an explicit polynomial-time construction of ϵ-nets that immediately implies an explicit construction of ϵ-t-nets. The size is far from optimal, but the construction is simpler than previous explicit constructions, as it does not rely on packing numbers nor on pseudo-random choices.

4.1 Deterministic construction of ϵ-nets

We start with the following definition:

▶ **Definition 12.** *Let A, B be two subsets of V. We say that A* stabs *B if for every hyperedge $S \in \mathcal{E}$ with $B \subseteq S$ we have $S \cap A \neq \emptyset$.*

Let $S \in \mathcal{E}$ be a hyperedge, $|S| \geq d+1$, and let $X \in \binom{S}{d+1}$. Since the VC-dimension is d the set X is not shattered. Notice that $X = X \cap S \in \Pi_H(X)$. We can also assume that $\emptyset \in \Pi_H(X)$, for otherwise X is a transversal for H of size $d+1$. Hence there exists at least one non-trivial, proper subset $A \subsetneq X$ such that $(X \setminus A) \notin \Pi_H(X)$. Equivalently, there is a non-trivial partition of X into A and $X \setminus A$ such that A stabs $X \setminus A$. We say that X is of type $|A| \in \{1, \ldots, d\}$. Note that X could have several types. By the pigeonhole principle, there is a type i and a subset $A \in \binom{S}{i}$ such that a fraction $d^{-1}\binom{|S|}{i}^{-1}$ of the elements of $\binom{S}{d+1}$ are stabbed by A, hence the following lemma holds:

▶ **Lemma 13.** *Let S be a hyperedge containing $\geq d+1$ vertices of V. Then there exists an integer $i \in \{1, \ldots, d\}$ and a subset $A \in \binom{S}{i}$ that stabs $\binom{|S|}{d+1} d^{-1}\binom{|S|}{i}^{-1}$ subsets of cardinality $d+1-i$.*

Constructing ϵ-nets

Put $n := |V|$. We construct an ϵ-net of size $O_d(\frac{1}{\epsilon^d})$ as follows. Start with $N = \emptyset$. As long as there is a hyperedge $S \in \mathcal{E}$ with $|S| \geq \epsilon n$ and $S \cap N = \emptyset$, Lemma 13 asserts that some i-subset from S stabs $\Omega_d((\epsilon n)^{d+1-i})$ subsets of S with cardinality $d+1-i$ for an appropriate $i \in \{1, \ldots, d\}$. Add all elements of this subset to N; we call this a type i iteration.

The resulting set is an ϵ-net by construction. It is left to show that $|N| = O_d(\frac{1}{\epsilon^d})$. As each step of the construction adds at most d vertices to N it is enough to bound the number of iterations T. By the pigeonhole principle, at least $\frac{T}{d}$ of the iterations have the same type, say i. After a type-i iteration N stabs an additional $\Omega_d((\epsilon n)^{d+1-i})$ subsets of cardinality $d+1-i$ none of which were previously stabbed. Since there are $\binom{n}{d+1-i}$ subsets of cardinality $d+1-i$ we have $\frac{T}{d} = O_d(\binom{n}{d+1-i}(\epsilon n)^{-(d+1-i)}) = O_d(\frac{1}{\epsilon^d})$.

Complexity analysis

We analyze the running time of the above algorithm. We assume that for the algorithm we have a data structure which is the incidence matrix of the hypergraph H. Without loss of generality, each hyperedge of $\mathcal{E}$ may be replaced with a subset of cardinality $\lceil \epsilon n \rceil$. This can be done in time $O(\epsilon n^{d+1})$ due to the fact that $|\mathcal{E}| = O(n^d)$.

We consider each $X \in \binom{V}{d+1}$. Firstly we check if there is a hyperedge $S \in \mathcal{E}$ which contains X, if not, we continue to the next subset. If yes, we consider each of the $2^{d+1} - 2$ proper subsets of X. Let $A \subset X$ be such a subset. We check if $X \setminus A$ is stabbed by A. We can do it by going over all $O(n^d)$ hyperedges of H. Hence, in total this pre-processing step takes $O(n^{d+1} \cdot 2^{d+1} \cdot n^d) = O_d(n^{2d+1})$ running time. While determining the type of any $(d+1)$-subset of X and scanning all the hyperedges of the hypergraph, we maintain for any i-subset $A \subset X$ $(1 \leq i \leq d)$, a list of all the $(d+1-i)$-subsets of X that A stabs and their number.

Consider some iteration of the algorithm and let $S \in \mathcal{E}$ be such that $|S| \geq \epsilon n$ and $S \cap N = \emptyset$ where N is the collection of elements found until this iteration. We find a subset $A \subset S$ of size at most d which stabs the most subsets of size $(d+1) - |A|$.

The running time of each iteration is $O(|S|^d \cdot n^d) = O(\epsilon^d n^{2d})$. Hence in total the running time of the algorithm after the pre-processing step is $O_d(\frac{\epsilon^d n^{2d}}{\epsilon^d}) = O_d(n^{2d})$. Hence the total running of the algorithm described in the previous section is $O_d(n^{2d})$.

Immediate applications to ϵ-t-nets

The construction of ϵ-nets in Section 4.1 gives two straightforward constructions of ϵ-t-nets.

1. *Trivial construction.* Use the above algorithm to explicitly construct t disjoint ϵ-nets of size $O_d(1/\epsilon^d)$, and take all t-subsets of elements in their union that contain one element from each net. The resulting ϵ-t-net is of size $O_d(1/\epsilon^{td})$.
2. *Construction via H^t_{lc}.* Use the above algorithm to explicitly construct an $\frac{\epsilon}{2}$-net for the hypergraph H^t_{lc}, which is an ϵ-t-net for H (as was shown in the proof of Theorem 2). The resulting ϵ-t-net is of size $O_{d,t}(1/\epsilon^{\dim H^t_{lc}})$. (The cycle with a low crossing number required for constructing the hypergraph H^t_{lc} can be found in polynomial time [37, 25]).

4.2 Deterministic construction of ϵ-t-nets

We present a direct construction of ϵ-t-nets without passing through ϵ-nets. For the sake of convenience, we present the method for $t = 2$, and extend it in the full version [3] for $t > 2$.

The following definition extends the classical notion of VC-dimension.

▶ **Definition 14.** *Let t be a positive integer. Also let $H = (V, \mathcal{E})$ be a hypergraph, and T', T such that $T' \subseteq T \subseteq V$. We say that T' is* t-realized *by H (with respect to T) if $T' \cup S \in \Pi_H(T)$ for some $S \subseteq T$ such that $|S| < t$. We say that T is* t-shattered *by H if every $T' \subseteq T$ is t-realized by H (with respect to T). The t-VC-dimension of H, denoted by $\dim_t H$, is the maximal size of a vertex set that is t-shattered by H.*

Note that the 1-VC-dimension is the standard VC-dimension. Moreover, the t-VC-dimension is at most the $(t+1)$-VC-dimension for any psitive integer t. We use the following adaptation of Definition 12:

▶ **Definition 15.** *Let $H = (V, \mathcal{E})$ be a hypergraph. Given two vertex sets $A, B \subseteq V$, we say that A* 2-stabs *B if each hyperedge of $\mathcal{E}$ that contains B also contains at least two vertices from A.*

▶ **Theorem 16.** *For a hypergraph $H = (V, \mathcal{E})$ with 2-VC-dimension d, one can construct explicitly an ϵ-2-net of size $O_d(1/\epsilon^{d-1})$.*

Proof. Let $S \in \mathcal{E}$ be a hyperedge and let $X \in \binom{S}{d+1}$. Since the 2-VC-dimension is d the set X is not 2-shattered. Notice that $X = X \cap S$ and so X and all elements of $\binom{X}{d}$ are 2-realized by H with respect to X. For our purpose, we can also assume that $\emptyset$ is 2-realized by H (with respect to X), for otherwise $\binom{X}{2}$ is a transversal for H of size $\binom{d+1}{2}$. This means that there is a partition, say $X = A \cup (X \setminus A)$, such that A 2-stabs $X \setminus A$. Let $i = |A|$. Note that $i \in \{2, \dots, d\}$. We say that $X = A \cup (X \setminus A)$ is a type i partition. We need the following lemma, whose proof is similar to that of Lemma 13.

▶ **Lemma 17.** *Let S be a hyperedge containing $\geq d+1$ vertices of V. Then there exists an integer $i \in \{2, \dots, d\}$ and a subset $A \subset S$ with cardinality i that 2-stabs $\frac{\binom{|S|}{d+1}}{(d-1)\binom{|S|}{i}}$ subsets B of cardinality $d+1-i$.*

Constructing ϵ-2-nets

Let $H = (V, \mathcal{E})$ be as above and let $\epsilon > 0$ be fixed. Put $n = |V|$. We construct an ϵ-2-net of size $O_d(\frac{1}{\epsilon^{d-1}})$ as follows. We start with a set $N = \emptyset$. As long as there is a hyperedge $S \in \mathcal{E}$ with $|S| \geq \epsilon n$ that does not contain any pair $\{v, w\} \in N$, for an appropriate $i \in \{2, \ldots, d\}$ we take an i-subset $A \subset S$ 2-stabbing $\Omega_d((\epsilon n)^{d+1-i})$ subsets of S with cardinality $d+1-i$, and add to N all $\binom{i}{2}$ elements of A. We call this a type i iteration. This is possible by Lemma 17.

The resulting set is an ϵ-2-net by construction. It is left to show that $|N| = O_d(\frac{1}{\epsilon^{d-1}})$. In each step of the construction we add at most $\binom{d}{2}$ pairs to N so it is enough to bound the number of iterations T. By the pigeonhole principle, at least $\frac{T}{d-1}$ of the iterations have the same type, say i. There are $\binom{n}{d+1-i}$ subsets of cardinality $d+1-i$, and in each of the at least $\frac{T}{d-1}$ type i iterations we 2-stab at least $\Omega_d((\epsilon n)^{d+1-i})$ additional subsets of cardinality $d+1-i$, so we have $\frac{T}{d-1} = O_d(\frac{\binom{n}{d+1-i}}{(\epsilon n)^{d+1-i}}) = O_d(\frac{1}{\epsilon^{d+1-i}})$ so $t = O_d(\frac{1}{\epsilon^{d-1}})$ (since $i \geq 2$). This completes the proof of Theorem 16. ◀

Complexity analysis

The only significant difference between the constructions of Section 4.1 and of Section 4.2 is the factor that depends on the size of the resulting net. Hence, the complexity of the algorithm in this section is bounded by $O_d(n^{2d})$, where d is the 2-VC-dimension of H.

4.3 t-VC-dimension versus classical VC-dimension

What can be said about the relation between VC-dimension and our newly introduced t-VC-dimension, for $t \geq 2$? By definition, $\dim H \leq \dim_2 H$. Ideas from Dudley's unpublished lecture notes [15, Th. 4.37] (see also the full version [3]) yield $\dim_2 H \leq 2 \dim H + 1$. This is sharp for some small hypergraphs, such as that with vertex set $\{a, b, c\}$ and hyperedges $\{a\}$, $\{b, c\}$, $\{a, c\}$, and $\{a, b, c\}$, which has VC-dimension 1 but 2-VC-dimension 3. For general t, we conjecture that $\dim_t H \leq 2 \dim H + 2t - 1$. The reasoning below gives roughly $\dim_t H \leq 9.09 \max\{\dim H, t-1\}$.

Let H be a hypergraph of finite VC-dimension with a largest t-shattered subset of vertices T. As T is t-shattered, we have $2^T = \{e \setminus S \colon e \in \Pi_H(T),\ S \subseteq T,\ |S| < t\}$. This yields

$$2^{\dim_t H} \leq |\Pi_H(T)| \cdot \sum_{i=0}^{t-1} \binom{\dim_t H}{i} \leq \sum_{i=0}^{\dim H} \binom{\dim_t H}{i} \cdot \sum_{i=0}^{t-1} \binom{\dim_t H}{i},$$

with the last inequality following from Lemma 4. When $\dim_t H \geq 2 \max\{t-1, \dim H\}$, applying Equation (1) gives

$$1 \leq h\left(\frac{\dim H}{\dim_t H}\right) + h\left(\frac{t-1}{\dim_t H}\right).$$

From this inequality we obtain:

▶ **Proposition 18.** *For $t \in \mathbb{N} \setminus \{0\}$, the t-VC-dimension of a hypergraph of VC-dimension d is at least d, at most $2\gamma_2 \max\{d, t-1\}$ (where $\gamma_2 \simeq 4.54$), and, as $d \to \infty$, at most $2d + o(d)$.*

An interesting geometric example is the hypergraph H whose vertex set is a finite subset of $\mathbb{R}^{d-1}$ and whose hyperedges are induced by half-spaces. It is well-known that $\dim H = d$.

More generally, we have $\dim_t H \leq td$ for all t. Indeed, by Tverberg's theorem (see, e.g., [26]), every set T of $td+1$ points in $\mathbb{R}^{d-1}$ admits a partition into $t+1$ pairwise disjoint and non-empty sets $T = X \cup Y_1 \cup \cdots \cup Y_t$ such that the intersection of their convex hulls is non-empty. No half-space can t-realize X since any half-space that contains X must contain at least one point from each Y_i, that is, at least t points of $T \setminus X$.

Therefore, for this hypergraph and $t = 2$, the direct construction yields an ϵ-2-net of size $O_d(1/\epsilon^{2d-1})$, while the trivial construction (described at the end of Section 4.1) yields only a weaker upper bound of $O_d(1/\epsilon^{2d})$. With good bounds on $\dim H^2_{lc}$, the construction via H^2_{lc} (see again Section 4.1) might provide even smaller ϵ-2-nets. In the plane (namely, where $d = 3$), it follows from [17] that $\dim H^2_{lc} \leq 5$, and so the upper bounds obtained using the direct construction and using H^2_{lc} are the same – $O(1/\epsilon^5)$.

5 Geometric ϵ-2-nets

For a fixed $\epsilon > 0$, any hypergraph with VC-dimension d and $n \geq \frac{C_d}{\epsilon^{2^{d+1}}}$ vertices admits, by Theorem 2, an ϵ-2-net of size $O(\frac{d}{\epsilon} \log \frac{1}{\epsilon})$. This leaves open two interesting questions:

1. In cases where the hypergraph admits an ϵ-net of small size, say $O(\frac{1}{\epsilon})$, does it also admit an $O(\frac{1}{\epsilon})$-sized ϵ-2-net (or, more generally, ϵ-t-nets)?
2. Does this extend to smaller values of n?

In this section we answer both in the affirmative for several classes of geometrically-defined hypergraphs.

▶ **Definition 19.** *Given two families B and R of sets, the intersection hypergraph $H(B, R)$ is the hypergraph on vertex set B, where any $r \in R$ defines a hyperedge $\{b \in B : b \cap r \neq \emptyset\}$.*

Note that $H(B, R)$ and $H(R, B)$ are (in general) not isomorphic but dual to each other. Intersection hypergraphs are ubiquitous in discrete and computational geometry. Particular attention is given to the case where either B or R is a set of points, with $H(B, R)$ respectively known as a *primal hypergraph* defined by R or a *dual hypergraph* defined by B. See the survey [28] and the references therein.

We present below and in the full version of the paper [3] several intersection hypergraphs that admit $O(\frac{1}{\epsilon})$-sized ϵ-nets, and prove that each of them has ϵ-2-nets of the same size. Furthermore, while Theorem 2 applies only to hypergraphs with a very large number of vertices, the geometric hypergraphs discussed do not have to contain "many" vertices in order to guarantee the existence of "small" ϵ-2-nets. In some cases (see, e.g., the full version of the paper [3]), the behavior is sharp: we can point out two constants $c_1 < c_2$ s.t. if the number of vertices satisfies $|V| \geq \frac{c_2}{\epsilon}$ the hypergraph admits an $O(\frac{1}{\epsilon})$-sized ϵ-2-net, while for $|V| \leq \frac{c_1}{\epsilon}$, there exist hypergraphs from the same family that admit only ϵ-2-nets of size $\Omega(\frac{1}{\epsilon^2})$.

5.1 Non-piercing regions

For our first example we consider a large class of geometric objects introduced by Raman and Ray [32]. A *family of non-piercing regions* is a family of regions of $\mathbb{R}^2$ such that for any two regions γ_1 and γ_2 the difference $\gamma_1 \setminus \gamma_2$ is connected. (Each region may contain holes. See [32] for the exact definitions.)

This extends the more familiar notion of pseudo-disks.

▶ **Theorem 20.** *The intersection hypergraph of two families B and R of non-piercing regions with B finite admits an ϵ-net of size $O(\frac{1}{\epsilon})$ and, if $\epsilon|B| \geq 2$, an ϵ-2-net of size $O(\frac{1}{\epsilon})$.*

The proof relies on several intermediary results. The first one is about an analogue of the Delaunay graph for non-piercing regions [32]. The important specific case where the regions are pseudo-disks had already been studied [4, 20, 21].

▶ **Definition 21.** *A* planar support *for the hypergraph $(V, \mathcal{E})$ is a planar graph G on the same vertex set V such that any hyperedge in $\mathcal{E}$ induces a connected subgraph of G.*

▶ **Theorem 22** ([32])**.** *Given two families B and R of non-piercing regions, B finite, their intersection hypergraph $H(B, R)$ admits a planar support.*

The following corollary has already been noted for families of pseudo-discs [4].

▶ **Corollary 23.** *Given two families B and R of non-piercing regions,* $\dim H(B, R) \leq 4$.

Proof. Let $B' \subseteq B$ be a shattered subset of vertices in $H(B, R)$. As the non-piercing property is clearly hereditary, the hypergraph $H(B', R)$ also admits a planar support. For every pair of vertices in B' there exists a hyperedge of $H(B', R)$ that contains these two vertices and no other. Following Definition 21 these two vertices must share an edge in any planar support of $H(B', R)$. Thus said *planar* support is a complete graph on B', forcing $|B'| \leq 4$. ◀

Proof of Theorem 20. First we observe that $H(B, R)$ has ϵ-nets of size $O(\frac{1}{\epsilon})$. Since $H(B, R)$ is finite, we may assume that R is finite as well. To paraphrase from Pyrga and Ray [30, Theorem 4], the following properties suffice:

- For any $0 < \epsilon < 1$ and any $B' \subseteq B$, $H(B', R)$ admits an ϵ-net whose size depends only on ϵ.
- There exist constants $\alpha > 0$, $\beta \geq 0$ and $\tau > 0$ s.t. for any $R' \subseteq R$ there is a graph $G_{R'} = (R', E_{R'})$ with $|E_{R'}| \leq \beta |R'|$ so that for any element $b \in B$ we have $m_b \geq \alpha n_b - \tau$, where n_b is the number of regions of R' intersecting b and m_b is the number of edges in $E_{R'}$ whose both endpoints (which are regions of R') intersect b.

The first condition is verified because $\dim H(B', R) \leq 4$ for every B'. For the second one, let $\alpha = \tau = 1$ and $\beta = 3$, and let $G_{R'}$ be a planar support of $H(R', B)$. (Note the use of duality!) The inequalities follow from its planarity and the connectedness of the subgraph "cut out" by each $b \in B$.

Finally, to obtain an ϵ-2-net, let $K_1 \subseteq B$ be an ϵ-net for $H(B, R)$ of size $O(\frac{1}{\epsilon})$. Let R' consist of the regions of R, if any, that intersect $\geq \epsilon |B|$ regions of B but only one of K_1, and let K_2 be an $\frac{\epsilon}{2}$-net for $H(B \setminus K_1, R')$ also of size $O(\frac{1}{\epsilon})$. Then the desired ϵ-2-net consists of all edges in a planar support of $H(K_1 \cup K_2, R)$. ◀

5.2 Small union complexity

Next, we prove the existence of a small ϵ-2-net for the intersection hypergraph of regions in the plane with linear union complexity and points (i.e. the dual hypergraph defined by the regions).

The union complexity of a family of objects is the function $\kappa : \mathbb{N} \to \mathbb{N}$ that sends each $n \in \mathbb{N}$ to the number of faces of all dimensions in the boundary of the union of $\leq n$ objects, maximized over all subsets of $\leq n$ objects. If $\kappa(n) = O(n)$, we say that the family has *linear union complexity*. Families with linear union complexity include, e.g., families of pseudo-discs: the boundary of the union of $n \geq 3$ pseudo-discs consists of at most $6n - 12$ arcs and as many vertices [19].

The $(\leq k)$-level complexity of the family is defined by counting all faces included in at most k objects (not just on the boundary). To make these definitions precise, one needs to define faces and their dimension; see the survey by Agarwal, Pach and Sharir [1].

A specific case of the following result could also be derived from previous results on Mnets [16], if one adds the additional assumption that the regions have bounded "semi-algebraic description complexity". (The proof of [16] is involved and uses algebraic arguments).

▶ **Theorem 24.** *Let L be a finite family of regions in $\mathbb{R}^2$ with linear union complexity and let $P \subseteq \mathbb{R}^2$ be a set of points. If $|L| \geq \frac{2}{\epsilon}$ then $H(L,P)$ admits an ϵ-2-net of size $O(\frac{1}{\epsilon})$.*

Proof. Let $n := |L|$. First, construct a set $K \subseteq L$ of size $O(\frac{1}{\epsilon})$ such that every "heavy" point of P is included in at least two elements of K, as in the proofs of Theorem 3 or Theorem 20. This relies on the existence of ϵ-nets of size $O(\frac{1}{\epsilon})$ for $H(L,P)$, a result of Aronov, Ezra and Sharir [5].

Since linear union complexity is a hereditary property, K as a subset of L also has linear union complexity. By a standard argument using the Clarkson–Shor theorem [14], the (≤ 2)-level complexity of K is linear as well. Hence, by Euler's formula, the number of hyperedges of size 2 in $H(K,P)$ (whose order of magnitude is equal to the number of (≤ 2)-level faces in the arrangement of K) is at most $c|K|$ for some constant c. By the pigeonhole principle, some region $d \in K$ participates in at most c such hyperedges (i.e., pairs of regions). We pick these at most c pairs of regions to be elements of the ϵ-2-net we construct, and repeat the process for $K \setminus \{d\}$.

We continue in this fashion until all elements of K are removed, and set the ϵ-2-net N to be the set of pairs we picked. Clearly, $|N| = O(|K|) = O(\frac{1}{\epsilon})$. To see that N is indeed an ϵ-2-net, let p be a point that belongs to at least ϵn regions of L. By construction, p belongs to at least two regions of K. Consider the process in which the elements of K are gradually removed, until none of them are left. As a single region is removed at every step, we can look at the step in which the number of remaining regions that contain p is reduced from 2 to 1. Since at that step p is included in exactly two regions of the arrangement, the corresponding pair of regions is added to the ϵ-2-net. Hence, p is covered by both elements of a pair in the ϵ-2-net, as asserted. This completes the proof. ◀

▶ Remark 25. By essentially the same argument, the hypergraph $H(L,P)$ admits an ϵ-t-net of size $O_t(\frac{1}{\epsilon})$ for any constant $t \leq \epsilon|L|$.

We can extend Theorem 24 to a family L with union complexity $\kappa(n) = n \cdot f(n)$. In this case, the size of the ϵ-2-net is $O(\frac{1}{\epsilon} \cdot \log f(\frac{1}{\epsilon}) \cdot f(\frac{1}{\epsilon} \cdot \log f(\frac{1}{\epsilon})))$. For example, if $\kappa(n) = n \log n$, then one obtains an ϵ-2-net of size $O(\frac{1}{\epsilon} \cdot \log \frac{1}{\epsilon} \cdot \log\log \frac{1}{\epsilon})$.

Indeed, by [5], the hypergraph $H(L,P)$ admits an ϵ-net of size $O(\frac{1}{\epsilon} \cdot \log f(\frac{1}{\epsilon}))$. Let $n' = \frac{1}{\epsilon} \cdot \log f(\frac{1}{\epsilon})$. By the Clarkson–Shor theorem [14], the (≤ 2)-level complexity is bounded by $O(n' \cdot f(n'))$, hence there exists a region that participates in at most $f(n')$ hyperedges of order 2. This means that the size of the obtained ϵ-2-net is bounded by $O(n' \cdot f(n'))$.

6 Discussion and open problems

A hypergraph H with finite VC-dimension d has ϵ-2-nets of size $O(\frac{d}{\epsilon} \log \frac{1}{\epsilon})$ when n is very large as a function of $\frac{1}{\epsilon}$. This upper bound is the best possible in general, and as we saw in Section 3 may also be best possible even if H admits smaller ϵ-nets. However, we conjecture that in any "reasonable" setting, (including, e.g., all the geometric scenarios discussed in Section 5, and all hypergraphs with hereditarily small ϵ-nets), the existence of an ϵ-net of some order of magnitude, implies the existence of an ϵ-2-net of roughly the same order of magnitude.

Furthermore, we are not aware of any hypergraph in which the dependence of n in $\frac{1}{\epsilon}$ has to be as large as in the assumption of Theorem 2. It may be interesting to extend our results to smaller values of n (as a function of $\frac{1}{\epsilon}$), and to understand whether (as in some of the geometric cases discussed above), there exists a sharp threshold (as a function of $\frac{1}{\epsilon}$) such that if n is above this threshold, then the hypergraph admits an ϵ-2-net of size $\tilde{O}(\frac{1}{\epsilon})$, but if n is below it, then any ϵ-2-net for the hypergraph contains at least $\Omega(\frac{1}{\epsilon^2})$ pairs.

References

1 Pankaj K. Agarwal, János Pach, and Micha Sharir. State of the union (of geometric objects). In *Proc. Joint Summer Research Conference on Discrete and Computational Geometry: 20 Years Later, Contemporary Mathematics 452, AMS*, pages 9–48, 2008.

2 Noga Alon, Graham Brightwell, H.A. Kierstead, A.V. Kostochka, and Peter Winkler. Dominating sets in k-majority tournaments. *Journal of Combinatorial Theory, Series B*, 96(3):374–387, 2006. doi:10.1016/j.jctb.2005.09.003.

3 Noga Alon, Bruno Jartoux, Chaya Keller, Shakhar Smorodinsky, and Yelena Yuditsky. The Epsilon-t-Net Problem, 2020. arXiv:2003.07061.

4 Boris Aronov, Anirudh Donakonda, Esther Ezra, and Rom Pinchasi. On pseudo-disk hypergraphs, 2018. arXiv:1802.08799.

5 Boris Aronov, Esther Ezra, and Micha Sharir. Small-size ϵ-nets for axis-parallel rectangles and boxes. *SIAM Journal on Computing*, 39(7):3248–3282, 2010. doi:10.1137/090762968.

6 Sunil Arya, Guilherme Dias da Fonseca, and David M. Mount. Polytope approximation and the Mahler volume. In Yuval Rabani, editor, *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 29–42. SIAM, 2012. doi:10.1137/1.9781611973099.3.

7 Patrick Assouad. Densité et dimension. *Annales de l'Institut Fourier*, 33(3):233–282, 1983. doi:10.5802/aif.938.

8 Amos Beimel. Secret-sharing schemes: A survey. In Yeow Meng Chee, Zhenbo Guo, San Ling, Fengjing Shao, Yuansheng Tang, Huaxiong Wang, and Chaoping Xing, editors, *Coding and Cryptology - Third International Workshop, IWCC 2011, Qingdao, China, May 30-June 3, 2011. Proceedings*, volume 6639 of *Lecture Notes in Computer Science*, pages 11–46. Springer, 2011. doi:10.1007/978-3-642-20901-7_2.

9 Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Learnability and the Vapnik–Chervonenkis dimension. *J. ACM*, 36(4):929–965, 1989. doi:10.1145/76359.76371.

10 Hervé Brönnimann, Bernard Chazelle, and Jiří Matoušek. Product range spaces, sensitive sampling, and derandomization. *SIAM Journal on Computing*, 28(5):1552–1575, 1999. doi:10.1137/S0097539796260321.

11 Chris Calabro. *The Exponential Complexity of Satisfiability Problems*. Phd thesis, University of California, San Diego, 2009. URL: https://escholarship.org/uc/item/0pk5w64k.

12 Timothy M. Chan. Improved deterministic algorithms for linear programming in low dimensions. *ACM Trans. Algorithms*, 14(3):30:1–30:10, June 2018. doi:10.1145/3155312.

13 Bernard Chazelle and Jiří Matoušek. On linear-time deterministic algorithms for optimization problems in fixed dimension. *Journal of Algorithms*, 21(3):579–597, 1996. doi:10.1006/jagm.1996.0060.

14 Kenneth L. Clarkson and Peter W. Shor. Application of random sampling in computational geometry, II. *Discrete & Computational Geometry*, 4:387–421, 1989. doi:10.1007/BF02187740.

15 Richard M. Dudley. Notes on empirical processes. Lecture notes, second printing, 2000.

16 Kunal Dutta, Arijit Ghosh, Bruno Jartoux, and Nabil H. Mustafa. Shallow packings, semialgebraic set systems, Macbeath regions, and polynomial partitioning. *Discrete & Computational Geometry*, 61(4):756–777, 2019. doi:10.1007/s00454-019-00075-0.

17 Nicolas Grelier, Saeed Gh. Ilchi, Tillmann Miltzow, and Shakhar Smorodinsky. On the VC-dimension of convex sets and half-spaces, 2019. arXiv:1907.01241.

18 David Haussler and Emo Welzl. Epsilon-nets and simplex range queries. *Discrete & Computational Geometry*, 2:127–151, 1987. doi:10.1007/BF02187876.

19 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*, 1:59–71, 1986. doi:10.1007/BF02187683.

20 Chaya Keller and Shakhar Smorodinsky. Conflict-free coloring of intersection graphs of geometric objects. *Discrete & Computational Geometry*, June 2019. doi:10.1007/s00454-019-00097-8.

21 Balázs Keszegh. Coloring intersection hypergraphs of pseudo-disks. *Discrete & Computational Geometry*, October 2019. doi:10.1007/s00454-019-00142-6.

22 János Komlós, János Pach, and Gerhard Woeginger. Almost tight bounds for ϵ-nets. *Discrete & Computational Geometry*, 7(2):163–173, February 1992. doi:10.1007/BF02187833.

23 Chung L. Liu. *Introduction to Combinatorial Mathematics*. McGraw-Hill, New York, 1968.

24 Jiří Matoušek. Approximations and optimal geometric divide-and-conquer. *Journal of Computer and System Sciences*, 50(2):203–208, 1995. doi:10.1006/jcss.1995.1018.

25 Jiří Matoušek. *Geometric Discrepancy: An Illustrated Guide*. Number 18 in Algorithms and Combinatorics. Springer, Berlin, New York, 1999. doi:10.1007/978-3-642-03942-3.

26 Jiří Matoušek. *Lectures on Discrete Geometry*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2002. doi:10.1007/978-1-4613-0039-7.

27 Nabil H. Mustafa and Saurabh Ray. ϵ-Mnets: Hitting geometric set systems with subsets. *Discrete & Computational Geometry*, 57(3):625–640, 2017. doi:10.1007/s00454-016-9845-8.

28 Nabil H. Mustafa and Kasturi Varadarajan. Epsilon-approximations and epsilon-nets. In Jacob E. Goodman, Joseph O'Rourke, and Csaba D. Tóth, editors, *Handbook of Discrete and Computational Geometry, 3rd Edition*, pages 1241–1267. CRC Press, 2018.

29 János Pach and Gábor Tardos. Tight lower bounds for the size of epsilon-nets. *J. Amer. Math. Soc.*, 26(3):645–658, 2013. doi:10.1090/S0894-0347-2012-00759-0.

30 Evangelia Pyrga and Saurabh Ray. New existence proofs for ϵ-nets. In *Proceedings of the Twenty-fourth Annual Symposium on Computational Geometry*, SCG '08, pages 199–207, New York, NY, USA, 2008. ACM. doi:10.1145/1377676.1377708.

31 Yuval Rabani and Amir Shpilka. Explicit construction of a small ϵ-net for linear threshold functions. *SIAM Journal on Computing*, 39(8):3501–3520, 2010. doi:10.1137/090764190.

32 Rajiv Raman and Saurabh Ray. Planar support for non-piercing regions and applications. In Yossi Azar, Hannah Bast, and Grzegorz Herman, editors, *26th Annual European Symposium on Algorithms, ESA 2018, August 20-22, 2018, Helsinki, Finland*, volume 112 of *LIPIcs*, pages 69:1–69:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.ESA.2018.69.

33 Norbert Sauer. On the density of families of sets. *Journal of Combinatorial Theory, Series A*, 13(1):145–147, 1972. doi:10.1016/0097-3165(72)90019-2.

34 Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979. doi:10.1145/359168.359176.

35 Saharon Shelah. A combinatorial problem; stability and order for models and theories in infinitary languages. *Pacific J. Math.*, 41(1):247–261, 1972. URL: https://projecteuclid.org:443/euclid.pjm/1102968432.

36 Vladimir N. Vapnik and Alexei 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.

37 Emo Welzl. Partition trees for triangle counting and other range searching problems. In *Proceedings of the Fourth Annual Symposium on Computational Geometry (Urbana, IL, 1988)*, pages 23–33. ACM, New York, 1988. doi:10.1145/73393.73397.

Terrain Visibility Graphs: Persistence Is Not Enough

Safwa Ameer
Department of Computer Science, The University of Texas at San Antonio, TX, USA
safwa.ameer@gmail.com

Matt Gibson-Lopez
Department of Computer Science, The University of Texas at San Antonio, TX, USA
matthew.gibson@utsa.edu

Erik Krohn
Department of Computer Science, The University of Wisconsin, Oshkosh, WI, USA
krohne@uwosh.edu

Sean Soderman
Department of Computer Science, The University of Texas at San Antonio, TX, USA
sean.soderman@my.utsa.edu

Qing Wang
Department of Computer Science, The University of Tennessee at Martin, TN, USA
qwang44@utm.edu

Abstract

In this paper, we consider the Visibility Graph Recognition and Reconstruction problems in the context of terrains. Here, we are given a graph G with labeled vertices $v_0, v_1, \ldots, v_{n-1}$ such that the labeling corresponds with a Hamiltonian path H. G also may contain other edges. We are interested in determining if there is a terrain T with vertices $p_0, p_1, \ldots, p_{n-1}$ such that G is the visibility graph of T and the boundary of T corresponds with H. G is said to be persistent if and only if it satisfies the so-called X-property and Bar-property. It is known that every "pseudo-terrain" has a persistent visibility graph and that every persistent graph is the visibility graph for some pseudo-terrain. The connection is not as clear for (geometric) terrains. It is known that the visibility graph of any terrain T is persistent, but it has been unclear whether every persistent graph G has a terrain T such that G is the visibility graph of T. There actually have been several papers that claim this to be the case (although no formal proof has ever been published), and recent works made steps towards building a terrain reconstruction algorithm for any persistent graph. In this paper, we show that there exists a persistent graph G that is not the visibility graph for any terrain T. This means persistence is not enough by itself to characterize the visibility graphs of terrains, and implies that pseudo-terrains are not stretchable.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Terrains, Visibility Graph Characterization, Visibility Graph Recognition

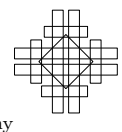
Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.6

Related Version A full version of the paper is available at `https://arxiv.org/abs/3112749`.

Funding *Matt Gibson-Lopez*: Supported by the National Science Foundation under Grant No. 1733874.
Sean Soderman: Supported by the National Science Foundation under Grant No. 1733874.

1 Introduction

The notion of geometric visibility plays a fundamental role in many applications such as robotics [8, 17] and shortest path computation in the presence of obstacles [16]. One of the most fundamental data structures in visibility is the *visibility graph* (VG). Let P be a simple



polygon in the plane with n vertices labeled $p_0, \ldots, p_{n-1}$ following the boundary of P in "counter-clockwise" order. P partitions the plane into two sets: "inside P" and "outside P". We say two vertices p_i and p_j see each other if and only if the line segment $\overline{p_i p_j}$ does not intersect the "outside P" region. The VG G of P has a vertex v_i for each point of p_i, and $\{v_i, v_j\}$ is an edge in G if and only if p_i and p_j see each other in P.

Given a simple polygon P, computing its VG in polynomial-time is a fairly trivial matter; however, if we are given a graph G, determining if it is the VG for some simple polygon has remained a tantalizing open problem for over 30 years. Along these lines, there are three main VG problems that have received much attention: 1) characterization, 2) recognition, and 3) reconstruction. In the *visibility graph characterization* problem, we seek to define a set of necessary and sufficient conditions that all VGs must satisfy. In the *visibility graph recognition* problem, we seek to design an algorithm that, given a graph G, determines if there is a simple polygon P such that G is the VG of P. In the *visibility graph reconstruction* problem, we are given a VG G and we wish to reconstruct a simple polygon P such that G is the VG of P.

1.1 Previous work

The history of simple polygon VG characterization dates back to 1988, when Ghosh gave three necessary conditions (NCs) that any VG must satisfy [12]. Shortly after, Everett and Corneil [11, 10] showed a counterexample to the sufficiency of NCs 1-3; that is, they gave an example of a graph that satisfies NCs 1-3 but is not the VG of any simple polygon. Everett [11] also showed that a NC might need to be strengthened to rule this example out. Srinivsraghavan and Mukhopadhyay [20] showed that a strengthening of this NC was in fact necessary, but a counterexample given by Abello, Lin, and Pisupati [5] showed that more NCs would be needed to complete the characterization. In 1997, Ghosh [13] gave a fourth NC that circumvents the latest counterexample, but in 2005 Streinu gave an example of a graph that satisfies the four NCs but is not a VG for any simple polygon [21].

Unfortunately, it is not known if simple polygon VG recognition is in NP. Even for special cases, characterization and recognition results have only been given in the extremely restricted special cases of simple polygons such as "spiral" polygons [10] and "tower polygons" [7].

1.2 Pseudo-visibility

Faced with the complexity of understanding simple polygon VGs, O'Rourke and Streinu [18] turned their attention to *pseudo-polygons*, a generalization of simple polygons where visibility is determined by a set of curves in the plane called pseudo-lines. An arrangement of *pseudo-lines* $\mathcal{L}$ is a collection of simple curves, each of which separates the plane, such that each pair of pseudo-lines in $\mathcal{L}$ intersects at exactly one point, where they cross. Given a set of n points in the plane and a set of pseudo-lines $\mathcal{L}$ such that every pair of points has a pseudo-line that contains them, a *pseudo-polygon* is determined similarly to a standard Euclidean simple polygon except that visibility is defined using $\mathcal{L}$ instead of straight line segments. Note that every simple polygon is a pseudo-polygon, where $\mathcal{L}$ is a set of straight line segments. Streinu showed that there are pseudo-polygons that cannot be *stretched* into a simple polygon [21]. That is, there is a pseudo-polygon such that its VG is not the VG for any simple polygon.

In 1997, O'Rourke and Streinu [18] gave a characterization of *vertex-edge* VGs of pseudo-polygons. In this setting, for any vertex v we are told which *edges* v sees rather than which vertices it sees. Unfortunately this does not extend to the desired characterization of regular

VGs, as O'Rourke and Streinu showed that vertex-edge VGs encode more information about a pseudo-polygon than a regular VG [19]. More recently, Gibson, Krohn, and Wang gave the desired characterization of the VGs of pseudo polygons [14] which has very recently been extended to a polynomial-time recognition and reconstruction algorithm [6].

1.3 The visibility graphs of terrains

One geometric structure that has gathered a lot of attention in the computational geometry community is the terrain. A *terrain* T is an x-monotone (a vertical line intersects it at most once) polygonal chain in the plane. Let T be a terrain with points labeled $p_0, \ldots, p_{n-1}$ from left to right. Let p_i^x denote the x-coordinate of the point p_i on T. Note that due to monotonicity, we have $p_i^x < p_{i+1}^x$ for each $i \in \{0, \ldots, n-2\}$. We say points p_i and p_j see each other if and only if the open line segment $\overline{p_i p_j}$ lies completely above T. Given this definition of vision, one can define the VG of a terrain similarly to that of a simple polygon.

Abello et al. [4] studied so-called "convex fans" which is essentially a simple polygon P that can be decomposed into a terrain T and one additional point p^* such that p^* sees every point of T (the boundary of P uses the boundary of T as well as the line segments $\overline{p^* p_0}$ and $\overline{p^* p_{n-1}}$). They show that every simple polygon can be decomposed into some number of convex fans, and therefore a potential strategy of tackling the simple polygon problem is to take such a decomposition and handle the fans individually. Since p^* sees every point of the convex fan, the complexity in understanding the convex fan lies almost entirely with the analysis of the "terrain portion" of the convex fan.

1.4 Persistent graphs

With a goal towards understanding the visibility graphs of convex fans, Abello et al. [3] defined a notion of so-called persistent graphs and established a connection with terrain visibility graphs and persistent graphs, which we will now describe. Suppose we are given a graph G with labeled vertices $v_0, v_1, \ldots v_{n-1}$ such that $\{v_i, v_{i+1}\}$ is an edge for each $i \in \{0, 1, \ldots, n-2\}$ (i.e., the labeling gives a Hamiltonian path). Let H denote this Hamiltonian Path. G also may contain other edges. We are interested in determining if there is a terrain T with points $p_0, p_1, \ldots, p_{n-1}$ such that G is the visibility graph of T and the boundary of T corresponds with H.

G is said to be *persistent* if and only if it satisfies the following two properties:

- **X-property:** for any set of four distinct integers $a, b, c, d \in \{0, \ldots, n-1\}$ such that $a < b < c < d$, if $\{v_a, v_c\}$ and $\{v_b, v_d\}$ are edges in G then $\{v_a, v_d\}$ is also an edge in G.
- **Bar-property**: for every edge $\{v_i, v_k\}$ in G such that $k \geq i + 2$, there exists a $j \in (i, k)$ such that $\{v_i, v_j\}$ and $\{v_j, v_k\}$ are edges in G.

Abello et al. [3] showed that for any terrain T, its visibility graph is persistent (albeit for a slightly different definition of persistence), and Evans and Saeedi [9] showed it for the definition of persistence being used here.

We now will help develop intuition for these properties (see [9] for a formal proof). For the X-property (sometimes referred to as the "order claim"), consider Figure 1. In part (a), we have a terrain such that: (1) p_0 sees p_3, and (2) p_1 sees p_4 (the blue dotted lines). Therefore no vertex between p_0 and p_4 is strictly above either of the blue dotted lines. Then the line segment connecting p_0 and p_4 will be "above" the blue dotted lines and therefore p_0 must see p_4. So now consider the graph in part (b). If the edges $\{v_0, v_3\}$ and $\{v_1, v_4\}$ are in the graph but $\{v_0, v_4\}$ is not an edge in the graph then it cannot be the visibility graph of a terrain.

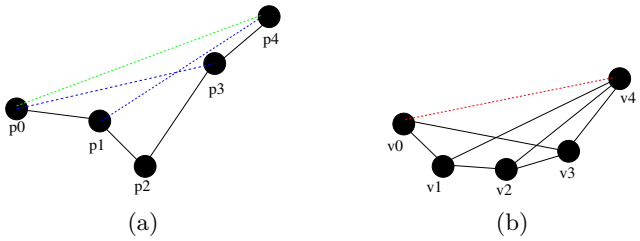


Figure 1 An illustration of the X-property.

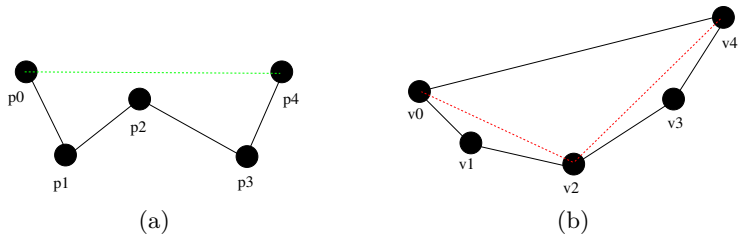


Figure 2 An illustration of the Bar-Property.

For the Bar-property, see Figure 2. In the terrain in part (a), we have that p_0 sees p_4. p_1 sees p_0, but it doesn't see p_4 because it is blocked by p_2. Then it must be that p_2 also sees p_0. Since p_2 also sees p_4 so we are done. In general, if p_i sees p_k, then p_{i+1} must see p_i. If p_{i+1} also sees p_k we are done, so suppose it doesn't see p_k because there is some point p_j for $j \in \{i+2, \ldots, k-1\}$ such that p_{i+1} sees p_j, and p_j is over the line segment $\overline{p_{i+1}, p_k}$. p_j must see p_i, and if it sees p_k we are done. Otherwise we repeat this argument with the point that blocks p_j from p_k, and eventually we find a point that sees both p_i and p_k. Therefore if the graph in part (b) only contains the black edges, it cannot be the visibility graph of a terrain, as the graph implies that p_0 should see p_4 but no other point between them should see both p_0 and p_4.

Abello et al. [3] showed a one-to-one correspondence between the VGs of pseudo-terrains (terrains using pseudo-lines to define visibilities rather than straight line segments) and persistent graphs. That is, they show that the VG of any pseudo-terrain is persistent, and they show that any persistent graph has a pseudo-terrain and give a polynomial-time algorithm to reconstruct it. Evans and Saeedi [9] give a simpler proof (and a faster reconstruction algorithm) of the same result.

It has remained an open problem to show that persistent graphs and the visibility graphs of (geometric) terrains are exactly the same set (i.e., to show that G is a persistent graph if and only if there is a terrain T such that G is the visibility graph of T). Several papers have made progress towards giving a reconstruction algorithm that can take a persistent graph G as input and construct a terrain T such that G is the visibility graph of T. In fact, there are papers [4, 1] that claim that there exists such a reconstruction algorithm although a formal proof of this has not been published. Evans and Saeedi [9] state that they ideally would like to reconstruct a terrain from a persistent graph but that it seems difficult. Most of the previous attempts to reconstruct terrains from a persistent graph involves an iterative placement of the points of the terrain (e.g., determining the x and y coordinates of the points of the terrain from left to right).

1.5 Our contribution

The main result of this paper is to prove that these two classes of graphs are in fact *not* the same.

▶ **Theorem 1.** *There is a persistent graph G such that there is no terrain T such that G is the visibility graph of T.*

We obtain this result by introducing a new style of reconstruction algorithm. We show that if one can compute a set of feasible x-coordinates for the points of the terrain, then the y-coordinates can be computed via linear programming (LP). Using standard LP analysis techniques, we identify a seven-vertex, persistent graph G' that must have its x-coordinates chosen carefully in order to be able to reconstruct a terrain with G' as its visibility graph. We then build a graph G^* that has thirty-five vertices which can be partitioned into five "copies" of G'. In order to represent G^* as a terrain, we would need to pick the thirty-five x-coordinates in a way where each "copy" of G' has its condition satisfied, and we show that this is not possible.

Since G^* is persistent, it is the visibility graph of some pseudo-terrain, and therefore our result also is a proof that pseudo-terrains are not stretchable.

1.5.1 Organization of the paper

In Section 2, we describe our LP-based reconstruction algorithm. In Section 3, we give our graph G' and show that it requires very specifically chosen x-coordinates in order to be realizable as a terrain. This critically uses our new LP-based reconstruction approach. In Section 4, we give our persistent graph G^* and prove that there is no terrain that has it as its visibility graph. We give a conclusion and some open problems in Section 5.

2 Reconstructing terrains via linear programming

Let G be a persistent graph with vertices $v_0, \dots v_{n-1}$. For any terrain T with points $p_0, \dots, p_{n-1}$, we let p_i^x denote the x-coordinate of p_i. Let $X = (x_0, x_1, \dots x_{n-1})$ be a vector of real numbers such that $x_i < x_{i+1}$ for each $i \in \{0, 1, \dots n-2\}$, and let $\mathcal{T}(G, X)$ be the set of all terrains T with n points such that:

1. $p_0^x = x_0, p_1^x = x_1, \dots p_{n-1}^x = x_{n-1}$ (i.e., it is the set of all terrains that have x-coordinates that correspond with X).
2. The boundary of T corresponds to the Hamiltonian path H.
3. G is the visibility graph of T.

For any two integers $i, j \in \{0, \dots, n-1\}$ such that $i < j$, let $d_{i,j} := |x_i - x_j|$. Intuitively, for a terrain $T \in \mathcal{T}(G, X)$, $d_{i,j}$ is the distance between the x-coordinates of p_i and p_j.

We will now show that given G and X, we can determine in polynomial-time if there is a terrain in $\mathcal{T}(G, X)$, and moreover if $\mathcal{T}(G, X) \neq \emptyset$ then we can compute in polynomial-time a feasible set of y-coordinates for some terrain $T \in \mathcal{T}(G, X)$. This algorithm is via a reduction to linear programming (LP) where the variables of the LP are the y-coordinates of the points of the terrain T. We show that given a fixed set of x-coordinates, we can model all of the visibility constraints that T must satisfy as inequalities that are linear in the y-coordinates of the points of T. It is not immediately obvious blocking constraints can be modeled as linear constraints (i.e, if $\{v_i, v_j\}$ is not an edge in G, ensuring that the y-coordinates are computed so that the points p_i and p_j do not see each other in T), but we will show that we can in fact model this as a linear constraint.

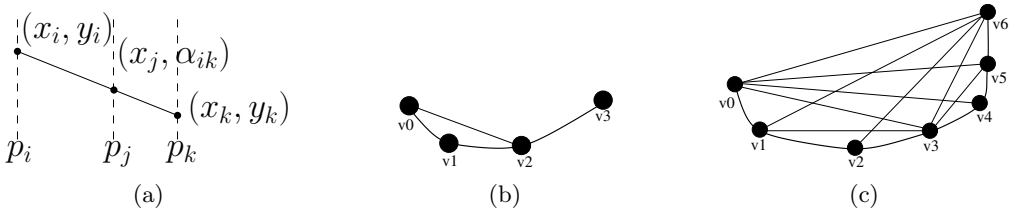


Figure 3 (a) LP constraint illustration. (b) A sample terrain VG. (c) The VG G'.

First let us consider a visibility constraint: let $\{v_i, v_k\}$ be an edge in G. We must ensure that the y-coordinates y_i and y_k for p_i and p_k respectively are such that the line segment $\overline{p_i p_k}$ "stays above" T. We can ensure this, by enforcing that for every $j \in (i, k)$, we choose the y-coordinate y_j such that p_j is underneath $\overline{p_i p_k}$. Let α_{ik}^j denote the y-coordinate of the intersection of $\overline{p_i p_k}$ and the vertical line $x = x_j$ (as illustrated in Figure 3 (a)). It is easy to see that $\alpha_{ik}^j = \frac{d_{j,k} \cdot y_i + d_{i,j} \cdot y_k}{d_{i,k}}$, a linear function of y_i and y_k since $d_{i,j}, d_{j,k}$, and $d_{i,k}$ are functions of the constant x-coordinates. Therefore the *visibility constraint* $y_j < \alpha_{ik}^j$ is a linear inequality. In our LP, we will write the constraint as $d_{j,k} \cdot y_i - d_{i,k} \cdot y_j + d_{i,j} \cdot y_k \geq \epsilon$ where ϵ is a positive constant. Note that $\{v_i, v_k\}$ can have many constraints in the LP associated with it (although some of them may be redundant and can be removed without affecting the set of feasible solutions to the LP, more on this later).

Now suppose v_i and v_k are such that $\{v_i, v_k\}$ is not an edge in G. Then we must enforce that the corresponding points p_i and p_k do not see each other in T. This means that $\overline{p_i p_k}$ must cross under the terrain T. We can do this by enforcing that *some* point p_j between p_i and p_k has its y-coordinate chosen to be large enough so that it is above $\overline{p_i p_k}$. Unfortunately the notion that *some* point must be over $\overline{p_i p_k}$ cannot directly be represented as a linear constraint (whereas in the previous case it had to be that *every* point must be under $\overline{p_i p_k}$). However we can see that by employing an analysis similar to the so-called *designated blocker* from the analysis of pseudo-polygon visibility graphs [14], we can identify a specific point (or two) that *must* be above $\overline{p_i p_k}$ which allows us to express the constraint as a linear inequality. To find the first such point, start at v_k and "walk to the left" along H towards v_i and let v_j be the first vertex encountered such that $\{v_i, v_j\}$ is an edge in G (note that such a vertex must exist; $\{v_i, v_{i+1}\}$ is an edge in G). We claim that for every $T \in \mathcal{T}(G, X)$, it must be that p_j is over $\overline{p_i p_k}$. Suppose for the sake of contradiction that p_j is under $\overline{p_i p_k}$. If there is a point p_z over $\overline{p_i p_k}$ such that $z < j$, then p_i doesn't see p_j, a contradiction, so suppose there is no such point over $\overline{p_i p_k}$. So now let p_z be the first point to the right of p_j that is over $\overline{p_i p_k}$. Then it must be that p_i sees p_z, but $\{v_i, v_z\}$ is not an edge in G by definition of p_j, a contradiction. So it is true that for every $T \in \mathcal{T}(G, X)$, it must be that p_j is over $\overline{p_i p_k}$, and we call p_j the designated blocker to block p_i from p_k. Therefore we can add the blocking constraint $y_j > \alpha_{ik}^j$ to our LP. We write this constraint $-d_{j,k} \cdot y_i + d_{i,k} \cdot y_j - d_{i,j} \cdot y_k \geq \epsilon$ where ϵ is a positive constant. We symmetrically compute the designated blocker to block p_k from seeing p_i. Note that this point $p_{j'}$ may not be the same point as the first designated blocker p_j (but it must be that $j \leq j'$ or else G violates the X-property and therefore is not persistent). If $j' \neq j$, then we add another blocking constraint for $p_{j'}$. We again remark that sometimes these blocking constraints are redundant and can be removed without altering the set of feasible solutions to the LP.

The choice of ϵ does not effect whether or not there is a feasible solution to the LP (as long as ϵ is positive). If there is a solution vector $\mathbf{y}$ that is feasible with right hand side ϵ', then one can obtain a feasible solution with right hand side ϵ by scaling $\mathbf{y}$ by a factor of $\frac{\epsilon}{\epsilon'}$.

To illustrate our approach, consider the example VG in Figure 3 (b). We will show how we construct the LP in order to reconstruct a terrain that has this graph as its VG. Suppose $X = (0, 1, 2, 3)$. First note that since $\{v_0, v_2\}$ is an edge, we need the visibility constraint $y_1 \leq \alpha^1_{0,2} = \frac{y_0+y_2}{2}$, which we can write as $y_0 - 2y_1 + y_2 \geq 1$. Secondly note that p_0 and p_3 do not see each other and p_2 is the designated blocker. Therefore we add the blocking constraint $y_2 > \alpha^2_{0,3} = \frac{y_0+2y_3}{3}$, which we state as $-y_0 + 3y_2 - 2y_3 \geq 1$. Note p_1 does not see p_3 and has designated blocker p_2, but this constraint is redundant with the other two constraints. Therefore our final LP is the following: $y_0 - 2y_1 + y_2 \geq 1; -y_0 + 3y_2 - 2y_3 \geq 1$. Any feasible solution to this LP will give y-coordinates for a terrain T such that G is the VG of T.

One of the advantages of the LP-based approach is we can use standard LP techniques to help us determine what (if any) constraints on x-coordinates need to be satisfied in order to reconstruct the terrain (or determine that no x-coordinates are possible). In particular, we will be using the well-known *Farkas' Lemma*. Let m denote the number of constraints in our LP, and let n be the number of points in the terrain. The LP can be represented as $\mathbf{Ay} \geq \mathbf{b}$, where $\mathbf{A}$ is an $m \times n$ matrix of coefficients, $\mathbf{y} \in \mathbb{R}^n$ is the vector of y-coordinate variables of the LP, and $\mathbf{b} = \{\epsilon\}^m$ for some $\epsilon > 0$. Then Farkas' Lemma [15] says that *exactly one* of the following two statements is true:

1. there exists a $\mathbf{y}$ satisfying $\mathbf{Ay} \geq \mathbf{b}$ (i.e., there exists a terrain in $\mathcal{T}(G, X)$)
2. there is a $\mathbf{z} \in \mathbb{R}^m$ such that $\mathbf{z} \leq 0$, $\mathbf{A}^{\mathrm{T}}\mathbf{z} \geq 0$ and $\mathbf{b}^{\mathrm{T}}\mathbf{z} < 0$.

Our result heavily relies on the use of Case 2 of Farkas' Lemma to determine exactly which X vectors create a non-empty $\mathcal{T}(G, X)$ for a given persistent graph G.

3 A picky persistent graph

In this section, we will prove one of the key lemmas that leads to our result: there is a persistent graph that requires its x-coordinates to satisfy a strict inequality in order for there to be a feasible solution to the LP. The same visibility graph was analyzed in [2] where they showed that this graph cannot be represented with "uniform step lengths" (which in our context means that for any $c > 0$ we have $d_{i,i+1} = c$). While this graph has been observed in previous works, what is new in this paper is the exact requirements that the x-coordinates must satisfy in order for there to be a terrain.

Let G' be the visibility graph in Figure 3 (c). A terrain that has G' as its visibility graph is shown in Figure 4. Consider the LP using the following constraints: (1) p_1 should be above $\overline{p_0p_2}$, (2) p_3 should be under $\overline{p_0p_4}$, (3) p_3 should be over $\overline{p_1p_5}$, (4) p_3 should be under $\overline{p_2p_6}$, (5) p_5 should be under $\overline{p_3p_6}$, and (6) p_5 should be over $\overline{p_4p_6}$. Note that there are other constraints we aren't explicitly stating here such as p_3 being under $\overline{p_0p_5}$ (we will show they are redundant and adding them does not affect the feasible region of the LP; removing the redundant constraints will simplify the later analyses). Here the number of constraints $m = 6$ and the number of points $n = 7$. We express this LP in the form $\mathbf{Ay} \geq \mathbf{b}$ where $\mathbf{A}$, $\mathbf{y}$, and $\mathbf{b}$ are as follows:

$$\mathbf{A} = \begin{vmatrix} -d_{1,2} & d_{0,2} & -d_{0,1} & 0 & 0 & 0 & 0 \\ d_{3,4} & 0 & 0 & -d_{0,4} & d_{0,3} & 0 & 0 \\ 0 & -d_{3,5} & 0 & d_{1,5} & 0 & -d_{1,3} & 0 \\ 0 & 0 & d_{3,6} & -d_{2,6} & 0 & 0 & d_{2,3} \\ 0 & 0 & 0 & d_{5,6} & 0 & -d_{3,6} & d_{3,5} \\ 0 & 0 & 0 & 0 & -d_{5,6} & d_{4,6} & -d_{4,5} \end{vmatrix} \quad \mathbf{y} = \begin{vmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \end{vmatrix} \quad \mathbf{b} = \begin{vmatrix} \epsilon \\ \epsilon \\ \epsilon \\ \epsilon \\ \epsilon \\ \epsilon \end{vmatrix}$$

Again, ϵ is a positive constant. Let $T(X, \mathbf{y})$ denote the n-point terrain whose x-coordinates correspond with X and y-coordinates correspond with $\mathbf{y}$. Clearly if T is a terrain in $\mathcal{T}(G', X)$ then the vector of y-coordinates of its points is a feasible solution to this LP. We will now argue that if $\mathbf{y}$ is a feasible solution to this LP then $T(X, \mathbf{y}) \in \mathcal{T}(G', X)$.

▶ **Lemma 2.** *Let $\boldsymbol{y}$ be a feasible solution to the LP. Then the visibility graph of $T(X, \boldsymbol{y})$ is G'.*

Proof. The combination of constraints 5 (p_5 should be under $\overline{p_3p_6}$) and 6 (p_5 should be over $\overline{p_4p_6}$) directly implies that the visibilities of p_i and p_j correctly match those given by G' for v_i and v_j for each pair when $i, j \geq 3$. In particular, p_4 must be under $\overline{p_3p_5}$ and $\overline{p_3p_6}$.

Now consider point p_0. Constraint 2 (p_3 should be under $\overline{p_0p_4}$) implies that p_0 will see p_3, p_4, p_5, and p_6 as long as p_1 and p_2 do not block them. Constraint 3 (p_3 should be over $\overline{p_1p_5}$) ensures that p_1 will be under $\overline{p_0p_3}$ and then Constraint 1 (p_1 should be above $\overline{p_0p_2}$) implies p_2 is under $\overline{p_0p_3}$ and $\overline{p_1p_3}$. Therefore p_0 will correctly see p_3, p_4, p_5, and p_6. Moreover, Constraint 1 directly implies that p_0 will not see p_2, and therefore all visibilities corresponding to p_0 are correct.

The fact that p_2 is under $\overline{p_1p_3}$ implies that p_1 and p_3 correctly see each other. Constraint 4 (p_3 should be under $\overline{p_2p_6}$) implies that p_2 will correctly see p_6 given the earlier configurations of p_4 and p_5. So using the fact that the visibility graph of any terrain satisfies the X-property, we can see that p_1 correctly sees p_6 (applying the X-property with $a = 1, b = 2, c = 3$, and $d = 6$).

Finally we need that p_i does not see p_j for $i \in \{1, 2\}$ and $j \in \{4, 5\}$. p_1 does not see p_5 as directly implied by Constraint 3, and we already showed the following: p_2 is under $\overline{p_1p_3}$, p_4 is under $\overline{p_3, p_5}$. This implies the remaining three pairs of points correctly do not see each other. ◀

The following lemma uses Farkas' Lemma to determine requirements on X (which in turn determines $\mathbf{A}$) in order to have $\mathcal{T}(G', X) \neq \emptyset$.

▶ **Lemma 3.** *There is a terrain $T \in \mathcal{T}(G', X)$ if and only if X satisfies $d_{0,1}d_{2,3}d_{3,4}d_{5,6} > d_{1,2}d_{4,5}d_{0,3}d_{3,6}$.*

Proof. Suppose that X satisfies $d_{0,1}d_{2,3}d_{3,4}d_{5,6} > d_{1,2}d_{4,5}d_{0,3}d_{3,6}$. Let ϵ (which appears in $\mathbf{b}$) be the minimum of $d_{3,5}(d_{0,1}d_{2,3}d_{3,4}d_{5,6} - d_{1,2}d_{4,5}d_{0,3}d_{3,6})$ and $d_{3,5}(d_{0,1}d_{3,4}d_{5,6}d_{3,5} + d_{3,4}d_{5,6}d_{0,2}d_{3,6} + d_{1,2}d_{5,6}d_{3,5}d_{3,6} + d_{3,4}d_{5,6}d_{3,5}d_{3,6} + d_{1,2}d_{0,3}d_{3,6}d_{3,5})$. Note that ϵ is strictly positive given our assumption on X. We show that the following vector $\mathbf{y}$ is a feasible solution to the LP (work shown in the full version):

$$\mathbf{y} = \begin{vmatrix} d_{0,1}d_{2,3}d_{3,5}d_{5,6} + d_{0,3}d_{3,5}d_{0,1}d_{2,3} + d_{0,3}d_{3,5}d_{0,1}d_{4,5} + d_{4,5}d_{0,3}d_{0,2}d_{3,6} + d_{0,3}d_{3,5}d_{4,5}d_{3,6} \\ d_{1,2}d_{4,5}d_{0,3}d_{3,6} - d_{0,1}d_{2,3}d_{3,4}d_{5,6} \\ -d_{2,3}d_{5,6}(d_{3,4}d_{0,2} + d_{3,5}(d_{1,2} + d_{3,4})) - d_{1,2}d_{0,3}d_{3,5}(d_{2,3} + d_{4,5}) \\ 0 \\ -d_{0,1}d_{3,4}d_{3,5}(d_{2,3} + d_{4,5}) - d_{4,5}d_{3,6}(d_{3,4}(d_{0,2} + d_{3,5}) + d_{1,2}d_{3,5}) \\ 0 \\ d_{0,1}d_{3,4}d_{5,6}d_{3,5} + d_{3,4}d_{5,6}d_{0,2}d_{3,6} + d_{1,2}d_{5,6}d_{3,5}d_{3,6} + d_{3,4}d_{5,6}d_{3,5}d_{3,6} + d_{1,2}d_{0,3}d_{3,6}d_{3,5} \end{vmatrix}$$

Now suppose X is such that $d_{0,1}d_{2,3}d_{3,4}d_{5,6} \leq d_{1,2}d_{4,5}d_{0,3}d_{3,6}$. We will show that $\mathcal{T}(G', X) = \emptyset$ by using Farkas' Lemma. In particular, we show that there is a vector $\mathbf{z} \in \mathbb{R}^m$ such that $\mathbf{z} \leq 0$, $\mathbf{A}^{\mathrm{T}}\mathbf{z} \geq 0$, and $\mathbf{b}^{\mathrm{T}}\mathbf{z} < 0$ for every $\epsilon > 0$. Our vector $\mathbf{z}$ is as follows:

$$\mathbf{z} = \begin{vmatrix} \frac{-d_{3,4}d_{5,6}}{d_{1,2}d_{0,3}} \\ \\ \frac{-d_{5,6}}{d_{0,3}} \\ \\ \frac{-d_{3,4}d_{5,6}d_{0,2}}{d_{2,1}d_{0,3}d_{3,5}} \\ \\ \frac{-d_{0,1}d_{3,4}d_{5,6}}{d_{1,2}d_{0,3}d_{3,6}} \\ \\ \frac{d_{0,1}d_{2,3}d_{3,4}d_{5,6} - d_{1,2}d_{4,5}d_{0,3}d_{3,6}}{d_{1,2}d_{0,3}d_{3,6}d_{3,5}} \\ \\ -1 \end{vmatrix}$$

Note that the next-to-last entry is at most 0 due to the assumption on X, and the rest are strictly negative for all X. Therefore it immediately follows that $\mathbf{z} \leq 0$ and $\mathbf{b}^{\mathrm{T}}\mathbf{z} < 0$ for every $\epsilon > 0$. We complete the proof by showing that $\mathbf{A}^{\mathrm{T}}\mathbf{z}$ is a zero vector (work shown in the full version). ◀

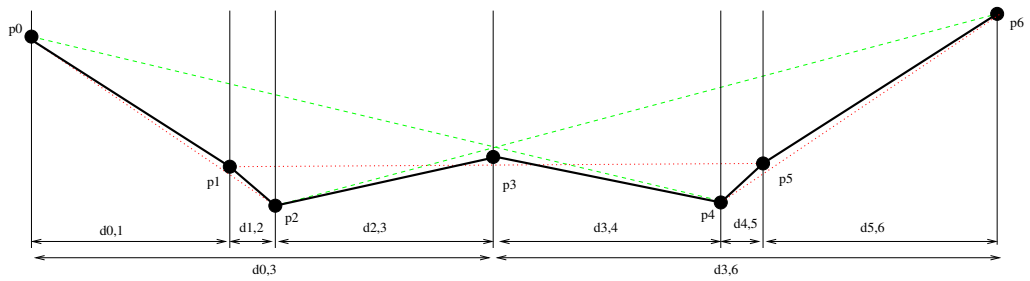


Figure 4 A terrain whose VG is G'.

We remark that Lemma 3 can illustrate the difficulty in designing an algorithm that reconstructs the terrain from left to right, placing the points of the terrain one at a time. Let G'' be the subgraph of G' induced by the first six vertices $\{v_0, \ldots, v_5\}$. It is not hard to see that G'' can be reconstructed using any vector of six, increasing x-coordinates. Suppose we take such a reconstruction and then try to extend the reconstruction to handle all of G'. If we reconstructed G'' using, say, $x_i = i$ for each $i \in \{0, \ldots, 5\}$ (implying that $d_{i,i+1} = 1$ for each $i \in \{0, \ldots, 4\}$), one can see that *every* choice of x_6 such that $x_6 > x_5$ will violate the inequality stated in Lemma 3 (note that the choice of x_6 impacts the $d_{5,6}$ term on the left side and impacts the $d_{3,6}$ term on the right side). This implies that a left-to-right style approach may need to shift both the x-coordinates and y-coordinates of the previously-placed points to accommodate the new point.

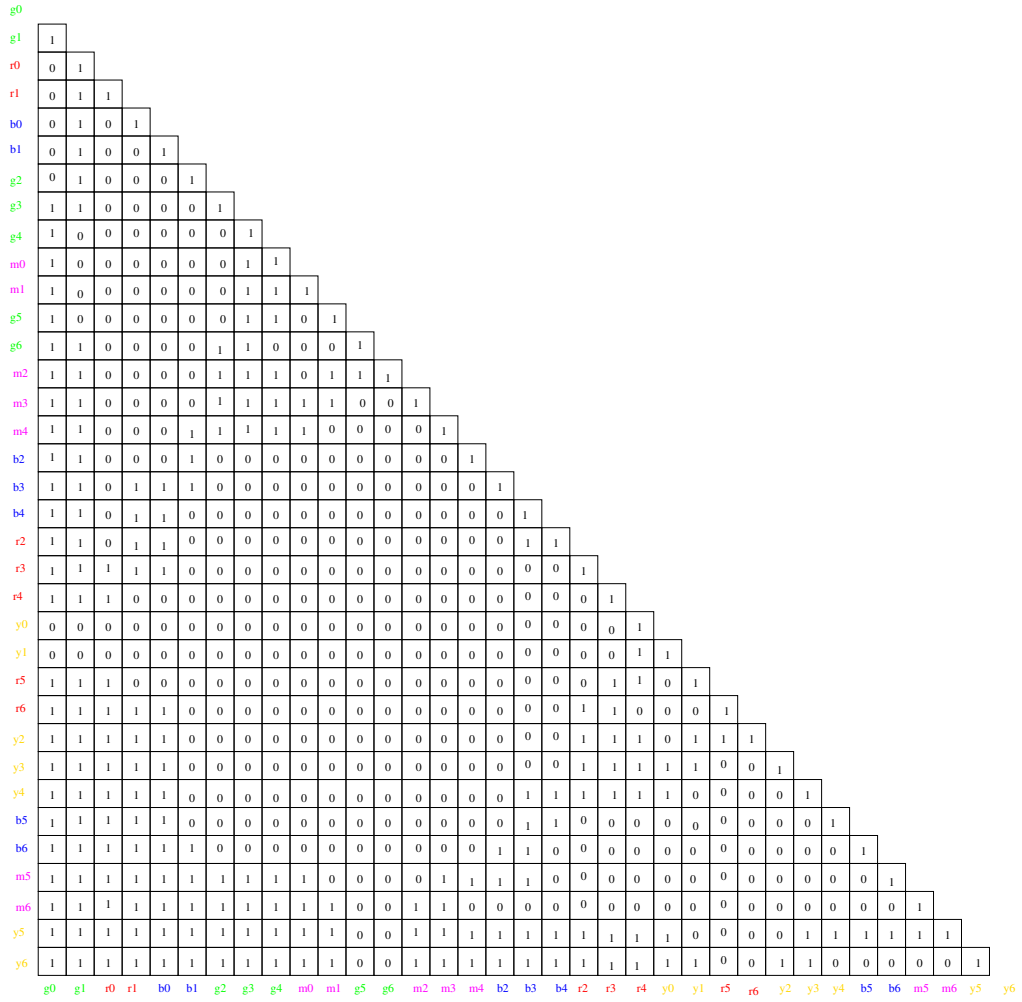


Figure 5 The adjacency matrix of G^*, a persistent graph that is not a terrain visibility graph.

4 A persistent graph that is not a terrain visibility graph

We are now ready to prove our main result of the paper, that there is a persistent graph G^* such that there is no terrain T such that G^* is the visibility graph of T. The adjacency matrix of G^* is given in Figure 5. There are 35 vertices in G^*, listed from left to right along the "horizontal axis" of the graph. The naming convention that we are using in this graph partitions the vertices into five color groups, each color containing seven vertices. There is green $(g_0, \ldots, g_6)$, red $(r_0, \ldots, r_6)$, blue $(b_0, \ldots, b_6)$, magenta $(m_0, \ldots, m_6)$, and yellow $(y_0, \ldots y_6)$. The key observation about each of these color classes is that the subgraph of G^* induced by each of the color classes is exactly the graph G' used in Lemma 3, and moreover the designated blockers are exactly the same. For example, g_1 must be over $\overline{g_0 g_2}$, because g_0 doesn't see any point between g_1 and g_2 (including points of different colors) and g_2 doesn't see any point between g_0 and g_1. This implies that in order to obtain a terrain T that has G^* as its visibility graph, the x-coordinates must be chosen so that each of the 5 color classes satisfy the inequality of Lemma 3, and we will show that this is not possible.

Proving that G^* is persistent via a direct proof involves a tedious case analysis, and we instead show it is persistent via a computer program. The program builds the adjacency matrix as it is shown in Figure 5 and then ensures that the graph satisfies both the X-

property and the Bar-property. It can be much more easily verified that the algorithm we used to check the properties is correct than it would be to analyze a direct proof that G^* is persistent. A copy of the C++ source code we use to perform the check can be found at `https://github.com/PySean/GraphChecker`.

The following lemma will be used to prove the main result.

▶ **Lemma 4.** *If X satisfies $d_{0,1}d_{2,3}d_{3,4}d_{5,6} > d_{1,2}d_{4,5}d_{0,3}d_{3,6}$, then at least one of the following two statements is true: 1) $d_{1,2} < \min\{d_{0,1}, d_{2,3}\}$, or 2) $d_{4,5} < \min\{d_{3,4}, d_{5,6}\}$.*

Proof. Suppose without loss of generality that X is such that $d_{1,2} \geq d_{0,1}$ and $d_{4,5} \geq d_{5,6}$. We will show that $d_{0,1}d_{2,3}d_{3,4}d_{5,6} < d_{1,2}d_{4,5}d_{0,3}d_{3,6}$. We have:

$$\begin{aligned} d_{0,1}d_{2,3}d_{3,4}d_{5,6} &\leq d_{1,2}d_{2,3}d_{3,4}d_{4,5} \\ &< d_{1,2}d_{0,3}d_{3,6}d_{4,5} \\ &= d_{1,2}d_{4,5}d_{0,3}d_{3,6} \end{aligned}$$

Note that the second inequality follows since for all X we have $x_0 < x_2 < x_3$ implying $d_{2,3} < d_{0,3}$, and similarly we have $d_{3,4} < d_{3,6}$.

The lemma follows by applying a similar analysis for the other 3 cases. For example, if $d_{1,2} \geq d_{2,3}$ and $d_{4,5} \geq d_{3,4}$ then we'd have:

$$\begin{aligned} d_{0,1}d_{2,3}d_{3,4}d_{5,6} &\leq d_{0,1}d_{1,2}d_{4,5}d_{5,6} \\ &< d_{0,3}d_{1,2}d_{4,5}d_{3,6} \\ &= d_{1,2}d_{4,5}d_{0,3}d_{3,6} \end{aligned}$$

◀

For any color c from our set of colors $\{g, r, b, m, y\}$ and any pair of distinct integers $i, j \in \{0, \ldots, 6\}$ such that $i < j$, we let $d^c_{i,j}$ denote the absolute value of the difference of x-coordinates of c_i and c_j. For example, $d^m_{2,3}$ is the absolute value of the difference of x-coordinates of m_2 and m_3. We next show that for any vector X of thirty-five, increasing x-coordinates, at least one color class has to violate the inequality from Lemma 3.

▶ **Lemma 5.** *Let X be any vector of 35 x-coordinates in increasing order. There is at least one color $c \in \{g, r, b, m, y\}$ such that the x-coordinates for the seven points of that color do not satisfy $d^c_{0,1}d^c_{2,3}d^c_{3,4}d^c_{5,6} > d^c_{1,2}d^c_{4,5}d^c_{0,3}d^c_{3,6}$.*

Proof. If blue does not satisfy the inequality then we are done, so suppose that blue does satisfy it. Then according to Lemma 4, it must be that either $d^b_{1,2} < d^b_{0,1}$ or $d^b_{4,5} < d^b_{5,6}$. Without loss of generality, suppose that $d^b_{1,2} < d^b_{0,1}$.

Now consider the green points. If green does not satisfy the inequality then we are done, so suppose it does. Since $g_1 < b_0 < b_1 < g_2 < g_3 < b_2$ and $d^b_{1,2} < d^b_{0,1}$, we must have that $d^g_{2,3} < d^g_{1,2}$. Then by Lemma 4 we have that that $d^g_{4,5} < d^g_{5,6}$.

Now consider the magenta points. If magenta does not satisfy the inequality then we are done, so suppose it does. Since $g_4 < m_0 < m_1 < g_5 < g_6 < m_2$ and $d^g_{4,5} < d^g_{5,6}$, we have that $d^m_{0,1} < d^m_{1,2}$. Therefore if magenta satisfies the inequality then we have $d^m_{4,5} < d^m_{3,4}$ and $d^m_{4,5} < d^m_{5,6}$ by Lemma 4.

Now consider the red points. If red does not satisfy the inequality then we are done, so suppose it does. Since $r_1 < m_3 < m_4 < r_2 < r_3 < m_5$ and $d^m_{4,5} < d^m_{3,4}$, we have $d^r_{2,3} < d^r_{1,2}$. Then by Lemma 4, we must have that $d^r_{4,5} < d^r_{5,6}$.

Now consider the yellow points. Since $m_4 < y_3 < y_4 < m_5 < m_6 < y_5$ and $d^m_{4,5} < d^m_{5,6}$ then it must be that $d^y_{3,4} < d^y_{4,5}$. Since $r_4 < y_0 < y_1 < r_5 < r_6 < y_2$ and $d^r_{4,5} < d^r_{5,6}$, we also have that $d^y_{0,1} < d^y_{1,2}$. Then by Lemma 4 we have that yellow must violate the inequality. ◀

We now show that G^* is not the visibility graph for any terrain, proving Theorem 1.

▶ **Lemma 6.** *For any choice X of thirty-five, increasing x-coordinates, $\mathcal{T}(G^*, X) = \emptyset$.*

Proof. By Lemma 5, there must be at least one color that does not satisfy the inequality from Lemma 3. Arbitrarily pick one such color with a violated inequality, and let c denote our choice.

Let $\mathbf{A}$ be the constraint matrix generated by our reconstruction approach for G^*. Note that for each of the 6 constraints that we used in the proof of Lemma 3, we must have a similar set of constraints for the points of color c here, namely: (1) p_1^c should be above $\overline{p_0^c p_2^c}$, (2) p_3^c should be under $\overline{p_0^c p_4^c}$, (3) p_3^c should be over $\overline{p_1^c p_5^c}$, (4) p_3^c should be under $\overline{p_2^c p_6^c}$, (5) p_5^c should be under $\overline{p_3^c p_6^c}$, and (6) p_5^c should be over $\overline{p_4^c p_6^c}$. The "under" constraints clearly must be satisfied, but it is not immediately clear that the "over" constraints must be satisfied: it must be verified that, for example, p_1^c is a designated blocker for p_0^c and p_2^c (for example, p_0^c shouldn't see any points of any color between p_1^c and p_2^c). One can easily verify that this is the case for G^* for each of the "over" constraints for each of the color classes.

We then prove that $\mathcal{T}(G^*, X) = \emptyset$ using Farkas' Lemma. That is, we show the existence of a vector $\mathbf{z}$ such that $\mathbf{z} \leq 0$, $\mathbf{A}^{\mathrm{T}}\mathbf{z} \geq 0$, and $\mathbf{b}^{\mathrm{T}}\mathbf{z} < 0$ for every $\epsilon > 0$. Note that each entry in $\mathbf{z}$ corresponds with one of the constraints of A. We can simply pick our $\mathbf{z}$ by allowing each of the entries in $\mathbf{z}$ that correspond with one of the six constraints associated with the vertices of color c to take the same value as the corresponding entry in our vector in the proof of Lemma 3. We set every other entry of $\mathbf{z}$ to be 0. The analysis to see that this vector satisfies the conditions of Case 2 of Farkas' Lemma is then identical to that of the proof of Lemma 3, completing the proof of this lemma. ◀

5 Conclusions and open problems

The visibility graphs of terrains have been studied for almost 30 years, and it was known that the visibility graph for any terrain must be persistent. Previous works tended to believe that persistence formed a characterization of the visibility graphs of terrains, that is that for any persistent graph G, there is a terrain T such that G is the visibility graph of T. Our main result in this paper is to show the existence of a persistent graph that is *not* the visibility graph for any terrain. This proves that pseudo-terrains are not stretchable (as every persistent graph is the visibility graph for some pseudo-terrain).

There is much left to be determined about the visibility graphs of terrains. This paper re-opens the question about obtaining a characterization of the visibility graphs of terrains. We now have that the X-property and Bar-properties are necessary but not sufficient properties for a graph to be the visibility graph of a terrain. What additional properties must the graph satisfy? We believe our linear programming approach to reconstructing terrains can shed some light on the reconstruction problem as well. Previous research attempted to perform an iterative placement of points from left to right. Our work shows that one needs not be concerned with the y-coordinates of points when reconstructing a terrain, as if one has a set of feasible x-coordinates then the y-coordinates can be computed in polynomial time using linear programming. Given a visibility graph for a terrain, is there a polynomial-time algorithm that can compute such a set of x-coordinates?

References

1 James Abello. The majority rule and combinatorial geometry (via the symmetric group), 2004.
2 James Abello and Ömer Eğecioğlu. Visibility graphs of staircase polygons with uniform step length. *International Journal of Computational Geometry & Applications*, 3(01):27–37, 1993.
3 James Abello, Ömer Eğecioğlu, and Krishna Kumar. Visibility graphs of staircase polygons and the weak bruhat order, i: from visibility graphs to maximal chains. *Discrete & Computational Geometry*, 14(3):331–358, 1995.
4 James Abello, Krishna Kumar, and Ömer Eğecioğlu. A combinatorial view of visibility graphs of simple polygons. In *Proceedings of ICCI'93: 5th International Conference on Computing and Information*, pages 87–92. IEEE, 1993.
5 James Abello, Hua Lin, and Sekhar Pisupati. On visibility graphs of simple polygons. *Congressus Numerantium*, 90:119–128, 1992.
6 Safwa Ameer, Matt Gibson, Erik Krohn, and Qing Wang. Recognizing and reconstructing pseudo-polygons from their visibility graphs. *Manuscript*, 2020.
7 Seung-Hak Choi, Sung Yong Shin, and Kyung-Yong Chwa. Characterizing and recognizing the visibility graph of a funnel-shaped polygon. *Algorithmica*, 14(1):27–51, 1995. doi:10.1007/BF01300372.
8 Peter Corke. *Robotics, vision and control: fundamental algorithms in MATLAB*, volume 73. Springer Science & Business Media, 2011.
9 William S. Evans and Noushin Saeedi. On characterizing terrain visibility graphs. *JoCG*, 6(1):108–141, 2015. URL: http://jocg.org/index.php/jocg/article/view/130, doi:10.20382/jocg.v6i1a5.
10 Hazel Everett and Derek G. Corneil. Negative results on characterizing visibility graphs. *Comput. Geom.*, pages 51–63, 1995. doi:10.1016/0925-7721(95)00021-Z.
11 Hazel Jane Margaret Everett. *Visibility graph recognition*. PhD thesis, University of Toronto, 1990.
12 Subir Kumar Ghosh. On recognizing and characterizing visibility graphs of simple polygons. In *SWAT*, pages 96–104, 1988.
13 Subir Kumar Ghosh. On recognizing and characterizing visibility graphs of simple polygons. *Discrete & Computational Geometry*, 17(2):143–162, 1997. doi:10.1007/BF02770871.
14 Matt Gibson, Erik Krohn, and Qing Wang. A characterization of visibility graphs for pseudo-polygons. In *ESA*, pages 607–618, 2015.
15 Jean B. Lasserre. A discrete farkas lemma. *Discrete Optimization*, 1(1):67–75, 2004. doi:10.1016/j.disopt.2004.04.002.
16 Tomás Lozano-Pérez and Michael A Wesley. An algorithm for planning collision-free paths among polyhedral obstacles. *Communications of the ACM*, 22(10):560–570, 1979.
17 Saeed B Niku. *Introduction to robotics: analysis, systems, applications*, volume 7. Prentice Hall New Jersey, 2001.
18 Joseph O'Rourke and Ileana Streinu. Vertex-edge pseudo-visibility graphs: Characterization and recognition. In *Symposium on Computational Geometry*, pages 119–128, 1997. doi:10.1145/262839.262915.
19 Joseph O'Rourke and Ileana Streinu. The vertex-edge visibility graph of a polygon. *Computational Geometry*, 10(2):105–120, 1998. doi:10.1016/S0925-7721(97)00011-4.
20 G. Srinivasaraghavan and Asish Mukhopadhyay. A new necessary condition for the vertex visibility graphs of simple polygons. *Discrete & Computational Geometry*, 12:65–82, 1994. doi:10.1007/BF02574366.
21 Ileana Streinu. Non-stretchable pseudo-visibility graphs. *Comput. Geom.*, 31(3):195–206, 2005. doi:10.1016/j.comgeo.2004.12.003.

On β-Plurality Points in Spatial Voting Games

Boris Aronov
Tandon School of Engineering, New York University, Brooklyn, NY 11201, USA
boris.aronov@nyu.edu

Mark de Berg
Department of Computing Science, TU Eindhoven, 5600 MB Eindhoven, The Netherlands
m.t.d.berg@tue.nl

Joachim Gudmundsson
School of Computer Science, University of Sydney, Sydney, NSW 2006, Australia
joachim.gudmundsson@sydney.edu.au

Michael Horton
Sportlogiq, Inc., Montreal, Quebec H2T 3B3, Canada
michael.horton@sportlogiq.com

Abstract

Let V be a set of n points in $\mathbb{R}^d$, called *voters*. A point $p \in \mathbb{R}^d$ is a *plurality point* for V when the following holds: for every $q \in \mathbb{R}^d$ the number of voters closer to p than to q is at least the number of voters closer to q than to p. Thus, in a vote where each $v \in V$ votes for the nearest proposal (and voters for which the proposals are at equal distance abstain), proposal p will not lose against any alternative proposal q. For most voter sets a plurality point does not exist. We therefore introduce the concept of *β-plurality points*, which are defined similarly to regular plurality points except that the distance of each voter to p (but not to q) is scaled by a factor β, for some constant $0 < \beta \leqslant 1$. We investigate the existence and computation of β-plurality points, and obtain the following results.

- Define $\beta_d^* := \sup\{\beta : \text{any finite multiset } V \text{ in } \mathbb{R}^d \text{ admits a } \beta\text{-plurality point}\}$. We prove that $\beta_2^* = \sqrt{3}/2$, and that $1/\sqrt{d} \leqslant \beta_d^* \leqslant \sqrt{3}/2$ for all $d \geqslant 3$.
- Define $\beta(V) := \sup\{\beta : V \text{ admits a } \beta\text{-plurality point}\}$. We present an algorithm that, given a voter set V in $\mathbb{R}^d$, computes an $(1-\varepsilon)\cdot\beta(V)$ plurality point in time $O(\frac{n^2}{\varepsilon^{3d-2}} \cdot \log \frac{n}{\varepsilon^{d-1}} \cdot \log^2 \frac{1}{\varepsilon})$.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

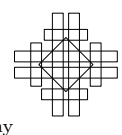
Keywords and phrases Computational geometry, Spatial voting theory, Plurality point, Computational social choice

Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.7

Related Version A full version of the paper is available at [1] `https://arxiv.org/abs/2003.07513`.

Funding *Boris Aronov*: Partially supported by NSF grant CCF-15-40656 and by grant 2014/170 from the US-Israel Binational Science Foundation.
Mark de Berg: Supported by the Netherlands' Organisation for Scientific Research (NWO) under project no. 024.002.003.
Joachim Gudmundsson: Supported under the Australian Research Council Discovery Projects funding scheme (project numbers DP150101134 and DP180102870).
Michael Horton: Part of the work performed on this paper was done while visiting NYU, supported by NSF grant CCF 12-18791.

Acknowledgements The authors would like to thank Sampson Wong for improving an earlier version of Lemma 2.6.



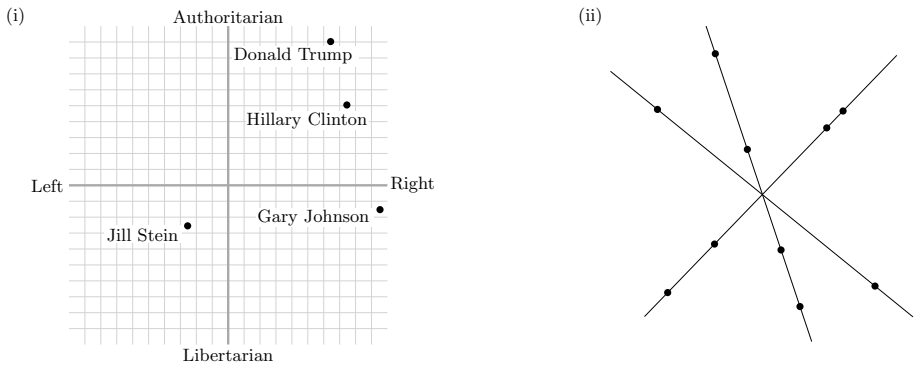


Figure 1 (i) The US presidential candidates 2016 modelled in the spatial voting model, according to The Political Compass (`https://politicalcompass.org/uselection2016`). Note that the points representing voters are not shown. (ii) The point set satisfies the generalized Plott symmetry conditions and therefore admits a plurality point.

1 Introduction

Background. Voting theory is concerned with mechanisms to combine preferences of individual voters into a collective decision. A desirable property of such a collective decision is that it is stable, in the sense that no alternative is preferred by more voters. In spatial voting games [6, 10] this is formalized as follows; see Fig. 1(i) for an example in a political context. The space of all possible decisions is modeled as $\mathbb{R}^d$ and every voter is represented by a point in $\mathbb{R}^d$, where the dimensions represent different aspects of the decision and the point representing a voter corresponds to the ideal decision for that voter. A voter v now prefers a proposed decision $p \in \mathbb{R}^d$ over some alternative proposal $q \in \mathbb{R}^d$ when v is closer to p than to q. Thus a point $p \in \mathbb{R}^d$ represents a stable decision for a given finite set V of voters if, for any alternative $q \in \mathbb{R}^d$, we have $\big|\{v \in V : |vp| < |vq|\}\big| \geqslant \big|\{v \in V : |vq| < |vp|\}\big|$. Such a point p is called a *plurality point*.[1]

For $d = 1$, a plurality point always exists, since in $\mathbb{R}^1$ a median of V is a plurality point. This is not true in higher dimensions, however. Define a *median hyperplane* for a set V of voters to be a hyperplane h such that both open half-spaces defined by h contain fewer than $|V|/2$ voters. For $d \geqslant 2$ a plurality point in $\mathbb{R}^d$ exists if and only if all median hyperplanes for V meet in a common point; see Fig. 1(ii). This condition is known as *generalized Plott symmetry* conditions [12, 23]; see also the papers by Wu *et al.* [28] and de Berg *et al.* [5], who present algorithms to determine the existence of a plurality point for a given set of voters.

It is very unlikely that voters are distributed in such a way that all median hyperplanes have a common intersection. (Indeed, if this happens, then a slightest generic perturbation of a single voter destroys the existence of the plurality point.) When a plurality point does not exist, we may want to find a point that is close to being a plurality point. One way to formalize this is to consider the center of the *yolk* (or *plurality ball*) of V, where the yolk [14, 17, 21, 22] is the smallest ball intersecting every median hyperplane of V. We introduce β-plurality points as an alternative way to relax the requirements for a plurality point, and study several combinatorial and algorithmic questions regarding β-plurality points.

[1] One can also require p to be strictly more popular than any alternative q. This is sometimes called a *strong* plurality point, in contrast to the *weak* plurality points that we consider.

β-Plurality points: definition and main questions. Let V be a multiset[2] of n voters in $\mathbb{R}^d$ in arbitrary, possibly coinciding, positions. In the traditional setting a proposed point $p \in \mathbb{R}^d$ wins a voter $v \in V$ against an alternative q if $|pv| < |qv|$. We relax this by fixing a parameter β with $0 < \beta \leqslant 1$ and letting p win v against q if $\beta \cdot |pv| < |qv|$. Thus we give an advantage to the initial proposal p by scaling distances to p by a factor $\beta \leqslant 1$. We now define

$$V[p \succ_\beta q] \coloneqq \{v \in V : \beta \cdot |pv| < |qv|\} \quad \text{and} \quad V[p \prec_\beta q] \coloneqq \{v \in V : \beta \cdot |pv| > |qv|\}$$

to be the multisets of voters won by p over q and lost by p against q, respectively. Finally, we say that a point $p \in \mathbb{R}^d$ is a *β-plurality point* for V when

$$\big|V[p \succ_\beta q]\big| \geqslant \big|V[p \prec_\beta q]\big|, \quad \text{for any point } q \in \mathbb{R}^d.$$

Observe that β-plurality is *monotone* in the sense that if p is a β-plurality point then p is also a β'-plurality point for all $\beta' < \beta$.

The spatial voting model was popularised by Black [6] and Down [10] in the 1950s. Stokes [26] criticized its simplicity and was the first to highlight the importance of taking non-spatial aspects into consideration. The reasoning is that voters may evaluate a candidate not only on their policies – their position in the policy space – but also take their so-called *valence* into account: charisma, competence, or other desirable qualities in the public's mind [13]. A candidate can also increase her valence by a stronger party support [27] or campaign spending [18]. Several models have been proposed to bring the spatial model closer to a more realistic voting approach; see [15, 16, 24] as examples. A common model is the multiplicative model, introduced by Hollard and Rossignol [19], which is closely related to the concept of a β-plurality point. The multiplicative model augments the existing spatial utility function by scaling the candidate's valence by a multiplicative factor. Note that in the 2-player game considered in this paper the multiplicative model is the same as our β-plurality model. From a computational point of view very little is known about the multiplicative model. We are only aware of a result by Chung [7], who studied the problem of positioning a new candidate in an existing space of voters and candidates, so that the valence required to win at least a given number of voters is minimized.

One reason for introducing β-plurality was that a set V of voters in $\mathbb{R}^d$, for $d \geqslant 2$, generally does not admit a plurality point. This immediately raises the question: Is it true that, for β small enough, any set V admits a β-plurality point? If so, we want to know the largest β such that any voter set V admits a β-plurality point, that is, we wish to determine

$$\beta_d^* \coloneqq \sup\{\beta : \text{any finite multiset } V \text{ in } \mathbb{R}^d \text{ admits a } \beta\text{-plurality point}\}.$$

Note that $\beta_1^* = 1$, since any set V in $\mathbb{R}^1$ admits a plurality point and 1-plurality is equivalent to the traditional notion of plurality.

After studying this combinatorial problem in Section 2, we turn our attention to the following algorithmic question: given a voter set V, find a point p that is a β-plurality point for the largest possible value β. In other words, if we define

$$\beta(V) \coloneqq \sup\{\beta : V \text{ admits a } \beta\text{-plurality point}\}$$

and

$$\beta(p, V) \coloneqq \sup\{\beta : p \text{ is a } \beta\text{-plurality point for } V\}$$

then we want to find a point p such that $\beta(p, V) = \beta(V)$.

[2] Even though we allow V to be a multiset, we sometimes refer to it as a "set" to ease the reading. When the fact that V is a multiset requires special treatment, we explicitly address this.

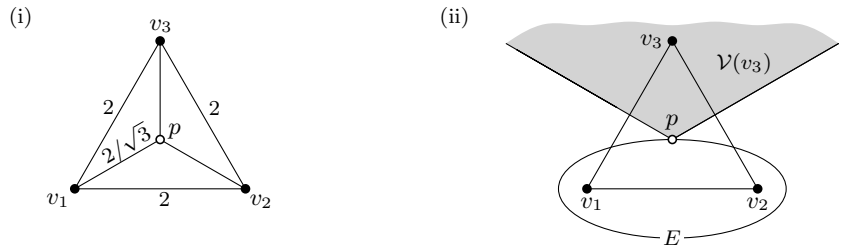


Figure 2 (i) The set $V = \{v_1, v_2, v_3\}$ of voters and the point p used in the proof of Lemma 2.2. (ii) The ellipse E is tangent to the Voronoi cell $\mathcal{V}(v_3)$.

Outline. In Section 2 we prove that $\beta_d^* \leqslant \sqrt{3}/2$ for all $d \geqslant 2$. To this end we first show that β_d^* is non-increasing in d, and then we exhibit a voter set V in $\mathbb{R}^2$ such that $\beta(V) \leqslant \sqrt{3}/2$. We also show how to construct, for any given V in $\mathbb{R}^2$, a $(\sqrt{3}/2)$-plurality point, thus proving that $\beta_2^* = \sqrt{3}/2$. For $d \geqslant 3$ we show how to construct a $(1/\sqrt{d})$-plurality point.

In Section 3 we study the problem of computing, for a given voter set V of n points in $\mathbb{R}^d$, a β-plurality point for the largest possible β. (Here we assume d to be a fixed constant.) While such a point can be found in polynomial time, the resulting running time is quite high. We therefore focus our attention on finding an approximately optimal point p, that is, a point p such that $\beta(p, V) \geqslant (1 - \varepsilon) \cdot \beta(V)$. We show that such a point can be computed in $O(\frac{n^2}{\varepsilon^{3d-2}} \cdot \log \frac{n}{\varepsilon^{d-1}} \cdot \log^2 \frac{1}{\varepsilon})$ time.

Notation. We denote the *open* ball of radius ρ centered at a point $q \in \mathbb{R}^d$ by $B(q, \rho)$ and, for a point $p \in \mathbb{R}^d$ and a voter v, we define $D_\beta(p, v) \coloneqq B(v, \beta \cdot |pv|)$. Observe that p wins v against a competitor q if and only if q is strictly outside $D_\beta(p, v)$, while q wins v if and only if q is strictly inside $D_\beta(p, v)$. Hence, $V[p \prec_\beta q] = \{v \in V : q \in D_\beta(p, v)\}$. We define $\mathcal{D}_\beta(p) \coloneqq \{D_\beta(p, v) : v \in V\}$ – here we assume V is clear from the context – and let $\mathcal{A}(\mathcal{D}_\beta(p))$ denote the arrangement induced by $\mathcal{D}_\beta(p)$. The competitor point q that wins the most voters against p will thus lie in the cell of $\mathcal{A}(\mathcal{D}_\beta(p))$ of the greatest depth or, more precisely, the cell contained in the maximum number of disks $D_\beta(p, v)$.

2 Bounds on β_d^*

In this section we will prove bounds on β_d^*, the supremum of all β such that any finite set $V \subset \mathbb{R}^d$ admits a β-plurality point. We start with an observation that allows us to apply bounds on β_d^* to those on $\beta_{d'}^*$ for $d' > d$. Let $\mathrm{conv}(V)$ denote the convex hull of V.

▶ **Observation 2.1.** *Let V be a finite multiset of voters in $\mathbb{R}^d$.*

(i) *Suppose a point $p \in \mathbb{R}^d$ is not a β-plurality point for V. Then there is a point $q \in \mathrm{conv}(V)$ such that $\big|V[p \succ_\beta q]\big| < \big|V[p \prec_\beta q]\big|$.*

(ii) *For any $p' \notin \mathrm{conv}(V)$, there is a point $p \in \mathrm{conv}(V)$ with $\beta(p, V) > \beta(p', V)$.*

(iii) *For any $d' > d$ we have $\beta_{d'}^* \leqslant \beta_d^*$.*

The proof is available in the full version [1]. We can now prove an upper bound on β_d^*.

▶ **Lemma 2.2.** *$\beta_d^* \leqslant \sqrt{3}/2$, for $d \geqslant 2$.*

Proof. By Observation 2.1(iii), it suffices to prove the lemma for $d = 2$. To this end let $V = \{v_1, v_2, v_3\}$ consist of three voters that form an equilateral triangle Δ of side length 2 in $\mathbb{R}^2$; see Fig. 2(i).

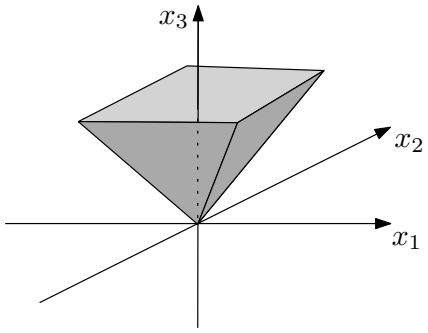


Figure 3 The cone C_3^+ used in the proof of Lemma 2.3.

Let p denote the center of Δ. We will first argue that $\beta(p, V) = \sqrt{3}/2$. Note that $|pv_i| = 2/\sqrt{3}$ for all three voters v_i. Hence, for $\beta = \sqrt{3}/2$, the open balls $D_\beta(v_i, p)$ are pairwise disjoint and touching at the mid-points of the edges of Δ. Therefore any competitor q either wins one voter and loses the remaining two, or wins no voter and loses at least one. The former happens when q lies inside one of the three balls $D_\beta(v_i, p)$; the later happens when q does not lie inside any of the balls, because in that case q can be on the boundary of at most two of the balls. Thus, for $\beta = \sqrt{3}/2$, the point p always wins more voters than q does. On the other hand, for $\beta > \sqrt{3}/2$, any two balls $D_\beta(v_i, p)$, $D_\beta(v_j, p)$ intersect and so a point q located in such a pairwise intersection wins two voters and beats p. We conclude that $\beta(p, V) = \sqrt{3}/2$, as claimed.

The lemma now follows if we can show that $\beta(p', V) \leqslant \sqrt{3}/2$ for any $p' \neq p$. Let $\mathrm{Vor}(V)$ be the Voronoi diagram of V, and let $\mathcal{V}(v_i)$ be the closed Voronoi cell of v_i, as shown in Fig. 2(ii). Assume without loss of generality that p' lies in $\mathcal{V}(v_3)$. Let E be the ellipse with foci v_1 and v_2 that passes through p. Thus

$$E \coloneqq \{z \in \mathbb{R}^2 : |zv_1| + |zv_2| = 4/\sqrt{3}\}.$$

Note that E is tangent to $\mathcal{V}(v_3)$ at the point p. Hence, any point $p' \neq p$ in $\mathcal{V}(v_3)$ has $|p'v_1| + |p'v_2| > 4/\sqrt{3}$. This implies that for $\beta \geqslant \sqrt{3}/2$ we have $\beta \cdot |p'v_1| + \beta \cdot |p'v_2| > 2$, and so the disks $D_\beta(p', v_1)$ and $D_\beta(p', v_2)$ intersect. It follows that for $\beta \geqslant \sqrt{3}/2$ there is a competitor q that wins two voters against p', which implies $\beta(p', V) \leqslant \sqrt{3}/2$ and thus finishes the proof of the lemma. ◀

We now prove lower bounds on β_d^*. We first prove that $\beta_d^* \geqslant 1/\sqrt{d}$ for any $d \geqslant 2$, and then we improve the lower bound to $\sqrt{3}/2$ for $d = 2$. The latter bound is tight by Lemma 2.2.

Let V be a finite multiset of n voters in $\mathbb{R}^d$. We call a hyperplane h *balanced* with respect to V, if both open half-spaces defined by h contain at most $n/2$ voters from V. Note the difference with median hyperplanes, which are required to have fewer than $n/2$ voters in both open half-spaces. Clearly, for any $1 \leqslant i \leqslant d$ there is a balanced hyperplane orthogonal to the x_i-axis, namely the hyperplane $x_i = m_i$, where m_i is a median in the multiset of all x_i-coordinates of the voters in V. (In fact, for any direction $\vec{d}$ there is a balanced hyperplane orthogonal to $\vec{d}$.)

▶ **Lemma 2.3.** *Let $d \geqslant 2$. For any finite multi-set V of voters in $\mathbb{R}^d$ there exists a point $p \in \mathbb{R}^d$ such that $\beta(p, V) = 1/\sqrt{d}$. Moreover, such a point p can be computed in $O(n)$ time.*

Proof. Let $\mathcal{H} \coloneqq \{h_1, \ldots, h_d\}$ be a set of balanced hyperplanes with respect to V such that h_i is orthogonal to the x_i-axis, and assume without loss of generality that $h_i \colon x_i = 0$. We will prove that the point p located at the origin is a β-plurality point for V for any $\beta < 1/\sqrt{d}$, thus showing that $\beta(p, V) \geqslant 1/\sqrt{d}$.

Let $q = (q_1, \ldots, q_d)$ be any competitor of p. We can assume without loss of generality that $\max_{1 \leqslant i \leqslant d} |q_i| = q_d > 0$. Thus q lies in the closed cone C_d^+ defined as

$$C_d^+ \coloneqq \{\ (x_1, \ldots, x_d) \in \mathbb{R}^d :\ x_d \geqslant |x_j| \text{ for all } j \neq d\ \}.$$

Note that C_d^+ is bounded by portions of the $2(d-1)$ hyperplanes $x_d = \pm x_j$ with $j \neq d$; see Fig. 3.

Because $h_d \colon x_d = 0$ is a balanced hyperplane, the open halfspace $h_d^+ \colon x_d > 0$ contains at most $n/2$ voters, which implies that the closed halfspace $\mathrm{cl}(h_d^-) \colon x_d \leqslant 0$ contains at least $n/2$ voters. Hence, it suffices to argue that for any $\beta < 1/\sqrt{d}$ the point p wins all the voters in $\mathrm{cl}(h_d^-)$ against q. For this we need the following claim, proved in the full version [1].

▷ Claim 2.4. For any voter $v \in \mathrm{cl}(h_d^-)$ with $v \neq p$, we have that $\sin(\angle qpv) \geqslant 1/\sqrt{d}$ with equality if and only if q lies on an edge of C_d^+ and v lies on the orthogonal projection of this edge onto h_d.

We can now use the Law of Sines and the claim above to derive that for any $\beta < 1/\sqrt{d}$ and any voter $v \in \mathrm{cl}(h_d^-)$ with $v \neq p$ we have

$$\beta \cdot |pv| < \frac{1}{\sqrt{d}} \cdot |pv| = \frac{1}{\sqrt{d}} \cdot \frac{|qv| \cdot \sin(\angle pqv)}{\sin(\angle qpv)} \leqslant |qv| \cdot \sin(\angle pqv) \leqslant |qv|\,.$$

Hence, p wins every point in $\mathrm{cl}(h_d^-)$. This proves the first part of the lemma since $\mathrm{cl}(h_d^-)$ contains at least $n/2$ voters, as already remarked.

Computing the point p is trivial once we have the balanced hyperplanes h_i, which can be found in $O(n)$ time by computing a median x_i-coordinate for each $1 \leqslant i \leqslant d$. ◀

In $\mathbb{R}^2$ we can improve the above bound: for any voter set V in the plane we can find a point p such that $\beta(p, V) = \sqrt{3}/2$. By Lemma 2.2 this bound is tight. The improvement is based on Lemma 2.5 below. This lemma – in fact a stronger version, stating that any two opposite cones defined by the three concurrent lines contain the same number of points – has been proved for even n by Dumitrescu *et al.* [11]. Our proof of Lemma 2.5 is similar to their proof. We give it because we also need it for odd n, and because we will need an understanding of the proof to describe our algorithm for computing the concurrent triple in the lemma. Our algorithm will run in $O(n \log^2 n)$ time, a significant improvement over the $O(n^{4/3} \log^{1+\varepsilon} n)$ running time obtained (for the case of even n) by Dumitrescu *et al.* [11].

▶ **Lemma 2.5.** *Given a multiset V of n voters in $\mathbb{R}^2$, there exists a triple of concurrent balanced lines (ℓ_1, ℓ_2, ℓ_3) such that the smaller angle between any two of them is $\frac{\pi}{3}$.*

Proof. Define the *orientation* of a line to be the counterclockwise angle it makes with the positive y-axis. Recall that for any given orientation θ there exists at least one balanced line with orientation θ. When n is odd this line is unique: it passes through the median of the voter set V when V is projected orthogonally onto a line orthogonal to the lines of orientation θ. In the rest of the proof it will be convenient to have a unique balanced line for any orientation θ. To achieve this when n is even, we simply delete an arbitrary voter from V. (If there are other voters at the same location, these voters are not deleted.) This is allowed because when $|V|$ is even, a balanced line for $V \setminus \{v\}$ is also a balanced line for V.

Now let μ be the function that maps an angle value θ to the unique balanced line $\mu(\theta)$; see Figure 4(i). Note that μ is continuous for $0 \leqslant \theta < \pi$. Let $\ell_1(\theta) \coloneqq \mu(\theta)$, and $\ell_2(\theta) \coloneqq \mu(\theta + \frac{\pi}{3})$, and $\ell_3(\theta) \coloneqq \mu(\theta + \frac{2\pi}{3})$. For $i \neq j$, let $p_{ij}(\theta) \coloneqq \ell_i(\theta) \cap \ell_j(\theta)$ be the intersection point

Figure 4 (i) The balanced line $\mu(\theta)$. (ii) If p_{23} is to the left of the directed line $\ell_1(0)$ then $p_{13}(0)$ is to the right of $\ell_2(0)$.

between $\ell_i(\theta)$ and $\ell_j(\theta)$. If $p_{23}(0) \in \ell_1(0)$ then the lines $\ell_1(0), \ell_2(0), \ell_3(0)$ are concurrent and we are done. Otherwise, consider the situation at $\theta = 0$ and imagine $\ell_1(0)$ and $\ell_2(0)$ to be directed in the positive y-direction, as in Fig. 4(ii). Clearly, if $p_{23}(0)$ is to the left of the directed line $\ell_1(0)$ then $p_{13}(0)$ is to the right of the directed line $\ell_2(0)$, and vice versa. Now increase θ from 0 to $\pi/3$, and note that $\ell_1(\pi/3) = \ell_2(0)$ and $p_{23}(\pi/3) = p_{13}(0)$. Hence, $p_{23}(\theta)$ lies to a different side of the directed line $\ell_1(\theta)$ for $\theta = 0$ than it does for $\theta = \pi/3$. Since both $\ell_1(\theta)$ and $p_{23}(\theta)$ move continuously, this implies that for some $\overline{\theta} \in (0, \pi/3)$ the point $p_{23}(\overline{\theta})$ crosses the line $\ell_1(\overline{\theta})$, and so the lines $\ell_1(\overline{\theta}), \ell_2(\overline{\theta}), \ell_3(\overline{\theta})$ are concurrent. ◀

Next we show how to efficiently compute a triple as in Lemma 2.5. We follow the definitions and notation from the proof of Lemma 2.5. We will assume that n is odd, which, as argued, is without loss of generality.

To find a concurrent triple of balanced lines, we first compute the lines $\ell_1(0), \ell_2(0), \ell_3(0)$ in $O(n)$ time. If they are concurrent, we are done. Otherwise, there is a $\overline{\theta} \in (0, \pi/3)$ such that $\ell_1(\overline{\theta}), \ell_2(\overline{\theta}), \ell_3(\overline{\theta})$ are concurrent. To find this value $\overline{\theta}$, we dualize the voter set V, using the standard duality transform that maps a point (a, b) to the line $y = ax + b$, and vice versa. Let v^* denote the dual line of the voter v, and let $V^* := \{v^* : v \in V\}$. Note that, for $\theta \in (0, \pi/3)$, the lines $\ell_1(\theta), \ell_2(\theta), \ell_3(\theta)$ are all non-vertical, therefore their duals $\ell_i^*(\theta)$ are well-defined.

Consider the arrangement $\mathcal{A}(V^*)$ defined by the duals of the voters. For $\theta \neq 0$, define slope(θ) to be the slope of the lines with orientation θ. Then $\mu^*(\theta)$, the dual of $\mu(\theta)$, is the intersection point of the vertical line $x = \text{slope}(\theta)$ with L_{med}, the median level in $\mathcal{A}(V^*)$. (The median level of $\mathcal{A}(V^*)$ is the set of points q such that there are fewer than $n/2$ lines below q and fewer than $n/2$ lines above q; this is well defined since we assume n is odd. The median level forms an x-monotone polygonal curve along edges of $\mathcal{A}(V^*)$.)

Now consider the duals $\ell_1^*(\theta), \ell_2^*(\theta), \ell_3^*(\theta)$, which all lie on L_{med}. For $\theta \in (0, \pi/3)$, the x-coordinate of $\ell_1^*(\theta)$ lies in $(-\infty, -1/\sqrt{3})$, the x-coordinate of $\ell_2^*(\theta)$ lies in $(-1/\sqrt{3}, 1/\sqrt{3})$, and the x-coordinate of $\ell_3^*(\theta)$ lies in $(1/\sqrt{3}, \infty)$. We split L_{med} into three pieces corresponding to these ranges of x-coordinate. Let E_1, E_2, and E_3 denote the sets of edges forming the parts of L_{med} in the first, second, and third range, respectively, where edges crossing the vertical lines $x = -1/\sqrt{3}$ and $x = 1/\sqrt{3}$ are split; see Fig. 5.

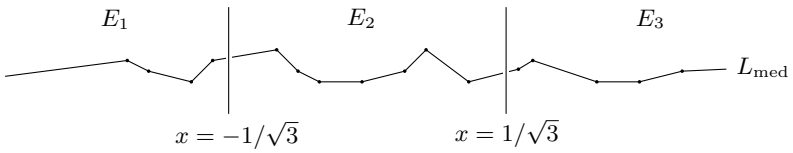


Figure 5 The edge sets E_1, E_2, and E_3 of L_{med}, the median level in $\mathcal{A}(V^*)$.

Recall that we want to find a value $\overline{\theta} \in (0, \pi/3)$ such that $\ell_1(\overline{\theta}), \ell_2(\overline{\theta}), \ell_3(\overline{\theta})$ are concurrent (or, in other words, such that the points $\ell_1^*(\overline{\theta}), \ell_2^*(\overline{\theta}), \ell_3^*(\overline{\theta})$ are collinear). Also recall that, for any $\theta \in (0, \pi/3)$, the point $\ell_i^*(\theta)$ lies on an edge in E_i, for $i = 1, 2, 3$. One way to find $\overline{\theta}$ would be to explicitly compute L_{med}, and then increase θ (starting at $\theta = 0$) and see how the points $\ell_i^*(\theta)$ move over E_i, until we reach a value where $\ell_1(\overline{\theta}), \ell_2(\overline{\theta}), \ell_3(\overline{\theta})$ are concurrent. Since the best known bounds on the complexity of the median level is $O(n^{4/3})$ [9] we will proceed differently, as follows.

1. Find an interval $(\theta_1, \theta_1') \subseteq (0, \pi/3)$ for which there is a $\overline{\theta}$ with the desired properties and such that $\ell_1^*(\theta)$ lies on the same edge of E_1 for all $\theta \in (\theta_1, \theta_1')$.
2. Find an interval $(\theta_2, \theta_2') \subseteq (\theta_1, \theta_1')$ for which there is a $\overline{\theta}$ with the desired properties and such that $\ell_2^*(\theta)$ lies on the same edge of E_2 for all $\theta \in (\theta_2, \theta_2')$.
3. Find an interval $(\theta_3, \theta_3') \subseteq (\theta_2, \theta_2')$ for which there is a $\overline{\theta}$ with the desired properties and such that $\ell_3^*(\theta)$ lies on the same edge of E_3 for all $\theta \in (\theta_3, \theta_3')$.
4. After Step 3 we have an interval $(\theta_3, \theta_3') \subseteq (0, \pi/3)$ for which there is a $\overline{\theta}$ with the desired properties and such that $\ell_1^*(\overline{\theta})$, $\ell_2^*(\overline{\theta})$, and $\ell_3^*(\overline{\theta})$ each lie on a fixed edge of L_{med}. Let v_1, v_2, and v_3 denote the voters whose dual lines contain these three edges. We know that for any $\theta \in (\theta_3, \theta_3')$, the line through v_1 with orientation θ is a balanced line. Similarly, for any $\theta \in (\theta_3, \theta_3')$ the line through v_2 with orientation $\theta + \pi/3$ is a balanced line, and the line through v_3 with orientation $\theta + 2\pi/3$ is a balanced line. Finding a $\overline{\theta} \in (\theta_3, \theta_3')$ with the desired properties thus only requires finding a $\overline{\theta} \in (\theta_3, \theta_3')$ for which these three lines are concurrent. Such a $\overline{\theta}$ is guaranteed to exist by construction, and finding it is a constant-time operation.

It remains to explain how to perform Steps 1–3. Below we describe this for Step 1; the other steps can be implemented in a similar way.

To implement Step 1 we perform a binary search over the x-coordinates of the vertices of $\mathcal{A}(V^*)$ in the slab $(-\infty, -1/\sqrt{3}) \times (-\infty, \infty)$, as follows. In a generic step of this binary search we have an interval $(\theta_{\min}, \theta_{\max})$ such that $p_{23}(\theta)$ lies to a different side of the directed line $\ell_1(\theta)$ for $\theta = \theta_{\min}$ than for $\theta = \theta_{\max}$. (Recall that this implies that there is a $\overline{\theta} \in (\theta_{\min}, \theta_{\max})$ with the desired property.) This interval corresponds to the slab $(\text{slope}(\theta_{\min}), \text{slope}(\theta_{\max})) \times (-\infty, \infty)$ in the dual plane. Let X be the set of x-coordinates of the vertices of $\mathcal{A}(V^*)$ inside this slab. We can find the median x_{med} of X in $O(n \log n)$ time using the algorithm by Cole *et al.* [8]. We then compute the three balanced lines $\ell_i(\theta_{\text{med}})$, where θ_{med} is such that $\text{slope}(\theta_{\text{med}}) = x_{\text{med}}$. If these three lines are concurrent we are immediately done, and we can stop. Otherwise we determine where $p_{23}(\theta_{\text{med}})$ lies relative to $\ell_1(\theta_{\text{med}})$, and based on that decide whether to recurse on $(\theta_{\min}, \theta_{\text{med}})$ or on $(\theta_{\text{med}}, \theta_{\max})$. We continue until the slab $(\text{slope}(\theta_{\min}), \text{slope}(\theta_{\max})) \times (-\infty, \infty)$ contains no more vertices of $\mathcal{A}(V^*)$. We then finish Step 1 by setting $(\theta_1, \theta_1') := (\theta_{\min}, \theta_{\max})$.

Each iteration of the binary search takes $O(n \log n)$ time, so Step 1 takes $O(n \log^2 n)$ time. Steps 2 and 3 can be done in a similar fashion, so we can find a concurrent triple of balanced lines as in Lemma 2.5 in $O(n \log^2 n)$ time. In [1] we show that the common intersection of these three lines is a $(\sqrt{3}/2)$-plurality point, thus proving the following lemma.

▶ **Lemma 2.6.** *For any finite multi-set V of voters in $\mathbb{R}^2$ there exists a point $p \in \mathbb{R}^2$ such that $\beta(p, V) \geqslant \sqrt{3}/2$. Moreover, such a point p can be computed in $O(n \log^2 n)$ time.*

The following theorem summarizes the results of this section.

▶ **Theorem 2.7.**

- **(i)** *We have $\beta_2^* = \sqrt{3}/2$. Moreover, for any multiset V of n voters in $\mathbb{R}^2$ we can compute a point p such that $\beta(p, V) = \sqrt{3}/2$ in $O(n \log^2 n)$ time.*
- **(ii)** *For $d \geqslant 2$, we have $1/\sqrt{d} \leqslant \beta_d^* \leqslant \sqrt{3}/2$. Moreover, for any multiset V of n voters in $\mathbb{R}^d$ we can compute a point p such that $\beta(p, V) = 1/\sqrt{d}$ in $O(n)$ time.*

3 Finding a point that maximizes $\beta(p, V)$

We know from Theorem 2.7 that, for any multiset V of n voters in $\mathbb{R}^d$, we can compute a point p with $\beta(p, V) \geq 1/\sqrt{d}$ (even with $\beta(p, V) \geq \sqrt{3}/2$, in the plane). However, a given voter multiset V may admit a β-plurality point for larger values of β – possibly even for $\beta = 1$. In this section we study the problem of computing a point p that maximizes $\beta(p, V)$, that is, a point p with $\beta(p, V) = \beta(V)$.

3.1 An exact algorithm

Below we sketch an exact algorithm to compute $\beta(V)$ together with a point p such that $\beta(p, V) = \beta(V)$. Our goal is to show that, for constant d, this can be done in polynomial time. We do not make a special effort to optimize the exponent in the running time; it may be possible to speed up the algorithm, but it seems clear that it will remain impractical, because of the asymptotic running time, and also because of algebraic issues.

Note that we can efficiently check whether a true plurality point exists (i.e., $\beta = 1$ can be achieved) in time $O(n \log n)$ by an algorithm of De Berg *et al.* [5], and if so, identify this point. Therefore, hereafter $\beta = 1$ is used as a sentinel value, and our algorithm proceeds on the assumption that $\beta(p, V) < 1$ for any point p.

For a voter $v \in V$, a candidate $p \in \mathbb{R}^d$, and an alternative candidate $q \in \mathbb{R}^d$, define $f_v(p, q) := \min(|qv|/|pv|, 1)$ when $p \neq v$, and define $f_v(p, q) := 1$ otherwise. Observe that for $f_v(p, q) < 1$ we have

- q wins voter v over p if and only if $\beta > f_v(p, q)$,
- q and p have a tie over voter v if and only if $\beta = f_v(p, q)$, and
- p wins voter v over q if and only if $\beta < f_v(p, q)$.

For $f_v(p, q) = 1$ this is not quite true: when $p = q = v$ we always have a tie, and when $|pv| < |qv|$ then p wins v even when $\beta = f_v(p, q) = 1$. When $p = q$ there is a tie for all voters, so the final conclusion (namely that $\big|V[p \succ_\beta q]\big| \geqslant \big|V[p \prec_\beta q]\big|$) is still correct. The fact that we incorrectly conclude that there is a tie when $|pv| < |qv|$ and $\beta = f_v(p, q) = 1$ does not present a problem either, since we assume $\beta(p, V) < 1$. Hence, we can pretend that checking if $\beta > f_v(p, q)$, or $\beta = f_v(p, q)$, or $\beta < f_v(p, q)$ tells us whether q wins v, or there's a tie, or p wins v, respectively.

Hereafter we identify $f_v \colon \mathbb{R}^{2d} \to \mathbb{R}$ with its graph $\{(p, q, f_v(p, q))\} \subset \mathbb{R}^{2d+1}$, which is a d-dimensional surface. Let f_v^+ be the set of points lying above this graph, and f_v^- be the set of points lying below it. Thus f_v^+ is precisely the set of combinations of (p, q, β) where q wins v over p, while f_v is the set where p ties with q, and f_v^- is the set where q loses v to p. Consider the arrangement $\mathcal{A} := \mathcal{A}(F)$ defined by the set of surfaces $F := \{f_v : v \in V\}$. Each face C in $\mathcal{A}$ is a maximal connected set of points with the property that all points of C are contained in, lie below, or lie above, the same subset of surfaces of F. (Note that we consider faces of all dimensions, not just full-dimensional cells.) Thus for all $(p, q, \beta) \in C$, exactly one of the following holds: $\big|V[p \succ_\beta q]\big| < \big|V[p \prec_\beta q]\big|$, or $\big|V[p \succ_\beta q]\big| = \big|V[p \prec_\beta q]\big|$, or $\big|V[p \succ_\beta q]\big| > \big|V[p \prec_\beta q]\big|$. Let L be the union of all faces C of $\mathcal{A}(F)$ such that $\big|V[p \succ_\beta q]\big| < \big|V[p \prec_\beta q]\big|$, that is, such that p loses against q for all (p, q, β) in C. We can construct $\mathcal{A}$ and L in time $O(n^{2d+1})$ using standard machinery, as $\mathcal{A}$ is an arrangement of degree-4 semi-algebraic surfaces of constant description complexity [3, 4]. We are interested in the set

$$W := \{(p, \beta) : \big|V[p \succ_\beta q]\big| \geqslant \big|V[p \prec_\beta q]\big| \text{ for any competitor } q \} \subset \mathbb{R}^{d+1}.$$

What is the relationship between W and L? A point (p, β) is in W precisely when, for every choice of $q \in \mathbb{R}^d$, p wins at least as many voters as q (for the given β). In other words,

$$W = \{(p, \beta) \mid \text{there is no } q \text{ such that } (p, q, \beta) \in L\}.$$

That is, W is the complement of the projection of L to the space $\mathbb{R}^{d+1}$ representing the pairs (p, β). The most straightforward way to implement the projection would involve constructing semi-algebraic formulas describing individual faces and invoking quantifier elimination on the resulting formulas [3]. Below we outline a more obviously polynomial-time alternative.

Construct the *vertical decomposition* $\mathrm{vd}(\mathcal{A})$ of $\mathcal{A}$, which is a refinement of $\mathcal{A}$ into pieces ("subfaces" τ), each bounded by at most $2(2d+1)$ surfaces of constant degree and therefore of constant complexity; see [25]. A vertical decomposition is specified by ordering the coordinates – we put the coordinates corresponding to q *last*. Since $\mathrm{vd}(\mathcal{A})$ is a refinement of $\mathcal{A}$, the set L is the union of subfaces τ of $\mathrm{vd}(\mathcal{A})$ fully contained in L. Since $\mathcal{A}$ is an arrangement of n well-behaved surfaces in $2d + 1 \geqslant 5$ dimensions, the complexity of $\mathrm{vd}(\mathcal{A})$ is $O(n^{2(2d+1)-4+\varepsilon}) = O(n^{4d-2+\varepsilon})$, for any $\varepsilon > 0$ [20]. In particular, L comprises $\ell \coloneqq O(n^{4d-2+\varepsilon})$ subfaces.

Since each $\tau \subset L$ is a subface of the vertical decomposition $\mathrm{vd}(\mathcal{A})$ in which the last d coordinates correspond to q, the projection τ' of τ to $\mathbb{R}^{d+1}$ is easy obtain (see [25]) in constant time; indeed it can be obtained by discarding the constraints on these last d coordinates from the description of τ. Thus, in time $O(\ell)$ we can construct the family of all the projections of the ℓ subfaces of L, each a constant-complexity semi-algebraic object in $\mathbb{R}^{d+1}$. We now construct the arrangement $\mathcal{A}'$ of the resulting collection and its vertical decomposition $\mathrm{vd}(\mathcal{A}')$. The complexity of $\mathrm{vd}(\mathcal{A}')$ is either $O(\ell^{d+1+\varepsilon})$ or $O(\ell^{2(d+1)-4+\varepsilon}) = O(\ell^{2d-2+\varepsilon})$, depending on whether $d + 1 \leqslant 4$ or not, respectively [20]. Each subface in $\mathrm{vd}(\mathcal{A}')$ is either fully contained in the projection of L or fully disjoint from it. Collecting all of the latter subfaces, we obtain a representation of W as a union of at most $O(\ell^{O(d)}) = O(n^{O(d^2)})$ constant-complexity semi-algebraic objects.

Now if $(p, \beta) \in W$ is the point with the highest value of β, then $\beta(V) = \beta(p, V) = \beta$. It can be found by enumerating all the subfaces of $\mathrm{vd}(\mathcal{A}')$ contained in the closure of W – we take the closure because $V(p, \beta)$ is defined as a supremum – and identifying their topmost point or points. Since each face has constant complexity, this can be done in $O(1)$ time per subface.[3] This completes our description of an $O(n^{O(d^2)})$-time algorithm to compute the best β that can be achieved for a given set of voters V, and the candidate p (or the set of candidates) that achieve this value.

3.2 An approximation algorithm

Since computing $\beta(V)$ exactly appears expensive, we now turn our attention to approximation algorithms. In particular, given a voter set V in $\mathbb{R}^d$ and an $\varepsilon \in (0, 1/2]$, we wish to compute a point $\bar{p}$ such that $\beta(\bar{p}, V) \geqslant (1 - \varepsilon) \cdot \beta(V)$.

Our approximation algorithm works in two steps. In the first step, we compute a set P of $O(n/\varepsilon^{2d-1} \log(1/\varepsilon))$ *candidates*. P may not contain the true optimal point p, but we will ensure that P contains a point $\bar{p}$ such that $\beta(\bar{p}, V) \geqslant (1 - \varepsilon/2) \cdot \beta(V)$. In the second step, we approximate $\beta(p', V)$ for each $p' \in P$, to find an approximately best candidate.

[3] Once again, the projection to the β coordinate is particularly easy to obtain if, when constructing $\mathrm{vd}(A')$, we set the coordinate corresponding to β *first*.

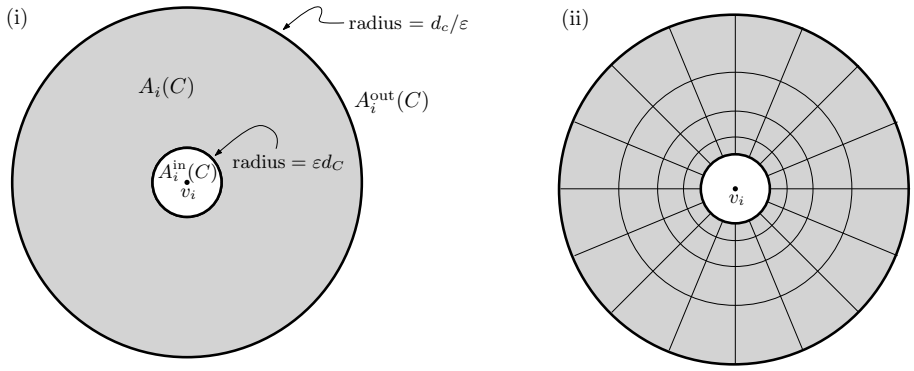


Figure 6 (i) The closed spherical shell $A_i(C)$ defined by the two balls of radii $\varepsilon \cdot d_C$ and d_C/ε around v_i. (ii) The exponential grid $G_i(C)$. The grid is defined by a collection of spheres centered at v_i, plus extreme rays of the cones with apex at v_i. The spheres have radii $(1+\varepsilon/4)^i \cdot \varepsilon \cdot d_C$ for $0 \leqslant i \leqslant \log_{(1+\varepsilon/4)}(1/\varepsilon^2) = O((1/\varepsilon)\log(1/\varepsilon))$, and the interior angle of a cone is $\varepsilon/2\sqrt{d}$.

Constructing the candidate set P. To construct the candidate set P, we will generate, for each voter $v_i \in V$, a set P_i of $O(1/\varepsilon^{2d-1}\log(1/\varepsilon))$ candidate points. Our final set P of candidates will be the union of the sets $P_1, \ldots, P_n$. Next we describe how to construct P_i.

Partition $\mathbb{R}^d$ into a set $\mathcal{C}$ of $O(1/\varepsilon^{d-1})$ simplicial cones with apex at v_i and opening angle $\varepsilon/(2\sqrt{d})$, so that for every pair of points u and u' in the same cone we have $\angle uv_iu' \leqslant \varepsilon/(2\sqrt{d})$. We assume for simplicity (and can easily guarantee) that no voter in V lies on the boundary of any of the cones, except for v_i itself and any voters coinciding with v_i. Let $\mathcal{C}(v_i)$ denote the set of all cones in $\mathcal{C}$ whose interior contains at least one voter. For each cone $C \in \mathcal{C}(v_i)$ we generate a candidate set $G_i(C)$ as explained next, and then we set $P_i := \bigcup_{C\in\mathcal{C}(v_i)} G_i(C) \cup \{v_i\}$.

Let d_C be the distance from v_i to the nearest other voter (not coinciding with v_i) in C. Let $A_i(C)$ be the closed spherical shell defined by the two spheres of radii $\varepsilon \cdot d_C$ and d_C/ε around v_i, as shown in Fig. 6(i). The open ball of radius $\varepsilon \cdot d_C$ is denoted by $A_i^{\text{in}}(C)$, and the complement of the closed ball of radius d_C/ε is denoted by $A_i^{\text{out}}(C)$. Let $G_i(C)$ be the vertices in an exponential grid defined by a collection of spheres centered at v_i, and the extreme rays of the cones in $\mathcal{C}$; see Fig. 6(ii). The spheres have radii $(1+\varepsilon/4)^i \cdot \varepsilon \cdot d_C$, for $0 \leqslant i \leqslant \log_{(1+\varepsilon/4)}(1/\varepsilon^2) = O((1/\varepsilon)\log(1/\varepsilon))$. Observe that $G_i(C)$ contains not only points in C, but in the entire spherical shell $A_i(C)$. The set $G_i(C)$ consists of $O(1/\varepsilon^d \log(1/\varepsilon))$ points, and it has the following property:

> Let p be any point in the spherical shell $A_i(C)$, and let p' be a corner of the grid cell containing p and nearest to p. Then $|p'p| \leqslant \varepsilon \cdot |pv_i|$. $(*)$

To prove the property, let q be the point on pv_i such that $|qv_i| = |p'v_i|$. From the construction of the exponential grid we have $|pq| \leqslant \frac{\varepsilon}{4}\cdot|pv_i|$. Since p' and q lie in the same cone $\angle p'v_iq \leqslant \frac{\varepsilon}{2\sqrt{d}}$ and, consequently, $|p'q| \leqslant \frac{\varepsilon}{2}\cdot|qv_i| \leqslant (1+\frac{\varepsilon}{4})\cdot\frac{\varepsilon}{2}\cdot|pv_i|$. The property is now immediate since $|pp'| \leqslant |pq| + |qp'| < \varepsilon \cdot |pv_i|$.

As mentioned above, $P_i := \bigcup_{C\in\mathcal{C}(v_i)} G_i(C) \cup \{v_i\}$, and the final candidate set P is defined as $P := \bigcup_{v_i\in V} P_i$. Computing the sets P_i is easy: for each of the $O(1/\varepsilon^{d-1})$ cones $C \in \mathcal{C}(v_i)$, determine the nearest neighbor of v_i in C in $O(n)$ time by brute force, and then generate $G_i(C)$ in $O((1/\varepsilon^{(d-1)})\log(1/\varepsilon))$ time. (It is not hard to speed up the nearest-neighbor computation using appropriate data structures, but this will not improve the final running time in Theorem 3.4.) We obtain the following lemma.

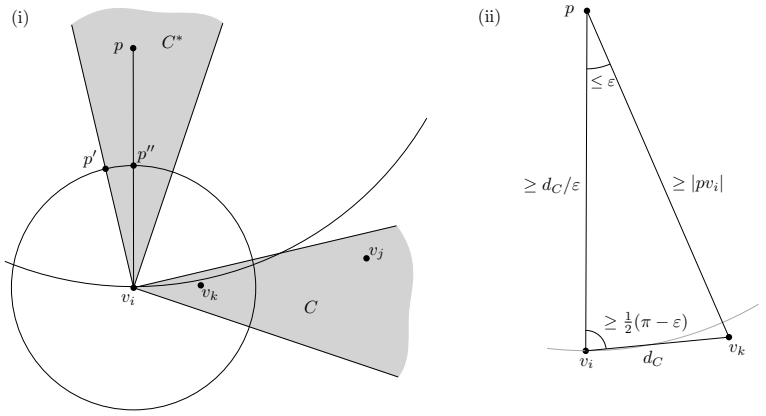


Figure 7 Illustration for Case III.

▶ **Lemma 3.1.** *The candidate set P has size $O(n/\varepsilon^{2d-1}\log(1/\varepsilon))$ and can be constructed in $O(n^2/\varepsilon^{d-1} + n/\varepsilon^{2d-1}\log(1/\varepsilon))$ time.*

The next lemma is crucial to show that P is a good candidate set.

▶ **Lemma 3.2.** *For any point $p \in \mathbb{R}^d$, there exists a point $p' \in P$ with the following property: for any voter $v_j \in V$, we have that $|p'v_j| \leqslant (1+2\varepsilon)\cdot|pv_j|$.*

Proof. Let v_i be a voter nearest to p. We will argue that the set P_i contains a point p' with the desired property. We distinguish three cases.

Case I: There is a cone $C \in \mathcal{C}(v_i)$ such that p lies in the spherical shell $A_i(C)$. In this case we pick p' to be a point of $G_i(C)$ nearest to p, that is, p' is a corner nearest to p of the grid cell containing p. By property $(*)$ we have

$$|p'v_j| \leqslant |p'p| + |pv_j| \leqslant \varepsilon\cdot|pv_i| + |pv_j| \leqslant (1+\varepsilon)\cdot|pv_j|,$$

where the last inequality follows from the fact that v_i is a voter nearest to p.

Case II: Point p lies in $A_i^{\text{in}}(C)$ for all $C \in \mathcal{C}(v_i)$. In this case we pick $p' := v_i$. Clearly $|p'v_j| = 0 \leqslant (1+\varepsilon)\cdot|pv_j|$ for $j = i$. For $j \neq i$, we argue as follows. Let $C \in \mathcal{C}(v_i)$ be the cone containing v_j. Since we are in Case II we know that $p \in A_i^{\text{in}}(C)$, and so

$$|p'v_j| \leqslant |p'p| + |pv_j| \leqslant \varepsilon d_C + |pv_j| \leqslant \varepsilon|p'v_j| + |pv_j|. \tag{1}$$

Moreover, we have

$$|pv_j| \geqslant |p'v_j| - |pp'| \geqslant |p'v_j| - \varepsilon d_C \geqslant |p'v_j|/2, \tag{2}$$

where the last step uses that $\varepsilon \leqslant 1/2$ and $d_C \leqslant |p'v_j|$. Combining (1) and (2), we obtain $|p'v_j| \leqslant (1+2\varepsilon)\cdot|pv_j|$.

Case III: Cases I and II do not apply. In this case there is at least one cone C such that $p \in A_i^{\text{out}}(C)$. Of all such cones, let C^* be the one whose associated distance d_{C^*} is maximized. Let p'' be the point on the segment pv_i at distance d_C/ε from v_i. Without loss of generality, we will assume that p and v_i only differ in the x_d coordinate; see Fig. 7(i).

We will prove that the point p' of $G_i(C^*)$ nearest to p'' (refer to Fig. 7(i)) has the desired property. Consider a voter v_j. We distinguish three cases.

- When $i = j$, then we have

$$|p'v_i| \leqslant |p'p''| + |p''v_i| \leqslant (1+\varepsilon)|p''v_i| \leqslant (1+\varepsilon)|pv_i|,$$

where the second inequality follows from $(*)$.
- When v_j lies in a cone C such that $p \in A_i^{\text{in}}(C)$, then we can use the same argument as in Case II to show that $|p'v_j| \leqslant (1+2\varepsilon)\cdot|pv_j|$.
- In the remaining case v_j lies in a cone C such that $p \in A_i^{\text{out}}(C)$. Let v_k be a voter in C nearest to v_i. Since $|v_iv_k| = d_C$, $|pv_i| \geqslant d_C/\varepsilon$, and $|pv_k| \geqslant |pv_i|$, we can deduce that $\angle pv_iv_k \geqslant \pi/2 - \varepsilon/2$, as illustrated in Fig. 7(ii). Furthermore, since v_k and v_j belong to the same cone C the angle $\angle v_kv_iv_j$ is bounded by $\varepsilon/2\sqrt{d} \leqslant \varepsilon/2$ according to the construction. Putting the two angle bounds together we conclude that $\angle pv_iv_j \geqslant \frac{\pi}{2} - \varepsilon$. Now consider the triangle defined by p, v_i and v_j. From the Law of Sines we obtain

$$\frac{|v_iv_j|}{\sin\angle v_ipv_j} = \frac{|pv_j|}{\sin\angle pv_iv_j}, \quad \text{or} \quad |v_iv_j| = |pv_j|\cdot\frac{\sin\angle v_ipv_j}{\sin\angle pv_iv_j} \leqslant \frac{|pv_j|}{\cos\varepsilon} \leqslant (1+\varepsilon)\cdot|pv_j|,$$

for $\varepsilon < 1/2$. Since p'' lies on the line between p and v_i we have:

$$|p''v_j| \leqslant \max\{|pv_j|, |v_iv_j|\} \leqslant (1+\varepsilon)\cdot|pv_j|.$$

Finally we get the claimed bound by noting that $|p'p''| \leqslant \varepsilon\cdot|p'v_i|$ (from $(*)$),

$$|p'v_j| \leqslant |p'p''| + |p''v_j| \leqslant \varepsilon\cdot|p'v_i| + (1+\varepsilon)\cdot|pv_j| \leqslant (1+2\varepsilon)\cdot|pv_j|. \quad \blacktriangleleft$$

An approximate decision algorithm. Given a point p, a positive real value ε and the voter multiset V, we say that an algorithm Alg is an *ε-approximate* decision algorithm if

- Alg answers yes if p is a β-plurality point, and
- Alg answers no if p is not a $(1-\varepsilon)\beta$-plurality point.

In the remaining cases, where $(1-\varepsilon)\beta < \beta(p,V) < \beta$, Alg may answer yes or no.

Next we propose an ε-approximate decision algorithm Alg. The algorithm will use the so-called Balanced Box-Decomposition (BBD) tree introduced by Arya and Mount [2]. BBD trees are hierarchical space-decomposition trees such that each node μ represents a region in $\mathbb{R}^d$, denoted by region(μ), which is a d-dimensional axis-aligned box or the difference of two such boxes. A BBD tree for a set P of n points in $\mathbb{R}^d$ can be built in $O(n\log n)$ time using $O(n)$ space. It supports $(1+\varepsilon)$-approximate range counting queries with convex query ranges in $O(\log n + \varepsilon^{1-d})$ time [2]. In our algorithm all query ranges will be balls, hence a $(1+\varepsilon)$-approximate range-counting query for a d-dimensional ball $s(v,r)$ with center at v and radius r returns an integer I such that $|P \cap s(v,r)| \leqslant I \leqslant |P \cap s(v,(1+\varepsilon)r)|$.

Our ε-approximate decision algorithm Alg works as follows.

1. Construct a set Q of $O(n/\varepsilon^{d-1})$ potential candidates competing against p, as follows. Let $Q(v)$ be a set of $O(1/\varepsilon^{d-1})$ points distributed uniformly on the boundary of the ball $s(v,(1-\varepsilon/2)\cdot\beta\cdot|pv|)$, such that the distance between any point on the boundary and its nearest neighbor in $Q(v)$ is at most $\frac{\varepsilon}{4\sqrt{d}}\cdot|pv| \leqslant \frac{\varepsilon}{4}\cdot\beta\cdot|pv|$. In the last step we use the fact that $\beta \geqslant 1/\sqrt{d}$, according to Lemma 2.3. Set $Q := Q(v_1)\cup\cdots\cup Q(v_n)$.
2. Build a BBD tree $\mathcal{T}$ on Q. Add a counter $c(\mu)$ to each node μ in $\mathcal{T}$, initialized to zero.
3. For each voter $v \in V$ perform a $(1+\varepsilon/4)$-approximate range-counting query with $s(v,(1-\varepsilon/4)\cdot\beta\cdot|pv|)$ in $\mathcal{T}$. We modify the search in $\mathcal{T}$ slightly as follows. If an internal node $\mu \in T$ is visited and expanded during the search, then for every non-expanded child μ' of μ with region(μ') entirely contained in $s(v,(1+\varepsilon/4)(1-\varepsilon/4)\cdot\beta\cdot|pv|)) \subset s(v,\beta\cdot|pv|)$ we increment the counter $c(\mu')$. Similarly, if a leaf is visited then the counter is incremented if the point stored in the leaf lies within $s(v,(1-\varepsilon/4)\cdot\beta\cdot|pv|)$.

4. For a leaf μ in $\mathcal{T}$, let $M(\mu)$ be the set of nodes in $\mathcal{T}$ on the path from the root to μ, and let $C(\mu) = \sum_{\mu' \in M(\mu)} c(\mu')$. Compute $C(\mu)$ for all leaves μ in $\mathcal{T}$ by a pre-order traversal of $\mathcal{T}$, and set $C \coloneqq \max_\mu C(\mu)$.
5. If $C \leqslant n/2$, then return YES, otherwise NO.

The proof of the following lemma can be found in the full version of the paper [1].

▶ **Lemma 3.3.** *Algorithm* ALG ε-approximately *decides if p is a β-plurality point in time* $O(\frac{n}{\varepsilon^{d-1}} \log \frac{n}{\varepsilon^{d-1}})$.

The algorithm. Now we have the tools required to approximate $\beta(V)$. First, generate the set P of $O(\frac{n}{\varepsilon^{2d-1}} \log \frac{1}{\varepsilon})$ candidate points. For each candidate point $p \in P$, perform a binary search for an approximate $\beta^*(p)$ in the interval $[1/\sqrt{d}, 1]$, until the remaining search interval has length at most $\varepsilon/2 \cdot 1/\sqrt{d}$. For each p and β^*, $(\varepsilon/2)$-approximately decide if p is a β^*-plurality point in V. Return the largest β^* and the corresponding point p on which the algorithm says YES.

▶ **Theorem 3.4.** *Given a multiset V of voters in $\mathbb{R}^d$, a $((1-\varepsilon)\cdot\beta(V))$-plurality point can be computed in* $O(\frac{n^2}{\varepsilon^{3d-2}} \cdot \log \frac{n}{\varepsilon^{d-1}} \cdot \log^2 \frac{1}{\varepsilon})$.

4 Concluding Remarks

We proved that any finite set of voters in $\mathbb{R}^d$ admits a β-plurality point for $\beta = 1/\sqrt{d}$ and that some sets require $\beta = \sqrt{3}/2$. For $d = 2$ we managed to close the gap by showing that $\beta_2^* = \sqrt{3}/2$. One of the main open problems is to close the gap for $d > 2$. We also presented an approximation algorithm that finds, for a given V, a $(1-\varepsilon)\cdot\beta(V)$-plurality point. The algorithm runs in $O^*(n^2/\varepsilon^{3d-2})$ time. Another open problem is whether a subquadratic approximation algorithm exists, and to prove lower bounds on the time to compute $\beta(V)$ or $\beta(p, V)$ exactly. Finally, it will be interesting to study β-plurality points in other metrics, for instance in the personalized L_1-metric [5] for $d > 2$.

References

1 Boris Aronov, Mark de Berg, Joachim Gudmundsson, and Michael Horton. On β-plurality points in spatial voting games, 2020. arXiv:2003.07513.
2 Sunil Arya and David M. Mount. Approximate range searching. *Computational Geometry – Theory & Applications*, 17(3):135–152, 2000.
3 Saugata Basu, Richard Pollack, and Marie-Françoise Roy. *Algorithms in Real Algebraic Geometry (Algorithms and Computation in Mathematics)*. Springer-Verlag, Berlin, Heidelberg, 2006.
4 Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark Overmars. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, 3rd edition, 2008.
5 Mark De Berg, Joachim Gudmundsson, and Mehran Mehr. Faster algorithms for computing plurality points. *ACM Transactions on Algorithms*, 14(3), 2018.
6 Duncan Black. On the rationale of group decision-making. *Journal of Political Economy*, 56(1):23–34, 1948.
7 Jonathan Chung. Optimally locating a new candidate in spatial and valence models of voting games, 2018. Honours thesis, University of Sydney.
8 Richard Cole, Jeffrey S. Salowe, William L. Steiger, and Endre Szemerédi. An optimal-time algorithm for slope selection. *SIAM Journal on Computing*, 18(4):792–810, 1989.
9 Tamal K. Dey. Improved bounds for planar k-sets and related problems. *Discrete and Computational Geometry*, 19(3):373–382, 1998.

10 Anthony Downs. An Economic Theory of Political Action in a Democracy. *Journal of Political Economy*, 65(2):135–135, 1957.

11 Adrian Dumitrescu, János Pach, and Géza Tóth. Drawing Hamiltonian cycles with no large angles. *The Electronic Journal of Combinatorics*, 19(2):P31, 2012.

12 James Enelow and Melvin Hinisch. On Plott's pairwise symmetry condition for majority rule equilibrium. *Public Choice*, 40(3):317–321, 1983.

13 Haldun Evrenk. Valence politics. In Roger D. Congleton, Bernard Grofman, Stefan Voigt, and Haldun Evrenk, editors, *Oxford Handbook of Public Choice*, chapter 13. Oxford University Press, 2019.

14 Scott Feld, Bernard Grofman, and Nicholas Miller. Centripetal forces in spatial voting: On the size of the yolk. *Public Choice*, 59:37–50, October 1988.

15 Noah Giansiracusa and Cameron Ricciardi. Computational geometry and the U.S. supreme court. *Math. Soc. Sci.*, 98:1–9, 2019.

16 Fabian Gouret, Guillaume Hollard, and Stéphane Rossignol. An empirical analysis of valence in electoral competition. *Social Choice and Welfare*, 37(2):309–340, 2011.

17 Joachim Gudmundsson and Sampson Wong. Computing the yolk in spatial voting games without computing median lines. In *The 33rd AAAI Conference on Artificial Intelligence*, pages 2012–2019. AAAI Press, 2019.

18 Helios Herrera, David K. Levine, and César Martinelli. Policy platforms, campaign spending and voter participation. *Journal of Public Economics*, 92(3):501–513, 2008.

19 Guillaume Hollard and Stéphane Rossignol. An alternative approach to valence advantage in spatial competition. *Journal of Public Economic Theory*, 10(3):441–454, 2008.

20 Vladlen Koltun. Almost tight upper bounds for vertical decompositions in four dimensions. *Journal of the ACM*, 51(5):699–730, 2004.

21 Richard D. McKelvey. Covering, dominance, and institution-free properties of social choice. *American Journal of Political Science*, 30(2):283–314, 1986.

22 Nicholas R. Miller. The spatial model of social choice and voting. In Jac C. Heckelman and Nicholas R. Miller, editors, *Handbook of Social Choice and Voting*, chapter 10, pages 163–181. Edward Elgar Publishing, 2015.

23 Charles R. Plott. A notion of equilibrium and its possibility under majority rule. *The American Economic Review*, 57(4):787–806, 1967.

24 David Sanders, Harold D. Clarke, Marianne C. Stewart, and Paul Whiteley. Downs, stokes and the dynamics of electoral choice. *British Journal of Political Science*, 41(2):287–314, 2011.

25 Micha Sharir and Pankaj K. Agarwal. *Davenport-Schinzel sequences and their geometric applications*. Cambridge University Press, 1995.

26 Donald E. Stokes. Spatial models of party competition. *American Political Science Review*, 57(2):368–377, 1963.

27 Alan E. Wiseman. A theory of partisan support and entry deterrence in electoral competition. *Journal of Theoretical Politics*, 18(2):123–158, 2006.

28 Yen-Wei Wu, Wei-Yin Lin, Hung-Lung Wang, and Kun-Mao Chao. Computing plurality points and condorcet points in Euclidean space. In *Proceedings of the 24th International Symposium on Algorithms and Computation (ISAAC)*, volume 8283 of *Lecture Notes in Computer Science*, pages 688–698. Springer, 2013.

Testing Polynomials for Vanishing on Cartesian Products of Planar Point Sets

Boris Aronov
Department of Computer Science and Engineering, Tandon School of Engineering, New York University, Brooklyn, NY 11201, USA
boris.aronov@nyu.edu

Esther Ezra
School of Computer Science, Bar Ilan University, Ramat Gan, Israel
ezraest@cs.biu.ac.il

Micha Sharir
School of Computer Science, Tel Aviv University, Israel
michas@tau.ac.il

Abstract

We present subquadratic algorithms, in the algebraic decision-tree model of computation, for detecting whether there exists a triple of points, belonging to three respective sets A, B, and C of points in the plane, that satisfy a certain polynomial equation or two equations. The best known instance of such a problem is testing for the existence of a collinear triple of points in $A \times B \times C$, a classical 3SUM-hard problem that has so far defied any attempt to obtain a subquadratic solution, whether in the (uniform) real RAM model, or in the algebraic decision-tree model. While we are still unable to solve this problem, in full generality, in subquadratic time, we obtain such a solution, in the algebraic decision-tree model, that uses only roughly $O(n^{28/15})$ constant-degree polynomial sign tests, for the special case where two of the sets lie on one-dimensional curves and the third is placed arbitrarily in the plane. Our technique is fairly general, and applies to any other problem where we seek a triple that satisfies a single polynomial equation, e.g., determining whether $A \times B \times C$ contains a triple spanning a unit-area triangle.

This result extends recent work by Barba et al. [4] and by Chan [7], where all three sets A, B, and C are assumed to be one-dimensional. While there are common features in the high-level approaches, here and in [4], the actual analysis in this work becomes more involved and requires new methods and techniques, involving polynomial partitions and other related tools.

As a second application of our technique, we again have three n-point sets A, B, and C in the plane, and we want to determine whether there exists a triple $(a, b, c) \in A \times B \times C$ that simultaneously satisfies two real polynomial equations. For example, this is the setup when testing for the existence of pairs of similar triangles spanned by the input points, in various contexts discussed later in the paper. We show that problems of this kind can be solved with roughly $O(n^{24/13})$ constant-degree polynomial sign tests. These problems can be extended to higher dimensions in various ways, and we present subquadratic solutions to some of these extensions, in the algebraic decision-tree model.

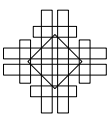
2012 ACM Subject Classification Theory of computation; Theory of computation → Computational geometry

Keywords and phrases Algebraic decision tree, Polynomial partition, Collinearity testing, 3SUM-hard problems, Polynomials vanishing on Cartesian products

Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.8

Related Version A full version of the paper is available at `https://arxiv.org/abs/2003.09533`.

Funding *Boris Aronov*: Partially supported by NSF grant CCF-15-40656 and by grant 2014/170 from the US-Israel Binational Science Foundation.
Esther Ezra: Partially supported by NSF CAREER under grant CCF:AF-1553354 and by Grant 824/17 from the Israel Science Foundation.



Micha Sharir: Partially supported by ISF Grant 260/18, by grant 1367/2016 from the German-Israeli Science Foundation (GIF), and by Blavatnik Research Fund in Computer Science at Tel Aviv University.

Acknowledgements The authors wish to thank Adam Sheffer and Frank de Zeeuw for suggesting the transformation used for collinearity testing in Theorem 6.1. We also thank Jean Cardinal, John Iacono, Stefan Langerman, and Aurélien Ooms for useful discussions.

1 Introduction

General background. Let A, B, and C be three n-point sets in the plane. We want to determine whether there exists a triple of points $(a, b, c) \in A \times B \times C$ that satisfy one or two prescribed polynomial equations. An example of such a scenario, with a single vanishing polynomial, is to determine whether $A \times B \times C$ contains a collinear triple of points. This classical problem is at least as hard as the 3SUM problem [14], in which we are given three sets A, B, and C, each consisting of n real numbers, and we want to determine whether there exists a triple of numbers $(a, b, c) \in A \times B \times C$ that add up to zero.

The 3SUM problem itself, conjectured for a long time to require $\Omega(n^2)$ time, has recently been shown by Grønlund and Pettie [16] (with further improvements by Chan [7]) to be solvable in very slightly subquadratic time. Moreover, in the *linear decision-tree model*, in which we only count linear sign tests performed on the input points (and do not allow any other operation to access the input explicitly), Grønlund and Pettie have improved the running time to nearly $O(n^{3/2})$ (see also [13, 15] for subsequent slight speedups), which has been drastically further improved (still in the linear decision-tree model) to $O(n \log^2 n)$ time by Kane et al. [18].

In contrast, no subquadratic algorithm is known for the collinearity detection problem, either in the standard real RAM model (also known as the *uniform* model) or in the decision-tree model; see [4] for a discussion. In the uniform model, the problem can be solved in $O(n^2)$ time. The primitive operation needed to test for collinearity of a specific triple is the so-called *orientation test*, in which we test for the sign of a quadratic polynomial in the six coordinates of a triple of points in $A \times B \times C$ (see Eq. (1) below for the concrete expression). Consequently, it is natural (and apparently necessary) to use the more general *algebraic decision-tree model*, in which each comparison is a sign test of some constant-degree polynomial in the coordinates of a constant number of input points; see [6, 20] and below.

The problems, in more detail

In this paper we consider several variants of testing a polynomial, or polynomials, for vanishing on a triple Cartesian product. The main motivation for the present study is the aforementioned collinearity testing question. We present the problem in a wider context, where we are given three sets A, B, and C, each consisting of n points in the plane, and we consider two scenarios:

(a) A single vanishing polynomial. Given a single constant-degree irreducible 6-variate polynomial F, determine whether there exists a triple $(a, b, c) \in A \times B \times C$ such that $F(a, b, c) = 0$.

(b) A pair of vanishing polynomials. Given a pair F, G of constant-degree irreducible 4-variate polynomials, determine whether there exists a triple $(a, b, c = (c_1, c_2)) \in A \times B \times C$ such that $c_1 = F(a, b)$ and $c_2 = G(a, b)$.

We begin by studying the vanishing pair problem in (b), because our results are stronger for this setup, and show that, as can be expected, requiring the triple (a, b, c) to satisfy two equalities facilitates a more efficient solution. In contrast, the collinearity testing problem, as well as more general instances of a single vanishing polynomial in (a), seem harder to solve efficiently. As we spell out below, we can solve problems of the latter kind in subquadratic time, in the algebraic decision-tree model, only for restricted input sets.

We note that the vanishing pair problem in (b) is a special case of a more general question, in which F and G are 6-variate real polynomials, and the equations that we want to satisfy are $F(a, b, c) = G(a, b, c) = 0$. This general setting can also be handled by a more involved variant of the technique presented here, using standard tools from real algebraic elimination theory (as in [5]), but we will not consider this extension in the paper. (See also [4] for the treatment of this issue in a different, and simpler, context.)

A special (but natural) case of the problem with two polynomial constraints is where each of the sets A, B, C consists of n complex numbers, and we want to test the vanishing of a *single* constant-degree bivariate polynomial $H : \mathbb{C}^2 \to \mathbb{C}$ defined over the complex numbers; this is an extension of the problem studied by Barba et al. [4] over the reals. That is, the problem is to determine whether there is a triple $(a, b, c) \in A \times B \times C$ such that[1] $c = H(a, b)$. Two concrete instances of this question, involving testing for the existence of similar triangles that are determined by A, B, and C, will be used to present our technique.

Comments on the purely one-dimensional setup. Questions of the type studied here are (extensions to higher dimensions of) the algorithmic counterparts of the classical problems in combinatorial geometry, studied by Elekes and Rónyai [10] and by Elekes and Szabó [11], and later improved in [22, 21]. We comment on this connection in some detail in the full version of the paper [3]. In the setup studied in [10, 11, 22, 21], all three sets A, B, C are one-dimensional. This purely one-dimensional setup has recently been studied by Barba et al. [4], both for the algebraic decision tree model and for the uniform model, whose algorithm in the algebraic decision tree model runs in close to $O(n^{12/7})$ time. The same approach, combined with more involved algorithmic techniques, yields an algorithm in the uniform model that runs in $O(n^2(\log\log n)^{3/2}/\log^{1/2} n)$ time, which has been slightly improved to $O(n^2(\log\log n)^{O(1)}/\log^2 n)$ by Chan [7].

Given this apparent (polynomial) hardness of computation, our goal is thus to obtain a significantly subquadratic solution in the *algebraic decision-tree model*. Here we only pay for sign tests that involve the input point coordinates, where each such test determines the sign of some real polynomial of constant degree in a constant number of variables. All other operations cost nothing in this model, and are assumed not to access the input explicitly. For example, each orientation test used in collinearity detection examines the sign of the determinant (a quadratic polynomial in $a_1, a_2, b_1, b_2, c_1, c_2$)

$$\begin{vmatrix} 1 & a_1 & a_2 \\ 1 & b_1 & b_2 \\ 1 & c_1 & c_2 \end{vmatrix}, \tag{1}$$

for some triple of points $(a_1, a_2) \in A$, $(b_1, b_2) \in B$, $(c_1, c_2) \in C$.

[1] Over the reals, H induces two polynomial equations, one for the real part and the other for the imaginary part, so this is indeed a special case of the polynomial vanishing pair problem. Here too one may consider the more general case where H is trivariate and we test for $H(a, b, c) = 0$.

Concrete problems in the two-dimensional setup. Each of the two general questions studied here (of one or two vanishing polynomials) arises in various concrete problems in computational geometry. For the case of a single vanishing polynomial, collinearity testing is a fairly famous (or should we say, notorious) example. Other problems include testing for the existence of a triangle Δabc, for $(a, b, c) \in A \times B \times C$, that has a given area, or perimeter, or circumscribing disk of a given radius, and so on.

We consider two simple instances of the vanishing polynomial pair problem. In the first one, we are given sets A, B, and C, each of n points in the plane, none of which contains the origin, and wish to determine whether there exists a triple $(a, b, c) \in A \times B \times C$ such that the triangle spanned by o (the origin), b and c is similar to the triangle spanned by o, $e_1 = (1, 0)$, and a (with e_1 corresponding to b and a to c). As a matter of fact, and as easily verified, this instance can also be interpreted as having three sets A, B, C of *complex* numbers, and the goal is to determine whether there exists a triple $(a, b, c) \in A \times B \times C$ such that $c = ab$. This is a single complex quadratic equation that (a, b, c) has to satisfy, which translates to two real quadratic equations in the coordinates of a, b, c, when treated as points in the real plane.

In the second instance, we are given sets A, B, and C, each of n points in the plane, and a fixed triangle $\Delta = \Delta uvw$, and we want to determine whether there exists a triple $(a, b, c) \in A \times B \times C$ that spans a triangle similar to Δ, with a corresponding to u, b to v, and c to w. This instance too can be interpreted as having three sets A, B, C of *complex* numbers, and the goal is to determine whether there exists a triple $(a, b, c) \in A \times B \times C$ that satisfies a certain single *linear* equation over complex numbers, determined by Δ; see the full version [3] for these details.

We note that the first instance can also be turned into an instance that involves a single complex *linear* equation, simply by replacing every $z \in A \cup B \cup C$ by $\ln z$. As it turns out, complex linear equations can be handled more efficiently (see below for details), but we use these examples as showcases of the more general technique that we develop.

In the full version of this paper [3] we show that both versions of the triangle similarity testing problem are 3SUM-hard.

▶ Remark 1.1. Since the underlying vanishing complex polynomial can be assumed to be *linear* (in both instances), the analysis in the very recent work of Aronov and Cardinal [2] implies that both problems can be reduced to the classical real-3SUM problem, via a random projection technique, and then solved, in the linear decision-tree model, in $O(n \log^2 n)$ time, using the technique of Kane et al. [18].

Our results. After setting up the technical machinery that our analysis requires, in Sections 2 and 3, we first consider, in Section 4, the problem of testing for a vanishing pair of polynomials, which includes the triangle similarity testing problems. We show that such problems can be solved, in the algebraic decision-tree model, with $O(n^{24/13+\varepsilon})$ polynomial sign tests, for any $\varepsilon > 0$ (with the constant of proportionality depending on ε), where each test involves a polynomial of constant degree in a constant number of variables, which in general might be more involved than just orientation tests. For the analysis, we need to assume that the pair of polynomials F, G have "good fibers" (which they do in the triangle similarity testing problems) – see Sections 2 and 4 for details.

We then consider, in Section 5, the problem of '$2 \times 1 \times 1$-dimensional' collinearity testing, meaning that A is an arbitrary set of points in the plane, but each of B and C lies on some respective constant-degree algebraic curve γ_B, γ_C. We show that this restricted problem can be solved in the algebraic decision-tree model with $O(n^{28/15+\varepsilon})$ polynomial sign tests, for any

$\varepsilon > 0$ (where again the constant of proportionality depends on ε). The technique extends naturally to similar problems involving a single vanishing polynomial, such as determining whether $A \times B \times C$ spans a unit-area triangle.

We still do not have a subquadratic solution, even in the algebraic decision-tree model, to the unconstrained (referred to as $2 \times 2 \times 2$-dimensional) collinearity testing problem, or even for the more restricted $2 \times 2 \times 1$-dimensional scenario, where only C is constrained to lie on a given curve. The techniques that we use for the $2 \times 1 \times 1$ version can be extended to the general unconstrained (or less constrained) case, but they actually result in a *superquadratic* algorithm; see the full version of the paper for more details. As shown by Erickson and Seidel [12], if the *only* sign tests that we allow in the decision tree are orientation tests, then $\Omega(n^2)$ tests are needed in the worst case. The solution presented here uses other sign tests, making it more powerful (and more efficient).

We also consider, in the full version, extensions of both problems to higher dimensions. Specifically, we study collinearity testing in d dimensions, where we assume that each of B and C lies on a hyperplane. Our solution is based on projections of the input onto lower-dimensional subspaces, and achieves the same asymptotic performance as in the plane. This result is presented in Section 6. More general extensions to higher dimensions are discussed in the appendix of the full version [3].

▶ Remark 1.2. We are not aware of a simple extension of the analysis in the earlier work of Barba et al. [4] or of Chan [7] to the problems studied in this paper. A main technique in our arsenal is to consider the Cartesian product of polynomial partitionings, which we believe to be essential, mainly for the triangle similarity problems and their higher-dimensional extensions, as well as to higher-dimensional extensions of collinearity testing.

The $2 \times 1 \times 1$ case of problems involving a single vanishing polynomial, considered in Section 5, has an alternative subquadratic, albeit less efficient, solution, using simpler considerations, which somewhat resemble the analysis in [4]. We sketch this alternative technique in the full version of the paper [3]

We also comment that Chan [7] addresses several related geometric 3SUM-hard problems, among which is a variant of dual collinearity testing: Given three sets A, B, and C of line segments in the plane, where the segments in A are pairwise disjoint, and so are the segments in B, decide whether there exist a triple of segments in $A \times B \times C$ that meet at a common point. Although Chan's technique results in a slightly subquadratic algorithm in the RAM model, and is also claimed to yield a truly subquadratic algorithm in the algebraic decision-tree model, the disjointness assumptions significantly restrict the problem, so, to quote [7], "it remains open whether there is a subquadratic algorithm for the degeneracy testing for n lines in $\mathbb{R}^2$."

2 Preliminaries

Our analysis relies on planar polynomial partitioning and on properties of Cartesian products of pairs of them. For a polynomial $f : \mathbb{R}^d \to \mathbb{R}$, for any $d \geq 2$, the *zero set* of f is $Z(f) := \{x \in \mathbb{R}^d \mid f(x) = 0\}$. We refer to an open connected component of $\mathbb{R}^d \setminus Z(f)$ as a *cell*. The classical Guth-Katz result is:

▶ **Proposition 2.1** (Polynomial partitioning; Guth and Katz [17]). *Let P be a finite set of points in $\mathbb{R}^d$, for any $d \geq 2$. For any real parameter D with $1 \leq D \leq |P|^{1/d}$, there exists a real d-variate polynomial f of degree $O(D)$ such that $\mathbb{R}^d \setminus Z(f)$ has $O(D^d)$ cells, each containing at most $|P|/D^d$ points of P.*

Agarwal, Matoušek, and Sharir [1] presented an algorithm that efficiently computes[2] such a polynomial f, whose expected running time is $O(nr + r^3)$, where $r = D^d$.

Note that the number of points of P on $Z(f)$ can be arbitrarily large. For planar polynomial partitions, though, this can be handled fairly easily, by partitioning the algebraic curve $Z(f)$ into subarcs, each containing at most $|P|/D^2$ points (as do the complementary cells). We state this property formally and spell out the easy details in the full version of the paper.

Polynomial partitioning for Cartesian products of point sets in the plane. Solymosi and De Zeeuw [23] studied polynomial partitioning for Cartesian products of planar point sets. Given two finite sets P_1 and P_2 of points in the plane, a natural strategy to construct a partitioning polynomial for $P_1 \times P_2 \subset \mathbb{R}^2 \times \mathbb{R}^2$, a space that we simply regard as $\mathbb{R}^4$, is to construct suitable bivariate partitioning polynomials φ_1 for P_1 and φ_2 for P_2, as provided in Proposition 2.1, and then take their product $\varphi(x, y, z, w) := \varphi_1(x, y)\varphi_2(z, w)$.

▶ **Corollary 2.2** (Polynomial partitioning of Cartesian product [23]). *The partition of $P_{1,2} := P_1 \times P_2$ just described results in overall $O(D^4)$ relatively open cells of dimensions 2, 3, and 4, each of which contains at most $|P_{1,2}|/D^4$ points of $P_{1,2}$. The zero- and one-dimensional cells do not contain any point of $P_{1,2}$.*

The analysis in [23] also bounds the number of partition cells intersected by a two-dimensional algebraic surface S in $\mathbb{R}^4$, provided it has "good fibers." We define this notion:

▶ **Definition 2.3** (Good fibers). *(i) A two-dimensional algebraic surface S in $\mathbb{R}^4$ has* good fibers *if, for every point $p \in \mathbb{R}^2$, the fibers $(\{p\} \times \mathbb{R}^2) \cap S$ and $(\mathbb{R}^2 \times \{p\}) \cap S$ are finite. (ii) A two-dimensional algebraic surface S in $\mathbb{R}^3$ has* good fibers *if, for every point $p \in \mathbb{R}^2$, except for $O(1)$ exceptional points, the fiber $(\{p\} \times \mathbb{R}) \cap S$ is finite (it is one-dimensional for an exceptional point p), and for every point $q \in \mathbb{R}$, the fiber $(\mathbb{R}^2 \times \{q\}) \cap S$ is a one-dimensional variety (i.e., an algebraic curve).*

Note that in this definition we are only concerned with one specific decomposition of the underlying space into a product of two subspaces.

▶ **Proposition 2.4** (Cells intersected by a surface [23]). *Let S be a constant-degree two-dimensional algebraic surface in $\mathbb{R}^4$ that has good fibers. Then S intersects at most $O(D^2)$ two-, three-, and four-dimensional cells in the partitioning induced by $P_{1,2}$.*

Both Corollary 2.2 and Proposition 2.4 have three-dimensional counterparts (for Cartesian products of a plane and a curve), presented in the full version of this paper.

3 Hierarchical Polynomial Partitioning

Even though we work in the algebraic decision-tree model, we still need to account for the cost of constructing the various polynomial partitionings (as it requires explicit access to the input points), which, if done by a straightforward application of the technique of [1], would be too expensive, as a naïve implementation of our technique needs to use polynomials of high, non-constant degree. We circumvent this issue by constructing a *hierarchical polynomial partitioning*, akin to the constructions of hierarchical cuttings of Chazelle [8] and Matoušek [19] from the 1990s. The material is rather technical, and we only state the resulting theorems, giving details in the full version.

[2] This polynomial forms a partition approximating the one in Proposition 2.1, and the constant of proportionality in the degree bound of [1] is slightly larger.

Roughly, we gain efficiency by constructing a hierarchical tree of partitions using constant-degree polynomials, until we reach subsets of the input point set of the desired size.

The actual hierarchical partitions that we will need are within a Cartesian product of either two planes or a plane and a one-dimensional curve, and are obtained by taking suitable Cartesian products of partitions constructed within each of these subspaces. We show that, up to n^ε factors, we achieve the same combinatorial properties as in a single-shot construction with a higher-degree polynomial, at a lower algorithmic cost. Specifically, we have the following results:

▶ **Theorem 3.1** (One set in the plane). *Let P be a set of n points in the plane, let $1 \le r \le n$ be an integer, and let $\varepsilon > 0$.*
(i) There is a hierarchical polynomial partition for P with $O((n/r)^{1+\varepsilon})$ bottom-level cells, each of which is associated *with at most r points of P which it contains. The hierarchy can be constructed in expected $O(n \log n)$ time.*
(ii) Any constant-degree algebraic curve γ reaches *at most $O((n/r)^{1/2+\varepsilon})$ cells at all levels of the hierarchy.*[3] *These cells can be computed within the same asymptotic time bound.*

▶ **Theorem 3.2** (Cartesian product of two planar point sets). *Let P_1, P_2 be two sets of points in the plane, each of size n, put $P_{1,2} = P_1 \times P_2 \subset \mathbb{R}^4$. Let $1 \le r \le n$ be an integer and $\varepsilon > 0$.*
(i) There is a hierarchical polynomial partition for $P_{1,2}$ with $O((n/r)^{2+\varepsilon})$ bottom-level cells, each of which is associated with a subset of at most r^2 points of $P_{1,2}$, which is the Cartesian product of a set of at most r points from P_1 and a set of at most r points from P_2, which it contains. The hierarchy can be constructed in expected $O(n \log n)$ time.
(ii) Any constant-degree two-dimensional algebraic surface S with good fibers reaches (in the same sense as in Theorem 3.1) at most $O((n/r)^{1+2\varepsilon})$ cells at all levels of the hierarchical partition of $P_{1,2}$. These cells can be computed within the same asymptotic time bound.

▶ **Theorem 3.3** (Cartesian product of a planar point set and a 1D set). *Let P be a set of n points in the plane, and let Q be a set of n points lying on a constant-degree algebraic curve $\gamma \subset \mathbb{R}^2$. Let $1 \le r, s \le \sqrt{n}$ be real parameters.*
(i) There is a hierarchical polynomial partition for $P \times Q \subset \mathbb{R}^2 \times \gamma$ into $O(n^{2+\varepsilon}/(rs)^{1+\varepsilon})$ bottom-level cells, for any $\varepsilon > 0$, each of which is associated with a subset of at most rs points of $P \times Q$, which is the Cartesian product of a set of at most r points from P and a set of at most s points from Q. The hierarchy can be constructed in expected $O(n \log n)$ time.
(ii) Any constant-degree two-dimensional surface S with good fibers reaches (in the same sense as above) at most $O\left(\frac{n^{3/2+\varepsilon}}{r^{1/2+\varepsilon}s^{1+\varepsilon}}\right)$ cells at all levels of the hierarchical partition of $P \times Q$. These cells can be computed within the same asymptotic time bound.

4 Testing for a Vanishing Pair of Polynomials

In this section we study problems of type (b), where A, B, and C are three sets of n points in the plane, and we seek a triple $(a, b, c) \in A \times B \times C$ that satisfies two polynomial equations. To simplify the presentation, we assume that they are of the form $c_1 = F(a, b)$, $c_2 = G(a, b)$, for $c = (c_1, c_2)$, where F and G are constant-degree 4-variate polynomials with *good fibers*, in the following sense: For any pair of real numbers κ_1, κ_2, the two-dimensional surface $\pi_{(\kappa_1,\kappa_2)} := \{(a, b) \in \mathbb{R}^4 \mid F(a, b) = \kappa_1,\ G(a, b) = \kappa_2\}$ has good fibers.

We fix a parameter $g \ll n$ (whose value will be set later), and apply Theorem 3.2(i) to the sets A, B, with $r = g$, to construct, in expected $O(n \log n)$ time two hierarchical planar polynomial partitionings, one for A and one for B, and combine them to obtain, a

[3] A curve γ is said to *reach* a cell τ if it intersects τ and all its ancestral cells – see the full version.

hierarchical four-dimensional polynomial partitioning for $A \times B$, so that each bottom-level cell ζ contains a Cartesian product $A_\zeta \times B_\zeta$ with $|A_\zeta|, |B_\zeta| \leq g$, and the overall number of cells is $O((n/g)^{2+\varepsilon})$, for any prescribed $\varepsilon > 0$.

Let τ (resp., τ') be a bottom-level cell at the hierarchical partition of A (resp., of B). Put $A_\tau := A \cap \tau$ and $B_{\tau'} := B \cap \tau'$. The high-level idea of the algorithm is to sort lexicographically each of the sets $H_{\tau,\tau'} := \{(F(a,b), G(a,b)) \mid (a,b) \in A_\tau \times B_{\tau'}\}$, over all pairs of cells (τ, τ'). We then search with each $c = (c_1, c_2) \in C$ through the sorted lists of those sets $H_{\tau,\tau'}$ that might contain (c_1, c_2). We show that each $c \in C$ has to be searched for in only a small number of sets. As in all works on this type of problems, starting from [16], sorting the sets explicitly is too expensive. We overcome this issue by considering the problem in the algebraic decision-tree model, and by using an algebraic variant of *Fredman's trick*, extending those used in the previous algorithms for one-dimensional point sets [4, 16]. (Also, rather than carrying out the sorting in the lexicographical order, we do it in a primary round, in which we only sort the values of $F(a,b)$, followed by secondary rounds, in which we sort the values of $G(a,b)$, for each maximal block of equal values of F. For clarity of presentation, we only focus on F in the discussion below, while G is treated analogously and implicitly.)

Consider the step of sorting $\{F(a,b) \mid (a,b) \in A_\tau \times B_{\tau'}\}$. It has to perform various comparisons of pairs of values $F(a,b)$ and $F(a',b')$, for $a, a' \in A_\tau$, $b, b' \in B_{\tau'}$.

We consider $A_\tau \times A_\tau$ as a set of g^2 points in $\mathbb{R}^4$, and associate, with each pair $(b,b') \in B_{\tau'} \times B_{\tau'}$, the 3-surface $\sigma_{b,b'} = \{(a,a') \in \mathbb{R}^4 \mid F(a,b) = F(a',b')\}$. Let $\Sigma_{\tau'}$ denote the set of these surfaces. The arrangement $\mathcal{A}(\Sigma_{\tau'})$ has the property that each of its cells ζ (of any dimension) has a fixed sign pattern with respect to all these surfaces. That is, each comparison of $F(a,b)$ with $F(a',b')$, for any $(a,b), (a',b') \in A_\tau \times B_{\tau'}$, has a fixed outcome for all points $(a,a') \in \zeta$ (for a fixed pair b, b'). In other words, if we locate the points of $A_\tau \times A_\tau$ in $\mathcal{A}(\Sigma_{\tau'})$, we have available the outcome of all the comparisons needed to sort the set $\{F(a,b) \mid (a,b) \in A_\tau \times B_{\tau'}\}$.

Doing what has just been described is still too expensive (takes $\Omega(n^2)$ steps, in the algebraic decision-tree model) if implemented naïvely, processing each pair $\tau \times \tau'$ separately. We circumvent this issue, in the algebraic decision-tree model, by forming the unions $P := \bigcup_\tau A_\tau \times A_\tau$, and $\Sigma := \bigcup_{\tau'} \Sigma_{\tau'}$; we have $|P|,\ |\Sigma| = O(g^2 \cdot (n/g)^{1+\varepsilon}) = O(n^{1+\varepsilon} g^{1-\varepsilon})$. By locating each point of P in $\mathcal{A}(\Sigma)$, we get all the signs that are needed to sort all the sets $\{F(a,b) \mid (a,b) \in A_\tau \times B_{\tau'}\}$, over all pairs τ, τ' of cells, and the actual sorting costs nothing in our model, once the answers to all the relevant comparisons are known.

Searching with the points of C. We next search the structure with every $c = (c_1, c_2) \in C$. We only want to visit subproblems (τ, τ') where there might exist $a \in \tau$ and $b \in \tau'$, such that $F(a,b) = c_1$ and $G(a,b) = c_2$. To find these cells, and to bound their number, we consider the two-dimensional surface $\pi_{c=(c_1,c_2)} := \{(a,b) \in \mathbb{R}^4 \mid F(a,b) = c_1,\ G(a,b) = c_2\}$, and our goal is to enumerate the bottom-level cells $\tau \times \tau'$ in the hierarchical partition of $A \times B$ crossed by π_c. By assumption, π_c has good fibers, so, by Theorem 3.2(ii) (with $r = g$), we can find, in time $O((n/g)^{1+\varepsilon})$, the $O((n/g)^{1+\varepsilon})$ cells $\tau \times \tau'$ that π_c intersects.

Summing over all the n possible values of c, the number of crossings between the surfaces π_c and the cells $\tau \times \tau'$ is $O(n^{2+\varepsilon}/g^{1+\varepsilon})$, for any $\varepsilon > 0$. In other words, denoting by $n_{\tau,\tau'}$ the number of surfaces π_c that cross $\tau \times \tau'$, we have $\sum_{\tau,\tau'} n_{\tau,\tau'} = O(n^{2+\varepsilon}/g^{1+\varepsilon})$. Thus computing all such surface-cell crossings, over all $c \in C$, costs $O(n^{2+\varepsilon}/g^{1+\varepsilon})$ time. The cost of searching with any specific c, in the structure of a cell $\tau \times \tau'$ crossed by π_c, is $O(\log g)$ (it is simply a binary search over the sorted lists). Hence the overall cost of searching with the elements of C through the structure is (with a slightly larger ε) $O\left(\frac{n^{2+\varepsilon}}{g^{1+\varepsilon}}\right)$.

Preprocessing: Sorting the F and G values. In order to sort the F and G values, we follow a similar batched point location strategy as the one taken in [4]. That is, we perform $O(n^{1+\varepsilon}g^{1-\varepsilon})$ point location queries in an arrangement of $O(n^{1+\varepsilon}g^{1-\varepsilon})$ algebraic 3-surfaces of constant degree in $\mathbb{R}^4$. The output of this algorithm is a compact representation for the signs $F(a,b)-F(a',b')$ (where $a,a'\in A_\tau$, $b,b'\in B_{\tau'}$ over all pairs of cells τ, τ'), given as a disjoint union of complete bipartite graphs of the form $(P_\alpha\times\Sigma_\beta,\sigma)$, where $P_\alpha\subseteq P$, $\Sigma_\beta\subseteq\Sigma$, and $\sigma\in\{-1,0,1\}$ is the fixed sign of all points in P_α with respect to the 3-surfaces in Σ_β, where the sign of a point (a,a') with respect to a surface $\sigma_{b,b'}$ is positive (resp., zero, negative) if $F(a,b)>F(a',b')$ (resp., $F(a,b)=F(a',b')$, $F(a,b)<F(a',b')$). We show, in the following lemma, that the overall complexity of this representation, measured by the total size of the *vertex sets* of these graphs, as well as the time to construct it, is only $O\left((ng)^{8/5+\varepsilon}\right)$, where the $\varepsilon>0$ here is slightly larger than the prescribed ε. Interestingly, as the proof of the lemma (in the full version) shows, this bound also holds in the uniform model. (G is handled by similar means.)

▶ **Lemma 4.1.** *One can perform batched point location of the points of P within the arrangement of Σ, and obtain the above complete bipartite graph representation of the output, in $O\left((ng)^{8/5+\varepsilon}\right)$ randomized expected time in the uniform model, for any prescribed $\varepsilon>0$, where the constant of proportionality depends on ε and on the degree of F (and G).*

The overall algorithm. Combining the cost of this preprocessing stage with that of the construction of the hierarchical partitions for A and B, and of searching with the elements of C in the sorted order obtained (for free) from the complete bipartite graph representation, we get total expected running time of $O\left(n\log n+(ng)^{8/5+\varepsilon}+\frac{n^{2+\varepsilon}}{g^{1+\varepsilon}}\right)$. We now choose $g=n^{2/13}$, and obtain expected running time of $O\left(n^{24/13+\varepsilon}\right)$, where the implied constant of proportionality depends on the degrees of F and G and on ε, and the final ε is a (small) constant multiple of the initially prescribed ε. That is, we have shown:

▶ **Theorem 4.2.** *Let A, B, C be three n-point sets in the plane, and let F, G be a pair of constant-degree 4-variate polynomials with good fibers (in the sense defined at the beginning of this section). Then one can test, in the algebraic decision-tree model, whether there exists a triple $a\in A$, $b\in B$, $c=(c_1,c_2)\in C$, such that $c_1=F(a,b)$ and $c_2=G(a,b)$, using only $O\left(n^{24/13+\varepsilon}\right)$ polynomial sign tests (in expectation), for any $\varepsilon>0$.*

▶ **Corollary 4.3.** *Let A, B, C be three sets, each of n complex numbers, and let H be a constant-degree bivariate polynomial defined over the complex numbers, with good fibers.*[4] *Then one can determine, in the algebraic decision-tree model, whether there exists a triple $(a,b,c)\in A\times B\times C$ such that $c=H(a,b)$, with only $O\left(n^{24/13+\varepsilon}\right)$ real-polynomial sign tests, in expectation, for any $\varepsilon>0$.*

We can demonstrate this result on both instances of the triangle similarity testing problem, and show that, if A, B, C are n-point sets in the plane, so that the origin does not belong to $A\cup B\cup C$, then the following holds: One can determine, in the algebraic decision-tree model, whether there exists a triple $(a,b,c)\in A\times B\times C$ such that the triangle Δoe_1a is similar to the triangle Δobc, or that the triangle Δabc is similar to a given triangle Δuvw, with $O\left(n^{24/13+\varepsilon}\right)$ polynomial sign tests, in expectation, for any $\varepsilon>0$. (In the full version [3] we show that the good-fiber property holds in this setting.)

[4] That is, the real and the imaginary parts of H are a pair of constant-degree 4-variate polynomials so that the intersection of their zero sets is a two-dimensional surface with good fibers.

▶ Remark 4.4. In the full version we describe a direct extension of the technique reviewed in this section to a single vanishing polynomial (which is different from the technique presented in the next section), and show that it results in a super-quadratic bound. In particular, this applies to the unconstrained collinearity testing problem in the plane.

5 Collinearity Testing and Related Problems: The Case of $2 \times 1 \times 1$ Dimensions

Let A, B, and C be three sets of points in the plane, but assume that B and C lie on respective constant-degree algebraic curves γ_B and γ_C. Our goal is to determine whether there exists a collinear triple of points in $A \times B \times C$, or more generally a triple (a, b, c) satisfying some prescribed constant-degree polynomial equation[5] $F(a, b, c) = 0$. To make the exposition easier to follow, we mostly focus on the collinearity testing problem.

Further simplifying, we assume that γ_B and γ_C are given in the parametric forms $\gamma_B(t) = (x_B(t), y_B(t))$ and $\gamma_C(s) = (x_C(s), y_C(s)))$, for $t, s \in \mathbb{R}$, where $x_B(t)$, $y_B(t)$, $x_C(s)$, $y_C(s)$ are constant-degree continuous algebraic functions, and that the sets A, B, and C are pairwise disjoint, for otherwise collinear triples exist trivially; the latter condition can be checked efficiently. A triple $(a = (a_1, a_2),\ b = (x_B(t), y_B(t)),\ c = (x_C(s), y_C(s)))$ is collinear if and only if

$$\begin{vmatrix} 1 & a_1 & a_2 \\ 1 & x_B(t) & y_B(t) \\ 1 & x_C(s) & y_C(s) \end{vmatrix} = 0, \text{ or}$$

$$x_B(t)y_C(s) - y_B(t)x_C(s) - a_1(y_C(s) - y_B(t)) + a_2(x_C(s) - x_B(t)) = 0.$$

While the theory can be developed for this general setting, we only consider here the special case where γ_C is a line, say the x-axis, so $\gamma_C(s) = (s, 0)$, and the last equality becomes:

$$-y_B(t)s + a_1 y_B(t) + a_2(s - x_B(t)) = 0 \quad \text{or} \quad s = \varphi(a, t) := \frac{a_1 y_B(t) - a_2 x_B(t)}{y_B(t) - a_2}.$$

Here φ is a constant-degree algebraic function; it is a linear rational function in a.

We fix a pair of parameters $g, h \ll n$ (whose values will be set later) and a parameter $\varepsilon > 0$, and apply Theorem 3.3(i) to the sets A, B, with the respective parameters $r = g, s = h$. Let τ (resp., τ') be a bottom-level cell in the resulting partition for A (resp., B). Put $A_\tau := A \cap \tau$ and $B_{\tau'} := B \cap \tau'$. In this analysis, somewhat abusing the notation, we regard B as a subset of $\mathbb{R}$, and denote by t the real parameter that parameterizes γ_B; in particular, we write $t \in B$ (resp., $t \in B_{\tau'}$) instead of $\gamma_B(t) \in B$ (resp., $\gamma_B(t) \in B_{\tau'}$). The number of bottom-level cells τ (and sets A_τ) is $O\left(\left(\frac{n}{g}\right)^{1+\varepsilon}\right)$, for any $\varepsilon > 0$, and the number of bottom-level cells τ' (and sets $B_{\tau'}$) is n/h.

The high-level idea of the algorithm is to sort each of the sets $\{\varphi(a, t) \mid (a, t) \in A_\tau \times B_{\tau'}\}$, over all pairs (τ, τ') of cells, and then to search with each $c = (s, 0) \in C$ (i.e., with the corresponding real s) through the sorted lists of only those sets that might contain s; this number is small, as argued below.

Again, sorting the sets explicitly is too expensive, and we use an instance of the algebraic variant of Fredman's trick, as in the previous section.

[5] Note that here F is (naturally) implicit, as opposed to the preceding section, where we used two explicit expressions $c_1 = F(a, b)$, $c_2 = G(a, b)$.

Preprocessing for batched point location. Consider the step of sorting $\{\varphi(a,t) \mid (a,t) \in A_\tau \times B_{\tau'}\}$, which has to perform various comparisons of pairs of values $\varphi(a,t)$ and $\varphi(a',t')$, for $a, a' \in A_\tau$, $t, t' \in B_{\tau'}$. We perform this task globally over all pairs (τ, τ') of cells.

We recurse by switching between a "primal" and a "dual" setups. In the primal, we view $P = \bigcup_\tau A_\tau \times A_\tau$ as a set of $O\left(\left(\frac{n}{g}\right)^{1+\varepsilon} \cdot g^2\right) = O(n^{1+\varepsilon} g^{1-\varepsilon})$ points in $\mathbb{R}^4$, and associate with each pair $(t,t') \in B_{\tau'} \times B_{\tau'}$, for each cell τ', the three-dimensional constant-degree algebraic surface $\sigma_{t,t'} = \{(a,a') \in \mathbb{R}^4 \mid \varphi(a,t) = \varphi(a',t')\}$. We let Σ be the collection of all these surfaces, over all cells τ', and have $|\Sigma| = n/h \cdot h^2 = O(nh)$.

In the dual, we view the pairs $(t,t') \in \bigcup_{\tau'} B_{\tau'} \times B_{\tau'}$ as points in the plane, and associate with each pair $(a,a') \in P$ the curve $\delta_{a,a'} = \{(t,t') \in \mathbb{R}^2 \mid \varphi(a,t) = \varphi(a',t')\}$. In each primal problem we need to perform batched point-location queries in an arrangement of (some subset of the) constant-degree algebraic 3-surfaces $\sigma_{t,t'}$ in $\mathbb{R}^4$, and in each dual problem we need to perform batched point location queries in an arrangement of (some subset of the) constant-degree algebraic curves $\delta_{a,a'}$ in $\mathbb{R}^2$. Initially we are in the primal, with $O(n^{1+\varepsilon} g^{1-\varepsilon})$ points and $O(nh)$ 3-surfaces.

If we could construct the full arrangement $\mathcal{A}$ of these surfaces and locate in it all these points, we would get the signs of all the differences $\varphi(a,t) - \varphi(a',t')$, for all (a,t), $(a',t') \in A_\tau \times B_{\tau'}$, over all pairs (τ,τ') of cells, from which we would get (for free) the sorted order of the sets $\{\varphi(a,t) \mid (a,t) \in A_\tau \times B_{\tau'}\}$, over all pairs (τ,τ'). However, a single-step construction of $\mathcal{A}$ is too expensive, so we replace it with the above "flip-flop" primal-dual processing, each time partitioning the (current version of the) arrangement using a polynomial of small degree, and thereby reduce the cost to that stated below.

The output of this preprocessing is a representation of $P \times \Sigma$ as a disjoint union of complete bipartite graphs, described in the following lemma (see the full version [3]):

▶ **Lemma 5.1.** *The above recursive batched point-location stage takes randomized expected time $O\left(n^{10/7+\varepsilon'} g^{6/7+\varepsilon'} h^{4/7}\right)$, also in the uniform model, where ε' is larger, by a small constant factor, than the prescribed ε.*

Searching with the points of C. We next search the structure with every $s \in C$ (identified with the point $(s,0)$ on the x-axis). For each $s \in C$, we only want to visit subproblems (τ,τ') where there might exist $a \in \tau$ and $t \in \tau'$ (not necessarily from $A_\tau \times B_{\tau'}$), such that $\varphi(a,t) = s$. We consider the two-dimensional surface $\pi_s := \{(a,t) \in \mathbb{R}^3 \mid \varphi(a,t) = s\}$ and show that it has good fibers (details appear in the full version [3]).

We next proceed as follows. By Theorem 3.3(ii), choosing g and h to satisfy $\left(\frac{n}{g}\right)^{1/2} = \frac{n}{h}$, or $h = n^{1/2} g^{1/2}$, we ensure that π_s reaches $O\left(\frac{n^{1+\varepsilon}}{g^{1+\varepsilon}}\right)$ cells $\tau \times \tau'$. Summing over all the n possible values of s, the number of crossings between the surfaces π_s and the cells $\tau \times \tau'$ is $O\left(\frac{n^{2+\varepsilon}}{g^{1+\varepsilon}}\right)$. Denoting by $n_{\tau,\tau'}$ the number of surfaces π_s that cross $\tau \times \tau'$, we have $\sum_{\tau,\tau'} n_{\tau,\tau'} = O\left(\frac{n^{2+\varepsilon}}{g^{1+\varepsilon}}\right)$ and we can enumerate all such crossings in $O(n^{2+\varepsilon}/g^{1+\varepsilon})$ time.

The cost of searching with any specific s in the structure of a cell $\tau \times \tau'$ crossed by π_s, is $O(\log g)$. Hence the overall cost of searching with the elements of C through the structure is $O(n^{2+\varepsilon}/g^{1+\varepsilon})$, where ε is slightly larger than the originally prescribed one.

Combining this cost with that of the construction of the hierarchical polynomial partitioning, and the point-location preprocessing stage, we get overall expected time of

$$O\left(n\log n + n^{10/7+\varepsilon}g^{6/7+\varepsilon}h^{4/7} + \frac{n^{2+\varepsilon}}{g^{1+\varepsilon}}\right) = O\left(n\log n + n^{12/7+\varepsilon}g^{8/7+\varepsilon} + \frac{n^{2+\varepsilon}}{g^{1+\varepsilon}}\right).$$

We roughly balance the two last terms by choosing $g = n^{2/15}$, making the overall cost of the procedure $O\left(\frac{n^{2+\varepsilon}}{g^{1+\varepsilon}}\right) = O\left(n^{28/15+\varepsilon}\right)$.

A similar analysis, albeit somewhat more complicated, can handle the case where C is contained in a general constant-degree algebraic curve, rather than a line.[6] In summary, we thus obtain (see the full version for more details [3]):

▶ **Theorem 5.2.** *Let A, B, C be n-point sets in the plane, where B and C are each contained in some respective constant-degree algebraic curve γ_B, γ_C, and assume $B \cap \gamma_C = \emptyset$. Then one can test whether $A \times B \times C$ contains a collinear triple, in the algebraic decision-tree model, using only $O\left(n^{28/15+\varepsilon}\right)$ polynomial sign tests (in expectation), for any $\varepsilon > 0$, where the constant of proportionality depends on ε and on the degrees of γ_B, γ_C.*

Unit area triangles and other problems. This analysis can be extended to the *unit area triangle* problem, where we want to test for the existence of a triangle spanned by $A \times B \times C$ that has unit area, as well as to many other similar problems (e.g., does there exist a triangle spanned by $A \times B \times C$ that has circumradius 1, or inradius 1, or unit perimeter, and so on). All these variants can be solved with the same technique and within the same asymptotic performance bound as in Theorem 5.2, provided that the good-fiber property is satisfied.

6 Higher Dimensions

Collinearity in higher dimensions: The $d \times (d-1) \times (d-1)$ case. Let A, B and C be three sets of n points each, so that A is a set of points in $\mathbb{R}^d$ and each of B and C lies in a *hyperplane*. The goal is to test, in the algebraic decision tree model, whether $A \times B \times C$ contains a collinear triple. Our approach is to use a recursive chain of projections, which ultimately map the points in A, B, and C to some plane, so that B and C is each mapped to a set of points on some respective line, collinearity is preserved, and no new collinearity appears among the projected points. This is a variant of a projection technique described by De Zeeuw [9]. Deferring all the details to the full version of the paper, we obtain:

▶ **Theorem 6.1.** *Let A, B and C be three sets of n points each, where A is a set of points in $\mathbb{R}^d$ and B, C each lies in a respective hyperplane h_1, h_2. Assume that $h_1 \neq h_2$, $A \subset \mathbb{R}^d \setminus (h_1 \cup h_2)$ and $(B \cup C) \cap h_1 \cap h_2 = \emptyset$. Then one can test whether $A \times B \times C$ contains a collinear triple, in the algebraic decision tree model, by a randomized algorithm that succeeds with probability 1, and uses only $O\left(n^{28/15+\varepsilon}\right)$ polynomial sign tests (in expectation), for any $\varepsilon > 0$, where the constant of proportionality depends on ε and on d.*

We present in the full version [3] several initial results for more general extensions of both the single-polynomial and the polynomial-pair vanishing problems to higher dimensions. In the former setup, each of B and C is contained in an algebraic surface of codimension 1 and constant degree. Unlike the bound in Theorem 6.1, the bounds that we obtain deteriorate with d, but remain subquadratic for every d.

[6] For this extension one can apply *quantifier elimination* (see, e.g., [5]) and use a similar batched point-location mechanism, albeit with more complicated semi-algebraic sets (but still of constant complexity and same dimensionality).

References

1 Pankaj K. Agarwal, Jiří Matoušek, and Micha Sharir. On range searching with semialgebraic sets. II. *SIAM J. Comput.*, 42(6):2039–2062, 2013. doi:10.1137/120890855.

2 Boris Aronov and Jean Cardinal. Geometric pattern matching reduces to k-SUM, 2020. arXiv:abs/2003.11890.

3 Boris Aronov, Esther Ezra, and Micha Sharir. Testing polynomials for vanishing on Cartesian products of planar point sets, 2020. Full version of this paper. arXiv:2003.09533.

4 Luis Barba, Jean Cardinal, John Iacono, Stefan Langerman, Aurélien Ooms, and Noam Solomon. Subquadratic algorithms for algebraic 3SUM. *Discrete Comput. Geom.*, 61(4):698–734, 2019. doi:10.1007/s00454-018-0040-y.

5 Saugata Basu, Richard Pollack, and Marie-Françoise Roy. *Algorithms in Real Algebraic Geometry*, volume 10 of *Algorithms and Computation in Mathematics*. Springer-Verlag, Berlin, second edition, 2006.

6 Michael Ben-Or. Lower bounds for algebraic computation trees (preliminary report). In *Proc. of the 15th Annual ACM Symposium on Theory of Computing*, pages 80–86, 1983. doi:10.1145/800061.808735.

7 Timothy M. Chan. More logarithmic-factor speedups for 3SUM, (median, +)-convolution, and some geometric 3SUM-hard problems. *ACM Trans. Algorithms*, 16(1):7:1–7:23, 2020. doi:10.1145/3363541.

8 Bernard Chazelle. Cutting hyperplanes for divide-and-conquer. *Discrete Comput. Geom.*, 9:145–158, 1993. doi:10.1007/BF02189314.

9 Frank de Zeeuw. Ordinary lines in space, 2018. arXiv:1803.09524.

10 György Elekes and Lajos Rónyai. A combinatorial problem on polynomials and rational functions. *J. Comb. Theory, Ser. A*, 89(1):1–20, 2000. doi:10.1006/jcta.1999.2976.

11 György Elekes and Endre Szabó. How to find groups? (and how to use them in Erdős geometry?). *Combinatorica*, 32(5):537–571, 2012. doi:10.1007/s00493-012-2505-6.

12 Jeff Erickson and Raimund Seidel. Better lower bounds on detecting affine and spherical degeneracies. *Discrete Comput. Geom.*, 13:41–57, 1995. doi:10.1007/BF02574027.

13 Ari Freund. Improved subquadratic 3SUM. *Algorithmica*, 77(2):440–458, 2017. doi:10.1007/s00453-015-0079-6.

14 Anka Gajentaan and Mark H. Overmars. On a class of $O(n^2)$ problems in computational geometry. *Comput. Geom.*, 5:165–185, 1995. doi:10.1016/0925-7721(95)00022-2.

15 Omer Gold and Micha Sharir. Improved bounds for 3SUM, k-SUM, and linear degeneracy. In *25th Annual European Symposium on Algorithms*, pages 42:1–42:13, 2017. Also in arXiv:1512.05279. doi:10.4230/LIPIcs.ESA.2017.42.

16 Allan Grønlund and Seth Pettie. Threesomes, degenerates, and love triangles. *J. ACM*, 65(4):22:1–22:25, 2018. doi:10.1145/3185378.

17 Larry Guth and Nets Hawk Katz. On the Erdős distinct distances problem in the plane. *Ann. Math. (2)*, 181(1):155–190, 2015. doi:10.4007/annals.2015.181.1.2.

18 Daniel M. Kane, Shachar Lovett, and Shay Moran. Near-optimal linear decision trees for k-SUM and related problems. *J. ACM*, 66(3):16:1–16:18, 2019. doi:10.1145/3285953.

19 Jiří Matoušek. Range searching with efficient hierarchical cuttings. *Discrete Comput. Geom.*, 10(2):157–182, 1993. doi:10.1007/BF02573972.

20 Franco P. Preparata and Michael Ian Shamos. *Computational Geometry—An Introduction*. Texts and Monographs in Computer Science. Springer, 1985. doi:10.1007/978-1-4612-1098-6.

21 Orit E. Raz, Micha Sharir, and Frank de Zeeuw. Polynomials vanishing on Cartesian products: the Elekes-Szabó theorem revisited. *Duke Math. J.*, 165(18):3517–3566, 2016 . doi:10.1215/00127094-3674103.

22 Orit E. Raz, Micha Sharir, and József Solymosi. Polynomials vanishing on grids: the Elekes-Rónyai problem revisited. *Amer. J. Math.*, 138(4):1029–1065, 2016. doi:10.1353/ajm.2016.0033.

23 József Solymosi and Frank de Zeeuw. Incidence bounds for complex algebraic curves on Cartesian products. In *New Trends in Intuitive Geometry*, volume 27 of *Bolyai Soc. Math. Stud.*, pages 385–405. János Bolyai Math. Soc., Budapest, 2018.

Extending Drawings of Graphs to Arrangements of Pseudolines

Alan Arroyo
IST Austria, Klosterneuburg, Austria
https://alanarroyo.github.io/
alanmarcelo.arroyoguevara@ist.ac.at

Julien Bensmail
Université Côte d'Azur, CNRS, Inria, I3S, Sophia-Antipolis, France
julien.bensmail.phd@gmail.com

R. Bruce Richter
Department of Combinatorics and Optimization, University of Waterloo, Canada
brichter@uwaterloo.ca

Abstract

In the recent study of crossing numbers, drawings of graphs that can be extended to an arrangement of pseudolines (pseudolinear drawings) have played an important role as they are a natural combinatorial extension of rectilinear (or straight-line) drawings. A characterization of the pseudolinear drawings of K_n was found recently. We extend this characterization to all graphs, by describing the set of minimal forbidden subdrawings for pseudolinear drawings. Our characterization also leads to a polynomial-time algorithm to recognize pseudolinear drawings and construct the pseudolines when it is possible.

2012 ACM Subject Classification Mathematics of computing → Graph algorithms; Mathematics of computing → Graphs and surfaces

Keywords and phrases graphs, graph drawings, geometric graph drawings, arrangements of pseudolines, crossing numbers, stretchability

Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.9

Related Version A full version of the paper is available at [4], https://arxiv.org/abs/1804.09317.

Funding *Alan Arroyo*: Supported by CONACYT. This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 754411.
Julien Bensmail: ERC Advanced Grant GRACOL, project no. 320812.
R. Bruce Richter: Supported by NSERC grant number 50503-10940-500.

1 Introduction

Since 2004, geometric methods have been used to make impressive progress for determining the crossing number of (certain classes of drawings of) the complete graph K_n. In particular, drawings that extend to straight lines, or, more generally, arrangements of pseudolines, have been central to this work, spurring interest in such drawings for arbitrary graphs, not just complete graphs [2, 5, 6, 7, 12].

In particular, for pseudolinear drawings, it is now known that, for $n \geq 10$, a pseudolinear drawing of K_n has more than

$$H(n) := \frac{1}{4}\left\lfloor \frac{n}{2} \right\rfloor \left\lfloor \frac{n-1}{2} \right\rfloor \left\lfloor \frac{n-2}{2} \right\rfloor \left\lfloor \frac{n-3}{2} \right\rfloor$$

crossings [1, 13]. The number $H(n)$ is conjectured by Harary and Hill to be the smallest number of crossings over all topological drawings of K_n; that is, the crossing number $\mathrm{cr}(K_n)$ is conjectured to be $H(n)$.

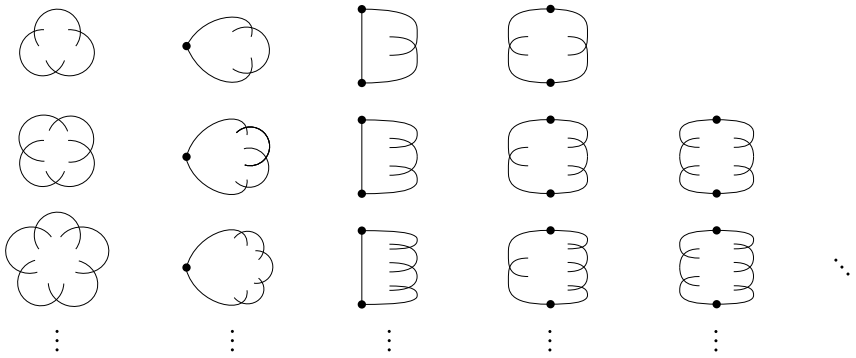


Figure 1 Obstructions to pseudolinearity.

A *pseudoline* is the image ℓ of a continuous injection from the real numbers $\mathbb{R}$ to the plane $\mathbb{R}^2$ such that $\mathbb{R}^2 \setminus \ell$ is not connected. An *arrangement of pseudolines* is a set Σ of pseudolines such that, if ℓ, ℓ' are distinct elements of Σ, then $|\ell \cap \ell'| = 1$ and the intersection is a crossing point. More on pseudolines and their importance for studying geometric drawings of graphs can be found in [10, 11].

A drawing D of a graph G is *pseudolinear* if there is an arrangment of pseudolines consisting of a different pseudoline ℓ_e for each edge e of G and such that $D[e] \subseteq \ell_e$.

In the study of crossing numbers, restricting the drawing to either straight lines or pseudolines yields the rectilinear crossing number $\overline{\text{cr}}(K_n)$ or the pseudolinear crossing number $\widetilde{\text{cr}}(K_n)$, respectively. Clearly $\overline{\text{cr}}(K_n) \geq \widetilde{\text{cr}}(K_n)$ and the geometric methods prove that $\widetilde{\text{cr}}(K_n) > H(n)$, for $n \geq 10$.

A *good drawing* is one where no edge self-intersects and any two edges share at most one point – either a crossing or a common end point – and no three edges share a common crossing. One somewhat surprising result is from Aichholzer et al.: *a good drawing of K_n in the plane is homeomorphic to a pseudolinear drawing if and only if it does not contain a non-planar drawing of K_4 whose crossing is incident with the unbounded face of the K_4* [2]. There are equivalent characterizations in [5, 6]. These conditions can be shown to be equivalent to not containing the *B-configuration* depicted as the third drawing of the first row of Figure 1.

Twenty-five years earlier, Thomassen proved a similar theorem for drawings in which each edge is crossed only once [16]. The B- and W-configurations are shown as the third and fourth drawings in the first row of Figure 1. Thomassen's theorem is: *if D is a planar drawing of a graph G in which each edge is crossed at most once, then D is homeomorphic to a rectilinear drawing of G if and only if D contains no B- or W-configuration.*

Thomassen presented in [16] the *clouds* (first column in Figure 1) as an infinite family of drawings that are minimally non-pseudolinear.

Shortly after Thomassen's paper, Bienstock and Dean proved that if $\text{cr}(G) \leq 3$, then $\overline{\text{cr}}(G) = \text{cr}(G)$ [8]. They also exhibited examples based on overlapping W-configurations to show the result fails for $\text{cr}(G) = 4$; such graphs can have arbitrarily large rectilinear crossing number.

Despite the existence of infinitely many obstructions to pseudolinearity, we characterize them all.

▶ **Theorem 1.** *A good drawing of a graph G is pseudolinear if and only if it does not contain one of the infinitely many obstructions shown in Figure 1.*

The drawings in Figure 1 are obtained from the *clouds* (first column) by replacing at most two crossings by vertices. The formal statement of Theorem 1 is Theorem 15 in Section 6; also a more general version of this statement, Theorem 2, is discussed below. That there is a result such as ours is somewhat surprising, because stretching an arrangement of pseudolines to a rectilinear drawing has been shown by Mnëv [14, 15] to be $\exists\mathbb{R}$-hard. In particular, recognizing a drawing as being homeomorphic to a rectilinear drawing is NP-hard.

The natural setting for our characterization is strings embedded in the plane. An *arc* σ is the image $f([0,1])$ of the compact interval $[0,1]$ under a continuous map $f : [0,1] \to \mathbb{R}^2$. Let $S(\sigma) = \{p \in \sigma \ : \ |f^{-1}(p)| \geq 2\}$ be the set of self-intersections of σ. A *string* is an arc σ for which $S(\sigma)$ is finite. If $S(\sigma) = \emptyset$, then σ is *simple*.

An intersection point between of two strings σ and σ' is *ordinary* if it is either an endpoint of σ or σ', or is a *crossing* (a crossing is a non-tangential intersection point in $\sigma \cap \sigma'$ that is not an end of σ or σ'). A set Σ of strings is *ordinary* if Σ is finite and any two strings in Σ have only finitely many intersections, all of which are ordinary. All the sets of strings considered in this paper are ordinary.

If Σ is an ordinary set of strings, then its *planarization* $G(\Sigma)$ is the plane graph obtained from Σ by inserting vertices at each crossing between strings and also at the endpoints of every string in Σ. To keep track of the information given by the strings, we will always assume that each string Σ has a different color and that each edge in $G(\Sigma)$ inherits the color of the string including it.

If Σ is an ordinary set of strings, then, for a cycle C in $G(\Sigma)$ (which is a simple closed curve in $\mathbb{R}^2$) and a vertex $v \in V(C)$, v is a *rainbow* for C if all the edges incident with v and drawn in the closed disk bounded by C (including the two edges of C at v) have different colours. The reader can verify that, for each drawing in Figure 1, if we let Σ be the edges of the drawing, then the unique cycle in $G(\Sigma)$ has at most two rainbows. Our main result characterizes these cycles as the only possible obstructions:

▶ **Theorem 2.** *An ordinary set of strings Σ can be extended to an arrangement of pseudolines if and only if every cycle C of $G(\Sigma)$ has at least three rainbows.*

Henceforth, we define any cycle C in $G(\Sigma)$ with at most two rainbows as an *obstruction*. A set of strings is *pseudolinear* if it has an extension to an arrangement of pseudolines.

Theorem 2 is our main contribution. In the next section, we show that the presence of an obstruction implies the set of ordinary strings is not pseudolinear. The converse is proved in Section 4 by extending, one small step at a time, the strings in Σ to get closer to an arrangement of pseudolines. After each extension, we must show that no obstruction has been introduced. This involves dealing with cycles in $G(\Sigma)$ that have precisely three rainbows (that we refer as *near-obstructions*). In Section 3 we show the key lemma that if G has two such near-obstructions that intersect nicely at a vertex v, then G has an obstruction. In Section 5 we present a polynomial-time algorithm for detecting obstructions and we argue why the proof of Theorem 2 implies a polynomial-time algorithm for extending a pseudolinear set of strings. Finally, in Section 6, we show how Theorem 1 follows from Theorem 2 and we present some concluding remarks.

2 A set of strings with an obstruction is not extendible

Let us start by showing the easy direction of Theorem 2:

▶ **Lemma 3.** *If the underlying graph $G(\Sigma)$ of a set Σ of strings has an obstruction, then Σ is not pseudolinear.*

Suppose that C is a cycle of $G(\Sigma)$ for some set of strings Σ. We define $\delta(C)$ as the set of vertices of C for which their two incident edges in C have different colours. In a set Σ of simple strings where no two intersect twice, $|\delta(C)| \geq 3$ for every cycle C of $G(\Sigma)$.

▶ **Lemma 4.** *Let Σ be a set of simple strings where every pair intersect at most once. Suppose that C is an obstruction with $|\delta(C)|$ as small as possible. Let $S = x_0, x_1, \ldots, x_\ell$ be a path of $G(\Sigma)$ representing a subsegment of some string $\sigma \in \Sigma$ such that $x_0x_1 \in E(C)$, $x_1 \in \delta(C)$ and x_1 is not a rainbow of C. Then $V(C) \cap V(S) = \{x_0, x_1\}$.*

Proof. By way of contradiction, suppose that there is a vertex $x_r \in V(C) \cap V(S)$ with $r \geq 3$. Assume that $r \geq 3$ is as small as possible. Let P be the subpath of S connecting x_1 to x_r. Since $x_0x_1 \in E(C)$ and $x_1 \in \delta(C)$ and $P \subseteq \sigma$, $x_1x_2 \notin E(C)$. Because x_1 is not a rainbow for C and no two strings tangentially intersect at x_1, the edge x_1x_2 is drawn in the closed disk bounded by C. By choice of r, P is an arc connecting x_1 to x_r in the interior of C.

Let C_1 and C_2 be the cycles obtained from the union of P and one of the two xy-subpaths in C. We may assume that $x_0x_1 \in E(C_1)$. Let $\rho(C)$ be either $\delta(C)$ or the set of rainbows in C. For $i = 1, 2$, let $Q_i = V(C_i) \setminus V(P)$. Then $\rho(C) \cap Q_i = \rho(C_i) \cap Q_i$. We see that $\rho(C_1) \setminus Q_1 \subseteq \{x_r\}$ and $\rho(C_2) \setminus Q_2 \subseteq \{x_1, x_r\}$.

For $\rho = \delta$, $|\delta(C_2)| \geq 3$, so $|\delta(C) \cap Q_2| \geq 1$. Since $x_1 \notin \delta(C_1)$, $|\delta(C_1)| \leq |\delta(C_1) \cap Q_2| + |\{x_r\}| \leq |\delta(C)| - 2 + |\{x_r\}| < |\delta(C)|$. Likewise, $|\delta(C) \cap Q_1| \geq 2$ and $x_1 \in \delta(C) \cap \delta(C_2)$. Therefore, $|\delta(C_2)| \leq |\delta(C)| - 2 + |\{x_r\}| < |\delta(C)|$. Thus, neither C_1 nor C_2 is an obstruction.

Now taking ρ to be the set of rainbows, the preceding paragraph shows $|\rho(C_1)| \geq 3$ and $|\rho(C_2)| \geq 3$. Therefore, $|\rho(C) \cap Q_1| = |\rho(C_1) \cap Q_1| \geq 2$ and $|\rho(C) \cap Q_2| = |\rho(C_2) \cap Q_2| \geq 1$. Thus, $|\rho(C)| \geq 3$, a contradiction. ◀

Proof of Lemma 3. By way of contradiction, suppose that Σ is pseudolinear and that $G(\Sigma)$ has an obstruction C.

Consider an extension of Σ to an arrangement of pseudolines, and then cut off the two infinite ends of each pseudoline to obtain a set of strings Σ' extending Σ, and in which every pair of strings in Σ' cross once. In $G(\Sigma')$, there is a cycle C' that represents the same simple closed curve as C. Because C' is obtained from subdividing some edges of C and the colours of a subdivided edge are the same, C' has fewer than three rainbows. Therefore, we may assume that $\Sigma = \Sigma'$ and $C = C'$. Now, the ends of every string in Σ are degree-1 vertices in the outer face of $G(\Sigma)$.

As every string in Σ is simple and no two strings intersect more than once, $|\delta(C)| \geq 3$. We will assume that C is chosen to minimize $|\delta(C)|$.

Since C is an obstruction, there exists $x_1 \in \delta(C)$ such that x_1 is not a rainbow in C. Consider a neighbour x_0 of x_1 in C. Let $S = x_0, x_1, \ldots x_\ell$ be the path obtained by traversing the string σ extending x_0x_1, such that x_ℓ is an end of σ. By Observation 4, $V(S) \cap V(C) = \{x_0, x_1\}$, and because x_ℓ is in the outer face of C, the segment of σ from x_1 to x_ℓ has its relative interior in the outer face of C.

However, since x_1 is not a rainbow, there exists a string $\sigma' \in \Sigma$ including two edges at x_1 drawn in the disk bounded by C. Thus, σ and σ' tangentially intersect at x_1, a contradiction. ◀

3 The key lemma

In this section we present the key lemma used in the proof of Theorem 2.

A plane graph G is *path-partitioned* if for $m \geq 1$, there exists a colouring $\chi : E(G) \to \{1, \ldots, m\}$ such that for each $i \in \{1, \ldots, m\}$, the edges in $\chi^{-1}(i)$ induce a path $P_i \subseteq G$ where any two distinct paths P_i and P_j do not tangentially intersect. Indeed, every underlying

planar graph $G(\Sigma)$ of a set of simple strings Σ is path-partitioned. Moreover, every path-partitioned plane graph can be obtained by subdividing a planarization of an ordinary set of simple strings. To extend the previously introduced notation we refer to each P_i as a string. The concepts of rainbow and obstruction naturally extend to the context of path-partitioned plane graphs.

Suppose that G is a path-partitioned plane graph. Given $v \in V(G)$, a *near-obstruction at* v is a cycle C with at most three rainbows and such that v is a rainbow of C. Understanding how near-obstructions behave is the key ingredient needed in the proof of Theorem 2:

▶ **Lemma 5.** *Let G be a path-partitioned plane graph and let $v \in V(G)$. Suppose that C_1 and C_2 are two near-obstructions at v such that the union of the closed disks bounded by C_1 and C_2 contains a small open ball centered at v. Suppose that one of the following two holds:*
1. *no obstruction of G contains v; or*
2. *the two edges of C_1 incident with v are the same as the two edges of C_2 incident with v.*
Then G has an obstruction not including v.

Given a plane graph G, a cycle $C \subseteq G$ and a vertex $v \in V(C)$, *the edges at v inside C* are the edges of G incident with v drawn in the disk bounded by C.

▶ **Useful Fact.** *Let G be planar path-partitioned graph. Suppose that for two cycles C and C', $v \in V(C) \cap V(C')$ is a vertex such that the edges at v inside C' are also edges at v inside C. If v is a rainbow for C, then v is a rainbow for C'.*

Proof of Lemma 5. By way of contradiction, suppose that G has no obstruction not including v. The "small ball" hypothesis implies that v is not in the outer face of the subgraph $C_1 \cup C_2$.

We claim that $|V(C_1) \cap V(C_2)| \geq 2$. Suppose not. Then C_1 and C_2 are edge-disjoint and $V(C_1) \cap V(C_2) = \{v\}$. For $i = 1, 2$, let e_i and f_i be the edges of C_i at v and let Δ_i be the closed disk bounded by C_i. From the "small ball" hypothesis it follows that (i) Δ_1 contains the edges e_2 and f_2; and (ii) the points near v in the exterior of Δ_2 are contained in Δ_1. These two properties imply that the path $C_2 - \{e_2, f_2\}$ intersects C_1 at least twice, and hence, $|V(C_1) \cap V(C_2)| \geq 2$.

From the last paragraph we know that $C_1 \cup C_2$ is 2-connected, and hence the outer face of $C_1 \cup C_2$ is bounded by a cycle C_{out}. We will assume that

(*) the cycles C_1 and C_2 satisfying the hypothesis of Lemma 5 are chosen so that the number of vertices of G in the disk bounded by C_{out} is minimal.

The Useful Fact applied to $C = C_{out}$ and to each $C' \in \{C_1, C_2\}$, shows that every vertex that is a rainbow in C_{out} is also a rainbow in each of the cycles in $\{C_1, C_2\}$ containing it. We can assume that C_{out} is not an obstruction or else we are done. We may relabel C_1 and C_2 so that two of the rainbows of C_{out}, say p and q, are also rainbows in C_1. Neither p nor q is v because $v \notin V(C_{out})$. Because C_1 is a near-obstruction, p, q and v are the only rainbows of C_1.

Since $v \notin V(C_{out})$, by following C_1 in the two directions starting at v, we find a path $P_v \subseteq C_1$ containing v in which only the ends u and w of P_v are in C_{out} (note that $u \neq v$ because $\{p, q\} \subseteq V(C_1) \cap V(C_{out})$). As v is in the interior face of C_{out}, P_v is also in the interior of C_{out}. Let Q^1_{out}, Q^2_{out} be the uw-paths of C_{out}. One of the two closed disks bounded by $P_v \cup Q^1_{out}$ and $P_v \cup Q^2_{out}$ contains C_1. By symmetry, we may assume that C_1 is contained in the first disk. Since $C_{out} \subseteq C_1 \cup C_2$, this implies that Q^2_{out} is a subpath of C_2.

Our desired contradiction will be to find three rainbows in C_2 distinct from v. We find the first: let $C_1 - (P_v)$ be the uw-path in C_1 distinct from P_v. The disk bounded by $(C_1 - (P_v)) \cup Q^2_{out}$ contains the one bounded by C_1. The Useful Fact applied to $C = (C_1 - (P_v)) \cup Q^2_{out}$ and $C' = C_1$ implies that each vertex in $C_1 - (P_v)$ that is rainbow in $(C_1 - (P_v)) \cup Q^2_{out}$ is also rainbow in C_1. Since C_1 has at most two rainbows in $C_1 - (P_v)$, namely p and q, $(C_1 - (P_v)) \cup Q^2_{out}$ has a third rainbow r_1 in the interior of Q^2_{out} (else $(C_1 - (P_v)) \cup Q^2_{out}$ is an obstruction and we are done). Note that r_1 is also a rainbow for C_2.

To find another rainbow in C_2, consider the edge e_u of C_2 incident to u and not in Q^2_{out}. We claim that either u is a rainbow in C_2 or that e_u is not included in the closed disk bounded by $P_v \cup Q^2_{out}$. Seeking a contradiction, suppose that u is not a rainbow of C_2 and that e_u is included in the disk. Then we can find two edges in the rotation at u, included in the disk bounded by $P_v \cup Q^2_{out}$, that belong to the same string σ. The vertex u is a rainbow in C_1, as else, we would find a string σ' with two edges inside $Q^1_{out} \cup P_v$, showing that σ and σ' tangentially intersect at u. As p and q are the only rainbows of C_1 in C_{out}, u is one of p and q. Therefore u is a rainbow in C_{out}, and hence, a rainbow in C_2, a contradiction.

If u is a rainbow in C_2, then this is the desired second one. Otherwise, e_u is not in the closed disk bounded by $P_v \cup Q^2_{out}$. Let $P_u \subseteq C_2$ be the path starting at u, continuing on e_u and ending on the first vertex u' in P_v that we encounter. Let C_u be the cycle consisting of P_u and the uu'-subpath uP_vu' of P_v.

▷ Claim 6. If P_u does not have a rainbow of C_u in its interior, then either C_u is an obstruction not containing v or:

(a) C_u and C_2 are near-obstructions at v satisfying the same conditions as C_1 and C_2 in Lemma 5; and

(b) the closed disk bounded by the outer cycle of $C_u \cup C_2$ contains fewer vertices than the disk bounded by C_{out}.

Proof. Suppose that all the rainbows of C_u are located in uP_vu'. If z is a rainbow of C_u, then $z \in \{u, v, u'\}$, as otherwise z is a rainbow of C_1 distinct from p, q and v, a contradiction. Thus, if $v \notin V(C_u)$, then C_u is the desired obstruction. We may assume that $v \in V(C_u)$.

If $u' = w$, then $C_2 = P_u \cup Q^2_{out}$, violating the assumption that $v \in V(C_2)$. Thus $u' \neq w$. If $u' = v$, then the rainbows of C_u are included in $\{u, u'\}$, and hence C_u is an obstruction. However, the existence of C_u shows that both alternatives (1) and (2) in Lemma 5 fail: condition (1) fails because C_u contains v and (2) fails because the edge of P_u incident with v is in $E(C_2) \setminus E(C_1)$. Thus $u' \neq v$.

The previous two paragraphs show that C_u is a near-obstruction at v with rainbows u, v and u'. Since the interior of C_u near v is the same as the interior of C_1 near v, the pair (C_u, C_2) satisfies the "small ball" hypothesis. Thus, (a) holds.

Let C'_{out} be the outer cycle of $C_u \cup C_2$. From the fact that $C_u \cup C_2 \subseteq C_1 \cup C_2$ it follows that the disk bounded by C_{out} includes the disk bounded by C'_{out}.

Since $p, q \in V(C_{out})$, p and q are in the disk bounded by C_{out}. If both p and q are in C_2, then p, q and r_1 are rainbows in C_2, and also distinct from v, contradicting that C_2 is a near-obstruction for v. If, say $p \notin V(C_2)$, then p is not in the disk bounded by C'_{out}, which implies (b). ◁

From Claim 6(b) and assumption (*) either C_u is the desired obstruction or P_u contains a rainbow r_2 of C_2 in its interior. We assume the latter as else we are done.

In the same way, the last rainbow r_3 comes by considering the edge of $C_2 - Q^2_{out}$ incident with w. It follows that v, r_1, r_2 and r_3 are four different rainbows in C_2, contradicting the fact that C_2 is a near-obstruction. ◀

4 Proof of Theorem 2

In this section we prove that a set of strings with no obstructions can be extended to an arrangement of pseudolines.

Proof of Theorem 2. It was shown in Observation 3 that the existence of obstructions implies non-extendibility. For the converse, suppose that Σ is a set of strings for which $G(\Sigma)$ has no obstructions.

We start by reducing to the case where the point set $\bigcup\Sigma$ is connected: iteratively add a new string in a face of $\bigcup\Sigma$ connecting two connected components of $\bigcup\Sigma$. No obstruction is introduced at each step (obstructions are cycles), and, eventually, the obtained set $\bigcup\Sigma$ is connected. An extension of the new set of strings contains an extension for the original set, thus we may assume that $\bigcup\Sigma$ is connected.

Our proof is algorithmic, and consists of repeatedly applying one of the three steps described below.

- **Disentangling Step.** If a string $\sigma \in \Sigma$ has an end a with degree at least 2 in $G(\Sigma)$, then we slightly extend the a-end of σ into one of the faces incident with a.
- **Face-Escaping Step.** If a string $\sigma \in \Sigma$ has an end a with degree 1 in $G(\Sigma)$, and is incident with an interior face, then we extend the a-end of σ until it intersects some point in the boundary of this face.
- **Exterior-Meeting Step.** Assuming that all the strings in Σ have their two ends in the outer face and these ends have degree 1 in $G(\Sigma)$, we extend the ends of two disjoint strings so that they meet in the outer face.

Each of these three steps either increases the number of pairs of strings that intersect, or increase the number crossings (recall that a crossing between σ and σ' is a non-tangential intersection point in $\sigma \cap \sigma'$ that is not an end of σ or σ'). Moreover, these steps can be performed as long as not all the strings have their ends in the outer face and they are pairwise crossing (in this case we extend their ends to infinity to obtain the desired arrangement of pseudolines). Henceforth, we will show that, if performed correctly, none of these steps introduces an obstruction. The proof for each step can be read independently.

▶ **Lemma 7** (Disentangling Step). *Suppose that $\sigma \in \Sigma$ has an end a with degree at least 2 in $G(\Sigma)$. Then we can extend the a-end of σ into one of the faces incident to a without creating an obstruction.*

Proof. A pair of different edges f and f' in $G(\Sigma)$ incident with a are *twins* if they belong to the same string in Σ. The edge $e \subseteq \sigma$ incident with a has no twin.

The fact that no pair of strings tangentially intersect at a tells us that if (f_1, f_1') and (f_2, f_2') are pairs of twins, then f_1, f_2, f_1', f_2' occur in this cyclic order for either the clockwise or counterclockwise rotation at a. Thus, we may assume that the counterclockwise rotation at a restricted to the twins and e is $e, f_1, \ldots, f_t, f_1', \ldots, f_t'$, where (f_i, f_i') is a twin pair for $i = 1, \ldots, t$.

To avoid tangential intersections, the extension of σ at a must be in the angle between f_t and f_1' not containing e. Let $e_1, \ldots, e_k$ be the counterclockwise ordered list of non-twin edges at a having an end in this angle (as depicted in Figure 2). We label $e_0 = f_t$ and $e_{k+1} = f_1'$. If there are no twins, then let $e_0 = e_{k+1} = e$.

Let us consider all the possible extensions: for $i \in \{0, \ldots, k\}$, let Σ_i be the set of strings obtained from Σ by slightly extending the a-end of σ into the face containing the angle between e_i and e_{i+1}. Let α_i be the new edge at a extending σ in Σ_i (see α_0 in Figure 2).

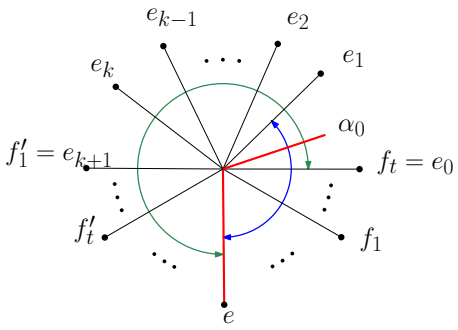


Figure 2 Substrings included in the disk bounded by C_0.

Seeking a contradiction, suppose that, for each $i \in \{0, ..., k\}$, $G(\Sigma_i)$ contains an obstruction C_i. Since α_i contains a degree-1 vertex, α_i is not in C_i. Hence C_i is a cycle of $G(\Sigma)$. Thus C_i is not an obstruction in $G(\Sigma)$ that becomes one in $G(\Sigma_i)$. This conversion has a simple explanation: in $G(\Sigma)$, C_i has exactly three rainbows, and one of them is a. After α_i is added, a is not a rainbow in C_i (witnessed by the edges e and α_i included in the new version of σ).

Recall from Section 3 that a *near-obstruction at* a is a cycle with exactly three rainbows, and one of them is a. Each of C_0, C_1,...,C_k is a near-obstruction at a in $G(\Sigma)$.

For a cycle $C \subseteq G$, let $\Delta(C)$ denote the closed disk bounded by C. Both e and α_0 are in $\Delta(C_0)$. Thus, either $\Delta(C_0) \supseteq \{e, f_1, f_2, \ldots, f_t, e_1\}$ (blue bidirectional arrow in Figure 2) or $\Delta(C_0) \supseteq \{f_t, e_1, \ldots, e_k, f'_1, f'_2, \ldots, f'_t, e\}$ (green bidirectional arrow). We rule out the latter situation as the second list contains f_t and f'_t, and this would imply that a is not a rainbow for C_0 in $G(\Sigma)$.

We just showed that $\{e, e_0, e_1\} \subseteq \Delta(C_0)$. By symmetry, $\{e_k, e_{k+1}, e\} \subseteq \Delta(C_k)$. Consider the largest index $i \in \{0, 1, \ldots, k-1\}$ for which $\{e, e_0, \ldots, e_{i+1}\} \subseteq \Delta(C_i)$. By the choice of i, and because $\{e, \alpha_{i+1}\} \subseteq \Delta(C_{i+1})$, $\{e, f'_t, \ldots, f'_1, e_k, \ldots, e_i\} \subseteq \Delta(C_{i+1})$. However, by applying Lemma 5 to the pair C_i and C_{i+1}, we obtain that $G(\Sigma)$ has an obstruction, a contradiction. ◀

▶ **Lemma 8** (Face-Escaping Step). *Suppose that there is a string σ that has an end a with degree 1 in $G(\Sigma)$, and a is incident to an interior face F. Then there is an extension σ' of σ from its a-end to a point in the boundary of F such that the set $(\Sigma \setminus \{\sigma\}) \cup \{\sigma'\}$ has no obstruction.*

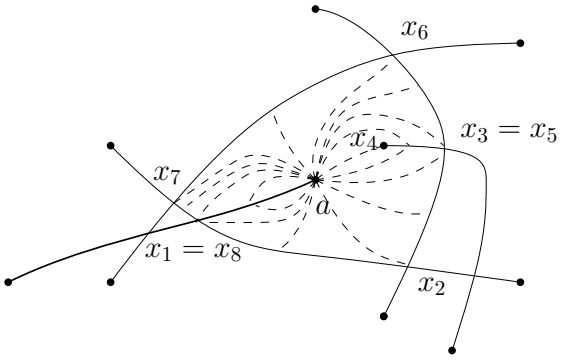


Figure 3 All possible extensions in the Face-Escaping Step.

Proof. Let W be the closed boundary walk $(x_0, e_1, x_1, e_2, \ldots, e_n, x_n)$ of F such that $x_0 = x_n = a$ and F is to the left as we traverse W (see Figure 3 for an illustration with $n = 9$). For $i = 1, \ldots, n$ we let m_i be a point in the relative interior of e_i, and let P be the list of non-necessarily distinct points m_1, x_1, m_2, $x_2 \ldots, m_n$, which are the potential ends for all the different extensions. For each $p \in P$, let Σ_p be the set of strings obtained from Σ by extending the a-end of σ by adding an arc α_p connecting a to p in F (see Figure 3). We assume that every two distinct arcs α_p and $\alpha_{p'}$ are internally disjoint.

Let f_p be the edge $e_1 \cup \alpha_p$ in $G(\Sigma_p)$; f_p has ends x_1 and p. Also, let $\sigma^p = \sigma \cup \alpha_p$. Seeking a contradiction, suppose that each $G(\Sigma_p)$ has an obstruction.

▷ Claim 9. Let $p \in P$. Then there exists an obstruction C_p in $G(\Sigma_p)$ including f_p. Moreover,
(1) if $p \in \sigma$, then C_p can be chosen so that all its edges are included in σ^p; and
(2) if $p \notin \sigma$, then every obstruction includes f_p.

Proof. First, if $p \in \sigma$, then the string σ^p self-intersects at p; thus σ^p has a simple close curve including f_p. In this case let C_p be the cycle in $G(\Sigma_p)$ representing this simple closed curve without rainbows, and thus (1) holds.

Second, assume that $p \notin \sigma$ and let C_p be any obstruction of $G(\Sigma_p)$. For (2), we will show that $f_p \in E(C_p)$.

Seeking a contradiction, suppose that $f_p \notin E(C_p)$.

If $p = m_i$ for $i \in \{1, \ldots, n\}$, since m_i is the only vertex whose rotation in $G(\Sigma)$ differs from its rotation in $G(\Sigma_{m_i})$, $m_i \in V(C_p)$. Consider the cycle C of $G(\Sigma)$ obtained from C_p by replacing the subpath (x_{i-1}, m_i, x_i) by the edge $x_{i-1}x_i$. For each vertex $v \in V(C)$ the colors of the edges of $G(\Sigma)$ at v included in the disk bounded by C are the same as in $G(\Sigma_p)$ for the disk bounded by $V(C_p)$. Thus, C is an obstruction for $G(\Sigma)$, a contradiction.

Suppose now that p is one of $x_1, \ldots, x_{n-1}$. The only vertex in $G(\Sigma)$ whose rotation is different in $G(\Sigma_p)$ is p. Therefore, p is a point that is a rainbow for C_p in $G(\Sigma)$, but not a rainbow in $G(\Sigma_p)$, witnessed by two edges included in σ^p. Since at least one of the two witnessing edges is in $G(\Sigma)$, $p \in \sigma$. This contradicts the assumption that $p \notin \sigma$. Hence $f_p \in E(C_p)$. ◁

Henceforth we assume that, for $p \in P$, C_p is an obstruction in $G(\Sigma_p)$ as in Claim 9.

More can be said about the obstructions in $G(\Sigma_p)$, but for this we need some terminology. If we orient an edge e in a plane graph, then the *sides* of e are either the points near e that are to the right of e, or the points near e to the left of e. For any cycle C of G through e, exactly one side of e lies inside C. This is the side of e *covered* by C. For the next claim and in the rest of the proof we will assume that for $p \in P$, f_p is oriented from x_1 to p.

▷ Claim 10. For $p \in P$ with $p \notin \sigma$, every obstruction in $G(\Sigma_p)$ covers the same side of f_p.

Proof. Suppose that for $p \in P$ there are obstructions C_p and C'_p covering both sides of f_p. Let G' be the plane graph obtained from $G(\Sigma_p)$ by subdividing f_p, and let v be the new degree-2 vertex inside f_p.

We consider the edge-colouring χ induced by the strings in Σ_p. Let χ' be a new colouring obtained from χ by replacing the colour of the edge vp by a new colour not used in χ. It is a routine exercise to verify that (i) χ' induces a path-partition in G' (defined in Section 3); and (ii) C_p and C'_p are near-obstructions for v with respect to χ'. By applying Lemma 5 to $C_1 = C_p$ and $C_2 = C'_p$, we obtain an obstruction in G' not containing v. However, this implies the existence of an obstruction in $G(\Sigma)$, a contradiction. ◁

Recall that the boundary walk of F is $W = (x_0, e_1, \ldots, e_n, x_n)$, with $x_0 = x_n = a$. Since x_1 and x_{n-1} are in σ, the extreme obstructions C_{x_1} and C_{x_2} cover the right of f_{x_1} and the left of $f_{x_{n-1}}$, respectively. Thus, there are two consecutive vertices x_{i-1}, x_i in $W - a$, such that the interior of $C_{x_{i-1}}$ covers the right of $f_{x_{i-1}}$ and the interior of C_{x_i} covers the left of f_{x_i}. Moreover, we may assume that the interior of C_{m_i} includes the left of f_{m_i} (otherwise we reflect our drawing).

The next claim (proved in the full version of this paper [4]) is the last ingredient to obtain a final contradiction.

▷ Claim 11. Exactly one of the following holds:

(a) $x_{i-1} \in \sigma$, $m_i \notin \sigma$ and $G(\Sigma_{m_i})$ has an obstruction covering the side of f_{m_i} not covered by C_{m_i}; or

(b) $x_{i-1} \notin \sigma$ and $G(\Sigma_{x_{i-1}})$ has an obstruction covering the side of $f_{x_{i-1}}$ not covered by $C_{x_{i-1}}$.

Claims 10 and 11 contradict each other. Thus, for some $p \in P$, $G(\Sigma_p)$ has no obstructions. ◀

▶ **Lemma 12** (Exterior-Meeting Step). *If all the strings in Σ have their ends on the outer face of $G(\Sigma)$ and the ends have degree 1 in $G(\Sigma)$, then we can extend a pair disjoint strings so that they intersect without creating an obstruction.*

Proof. By following the outer boundary of $\bigcup \Sigma$, we obtain a simple closed curve $\mathcal{O}$ containing all the ends of the strings in Σ, but otherwise disjoint from $\bigcup \Sigma$.

Suppose σ_1, σ_2 are two disjoint strings in Σ. For $i = 1, 2$, let a_i, b_i be the ends of σ_i; since σ_1 and σ_2 do not cross, we may assume that these ends occur in the cyclic order a_1, b_1, b_2, a_2. We extend the a_i-ends of σ_1 and σ_2 so that they meet in a point p in the outer face, and so that all the ends of σ_1 and σ_2 remain incident with the outer face (Figure 4). Let Σ' be the obtained set of strings.

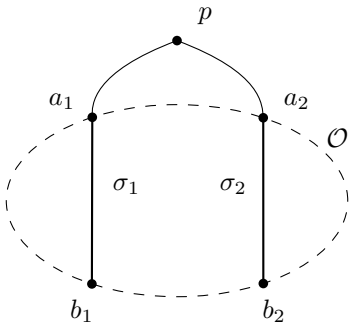


Figure 4 Exterior-Meeting Step.

Seeking a contradiction, suppose that $G(\Sigma')$ has an obstruction C. Since $G(\Sigma)$ has no obstruction, $p \in V(C)$. Our contradiction will be to find three rainbows in C. Note that p is a rainbow. To obtain a second rainbow, traverse C starting from p towards a_1. Let d_1 be the first vertex during our traversal that is not in the extended σ_1, and let c_1 be its neighbour in σ_1, one step before we reach d_1. Since b_1 has degree one, $c_1 \neq b_1$.

▷ Claim 13. The cycle C has a rainbow included in the closed disk Δ_1 bounded by σ_1 and the a_1b_1-arc of $\mathcal{O}$ disjoint from σ_2.

Proof. First, suppose that $d_1 \notin \Delta_1$. In this case, c_1 is a rainbow because otherwise there would be a string σ that tangentially intersects σ_1 at c_1. Thus, if $d_1 \notin \Delta_1$, then c_1 is the desired rainbow.

Second, suppose that $d_1 \in \Delta_1$. Let P_1 be the path of C starting at c_1, continuing on the edge c_1d_1, and ending at the first vertex we encounter in σ_1. Since the cycle C' enclosed by $P_1 \cup \sigma_1$ is not an obstruction, there is one rainbow of C' that is an interior vertex of P_1; this is the desired rainbow of C. This concludes the proof of Claim 13. ◁

Considering σ_2 instead of σ_1, Claim 13 yields a third rainbow in C inside an analogous disk Δ_2 disjoint from Δ_1, contradicting that C is an obstruction. Hence Lemma 12 holds. ◀

We iteratively apply the Disentangling Step, Face-Escaping Step or Exterior-Meeting Step without creating obstructions. Each step increases the number of pairwise intersecting strings in Σ until we reach a stage where the strings are pairwise intersecting and all of them have their two ends in the unbounded face. From this we extend them into an arrangement of pseudolines. This concludes the proof of Theorem 2. ◀

5 Finding obstructions and extending strings in polynomial time

We start this section by describing an algorithm to detect obstructions. Henceforth, we assume that the input to the problem is the planarization $G(\Sigma)$ of an ordinary set of s strings Σ. For the running-time analysis, we assume that n and m are the number of vertices and edges in $G(\Sigma)$, respectively. Since $G(\Sigma)$ is planar, $m = O(n)$. Moreover, if Σ is pseudolinear, then $n \leq \binom{s}{2} + 2s = \binom{s+2}{2} - 1$. At the end of this section we explain how to extend Σ (if possible) in polynomial time.

Recall that each string in Σ receives a different colour; this induces an edge-colouring on $G(\Sigma)$ where each string is a monochromatic path. An *outer-rainbow* is a vertex $x \in V(G(\Sigma))$ incident with the outer face and for which the edges incident with x have different colours. Next we describe the basic operation in our obstruction-detecting algorithm.

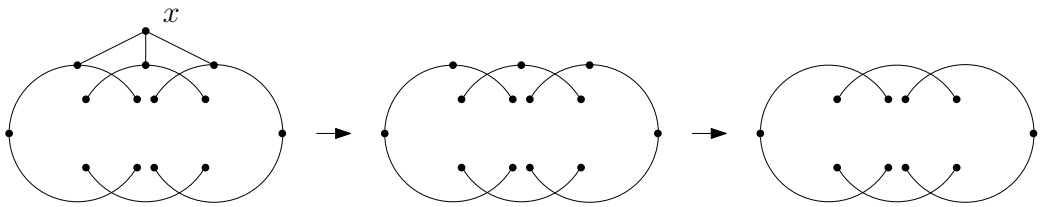


Figure 5 From Σ to $\Sigma - x$.

Outer-rainbow deletion. Given an outer-rainbow $x \in V(G(\Sigma))$, the instance $G(\Sigma - x)$ is defined by: first, removing x and the edges incident to x; second, suppressing the degree-2 vertices incident with edges of the same colour; and third, removing remaining degree-0 vertices (Figure 5 illustrates this process). Edge colours are preserved.

It is easy to verify that $G(\Sigma - x)$ is the planarization of an arrangement of strings. The colours removed by this operation are those belonging to strings that are paths of length 1 in $G(\Sigma)$ incident with x. Our obstruction-detecting algorithm relies on the following property:

(**) if x is an outer-rainbow of $G(\Sigma)$, then there is an obstruction in $G(\Sigma)$ not including x if and only if there is an obstruction in $G(\Sigma - x)$.

This property holds because cycles in $G(\Sigma) - x$ and in $G(\Sigma - x)$ are in 1-1 correspondence: two cycles correspond to each other if they are the same simple closed curve. This correspondence is obstruction-preserving.

Let us now describe the two subroutines in our algorithm. For this, we remark that an outer-rainbow of $G(\Sigma)$ is a rainbow for any cycle containing it.

Algorithm 1 Subroutine for detecting an obstruction through two outer-rainbows x and y.

(1) Find a cycle C through x and y whose edges are incident with the outer face of $G(\Sigma)$. If no such C exists, then output *No obstruction through x and y*. Else, go to Step 2.
(2) Find whether there is a third outer-rainbow $z \in V(C) \setminus \{x, y\}$. If such z exists, update $G(\Sigma) \longleftarrow G(\Sigma - z)$ and go to Step 1. If no such z exists, output C.

Correctness and running-time of Algorithm 1: If an obstruction through x and y exists, then x and y are in the same block (some authors use the term "biconnected component"). Since x and y are incident with the outer face, the outer boundary of the block containing x and y is the cycle C from Step 1. This C can be found by considering outer boundary walk W of $G(\Sigma)$ and then by finding whether x and y belong to the same non-edge block of W. Finding W is $O(m)$ and computing the blocks of W via a DFS takes $O(m)$ time.

In Step 2, if there is a third outer rainbow z in C, then no obstruction through x and y contains z. Property (**) justifies the update that takes $O(m)$ time.

A full run from Step 1 to Step 2 takes $O(m)$. Moving from Step 2 to Step 1 occurs $O(n)$ times. Thus, the total time for Algorithm 1 is $O(mn) = O(n^2)$.

Algorithm 2 Subroutine for detecting an obstruction through a single outer-rainbow x.

(1) Find a cycle C through x whose edges are incident with the outer face of $G(\Sigma)$. If no such C exists, output *No obstruction through x*. Else, go to Step 2.
(2) Find whether there is an outer-rainbow y in $V(C) \setminus \{x\}$. If no such y exists, output C. Else, apply Algorithm 1 to x and y; if there is an obstruction C' through x and y, then output C'. Else, update $G(\Sigma) \longleftarrow G(\Sigma - y)$ and go to Step 1.

Correctness and running-time of Algorithm 2: If $G(\Sigma)$ has an obstruction through x, then there is a non-edge block in $G(\Sigma)$ containing x. The outer boundary of this block is a cycle C through x having all edges incident with the outer face. As in Algorithm 1, Step 1 takes $O(m)$ time.

Detecting the existence of y in Step 2 is $O(m)$ because to detect rainbows in C, each edge incident with a vertex in $V(C)$ is verified at most twice. The update in Step 2 is justified by Property (**). Since Step 2 may use Algorithm 1, Step 2 takes $O(n^2)$ time. As moving from Step 2 to Step 1 occurs $O(n)$ times, the total running-time for Algorithm 2 is $O(n^3)$.

We are now ready for the algorithm to detect obstructions.

Algorithm 3 Detecting obstructions in $G(\Sigma)$.

(1) Find a cycle C having all edges incident with the outer face. If no such C exists, output *No obstruction*. Else, go to step 2.
(2) Find whether there is an outer rainbow $x \in V(C)$. If not, output C. Else apply Algorithm 2 to x; if there is an obstruction C' through x, output C'. Else, update $G(\Sigma) \longleftarrow G(\Sigma - x)$ and go to Step 1.

Correctness and running-time of Algorithm 3: If $G(\Sigma)$ has an obstruction, then it has a non-trivial block whose outer boundary is a cycle C as in Step 1. As before, C and x as in Step 2 can be found in $O(m)$ steps. If C has not outer rainbow x, then C is an obstruction; Property (**) justifies the update in Step 2.

Since Step 2 may use Algorithm 2, a full run of Steps 1 and 2 takes $O(n^3)$ time. Since Step 2 goes to Step 1 $O(n)$ times, the running-time of Algorithm 3 is $O(n^4)$.

Algorithm 3 and the constructive proof of Theorem 2 imply the following result (proved in the full version of this paper [4]).

▶ **Theorem 14.** *There is a polynomial-time algorithm to recognize and extend an ordinary set of strings that are extendible to an arrangement of pseudolines.*

6 Concluding remarks

In this work we characterized in Theorem 2 sets of strings that can be extended into arrangements of pseudolines. Moreover, we showed that the obstructions to pseudolinearity can be detected in $O(n^4)$ time, where n is the number of vertices in the planarization of the set of strings.

An easy consequence of Theorem 2 is the following (presented before as Theorem 1). We prove this result in the full version of this paper [4].

▶ **Theorem 15.** *Let D be a non-pseudolinear good drawing of a graph H. Then there is a subset S of edge-arcs in $\{D[e] : e \in E(H)\}$, such that each $\sigma \in S$ has a substring $\sigma' \subseteq \sigma$ for which $\bigcup_{\sigma \in S} \sigma'$ is one of the drawings represented in Figure 1.*

Theorem 2 can also be applied to find a short proof that pseudolinear drawings of K_n are characterized by forbidding the B-configuration (see Theorem 2.5.1 in [3]). This implies the characterizations of pseudolinear drawings of K_n presented in [2, 5, 6].

A drawing is *stretchable* if it is homeomorphic to a rectilinear drawing. There are pseudolinear drawings that are not stretchable. For instance, consider the Non-Pappus configuration in Figure 6. Nevertheless, as an immediate consequence of Thomassen's main result in [16], pseudolinear and stretchable drawings are equivalent, under the assumption that every edge is crossed at most once.

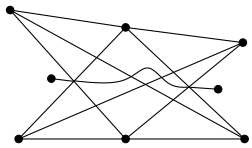


Figure 6 Non-Pappus configuration.

▶ **Corollary 16.** *A drawing D of a graph in which every edge is crossed at most once is stretchable if and only if it is pseudolinear.*

Proof. If a drawing D is stretchable then clearly it is pseudolinear. To show the converse, suppose that D is pseudolinear. Then D does not contain any obstruction, and in particular, neither of the B- and W-configurations in Figure 1 occurs in D. This condition was shown in [16] to be equivalent to being stretchable. ◀

One can construct more general examples of pseudolinear drawings that are not stretchable by considering non-strechable arrangements of pseudolines. However, such examples seem to inevitably have some edge with multiple crossings. This leads to a natural question.

▶ **Question 17.** *Is it true that if D is a pseudolinear drawing in which every edge is crossed at most twice, then D is stretchable?*

We believe that there are other instances where pseudolinearity characterizes stretchability of drawings. A drawing is *near planar* if the removal of one edge produces a planar graph. One instance, is the following result by Eades et al. that can be translated to the language of pseudolines:

▶ **Theorem 18** ([9])**.** *A drawing of a near-planar graph is stretchable if and only if the drawing induced by the crossed edges is pseudolinear.*

References

1 Bernardo M Ábrego and Silvia Fernández-Merchant. A lower bound for the rectilinear crossing number. *Graphs and Combinatorics*, 21(3):293–300, 2005.

2 Oswin Aichholzer, Thomas Hackl, Alexander Pilz, Birgit Vogtenhuber, and G Salazar. Deciding monotonicity of good drawings of the complete graph. In *Encuentros de Geometría Computacional*, pages 33–36. ., 2015.

3 Alan Arroyo. *On Geometric Drawings of Graphs*. PhD thesis, University of Waterloo, 2018.

4 Alan Arroyo, Julien Bensmail, and R Bruce Richter. Extending drawings of graphs to arrangements of pseudolines. *arXiv preprint*, 2018. `arXiv:1804.09317`.

5 Alan Arroyo, Dan McQuillan, R Bruce Richter, and Gelasio Salazar. Levi's lemma, pseudolinear drawings of, and empty triangles. *Journal of Graph Theory*, 87(4):443–459, 2018.

6 Martin Balko, Radoslav Fulek, and Jan Kynčl. Crossing numbers and combinatorial characterization of monotone drawings of K_n. *Discrete & Computational Geometry*, 53(1):107–143, 2015.

7 József Balogh, Jesús Leaños, Shengjun Pan, R Bruce Richter, and Gelasio Salazar. The convex hull of every optimal pseudolinear drawing of K_n is a triangle. *Australasian Journal of Combinatorics*, 38:155, 2007.

8 Daniel Bienstock and Nathaniel Dean. Bounds for rectilinear crossing numbers. *Journal of Graph Theory*, 17(3):333–348, 1993.

9 Peter Eades, Seok-Hee Hong, Giuseppe Liotta, Naoki Katoh, and Sheung-Hung Poon. Straight-line drawability of a planar graph plus an edge. *arXiv preprint*, 2015. `arXiv:1504.06540`.

10 Stefan Felsner. *Geometric graphs and arrangements: some chapters from combinatorial geometry*. Springer Science & Business Media, 2012.

11 Stefan Felsner and Jacob E Goodman. Pseudoline arrangements. In *Handbook of Discrete and Computational Geometry*, pages 125–157. Chapman and Hall/CRC, 2017.

12 César Hernández-Vélez, Jesús Leaños, and Gelasio Salazar. On the pseudolinear crossing number. *Journal of Graph Theory*, 84(3):155–162, 2017.

13 László Lovász, Katalin Vesztergombi, Uli Wagner, and Emo Welzl. Convex quadrilaterals and k-sets. *Contemporary Mathematics*, 342:139–148, 2004.

14 Nicolai E Mnëv. Varieties of combinatorial types of projective configurations and convex polytopes. *Doklady Akademii Nauk SSSR*, 283(6):1312–1314, 1985.

15 Nikolai E Mnëv. The universality theorems on the classification problem of configuration varieties and convex polytopes varieties. In *Topology and geometry – Rohlin seminar*, pages 527–543. Springer, 1988.

16 Carsten Thomassen. Rectilinear drawings of graphs. *Journal of Graph Theory*, 12(3):335–341, 1988.

Dimensionality Reduction for k-Distance Applied to Persistent Homology

Shreya Arya
Duke University, Durham, NC, USA
shreya.arya14@gmail.com

Jean-Daniel Boissonnat
Université Côte d'Azur, INRIA, Sophia-Antipolis, France
jean-daniel.boissonnat@inria.fr

Kunal Dutta
Faculty of Mathematics, Informatics, and Mechanics, University of Warsaw, Warsaw, Poland
K.Dutta@mimuw.edu.pl

Martin Lotz
Mathematics Institute, University of Warwick, Coventry, United Kingdom
martin.lotz@warwick.ac.uk

Abstract

Given a set P of n points and a constant k, we are interested in computing the persistent homology of the Čech filtration of P for the k-distance, and investigate the effectiveness of dimensionality reduction for this problem, answering an open question of Sheehy [*Proc. SoCG, 2014*]. We show that *any* linear transformation that preserves pairwise distances up to a $(1 \pm \varepsilon)$ multiplicative factor, must preserve the persistent homology of the Čech filtration up to a factor of $(1 - \varepsilon)^{-1}$. Our results also show that the Vietoris-Rips and Delaunay filtrations for the k-distance, as well as the Čech filtration for the approximate k-distance of Buchet et al. are preserved up to a $(1 \pm \varepsilon)$ factor.

We also prove extensions of our main theorem, for point sets (i) lying in a region of bounded Gaussian width or (ii) on a low-dimensional manifold, obtaining the target dimension bounds of Lotz [*Proc. Roy. Soc.* , 2019] and Clarkson [*Proc. SoCG, 2008*] respectively.

2012 ACM Subject Classification Theory of computation; Theory of computation → Computational geometry

Keywords and phrases Dimensionality reduction, Johnson-Lindenstrauss lemma, Topological Data Analysis, Persistent Homology, k-distance, distance to measure

Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.10

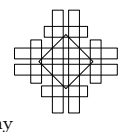
Funding The research leading to these results has received funding from the European Research Council (ERC) under the European Union's Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreement No. 339025 GUDHI (Algorithmic Foundations of Geometry Understanding in Higher Dimensions).

Acknowledgements We thank the reviewers for their helpful comments and suggestions.

1 Introduction

Persistent homology is one of the main tools used to extract information from data in topological data analysis. Given a data set as a point cloud in some ambient space, the idea is to construct a filtration sequence of topological spaces from the point cloud, and extract topological information from this sequence. The topological spaces are usually constructed by considering balls around the data points, in some given metric of interest, as the open sets. However the usual distance function is highly sensitive to the presence of outliers and

36th International Symposium on Computational Geometry (SoCG 2020).
Editors: Sergio Cabello and Danny Z. Chen; Article No. 10; pp. 10:1–10:15



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

noise. One approach is to use distance functions that are more robust to outliers, such as the *distance-to-a-measure* and the related *k-distance* (for finite data sets), proposed recently by Chazal et al. [9] Although this is a promising direction, an exact implementation is extremely costly. To overcome this difficulty, approximations of the k-distance have been proposed recently that led to certified approximations of persistent homology [22, 7]. Other approaches involve using kernels [28], de-noising algorithms [8, 33], etc.

In all the above settings, the sub-routines required for computing persistent homology have exponential or worse dependence on the ambient dimension, and rapidly become unusable in real-time once the dimension grows beyond a few dozens - which is indeed the case in many applications, for example in image processing, neuro-biological networks, data mining (see e.g. [20]), a phenomenon often referred to as the *curse of dimensionality*.

The Johnson-Lindenstrauss Lemma. One of the simplest and most commonly used mechanisms to mitigate this curse, is that of *random projections*, as applied in the celebrated Johnson and Lindenstrauss lemma (JL Lemma for short) [24]. The JL Lemma states that any set of n points in Euclidean space can be embedded in a space of dimension $O(\varepsilon^{-2} \log n)$ with $(1 \pm \varepsilon)$ distortion. Since the initial non-constructive proof of this fact by Johnson and Lindenstrauss [24], several authors have given successive improvements, e.g. Indyk, Motwani, Raghavan and Vempala [23], Dasgupta and Gupta [14], Achlioptas [1], Ailon and Chazelle [2], Matoušek [26] and others, which address the issues of *efficient* constructivization and implementation, using random linear transformations. Dirksen [15] gave a unified theory for dimensionality reduction using subgaussian matrices.

In a different direction, variants of the Johnson-Lindenstrauss lemma with better target dimension have been given under several specific settings. For point sets lying in regions of bounded Gaussian width, a theorem of Gordon [21] implies that the target dimension can be reduced to a function of the Gaussian width, independent of the number of points. Sarlos [29] showed that points lying on a d-flat can be mapped on to $O(d/\varepsilon^2)$ dimensions independently of the number of points. Baraniuk and Wakin [5] proved an analogous result for points on a smooth manifold, which was subsequently sharpened by Clarkson [12]. Verma [31] gave a further improvement, directly preserving geodesic distances on the manifold. Other related results include those of Indyk and Naor [12] for sets of bounded doubling dimension, with additive errors, and Alon and Klartag [3] preserving general inner products, again with additive error only.

Dimension Reduction and Persistent Homology. The JL Lemma has also been used by Sheehy [30] and Lotz [25] to reduce the complexity of computing persistent homology. Both Sheehy and Lotz show that the persistent homology of a point cloud is approximately preserved under random projections [30, 25], up to a $(1 \pm \varepsilon)$ multiplicative factor, for any $\varepsilon \in [0, 1]$. Sheehy proves this for an n-point set, whereas Lotz's generalization applies to sets of bounded Gaussian width. However, their techniques involve only the usual distance to a point set and therefore remain sensitive to outliers and noise as mentioned earlier. The question of adapting the method of random projections in order to reduce the complexity of computing persistent homology using the k-distance, is therefore a natural one, and has been raised by Sheehy [30], who observed that *"One notable distance function that is missing from this paper* [i.e. [30]] *is the so-called distance to a measure or ... k-distance ... it remains open whether the k-distance itself is* $(1 \pm \varepsilon)$*-preserved under random projection."*

Our Contribution

In this paper, we combine the method of random projections with the k-distance and show its applicability in computing persistent homology. It is not very hard to see that for a given point set P, the random Johnson-Lindenstrauss mapping preserves the pointwise k-distance to P (Theorem 13). However, this is not enough to preserve intersections of balls at varying scales of the radius parameter and thus does not suffice to preserve the persistent homology of Čech filtrations, as noted by Sheehy [30] and Lotz [25]. We show how the squared radius of a set of weighted points can be expressed as a convex combination of pairwise squared distances. From this, it follows that the Čech filtration under the k-distance, will be preserved by *any* linear mapping that preserves pairwise distances.

Extensions

Further, as our main result applies to any linear mapping that approximately preserves pairwise distances, the theorems of Lotz, Baraniuk and Wakin, and others apply immediately. Thus, we give two extensions of our results. The first one, analogous to Lotz [25], shows that the persistent homology with respect to the k-distance, of point sets contained in regions having bounded Gaussian width, can be preserved via dimensionality reduction, with target dimension a function of the Gaussian width. Another result is that for points lying in a low-dimensional submanifold of a high-dimensional Euclidean space, the target dimension for preserving the persistent homology with k-distance depends linearly on the dimension of the manifold. Both these settings are commonly encountered in high-dimensional data analysis, machine learning, etc. (see e.g. the *manifold hypothesis* [18]).

▶ Remark 1. It should be noted that the approach of using dimensionality reduction for the k-distance, is complementary to denoising techniques such as [8] as we do not try to remove noise, only to be more robust to noise. Therefore, it can be used in conjunction with denoising techniques, as a pre-processing tool when the dimensionality is high.

Outline

The rest of this paper is as follows. In Section 2, we briefly summarize some basic definitions and background. Our theorems are stated in Section 3 and proved in Section 4. Some applications of our results are proved in Section 5. We end with some final remarks and open questions in Section 6.

2 Background

We begin with some preliminary background.

We shall need a well-known identity for the variance of bounded random variables, which will be crucial in the proof of our main theorem. Let $\lambda_1, \ldots, \lambda_k \geq 0$ be such that $\sum_{i=1}^{k} \lambda_i = 1$. Let $p_1, \ldots, p_k \in \mathbb{R}^D$ be given points. and let $b = \sum_{i=1}^{k} \lambda_i p_i$. Then for any point $x \in \mathbb{R}^D$, the following holds

$$\sum_{i=1}^{k} \lambda_i \|x - p_i\|^2 \quad = \quad \|x - b\|^2 + \sum_{i=1}^{k} \lambda_i \|b - p_i\|^2. \tag{1}$$

In particular, for $\lambda_i = 1/k$ for all i, we have

$$\frac{1}{k} \sum_{i=1}^{k} \|x - p_i\|^2 \quad = \quad \|x - b\|^2 + \sum_{i=1}^{k} \frac{1}{k} \|b - p_i\|^2. \tag{2}$$

2.1 Random Projections

The Johnson-Lindenstrauss lemma [24] states that any subset of n points of Euclidean space can be embedded in a space of dimension $O(\varepsilon^{-2}\log n)$ with $(1\pm\varepsilon)$ distortion. In order to separate the technical aspects of our result from the issues of implementation, we use the notion of an *ε-distortion map with respect to P* (also commonly called a Johnson-Lindenstrauss map).

▶ **Definition 2.** *Given a point set $P\subset\mathbb{R}^D$, and $\varepsilon\in(0,1)$, a mapping $f:\mathbb{R}^D\to\mathbb{R}^d$ for some $d\leq D$ is an* ε-distortion map with respect to P, *if for all $x,y\in P$,*

$$(1-\varepsilon)\|x-y\|\leq\|f(x)-f(y)\|\leq(1+\varepsilon)\|x-y\|.$$

A random variable X with mean zero, is said to be *subgaussian* with *subgaussian norm* K if $\mathbb{E}\left[\exp X^2/K^2\right]\leq 2$. In this case, the tails of the random variable satisfy

$$\mathbb{P}\left[|X|\geq t\right]\leq 2\exp\left(-t^2/2K^2\right).$$

We focus on the case where the Johnson-Lindenstrauss embedding is carried out via random subgaussian matrices, i.e. matrices where for some given $K>0$, each entry is an independent subgaussian random variable with subgaussian norm K. This case is general enough to include the mappings of e.g. Achlioptas [1], Ailon and Chazelle [2], Dasgupta and Gupta [14], Indyk, Motwani, Raghavan, and Vempala [23], and Matoušek [26] (see e.g. Dirksen for a unified treatment [15]).

▶ **Lemma 3** (JL Lemma). *Given $0<\varepsilon,\delta<1$, and a finite point set $P\subset\mathbb{R}^D$ of size $|p|=n$. Then a random linear mapping $f\colon\mathbb{R}^D\to\mathbb{R}^d$ where $d=O(\varepsilon^{-2}\log n)$ given by $f(v)=\sqrt{\frac{D}{d}}Gv$ where G is a $d\times D$ subgaussian random matrix, is an ε-distortion map with respect to P, with probability at least $1-\delta$.*

2.2 k-Distance

The distance to a finite point set P is usually taken to be the minimum distance to a point in the set. For the computations involved in geometric and topological inference, however, this distance is extremely sensitive to outliers and noise. To handle this problem of sensitivity, Chazal et al. in [9] introduced the *distance to a probability measure* which, in the case of a uniform probability on P, is called the *k-distance*.

▶ **Definition 4** (k-distance). *For $k\in\{1,...,n\}$ and $x\in\mathbb{R}^D$, the k-distance of x to P is*

$$d_{P,k}(x)=\min_{S_k\in\binom{P}{k}}\sqrt{\frac{1}{k}\sum_{p\in S_k}\|x-p\|^2}=\sqrt{\frac{1}{k}\sum_{p\in NN_P^k(x)}\|x-p\|^2}\tag{3}$$

where $NN_P^k(x)\subset P$ denotes the k nearest neighbours in P to the point $x\in\mathbb{R}^D$.

It was shown in [4], that the k-distance can be expressed in terms of weighted points and power distance. A weighted point $\hat{p}$ is a point p of $\mathbb{R}^D$ together with a (not necessarily positive) real number called its weight and denoted by $w(p)$. The distance between two weighted points $\hat{p}_i=(p_i,w_i)$ and $\hat{p}_j=(p_j,w_j)$ is defined as $D(\hat{p}_i,\hat{p}_j)=\|p_i-p_j\|^2-w_i-w_j$. This definition encompasses the case where the two weights are 0, in which case we have the squared euclidean distance and the case where one of the points has weight 0, in which case, we have the power distance of a point to a ball. We say that two weighted points are *orthogonal* when their distance is 0.

Let $B_{P,k}$ be the set of iso-barycentres of all subsets of k points in P. To each barycenter $b = (1/k)\sum_i p_i \in B_{P,k}$, we associate the weight $w(b) = -\frac{1}{k}\sum_i \|b - p_i\|^2$. Writing $\hat{B}_{P,k} = \{\hat{b} = (b, w(b)), b \in B_{P,k}\}$, we see from (2) that the k-distance is the square root of a power distance [4]

$$d_{P,k}(x) = \min_{\hat{b}\in\hat{B}_{P,k}} \sqrt{D(x,\hat{b})}. \tag{4}$$

Observe that in general the squared distance between a pair of weighted points can be negative, but the above assignment of weights ensures that the k-distance $d_{P,k}$ is a real function. Since $d_{P,k}$ is the square root of a non-negative power distance, the α-sublevel set of $d_{P,k}$, $d_{P,k}([-\infty, \alpha])$, $\alpha \in \mathbb{R}$, is the union of $\binom{n}{k}$ balls $B(b, \sqrt{\alpha^2 + w(b)})$, $b \in B_{P,k}$. However, some of the balls may be included in the union of others and be redundant. In fact, the number of barycenters (or equivalently of balls) required to define a level set of $d_{P,k}$ is equal to the number of the non-empty cells in the kth-order Voronoi diagram of P. Hence the number of non-empty cells is $\Omega\left(n^{\lfloor (D+1)/2 \rfloor}\right)$ [13] and computing them in high dimensions is intractable. It is then natural to look for approximations of the k-distance, e.g., the following definition has been proposed [7]:

▶ **Definition 5** (Approximation). *Let $P \subset \mathbb{R}^D$ and $x \in \mathbb{R}^D$. The approximate k-distance $\tilde{d}_{P,k}(x)$ is defined as*

$$\tilde{d}_{P,k}(x) \quad := \quad \min_{p\in P} \sqrt{D(x,\hat{p})} \tag{5}$$

where $\hat{p} = (p, w(p))$ with $w(p) = -d^2_{P,k}(p)$, the opposite of the squared k-distance of p.

As in the exact case, $\tilde{d}_{P,k}$ is the square root of a power distance and its α-sublevel set, $\alpha \in \mathbb{R}$, is a union of balls, specifically the balls $B(p, \sqrt{\alpha^2 - d^2_{P,k}(p)})$, $p \in P$. The major difference with the exact case is that, since we consider only balls around the points of P, their number is n instead of $\binom{n}{k}$ in the exact case (compare Eq. (5) and Eq. (4)). Still, $\tilde{d}_{P,k}(x)$ approximates the k-distance [7]:

$$\frac{1}{\sqrt{2}}\, d_{P,k} \le \tilde{d}_{P,k} \le \sqrt{3}\, d_{P,k}. \tag{6}$$

We now make an observation for the case when the weighted points are barycenters, which will be very useful in proving our main theorem.

▶ **Lemma 6.** *Given $b_1, b_2 \in B_{P,k}$, and $p_{i,1}, \ldots, p_{i,k} \in P$ for $i = 1, 2$, such that $b_i = \frac{1}{k}\sum_{l=1}^k p_{i,l}$, and $w(b_i) = \frac{1}{k}\sum_{l=1}^k \|b_i - p_{i,l}\|^2$ for $i = 1, 2$, then it holds that*

$$D(\hat{b}_1, \hat{b}_2) = \frac{1}{k^2}\sum_{l,s=1}^{k} \|p_{1,l} - p_{2,s}\|^2.$$

Proof of Lemma 6. We have

$$D(\hat{b}_1, \hat{b}_2) \;=\; \|b_1 - b_2\|^2 - w(b_1) - w(b_2) \;=\; \|b_1 - b_2\|^2 + \frac{1}{k}\sum_{l=1}^{k} \|b_1 - p_{1,l}\|^2 + \frac{1}{k}\sum_{l=1}^{k} \|b_2 - p_{2,l}\|^2.$$

Applying the identity (2), we get $\|b_1 - b_2\|^2 + \frac{1}{k}\sum_{l=1}^k \|b_2 - p_{2,l}\|^2 = \frac{1}{k}\sum_{l=1}^k \|b_1 - p_{2,l}\|^2$, so that

$$\begin{aligned} D(\hat{b}_1, \hat{b}_2) &= \frac{1}{k}\sum_{l=1}^{k} \|b_1 - p_{2,l}\|^2 + \frac{1}{k}\sum_{l=1}^{k} \|b_1 - p_{1,l}\|^2 \\ &= \frac{1}{k}\sum_{l=1}^{k} \|b_1 - p_{2,l}\|^2 + \frac{1}{k^2}\sum_{s=1}^{k}\sum_{l=1}^{k} \|b_1 - p_{1,l}\|^2 \\ &= \frac{1}{k}\sum_{l=1}^{k} \left(\|b_1 - p_{2,l}\|^2 + \frac{1}{k}\sum_{s=1}^{k} \|b_1 - p_{1,s}\|^2 \right) \\ &= \frac{1}{k}\sum_{l=1}^{k} \left(\frac{1}{k}\sum_{s=1}^{k} \|p_{1,s} - p_{2,l}\|^2 \right) = \frac{1}{k^2}\sum_{l,s=1}^{k} \|p_{1,s} - p_{2,l}\|^2, \end{aligned} \tag{7}$$

where in (7), we again applied (2) to each of the points $p_{2,s}$, with respect to the barycenter b_1. ◀

2.3 Persistent Homology

Simplicial Complexes and Filtrations. Let V be a finite set. An (abstract) simplicial complex with vertex set V is a set K of finite subsets of V such that if $A \in K$ and $B \subseteq A$, then $B \in K$. The sets in K are called the simplices of K. A simplex $F \in K$ that is strictly contained in a simplex $A \in K$, is said to be a *face* of A.

A simplicial complex K with a function $f : K \to \mathbb{R}$ such that $f(\sigma) \leq f(\tau)$ whenever σ is a face of τ is a filtered simplicial complex. The sublevel set of f at $r \in \mathbb{R}$, $f^{-1}(-\infty, r]$, is a subcomplex of K. By considering different values of r, we get a nested sequence of subcomplexes (called a filtration) of K, $\emptyset = K^0 \subseteq K^1 \subseteq ... \subseteq K^m = K$, where K^i is the sublevel set at value r_i.

The Čech filtration associated to a finite set P of points in $\mathbb{R}^D$ plays an important role in Topological Data Analysis.

▶ **Definition 7** (Čech Complex). *The Čech complex $\check{C}_\alpha(P)$ is the set of simplices $\sigma \subset P$ such that rad(σ) $\leq \alpha$, where* $\mathrm{rad}(\sigma)$ *is the radius of the smallest enclosing ball of σ, i.e.*

$$\mathrm{rad}(\sigma) \leq \alpha \Leftrightarrow \exists x \in \mathbb{R}^D,\ \forall p_i \in \sigma,\ \|x - p_i\| \leq \alpha.$$

When α goes from 0 to $+\infty$, we obtain the Čech filtration of P. $\check{C}_\alpha(P)$ can be equivalently defined as the nerve of the closed balls $\overline{B}(p, \alpha)$, centered at the points in P and of radius α:

$$\check{C}_\alpha(P) = \{\sigma \subset P | \cap_{p\in\sigma} \overline{B}(p, \alpha) \neq \emptyset\}.$$

By the nerve lemma, we know that the union of balls $U_\alpha = \cup_{p\in P}\overline{B}(p, \alpha)$, $p \in P$, and $\check{C}_\alpha(P)$ have the same homotopy type.

Persistence Diagrams. Persistent homology is a means to compute and record the changes in the topology of the filtered complexes as the parameter α increases from zero to infinity. Edelsbrunner, Letscher and Zomorodian [17] gave an algorithm to compute the persistent homology, which takes a filtered simplicial complex as input, and outputs a sequence $(\alpha_{birth}, \alpha_{death})$ of pairs of real numbers. Each such pair corresponds to a topological feature, and records the values of α at which the feature appears and disappears, respectively, in the filtration. Thus the topological features of the filtration can be represented using this sequence of pairs, which can be represented either as points in the extended plane

$\bar{\mathbb{R}}^2 = (\mathbb{R} \cup \{-\infty, \infty\})^2$, called the *persistence diagram* or as a sequence of barcodes (the *persistence barcode*) (see, e.g., [16]). A pair of persistence diagrams $\mathbb{G}$ and $\mathbb{H}$ corresponding to the filtrations (G_α) and (H_α) respectively, are *multiplicatively β-interleaved*, $(\beta \geq 1)$, if for all α, we have that $G_{\alpha/\beta} \subseteq H_\alpha \subseteq G_{\alpha\beta}$. We shall crucially rely on the fact that a given persistence diagram is closely approximated by another one if they are multiplicatively c-interleaved, with c close to 1 (see e.g. [10]).

The Persistent Nerve Lemma [11] shows that the persistent homology of the Čech complex is the same as the homology of the α-sublevel filtration of the distance function.

The Weighted Case. Our goal is to extend the above definitions and results to the case of the k-distance. As we observed earlier, the k-distance is a power distance in disguise. Accordingly, we need to extend the definition of the Čech complex to sets of weighted points.

▶ **Definition 8** (Weighted Čech Complex). *Let $\hat{P} = \{\hat{p}_1, ..., \hat{p}_n\}$ be a set of weighted points, where $\hat{p}_i = (p_i, w_i)$. The α-Čech complex of $\hat{P}$, $\check{C}_\alpha(\hat{P})$, is the set of all simplices σ satisfying*

$$\exists x,\ \forall p_i \in \sigma,\ \|x - p_i\|^2 \leq w_i + \alpha^2 \quad \Leftrightarrow \quad \exists x,\ \forall p_i \in \sigma,\ D(x, \hat{p}_i) \leq \alpha^2.$$

In other words, the α-Čech complex of $\hat{P}$ is the nerve of the closed balls $\overline{B}(p_i, r_i^2 = w_i + \alpha^2)$, centered at the p_i and of squared radius $w_i + \alpha^2$ (if negative, $\overline{B}(p_i, r_i^2)$ is imaginary).

The notions of weighted Čech filtrations and their persistent homology now follow naturally. Moreover, it follows from (4) that the Čech complex $\check{C}_\alpha(P)$ for the k-distance is identical to the weighted Čech complex $\check{C}_\alpha(\hat{B}_{P,k})$, where $\hat{B}_{P,k}$ is, as above, the set of iso-barycenters of all subsets of k points in P.

In the Euclidean case, we equivalently defined the α-Čech complex as the collection of simplices whose smallest enclosing balls have radius at most α. We can proceed similarly in the weighted case. Let $\hat{X} \subseteq \hat{P}$. We define the *radius of* $\hat{X}$ as $\mathrm{rad}^2(\hat{X}) = \min_{x \in \mathbb{R}^D} \max_{\hat{p}_i \in \hat{X}} D(x, \hat{p}_i)$, and the weighted center or simply the *center* of $\hat{X}$ as the point, noted $c(\hat{X})$, where the minimum is reached.

Our goal is to show that preserving smallest enclosing balls in the weighted scenario under a given mapping, also preserves the persistent homology. Sheehy [30] and Lotz [25], proved this for the unweighted case. Their proofs also work for the weighted case but only under the assumption that the weights stay unchanged under the mapping. In our case however, the weights need to be recomputed in $f(\hat{P})$. We therefore need a version of [25, Lemma 2.2] for the weighted case which does not assume that the weights stay the same under f. This is Lemma 12, which follows at the end of this section. The following lemmas will be instrumental in proving Lemma 12 and in proving our main result. Let $\hat{X} \subseteq \hat{P}$ and assume without loss of generality that $\hat{X} = \{\hat{p}_1, ..., \hat{p}_m\}$, where $\hat{p}_i = (p_i, w_i)$.

▶ **Lemma 9.** *$c(\hat{X})$ and $\mathrm{rad}(\hat{X})$ are uniquely defined.*

▶ **Lemma 10.** *Let I be the set of indices for which $D(c, \hat{p}_i) = \mathrm{rad}(\hat{X})$ and let $\hat{X}_I = \{\hat{p}_i, i \in I\}$. $c(\hat{X})$ is a convex combination of the points in X_I, i.e. $c(\hat{X}) = \sum_{i=1}^m \lambda_i p_i$ with $\sum_{i=1}^m \lambda_i = 1$, $\lambda_i \geq 0$ for all i, and $\lambda_i = 0$ for all $i \notin I$.*

Combining the above lemmas with [25, Lemma 4.2] gives the following lemma.

▶ **Lemma 11.** $\mathrm{rad}^2(\hat{X}) = \frac{1}{2} \sum_{i \in I} \sum_{j \in I} \lambda_i \lambda_j D(\hat{p}_i, \hat{p}_j)$.

Let $X \in \mathbb{R}^D$ be a finite set of points and $\hat{X}$ be the associated weighted points where the weights are computed according to a weighting function $w : X \to \mathbb{R}^-$. Given a mapping $f : \mathbb{R}^D \to \mathbb{R}^d$, we define $\widehat{f(X)}$ as the set of weighted points $\{(f(x), w(f(x))), x \in X\}$. Note that the weights are recomputed in the image space $\mathbb{R}^d$.

▶ **Lemma 12.** *In the above setting, if f is such that for some $\varepsilon \in (0,1)$ and for all subsets $\hat{S} \subseteq \hat{X}$ we have*

$$(1-\varepsilon)\mathrm{rad}^2(\hat{S}) \leq \mathrm{rad}^2(\widehat{f(S)}) \leq (1+\varepsilon)\mathrm{rad}^2(\hat{S}),$$

then the weighted Čech filtrations of $\hat{X}$ and $f(\hat{X})$ are multiplicatively $(1-\varepsilon)^{-1/2}$ interleaved.

3 Results

For the subsequent theorems, we denote by P a set of n points in $\mathbb{R}^D$.

Our first theorem shows that for the points in P, the pointwise k-distance $d_{P,k}$ is preserved by a random subgaussian matrix satisfying Lemma 3.

▶ **Theorem 13.** *Given $\varepsilon \in (0,1]$, an ε-distortion map with respect to P $f\colon \mathbb{R}^D \to \mathbb{R}^d$, where $d = O(\varepsilon^{-2}\log n)$, satisfies for all points $x \in P$:*

$$(1-\varepsilon)d^2_{P,k}(x) \leq d^2_{f(P),k}(f(x)) \leq (1+\varepsilon)d^2_{P,k}(x).$$

Moreover, given any $\delta \in (0,1)$, the above inequality holds with probability at least $1-\delta$ for a random function $f : \mathbb{R}^D \to \mathbb{R}^d$ given by $f(x) = \sqrt{D/d}Gx$, where G is a random subgaussian matrix, and $d = O\left(\frac{\log n}{\varepsilon^2}\right)$, where the constant in the O-notation depends on δ.

As mentioned previously, the preservation of the pointwise k-distance does not imply the preservation of the Čech complex formed using the points in P. Nevertheless, the following theorem shows that this can always be done in dimension $O(\log n/\varepsilon^2)$.

Let $\hat{B}_{P,k}$ be the set of iso-barycenters of every k-subset of P, weighted as in Section 2.2. Recall from Section 2.3 that the weighted Čech complex $\check{C}_\alpha(\hat{B}_{P,k})$ is identical to the Čech complex $\check{C}_\alpha(P)$ for the k-distance.

▶ **Theorem 14** (k-distance)**.** *Let $\hat{\sigma} \subseteq \hat{B}_{P,k}$ be a simplex in the weighted Čech complex $\check{C}_\alpha(\hat{B}_{P,k})$. Then, given $d \leq D$ such that there exists a ε-distortion map with respect to P $f : \mathbb{R}^D \to \mathbb{R}^d$, the following holds:*

- **(i)** $(1-\varepsilon)\mathrm{rad}^2(\hat{\sigma}) \leq \mathrm{rad}^2(\widehat{f(\sigma)}) \leq (1+\varepsilon)\mathrm{rad}^2(\hat{\sigma})$.
- **(ii)** *In particular, for a n-point set P, given $\delta \in (0,1)$, the function $f : \mathbb{R}^D \to \mathbb{R}^d$ given by $f(x) = \left(\sqrt{D/d}\right)Gx$, where G is a random $d \times D$ Gaussian matrix G where $d = O\left(\frac{\log n}{\varepsilon^2}\right)$, satisfies the above inequality with probability at least $1-\delta$.*

For the approximation of the k-distance given by [7] also, we get an optimal target dimension, as the number of weighted points needed to compute the approximate k-distance, is just n.

▶ **Theorem 15** (Approximate k-distance)**.** *Let $\hat{P}$ be the weighted points associated with P, introduced in Definition 5 (Equ. 5). Let, in addition, $\hat{\sigma} \subseteq \hat{P}$ be a simplex in the associated weighted Čech complex $\check{C}_\alpha(\hat{P})$. Then an ε-distortion mapping with respect to P, $f : \mathbb{R}^D \to \mathbb{R}^d$ satisfies: $(1-\varepsilon)\mathrm{rad}^2(\hat{\sigma}) \leq \mathrm{rad}^2(\widehat{f(\sigma)}) \leq (1+\varepsilon)\mathrm{rad}^2(\hat{\sigma})$. Moreover, the function $f : \mathbb{R}^D \to \mathbb{R}^d$ given by $f(x) = \left(\sqrt{D/d}\right)Gx$, where G is a random $d \times D$ Gaussian matrix G where $d = O(\log n/\varepsilon^2)$, satisfies the above inequality, with probability at least $1-\delta$.*

Applying Lemma 12 to the theorems 14 and 15, we get the following corollary.

▶ **Corollary 16.** *The persistent homology for the Čech filtrations of P and its image $f(P)$ under any ε-distortion mapping with respect to P, using the (i) exact k-distance, as well as the (ii) approximate k-distance, are multiplicatively $(1-\varepsilon)^{-1/2}$-interleaved with probability $1-\delta$.*

However, note that the approximation in Corollary 16 (ii) is with respect to the *approximate* k-distance, which is itself an $O(1)$ approximation of the k-distance (see (6)).

4 Proofs

We begin with the proofs of the auxiliary lemmas.

Proof of Lemma 9. The proof follows from the convexity of D (see Lemma 6). Assume, for a contradiction, that there exists two centers c_0 and $c_1 \neq c_0$ for $\hat{X}$. For convenience, write $r = \mathrm{rad}(\hat{X})$. By the definition of the center of $\hat{X}$, we have

$$\begin{aligned} \exists \hat{p}_0, \forall \hat{p}_i : D(c_0, \hat{p}_i) &\leq D(c_0, \hat{p}_0) = \|c_0 - p_0\|^2 - w_0 = r^2 \\ \exists \hat{p}_1, \forall \hat{p}_i : D(c_1, \hat{p}_i) &\leq D(c_1, \hat{p}_1) = \|c_1 - p_1\|^2 - w_1 = r^2. \end{aligned}$$

Consider $D_\lambda(\hat{p}_i) = (1-\lambda)D(c_0, \hat{p}_i) + \lambda D(c_1, \hat{p}_i)$ and write $c_\lambda = (1-\lambda)c_0 + \lambda c_1$. For any $\lambda \in (0,1)$, we have

$$\begin{aligned} D_\lambda(\hat{p}_i) &= (1-\lambda)D(c_0, \hat{p}_i) + \lambda D(c_1, \hat{p}_i) \\ &= (1-\lambda)(c_0 - p_i)^2 + \lambda(c_1 - p_i)^2 - w_i \\ &= D(c_\lambda, \hat{p}_i) - c_\lambda^2 + (1-\lambda)c_0^2 + \lambda c_1^2 \\ &= D(c_\lambda, \hat{p}_i) + \lambda(1-\lambda)(c_0 - c_1)^2 \\ &> D(c_\lambda, \hat{p}_i). \end{aligned}$$

Moreover, for any i,

$$D_\lambda(\hat{p}_i) = (1-\lambda)D(c_0, \hat{p}_i) + \lambda D(c_1, \hat{p}_i) \leq r^2.$$

Thus, for any i and any $\lambda \in (0,1)$, $D(c_\lambda, \hat{p}_i) < r^2$. Hence c_λ is a better center than c_0 and c_1, and r is not the minimal possible value for $\mathrm{rad}(\hat{X})$. We have obtained a contradiction. ◀

Proof of Lemma 10. We write for convenience $c = c(\hat{X})$ and $r = \mathrm{rad}(\hat{X})$ and prove that $c \in \mathrm{conv}(X_I)$ by contradiction. Let $c' \neq c$ be the point of $\mathrm{conv}(X_I)$ closest to c, and $\tilde{c} \neq c$ be a point on $[cc']$. Since $\|\tilde{c} - p_i\| < \|c - p_i\|$ for all $i \in I$, $D(\tilde{c}, \hat{p}_i) < D(c, \hat{p}_i)$ for all $i \in I$. For $\tilde{c}$ sufficiently close to c, $\tilde{c}$ remains closer to the weighted points $\hat{p}_j$, $j \notin I$, than to the $\hat{p}_i$, $i \in I$. We thus have

$$D(\tilde{c}, \hat{p}_j) < D(\tilde{c}, \hat{p}_i) < D(c, \hat{p}_i) = r^2.$$

It follows that c is not the center of $\hat{X}$, a contradiction. ◀

Proof of Lemma 11. From Lemma 10, and writing $c = c(\hat{X})$ for convenience, we have

$$\mathrm{rad}^2(\hat{X}) = \sum_{i \in I} \lambda_i \big(\|c - p_i\|^2 - w_i \big).$$

We use the following simple fact from [25, Lemma 4.5]

$$\sum_{i \in I} \lambda_i \|c - p_i\|^2 = \frac{1}{2} \sum_{i \in I} \sum_{j \in I} \lambda_i \lambda_j \|p_i - p_j\|^2.$$

Substituting in the expression for $\mathrm{rad}^2(\hat{X})$,

$$\begin{aligned}
\mathrm{rad}^2(\hat{X}) &= \frac{1}{2}\sum_{j\in I}\sum_{i\in I}\lambda_j\lambda_i\|p_i-p_j\|^2 - \frac{1}{2}\sum_{i\in I}2\lambda_i w_i \\
&= \frac{1}{2}\sum_{i,j\in I}\lambda_j\lambda_i\|p_i-p_j\|^2 - \frac{1}{2}\sum_{i,j\in I}2\lambda_i\lambda_j w_i \ \ (\text{since} \ \ \sum_{j\in I}\lambda_j = 1) \\
&= \frac{1}{2}\sum_{i,j\in I}\lambda_j\lambda_i\|p_i-p_j\|^2 - \frac{1}{2}\sum_{i,j\in I}\lambda_i\lambda_j(w_i+w_j) \\
&= \frac{1}{2}\sum_{i,j\in I}\lambda_i\lambda_j\left(\|p_i-p_j\|^2 - w_i - w_j\right) \\
&= \frac{1}{2}\sum_{i,j\in I}\lambda_i\lambda_j D(\hat{p}_i,\hat{p}_j).
\end{aligned}$$

◀

Proof of Theorem 13. The proof follows from the observation that the squared k-distance from any point $p \in P$ to the point set P, is a convex combination of the squares of the Euclidean distances to the k nearest neighbours of p. Since the mapping in the JL Lemma 3 is linear, and it $(1\pm\varepsilon)$-preserves squared pairwise distances, their convex combinations also get $(1\pm\varepsilon)$-preserved. ◀

Proof of Theorem 14. Let $\hat{\sigma} = \{\hat{b}_1, \hat{b}_2, ..., \hat{b}_m\}$, where $\hat{b}_i$ is the weighted point defined in Section 2.3, i.e. $\hat{b}_i = (b_i, w(b_i))$ with $b_i \in B_{P,k}$ and $w(b_i) = -\frac{1}{k}\sum_{l=1}^{k}\|b_i - p_{il}\|^2$, where $p_{i,1},\ldots,p_{i,k} \in P$ are such that $b_i = \frac{1}{k}\sum_{j=1}^{k} p_{i,j}$. Applying Lemma 11 to $\hat{\sigma}$, we have that

$$\mathrm{rad}^2(\hat{\sigma}) \quad = \quad \frac{1}{2}\sum_{i,j\in I}\lambda_i\lambda_j D(\hat{b}_i,\hat{b}_j). \tag{8}$$

By Lemma 6, the distance between $\hat{p}_i$ and $\hat{p}_j$ is $D(\hat{b}_i,\hat{b}_j) = \frac{1}{k^2}\sum_{l,s=1}^{k}\|p_{i,l}-p_{j,s}\|^2$. As this last expression is a convex combination of squared pairwise distances of points in P, it is $(1\pm\varepsilon)$-preserved by any ε-distortion map with respect to P, which implies that the convex combination $\mathrm{rad}^2(\hat{\sigma}) = \frac{1}{2}\sum_{i,j\in I}\lambda_i\lambda_j D(\hat{p}_i,\hat{p}_j)$ corresponding to the squared radius of σ in $\mathbb{R}^D$, will be $(1\pm\varepsilon)$-preserved.

Let $f : \mathbb{R}^D \to \mathbb{R}^d$ be an ε-distortion map with respect to P, from $\mathbb{R}^D$ to $\mathbb{R}^d$, where d will be chosen later. By Lemma 11, the centre of $\widehat{f(\sigma)}$ is a convex combination of the points $(f(b_i))_{i=1}^m$. Let the centre $c(\widehat{f(\sigma)})$ be given by $c(\widehat{f(\sigma)}) = \sum_{i\in I}\nu_i D(\widehat{f(b_i)})$. where for $i \in I$, $\nu_i \geq 0$, $\sum_i \nu_i = 1$. Consider the convex combination of power distances $\sum_{i,j\in I}\nu_i\nu_j D(\hat{b}_i,\hat{b}_j)$. Since f is an ε-distortion map with respect to P, by Lemmas 6 and 3 we get

$$\frac{1}{2}(1-\varepsilon)\sum_{i,j\in I}\nu_i\nu_j D(\hat{b}_i,\hat{b}_j) \quad \leq \quad \frac{1}{2}\sum_{i,j\in I}\nu_i\nu_j D(\widehat{f(b_i)},\widehat{f(b_j)}) \quad = \quad \mathrm{rad}^2(\widehat{f(\sigma)}). \tag{9}$$

On the other hand, since the squared radius is a minimizing function by definition, we get that

$$\begin{aligned}
\mathrm{rad}^2(\hat{\sigma}) &= \frac{1}{2}\sum_{i,j\in I}\lambda_i\lambda_j D(\hat{b}_i,\hat{b}_j) \;\leq\; \frac{1}{2}\sum_{i,j\in I}\nu_i\nu_j D(\hat{b}_i,\hat{b}_j) && (10)\\
&\leq \frac{1}{(1-\varepsilon)}\mathrm{rad}^2(f(\sigma)),\ \text{by (9)}\\
\mathrm{rad}^2(\widehat{f(\sigma)}) &= \frac{1}{2}\sum_{i,j\in I}\nu_i\nu_j D(\widehat{f(b_i)},\widehat{f(b_j)}) && (11)\\
&\leq \frac{1}{2}\sum_{i,j\in I}\lambda_i\lambda_j D(\widehat{f(b_i)},\widehat{f(b_j)}). && (12)
\end{aligned}$$

Combining the inequalities (9), (10), (12) gives

$$(1-\varepsilon)\mathrm{rad}^2(\hat{\sigma}) \;\leq\; \mathrm{rad}^2(\widehat{f(\sigma)}) \;\leq\; \frac{1}{2}\sum_{i,j\in I}\lambda_i\lambda_j D(\widehat{f(b_i)},\widehat{f(b_j)}) \;\leq\; (1+\varepsilon)\mathrm{rad}^2(\hat{\sigma}).$$

where the final inequality follows by Lemma 3, since f is an ε-distortion map with respect to P. Thus, we have that

$$(1-\varepsilon)\mathrm{rad}^2(\hat{\sigma}) \;\leq\; \mathrm{rad}^2(\widehat{f(\sigma)}) \;\leq\; (1+\varepsilon)\mathrm{rad}^2(\hat{\sigma}),$$

which completes the proof of the theorem. ◀

Proof of Theorem 15. Recall that, in Section 2.2, we defined the approximate k-distance to be $\tilde{d}_{P,k}(x) := \min_{p\in P}\sqrt{D(x,\hat{p})}$, where $\hat{p} = (p, w(p))$ is a weighted point, having weight $w(p) = -d^2_{P,k}(p)$. So, the Čech complex would be formed by the intersections of the balls around the weighted points in P. The proof follows on the lines of the proof of Theorem 14. Let $\hat{\sigma} = \{\hat{p}_1, \hat{p}_2, ..., \hat{p}_m\}$, where $\hat{p}_1, \ldots, \hat{p}_m$ are weighted points in $\hat{P}$, and let $c(\hat{\sigma})$ be the center of $\hat{\sigma}$. Applying again Lemma 11, we get

$$\mathrm{rad}^2(\hat{\sigma}) = \frac{1}{2}\sum_{i,j\in I}\lambda_i\lambda_j\|p_i-p_j\|^2 + \sum_{i\in I}\lambda_i w(p_i) = \sum_{i,j\in I; i<j}\lambda_i\lambda_j\|p_i-p_j\|^2 + \sum_{i\in I}\lambda_i w(p_i),$$

where $w(p) = d^2_{P,k}(p)$. In the second equality, we used the fact that the summand corresponding to a fixed pair of distinct indices $i < j$ is being counted twice and that the contribution of the terms corresponding to indices $i = j$ is zero. An ε-distortion map with respect to P preserves pairwise distances and the k-distance in dimension $O(\varepsilon^{-2}\log n)$. The result then follows as in the proof of Theorem 14. ◀

5 Extensions

In this section we state and prove some extensions of Theorem 14 for dimensionality reduction, obtaining better bounds for the target dimension than in Section 3, in certain settings like point sets contained in regions of bounded Gaussian width, or in low-dimensional submanifolds of Euclidean space.

5.1 Sets of Bounded Gaussian Width

The first result in this section, is analogous to a theorem [25] for point sets contained in a region of bounded Gaussian width.

▶ **Definition 17.** *Given a set $S \subset \mathbb{R}^D$, the* Gaussian width *of S is*

$$w(S) := \mathbb{E}\left[\sup_{x \in S} \langle x, g \rangle\right],$$

where $g \in \mathbb{R}^D$ is a random standard D-dimensional Gaussian vector.

In several areas like geometric functional analysis, compressed sensing, machine learning, etc. the Gaussian width is a very useful measure of the width of a set in Euclidean space (see e.g. [19] and the references therein). It is also closely related to the *statistical dimension* of a set (see e.g. [32, Chapter 7]).

▶ **Theorem 18.** *Let $P \subset \mathbb{R}^D$ be a finite set of points, and define $S := \{(x-y)/\|x-y\| : x, y \in P\}$. Let $w(S)$ denote the Gaussian width of S. Then, given any $\varepsilon, \delta \in (0,1)$, for any $d \geq \frac{\left(w(S)+\sqrt{2\log(2/\delta)}\right)^2}{\varepsilon^2} + 1$, the map from $\mathbb{R}^D \to \mathbb{R}^d$, given by $x \mapsto \sqrt{D/d}Gx$, where $d = O\left(\frac{\log n}{\varepsilon^2}\right)$ and G is a $d \times D$ random Gaussian matrix, preserves the persistent homology of the Čech filtration associated to P, up to a multiplicative factor of $(1-\varepsilon)^{-1/2}$, with probability at least $1-\delta$.*

Note that since the Gaussian width of an n-point set is at most $O(\log n)$ (using e.g. the Gaussian concentration inequality, see e.g. [6, Section 2.5]), Theorem 18 strictly generalizes Theorem 14 (*ii*).

Proof of Theorem 18. We state an analogue of the Johnson Lindenstrauss lemma for sets of bounded Gaussian width, given in [21, Theorem 3.1], which essentially follows from a result of Gordon [21].

▶ **Theorem 19** ([25], Theorem 3.1)**.** *Given $\varepsilon,\ \delta \in (0,1)$, $P \subset \mathbb{R}^D$, let $S := \{(x-y)/\|x-y\| : x, y \in S\}$. Then for any $d \geq \frac{\left(w(S)+\sqrt{2\log(2/\delta)}\right)^2}{\varepsilon^2} + 1$, the function $f : \mathbb{R}^D \to \mathbb{R}^d$ given by $f(x) = \left(\sqrt{D/d}\right)Gx$, where G is a random $d \times D$ Gaussian matrix G, is an ε-distortion map with respect to P, with probability at least $1-\delta$.*

By Theorem 19, the scaled random Gaussian matrix $f : x \mapsto \left(\sqrt{D/d}\right)Gx$ is an ε-distortion map with respect to P, with target dimension $d \geq \frac{\left(w(S)+\sqrt{2\log(2/\delta)}\right)^2}{\varepsilon^2} + 1$. Now applying the first statement in Theorem 14 to the point set P with the mapping f, immediately gives us that for any simplex $\hat{\sigma} \in \check{C}_\alpha(\hat{B}_{P,k})$, where $\check{C}_\alpha(\hat{B}_{P,k})$ is the weighted Čech complex with parameter α, the squared radius $\mathrm{rad}^2(\hat{\sigma})$ is preserved up to a multiplicative factor of $(1 \pm \varepsilon)$. By Lemma 12, this implies that the persistent homology for the Čech filtration is $(1-\varepsilon)^{-1/2}$-multiplicatively interleaved. ◀

5.2 Submanifolds of Euclidean Space

For point sets lying on a low-dimensional manifold in a high-dimensional Euclidean space, one can obtain a better target dimension using the bounds of Baraniuk and Wakin [5] or Clarkson [12], which will depend only on the parameters of the manifold.

▶ **Theorem 20.** *There exists an absolute constant $c > 0$ such that, given a finite point set P lying on a connected, compact, orientable, differentiable μ-dimensional manifold $M \subset \mathbb{R}^D$, and $\varepsilon, \delta \in (0,1)$, a random projection map $f : \mathbb{R}^D \to \mathbb{R}^d$ preserves the persistent homology of the Čech filtraton computed on P, using the k-distance, with probability at least $1-\delta$, provided*

$$d \geq c\left(\frac{\mu \log(1/\varepsilon) + \log(1/\delta)}{\varepsilon^2} + \frac{C(M)}{\varepsilon^2}\right),$$

where $C(M)$ depends only on M.

Proof of Theorem. The proof is a direct application of Clarkson's bound [12] to Theorem 14 (i). Clarkson's theorem is stated below.

▶ **Theorem 21** (Clarkson [12])**.** *There exists an absolute constant $c > 0$ such that, given a connected, compact, orientable, differentiable μ-dimensional manifold $M \subset \mathbb{R}^D$, and $\varepsilon, \delta \in (0,1)$, any random projection map $f : \mathbb{R}^D \to \mathbb{R}^d$, is an ε-distortion map with respect to P, with probability at least $1-\delta$, for*

$$d \geq c\left(\frac{\mu \log(1/\varepsilon) + \log(1/\delta)}{\varepsilon^2} + \frac{C(M)}{\varepsilon^2}\right),$$

where $C(M)$ depends only on M.

Now the statement of Theorem 20 follows directly by applying Clarkson's theorem to Theorem 14 (i). ◀

6 Conclusion and Future Work

Vietoris-Rips and Delaunay filtrations. Since the Vietoris-Rips filtration [27, Chapter 4] depends only on pairwise distances, it follows from Theorem 13 that this filtration is preserved upto a multiplicative factor of $(1-\varepsilon)^{-1/2}$, under a Johnson-Lindenstrauss mapping. Furthermore, the Delaunay and the Čech filtrations [27, Chapter 4] have the same persistent homology. Theorems 14 (i) therefore implies that the Delaunay filtration of a given finite point set P is also $(1-\varepsilon)^{-1/2}$-preserved under an ε-distortion map with respect to P. Thus, theorems 14 (ii), 15, 18 and 20 apply also to the Vietoris-Rips and Delaunay filtrations.

Kernels. Other distance functions defined using kernels have proved successful in overcoming issues due to outliers. Using a result analogous to Theorem 13, we can show that random projections preserve the persistent homology for kernels up to a $C(1-\varepsilon)^{-1/2}$ factor where C is a constant. We don't know if we can make $C = 1$ as for the k-distance.

References

1 Dimitris Achlioptas. Database-friendly random projections. In Peter Buneman, editor, *Proceedings of the Twentieth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, May 21-23, 2001, Santa Barbara, California, USA*. ACM, 2001. doi:10.1145/375551.375608.

2 Nir Ailon and Bernard Chazelle. The fast johnson–lindenstrauss transform and approximate nearest neighbors. *SIAM J. Comput.*, 39(1):302–322, 2009. doi:10.1137/060673096.

3 Noga Alon and Bo'az Klartag. Optimal compression of approximate inner products and dimension reduction. In Chris Umans, editor, *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 639–650. IEEE Computer Society, 2017. doi:10.1109/FOCS.2017.65.

4 Franz Aurenhammer. A new duality result concerning voronoi diagrams. *Discret. Comput. Geom.*, 5:243–254, 1990. doi:10.1007/BF02187788.

5 Richard G. Baraniuk and Michael B. Wakin. Random projections of smooth manifolds. *Found. Comput. Math.*, 9(1):51–77, 2009. doi:10.1007/s10208-007-9011-z.

6 Stéphane Boucheron, Gábor Lugosi, and Pascal Massart. *Concentration Inequalities - A Nonasymptotic Theory of Independence*. Oxford University Press, 2013. doi:10.1093/acprof:oso/9780199535255.001.0001.

7 Mickaël Buchet, Frédéric Chazal, Steve Y. Oudot, and Donald R. Sheehy. Efficient and robust persistent homology for measures. *Comput. Geom.*, 58:70–96, 2016. doi:10.1016/j.comgeo.2016.07.001.

8 Mickaël Buchet, Tamal K. Dey, Jiayuan Wang, and Yusu Wang. Declutter and resample: Towards parameter free denoising. *JoCG*, 9(2):21–46, 2018. doi:10.20382/jocg.v9i2a3.

9 Frédéric Chazal, David Cohen-Steiner, and Quentin Mérigot. Geometric inference for probability measures. *Found. Comput. Math.*, 11(6):733–751, 2011. doi:10.1007/s10208-011-9098-0.

10 Frédéric Chazal, Vin de Silva, Marc Glisse, and Steve Y. Oudot. *The Structure and Stability of Persistence Modules*. Springer Briefs in Mathematics. Springer, 2016. doi:10.1007/978-3-319-42545-0.

11 Frédéric Chazal and Steve Oudot. Towards persistence-based reconstruction in euclidean spaces. In Monique Teillaud, editor, *Proceedings of the 24th ACM Symposium on Computational Geometry, College Park, MD, USA, June 9-11, 2008*, pages 232–241. ACM, 2008. doi:10.1145/1377676.1377719.

12 Kenneth L. Clarkson. Tighter bounds for random projections of manifolds. In Monique Teillaud, editor, *Proceedings of the 24th ACM Symposium on Computational Geometry, College Park, MD, USA, June 9-11, 2008*, pages 39–48. ACM, 2008. doi:10.1145/1377676.1377685.

13 Kenneth L. Clarkson and Peter W. Shor. Application of random sampling in computational geometry, II. *Discret. Comput. Geom.*, 4:387–421, 1989. doi:10.1007/BF02187740.

14 Sanjoy Dasgupta and Anupam Gupta. An elementary proof of a theorem of johnson and lindenstrauss. *Random Struct. Algorithms*, 22(1):60–65, 2003. doi:10.1002/rsa.10073.

15 Sjoerd Dirksen. Dimensionality reduction with subgaussian matrices: A unified theory. *Found. Comput. Math.*, 16(5):1367–1396, 2016. doi:10.1007/s10208-015-9280-x.

16 Herbert Edelsbrunner and John Harer. *Computational Topology - an Introduction*. American Mathematical Society, 2010. URL: http://www.ams.org/bookstore-getitem/item=MBK-69.

17 Herbert Edelsbrunner, David Letscher, and Afra Zomorodian. Topological persistence and simplification. *Discret. Comput. Geom.*, 28(4):511–533, 2002. doi:10.1007/s00454-002-2885-2.

18 Charles Fefferman, Sanjoy Mitter, and Hariharan Narayanan. Testing the manifold hypothesis. *Journal of the American Mathematical Society*, 29(4):983–1049, 2016. doi:10.1090/jams/852.

19 Simon Foucart and Holger Rauhut. *A Mathematical Introduction to Compressive Sensing*. Applied and Numerical Harmonic Analysis. Birkhäuser, 2013. doi:10.1007/978-0-8176-4948-7.

20 C. Giraud. *Introduction to High-Dimensional Statistics*. Chapman & Hall/CRC Monographs on Statistics & Applied Probability. Taylor & Francis, 2014. URL: https://www.crcpress.com/Introduction-to-High-Dimensional-Statistics/Giraud/p/book/9781482237948.

21 Y. Gordon. On milman's inequality and random subspaces which escape through a mesh in rn. In Joram Lindenstrauss and Vitali D. Milman, editors, *Geometric Aspects of Functional Analysis*, pages 84–106, Berlin, Heidelberg, 1988. Springer Berlin Heidelberg. doi:10.1007/BFb0081737.

22 Leonidas J. Guibas, Dmitriy Morozov, and Quentin Mérigot. Witnessed k-distance. *Discret. Comput. Geom.*, 49(1):22–45, 2013. doi:10.1007/s00454-012-9465-x.

23 Piotr Indyk, Rajeev Motwani, Prabhakar Raghavan, and Santosh S. Vempala. Locality-preserving hashing in multidimensional spaces. In Frank Thomson Leighton and Peter W. Shor, editors, *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, El Paso, Texas, USA, May 4-6, 1997*, pages 618–625. ACM, 1997. doi:10.1145/258533.258656.

24 William B Johnson and Joram Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. *Contemporary Mathematics*, 26(189-206):1, 1984. doi:10.1007/BF02764938.

25 Martin Lotz. Persistent homology for low-complexity models. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 475(2230):20190081, 2019. doi:10.1098/rspa.2019.0081.

26 Jiří Matoušek. On variants of the Johnson Lindenstrauss lemma. *Random Structures & Algorithms*, 33(2):142–156, 2008. doi:10.1002/rsa.20218.

27 Steve Y. Oudot. *Persistence Theory - From Quiver Representations to Data Analysis*, volume 209 of *Mathematical surveys and monographs*. American Mathematical Society, 2015. URL: http://bookstore.ams.org/surv-209/.

28 Jeff M. Phillips, Bei Wang, and Yan Zheng. Geometric inference on kernel density estimates. In Lars Arge and János Pach, editors, *31st International Symposium on Computational Geometry, SoCG 2015, June 22-25, 2015, Eindhoven, The Netherlands*, volume 34 of *LIPIcs*, pages 857–871. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2015. doi:10.4230/LIPIcs.SOCG.2015.857.

29 Tamás Sarlós. Improved approximation algorithms for large matrices via random projections. In *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006), 21-24 October 2006, Berkeley, California, USA, Proceedings*, pages 143–152. IEEE Computer Society, 2006. doi:10.1109/FOCS.2006.37.

30 Donald R. Sheehy. The persistent homology of distance functions under random projection. In Siu-Wing Cheng and Olivier Devillers, editors, *30th Annual Symposium on Computational Geometry, SOCG'14, Kyoto, Japan, June 08 - 11, 2014*, page 328. ACM, 2014. doi:10.1145/2582112.2582126.

31 Nakul Verma. A note on random projections for preserving paths on a manifold. Technical report, UC San Diego, 2011. URL: https://csetechrep.ucsd.edu/Dienst/UI/2.0/Describe/ncstrl.ucsd_cse/CS2011-0971.

32 Roman Vershynin. *High-Dimensional Probability: An Introduction with Applications in Data Science*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 2018. doi:10.1017/9781108231596.

33 Ji Zhang. Advancements of outlier detection: A survey. *EAI Endorsed Trans. Scalable Information Systems*, 1(1):e2, 2013. doi:10.4108/trans.sis.2013.01-03.e2.

Persistent Homology Based Characterization of the Breast Cancer Immune Microenvironment: A Feasibility Study

Andrew Aukerman
Department of Pathology & Cell Biology, Columbia University, New York, NY, United States
aa4542@cumc.columbia.edu

Mathieu Carrière
Department of Systems Biology, Columbia University, New York, NY, United States
mc4660@cumc.columbia.edu

Chao Chen
Department of Biomedical Informatics, Stony Brook University, NY, United States
chao.chen.1@stonybrook.edu

Kevin Gardner
Department of Pathology & Cell Biology, Columbia University, New York, NY, United States
klg2160@cumc.columbia.edu

Raúl Rabadán
Department of Systems Biology, Columbia University, New York, NY, United States
rr2579@cumc.columbia.edu

Rami Vanguri
Department of Pathology & Cell Biology, Columbia University, New York, NY, United States
r.vanguri@columbia.edu

Abstract

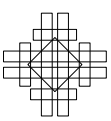
Persistent homology is a common tool of topological data analysis, whose main descriptor, the persistence diagram, aims at computing and encoding the geometry and topology of given datasets. In this article, we present a novel application of persistent homology to characterize the spatial arrangement of immune and epithelial (tumor) cells within the breast cancer immune microenvironment. More specifically, quantitative and robust characterizations are built by computing persistence diagrams out of a staining technique (quantitative multiplex immunofluorescence) which allows us to obtain spatial coordinates and stain intensities on individual cells. The resulting persistence diagrams are evaluated as characteristic biomarkers of cancer subtype and prognostic biomarker of overall survival. For a cohort of approximately 700 breast cancer patients with median 8.5-year clinical follow-up, we show that these persistence diagrams outperform and complement the usual descriptors which capture spatial relationships with nearest neighbor analysis. This provides new insights and possibilities on the general problem of building (topology-based) biomarkers that are characteristic and predictive of cancer subtype, overall survival and response to therapy.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Topological data analysis, persistence diagrams

Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.11

Funding *Mathieu Carrière*: research was partially supported by NIH T15LM007079-28.
Chao Chen: research was partially supported by NSF IIS-1909038, IIS-1855759, CCF-1855760.



36th International Symposium on Computational Geometry (SoCG 2020).
Editors: Sergio Cabello and Danny Z. Chen; Article No. 11; pp. 11:1–11:20

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

1 Introduction

Descriptors computed with topological data analysis (TDA), such as persistence diagrams [21, 60] and Mapper [56], have shown strong analytical power in many real world biological data. Examples include (but are not limited to) neuronal structures [41, 33], cardiac trabeculae [24, 59], brain images [46, 39] and genomics data [44, 12, 51]. These methods capture multi-scale geometric and structural patterns of the data with guaranteed robustness against potential noise introduced in measurement [17, 18] and in upstream preprocessing steps [7]. As such, they provide a systematic way to quantify complex biomedical systems. Furthermore, state-of-the-art discriminative models (i.e., classifiers) [11, 30, 34] and unsupervised models (i.e., clustering methods) [36] have been recently introduced, and are able to effectively connect topological features and clinical/biological outcomes of interest.

In this paper, we present a new application of topological data analysis, namely, the characterization of breast cancer immune microenvironment using persistence diagrams. Despite tremendous advancements in cancer screening, diagnostic methods and treatment, breast cancer remains the second leading cause of cancer death in women with projections of 270,000 new cases and approximately 42,000 deaths from invasive breast cancer in 2019 [53]. Therefore, identifying descriptors that indicate potential therapeutic targets and predict outcome is a critical yet unmet need in breast cancer [20]. The goal of this article is to show how persistence diagrams can help in fulfilling this task.

Cancer research and characterization of spatial cell arrangement. In the past decade, a major focus of cancer research has been on the interplay between the tumor and the immune environment, referred to as the *tumor immune microenvironment* [8]. By characterizing host-specific functional anti-tumor immune responses and their correlation to cancer subtype and overall survival, patient specific immunotherapeutic targets can be identified [49] with higher precision. To achieve the goal, it is necessary to characterize the complex spatial arrangement between cancer cells and a mixture of different immune cells, e.g., T-cells and macrophages, both of which play a versatile biological role and are believed to be crucially relevant to initiation and regulation of the immune response. This task involves two important steps: cell detection and characterization.

Thanks to the rapid development of imaging technology and deep learning methods, we are able to detect not only locations, but also types of different cells within a slide of tumor biopsy sample from a cancer patient. By staining the slide using immunohistochemical (IHC) markers, we are able to tag different types of cells with different stains, i.e., colors bounded with different protein biomarkers. Using a brightfield image scanner, we convert the stained slide into a whole slide image in which various cells can be identified by their respective stains [47, 32]. The identification of cells is referred to as *phenotyping*. Advanced deep learning methods [23, 1] have been developed to unmix the stains and to detect cells and their types. This approach, called multiplex IHC, is scalable but less precise as noise is introduced due to the additional deep learning cell detector. Alternatively, we may use quantitative multiplex immunofluorescence (qmIF), which stains different cells with different fluorescent stains and detect them using lenses with specific filters. The qmIF approach is highly reliable, albeit costly in material and in time.

Once cells of different types are detected, we need to quantitatively characterize their spatial arrangements in order to evaluate correlations with various outcomes of interest. There are two major challenges. First, the spatial arrangement is highly heterogeneous across different patients and even within a single tissue sample. Second, stain intensity is

relative, and phenotype thresholds must be manually determined. Discerning true signal from background isn't always clear, and currently is done in relation to other tissue samples. Nonetheless, qmIF imaging provides rich data for study; see Figure 1 for an example of the raw image data.

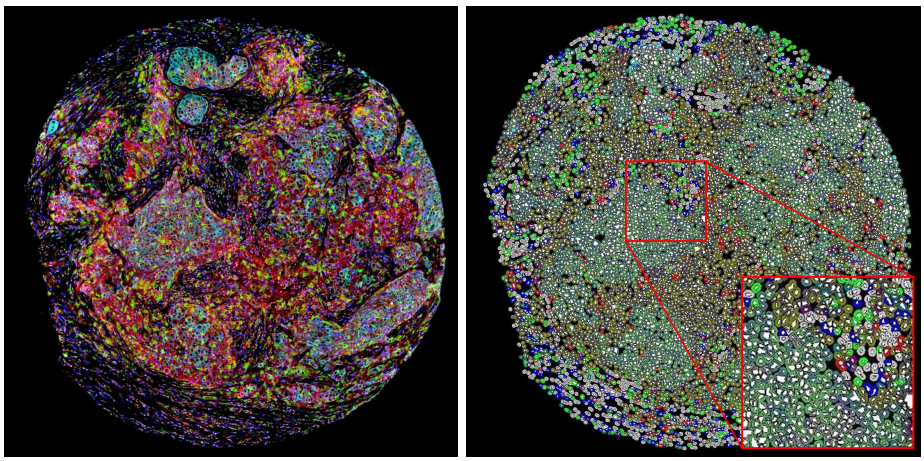


Figure 1 An example input data. Left: The raw microscopic image of a stained tissue sample. The sample is approximately 1x1 mm^2 large. The image is 2,000x2,000 pixels, 0.5x0.5 $micron^2$ per pixel. A sample usually contains 3,000 to 5,000 cells. Right: The processed results. Cells are identified by localizing their nuclei with a special stain (shown as white regions). The phenotype of each cell can be identified by the stain intensity of its cytoplasm and nucleus: T cells are tagged with CD8 (blue), macrophages are tagged with CD68 (green), tumor cells are tagged with pancytokeratin (cyan). Any cell may additionally be tagged with PD-L1 (red). The cells are abstracted into point clouds with different stain intensities, as shown in Figure 3.

Related work. Previous methods [25, 54] focus on using nearest neighbor distances from cells of one type (obtained by thresholding the stain intensities) to cells of a second type. Unfortunately, this approach is sensitive to noise and lacks the ability to model stain concentration variations due to the thresholding. Moreover, it can only characterize fixed neighborhoods around the cells and is oblivious to larger cell arrangements.

Persistent homology has recently been used to characterize cellular architecture in pathology images in [37], where these descriptors were shown to successfully detect and quantify circular cell structures corresponding to glands. In contrast, our work operates on coordinates of phenotyped cells and deals with the global characterization of complex interactions between these cellular phenotypes.

Contributions. In this article, *we propose the first topological analysis of tumor immune microenvironment.* More specifically, we provide empirical evidence that persistence diagrams are suitable descriptors by experimentally demonstrating the following points:

- First, stain concentration levels, or stain intensities, that are usually used by practitioners to filter cells, are natural candidates for defining *filtrations* (in the TDA vocabulary) from which persistence diagrams can be computed. This way, the whole range of stain intensities is taken into account instead of thresholding. We hypothesize that the stain intensity is biologically meaningful and the resulting persistence diagrams will be more predictive than just using cell coordinates from thresholding.

- Second, persistence diagrams are able to capture topological and structural features that are characteristic of the arrangement of the cells. This is because the structures encoded by persistence diagrams are robust to spatial deformation and other types of noise introduced in detection, which prevents the analysis from being biased by measurement errors, contrarily to other descriptors used in the literature.

Our study, although preliminary, demonstrates the potential of persistence homology being a novel tool to characterize the tumor immune microenvironment. With rich computation and learning tools available for persistence-derived features, we are confident that topological characterization will lead to powerful diagnostic and prognostic cancer biomarkers.

Plan of the article. We introduce our biological data, and briefly recall the basics of topological data analysis in Section 2. Then, we explain our methods for computing and running statistical tests on persistence diagrams in Section 3. Finally, we conclude and summarize future investigations and open questions in Section 4.

2 Data and Background

In this section, we introduce our biological data (Section 2.1), and briefly recall the rationale for nearest neighbor analysis (Section 2.2) and topological data analysis (Section 2.3).

2.1 Biological Data

We analyze a large cohort of patients with extensive 8.5 years of follow-up. For each tissue sample, qmIF imaging was obtained with a panel of immune markers for phenotyping the tumor immune microenvironment, including: CD8 (T-cells), CD68 (macrophages) and pancytokeratin (cancer cells). Then, a commercial software package (HALO, Indica Labs) was used to perform nuclear segmentation, cytoplasmic definition, and stain quantification. Cell phenotypes, based on a threshold applied to the stain intensity, were defined manually. See Figures 1 for the conventional threshold-based phenotype analysis. Let us now provide details on the important steps that were necessary to collect our data.

Patient Cohort. Our raw data is comprised of high-throughput tissue microarrays (TMA) consisting of 1mm $\times$ 1mm cores of tissue. The TMA were assembled with tissues from a cohort of 900 patients that underwent tumor resection following a diagnosis of breast cancer at Pitt County Memorial Hospital (now Vidant Hospital) in Greenville, North Carolina. Patient samples and clinicopathological data were collected under an IRB approved protocol at the Brody School of Medicine, East Carolina University [9]. The cohort is uniquely valuable for research as there is median 8.5 year follow-up data which allows for in depth evaluation for topological biomarkers with patient attributes and clinical outcomes.

Quantitative Multiplex Immunofluorescence. Unlike traditional immunohistochemistry, qmIF enables simultaneous staining of multiple markers in a single piece of tissue. We use the Ultivue UltiMapper I/O PD-L1 assay consisting of the following markers: CD8 (cytotoxic T-cells), CD68 (macrophages), PD-L1 (an immune suppressive protein), pancytokeratin (epithelial cells), and DAPI (DNA marker) for identification of cell nuclei. In our data, positively stained epithelial cells via pancytokeratin are considered to be tumor cells. Every cell in the tissue is designated with a PD-L1 status being either positive or negative corresponding to above or below threshold stain intensity. All staining thresholds are adaptively determined to

enhance signal (consistent with a positive staining pattern assessed visually) to background. The result of the phenotyping analysis is a text file for each tissue sample consisting of entries listing information about each cell location, including the manual phenotyping result and raw stain intensities. Each tissue sample consists of 3,000-5,000 cells.

2.2 Nearest Neighbor Analysis

Nearest neighbor analysis is commonly performed with qmIF data [25]. We perform a nearest neighbor search between combinations of phenotypes for all possible phenotype pairs. More specifically, for a given pair of phenotypes P_1, P_2, each composed of cells (detected with thresholds on their stain intensities) with two coordinates, we compute, for a given cell C_i belonging to P_1, the Euclidean distances to all cells belonging to P_2, excluding those whose distance is less than 0.05 microns to prevent cell-overlap. We keep the minimum distance value among those, which we call the nearest neighbor distance, and repeat this process for each cell in P_1 to form a distribution of nearest neighbor distances, d_i. The mean and standard deviation of d_i are then derived. We apply the same process for all pairs of phenotypes and used the corresponding means and deviations as features of biomarkers, potentially predictive of triple-negative status and prognostic of overall survival. This can also be written as a function of matrix operations involving the similarity matrix of cell coordinates between P_1 and P_2:

$$\begin{aligned}
\mathbf{q}_{ik} &= \{C_{1k}, C_{2k}, ..., C_{ik}\},\ \mathbf{t}_{kj} = \{C_{1k}, C_{2k}, ..., C_{jk}\}^{\mathrm{T}},\ k = 1, 2 \\
\mathbf{N}_{ij} &= (\mathrm{q}_k^2)_i + (\mathrm{t}_k^2)_j - 2\mathbf{q}_{ik}\mathbf{t}_{kj} \\
\mathrm{d}_i &= \sqrt{\min_j \mathbf{N}_{ij}}\ (\mathbf{N}_{ij} \geq 0.05)
\end{aligned}$$

2.3 Topological Data Analysis

In this article, we aim at characterizing the spatial arrangement of phenotypes using persistence diagrams, which are common descriptors of topological data analysis. Thus, we briefly recall, in this section, the basics of persistent homology and persistence diagrams. The interested reader can find a thorough treatment of persistence in several computational topology and algebraic topology textbooks such as [22, 14, 45].

Persistent homology. The aim of persistent homology is to encode the topological information contained in a dataset X through the lens of a *filter function* $f : X \to \mathbb{R}$. This is achieved by considering the *sublevel sets* of f: $F_\alpha = \{x \in X : f(x) \leq \alpha\}$. The family of sublevel sets $\mathcal{F} = \{F_\alpha\}_{\alpha \in \mathbb{R}}$ defines a *filtration*, i.e., a family of subsets of X that are nested with respect to the inclusion: $F_\alpha \subseteq F_\beta$ if $\alpha \leq \beta$. The idea of persistence is to track the topological changes occurring in the filtration as the sublevel set threshold α increases from $-\infty$ to $+\infty$. For instance, each time a *topological structure* such as a connected component, a handle or a void, appears in the sublevel set, we use the corresponding threshold as the so-called *birth time* for this structure. Similarly, each time a structure disappears in the sublevel set (think for instance of a handle being filled in after data points inside the handle were added to the sublevel set), we use the corresponding threshold as the *death time*. This tracking is eventually encoded in a *persistence diagram*, that we denote by $D(f)$, which is a set of dots in the Euclidean plane $\mathbb{R}^2$, each dot representing a topological structure whose birth and death times can be retrieved from the coordinates of the dot.

Persistence on images. In Figure 2, we provide an example of persistent homology computation performed on an image taken from the MNIST [38] dataset using the opposite of the pixel stain intensity as the filter function, so that it increases from white to black. Given a specific filter function value, the black pixels displayed in the top row of Figure 2 are those constituting the sublevel sets. One can see that at values b and d, handles are created in the union of black pixels, and they are eventually filled in at value e, for which the corresponding sublevel set includes all pixels. Other examples on our biological data are also displayed in Figures 5 and 6.

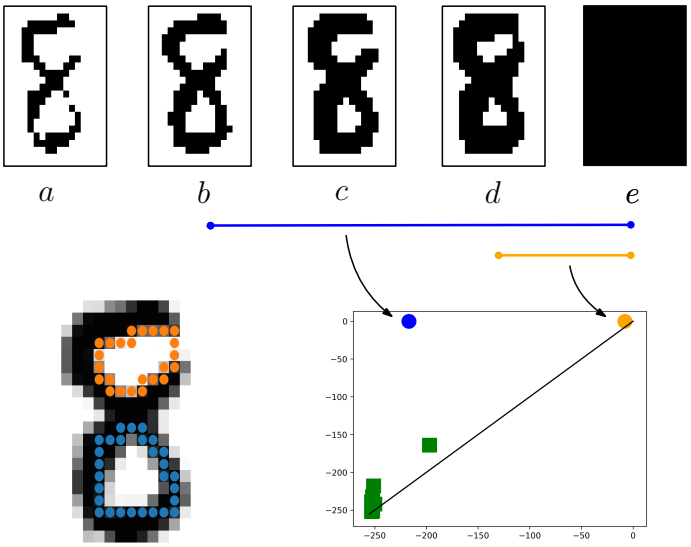


Figure 2 Example of a persistence diagram (lower right) computed on an image taken from the MNIST [38] dataset (lower left) using the opposite of the pixel stain intensity whose sublevel sets are displayed in the top row. Green squares represent connected components while the blue and orange circles represent handles, whose representative cycles are displayed on the original image.

Stability of persistence diagrams. One of the most useful properties of persistence diagrams is their *stability*: persistence diagrams computed from similar images must be similar themselves, w.r.t. the so-called *Wasserstein distances* between them.

▶ **Definition 1** ([14, 17]). *The* p-Wasserstein distance d_p *between two persistence diagrams* D, D' *is defined as:*

$$d_p(D, D')^p = \inf_\gamma \sum_{\mathrm{pt} \in D \cup \Delta} \|\mathrm{pt} - \gamma(\mathrm{pt})\|_\infty^p,$$

where Δ *is made of an infinite number of copies of the diagonal* $\{(x, x) : x \in \mathbb{R}\}$ *and* γ *ranges over all matchings between* $D \cup \Delta$ *and* $D' \cup \Delta$.

When the sum in Definition 1 is replaced by a maximum, the Wasserstein distance becomes the so-called *bottleneck distance* d_∞. Using this distance, one can state the *stability property* of persistence diagrams, which shows that the Wasserstein distance between persistence diagrams is upper bounded by the distance (in the $\|\cdot\|_\infty$ norm) between filter functions.

▶ **Theorem 2** ([13, 17]). *Given a topological space X and two continuous functions $f, g : X \to \mathbb{R}$, the following inequality is true:*

$$d_\infty(D(f), D(g)) \leq \|f - g\|_\infty \tag{1}$$

Note that similar stability results can be obtained with p-Wasserstein distances, with different upper bounds [45].

3 Methods and Results

In this section, we detail our methods to compute and analyze persistence diagrams from point clouds representing cells with different stain intensities. More specifically, we show how to discretize the cell domain into an image with stain intensity-valued pixels, from which we calculate the corresponding persistence diagrams (in homological dimension zero and one) in Section 3.1. Then, we show how to run statistical tests on persistence diagrams between different populations using Hilbert space embeddings with the *Sliced Wasserstein kernel* [11] in Section 3.2. Finally, we provide and discuss results for different patient groups (patients with different molecular subtypes, patients that survived after 8.5 years vs. deceased) in Section 3.3.

3.1 Persistence Diagrams of Cells with Stain Intensity values

In this section, we explain how persistence diagrams were computed on our point clouds representing cells so as to make use of the associated stain intensities.

Point clouds. As mentioned above, the image data for each patient is summarized in a point cloud, where the points represent cells, and have four associated stain intensities, corresponding to the CD8, CD68, PD-L1, and pancytokeratin (tumor) stains (see Section 2.1). Each patient also has two binary labels corresponding to overall survival and whether the cancer subtype is triple-negative. After removing samples with bad quality or missing labels, our final dataset is comprised of 671 point clouds. See Figure 3 for an example of such point clouds, where we only kept the cells with stain intensities above a certain threshold to ease visualization. One can see from these point clouds that different topological structures seem to emerge depending on the stain being considered: structures can be either isolated connected components corresponding to the scattered spots of cells exhibiting large stain intensity values (such as pancytokeratin (tumor) in Figure 3) or small cycles corresponding to regions where there are no cells with large stain intensity (such as CD8 in Figure 3). The lack of any discernible structure is also a possible feature if the stain intensity is diffuse across the whole tissue (such as PD-L1 in Figure 3).

Persistence Diagrams. It is common in topological data analysis to use Rips, Cech or Alpha filtrations [13] when dealing with point clouds. However, this would leave the stain intensity values aside and only provide information about the shape of the whole point cloud, which might not be sufficient to successfully encode the spatial and geometrical relationships between phenotypes.

Hence, in order to take the stain intensities into account when computing topological descriptors, we first discretized the plane into a grid of 40×40 pixels. Next, we binned the stain intensity values on this grid, so as to obtain an image. Note that the choice of resolution (i.e. the number of pixels) has to be carefully done: if the number of pixels is too

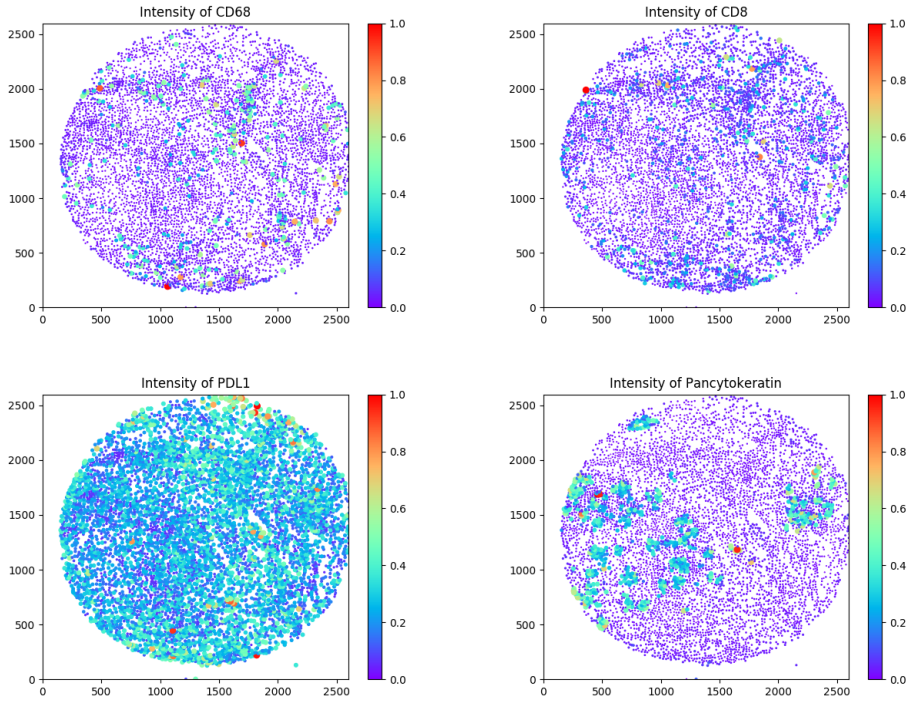


Figure 3 Illustration of the point clouds corresponding to the different stains (cell color and size is proportional to stain intensity to ease visualization). One can see that the different stain intensities induce different geometric patterns.

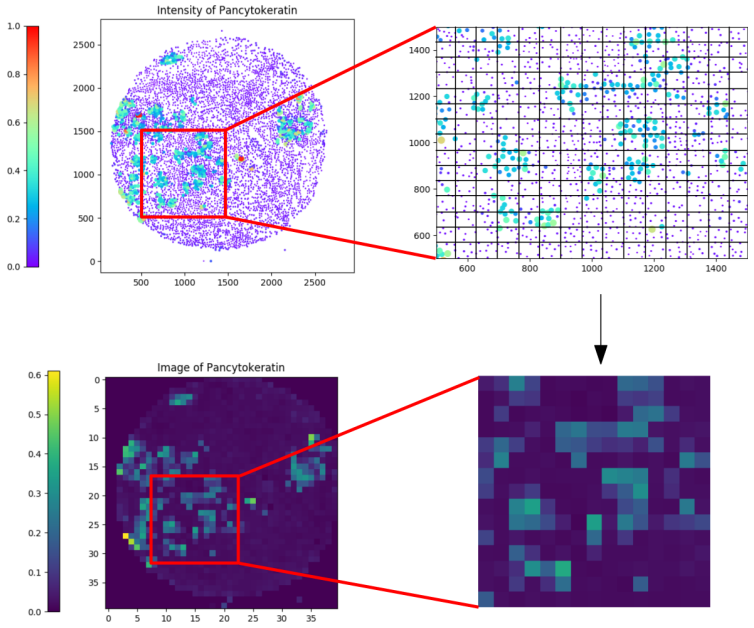


Figure 4 Discretization process turning a point cloud with stain intensity values into an image. We start with the full point cloud with the corresponding stain intensity values (upper left). Note that we only show cells above a certain stain intensity threshold to ease visualization. The cells are then placed into pixels of a grid drawn on top of the plane (upper right). These pixels with the corresponding stain intensity values are then turned into an image (down right and left).

small, one might not be able to see and compute the topological structures, but in the other hand, a resolution that is too large would induce artifacts, in the sense that all cells would be isolated, and no interesting topology could be computed. Our resolution of 40×40 pixels was manually chosen and seemed to be the best tradeoff on our data. See Figure 4 for an illustration of this process. Note also that it would be interesting to use Nadaraya-Watson kernel-based estimators (see Chapter 6 in [29]) to smooth the stain intensities of the pixels, but we left this possibility for future work.

Finally, we used persistent homology (see Section 2.3) to produce persistence diagrams out of our stain intensity-based images, by filtering the pixels with the opposite of the stain intensity (so that pixels with large stain intensity appear first). Note that points with ordinate 0 corresponds to topological structures that disappeared when adding the pixels with stain intensity 0, i.e., the pixels corresponding either to the cells not belonging to the corresponding phenotype or to pixels with no associated cells. These points should thus not be considered characteristic of the corresponding phenotype. See Figure 5 for examples of such persistence diagrams. One can see from these images that some patterns in the persistence diagrams, such as the distance-to-the-diagonal of points in homological dimension 0, or the number of points in homological dimension 1, seem to be correlated with how diffuse the cells with large stain intensity values are within the image.

Pairs of phenotypes. As mentioned in Section 1, characterizing the interactions, or co-localizations, between pairs of phenotypes might be as important, if not more, as characterizing them alone. Hence, we also computed persistence diagrams out of images with pixels colored by the average of pairs of phenotypes. This can be thought of as a similar but quite more general measure of co-localization than the one given by nearest neighbors (see Section 2.2). Indeed, the standard nearest neighbors analysis basically ranks the cells with respect to the distance to their closest neighbor. In terms of persistence, this ranking can be retrieved from the pixel filtration values: the lower they are, the more the corresponding pixels are likely to contain cells that co-localize from the two phenotypes. However, persistence diagrams also encode the interactions between the topological structures that are born from these co-localization spots. See Figure 6 for examples of such persistence diagrams. One can see from these images that the topological structures that are present in the image of a pair of phenotypes roughly include those of each phenotype alone, and that the structures that co-localize are emphasized.

Robustness. From a theoretical point of view, the stability property that persistence diagrams enjoy (see Section 2.3 and Proposition 2) is very advantageous. Indeed, it is well-known that any nearest neighbors analysis is sensitive to measurement errors: even a slight mistake in the measurement of stain intensity can induce different phenotype assignments for the cells, and thus different outputs from a nearest neighbor analysis. Since we do not depend on thresholding to compute persistence diagrams, we avoid this issue. On the other hand, the stability theorem for persistence diagrams ensures that any measurement error only has a small effect, provided that the error is small itself.

3.2 Statistics on Persistence Diagrams

In this section, we provide details about the statistical methods we used to assess the efficiency of persistence diagrams as characteristic and predictive biological descriptors.

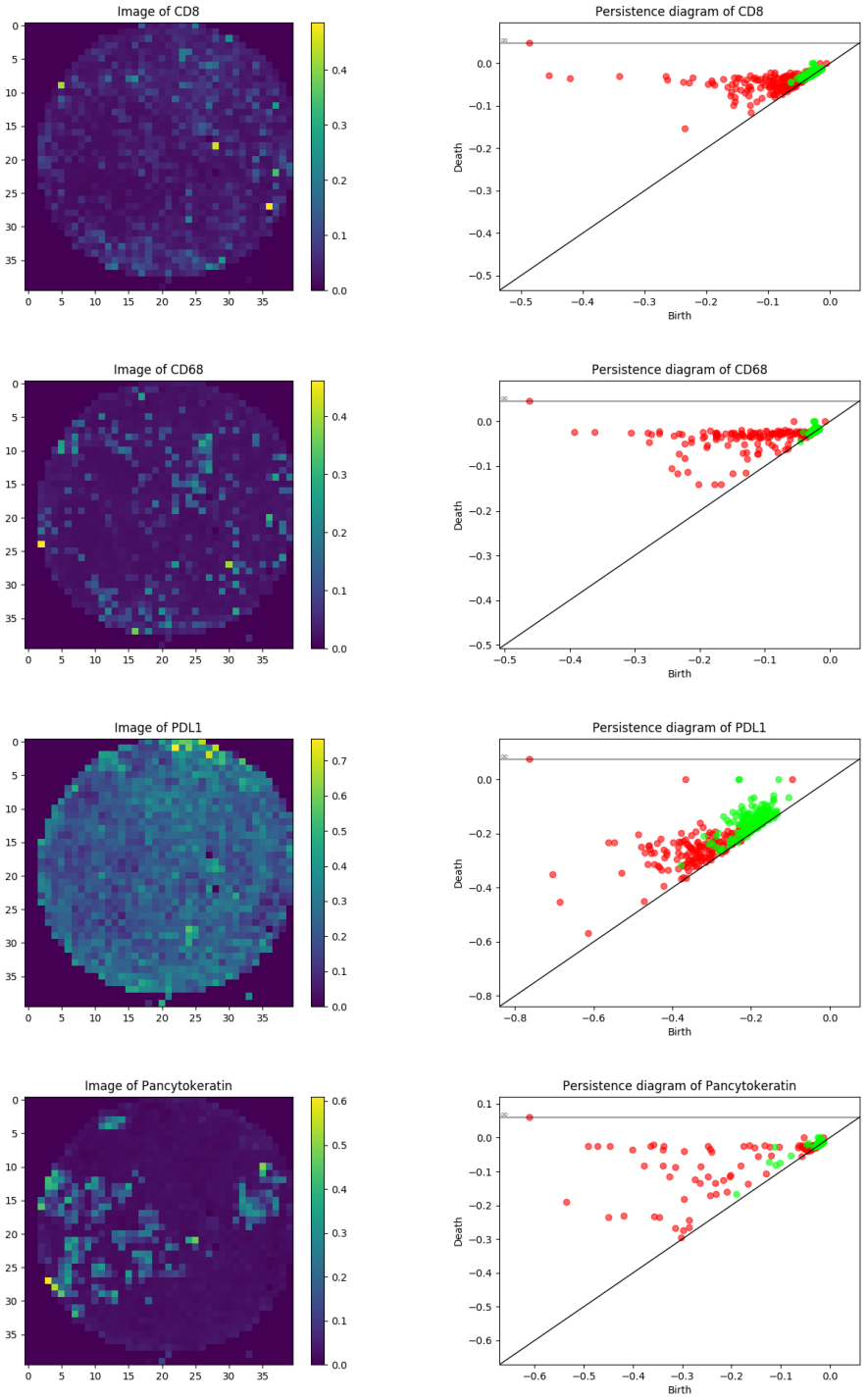


Figure 5 Examples of images with stain intensity-based pixels computed from point clouds (left) and their corresponding persistence diagrams (right). Points in homological dimension 0 are displayed in red and points in homological dimension 1 are displayed in green. From top to bottom: stains of CD8, CD68, PD-L1 and pancytokeratin.

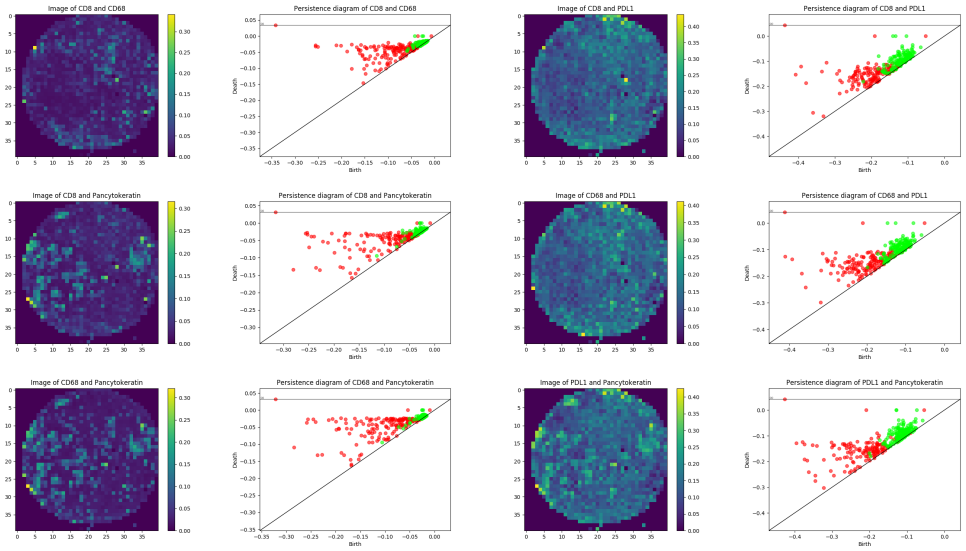


Figure 6 Examples of images and associated persistence diagrams computed from pairs of phenotypes/stains. Points in homological dimension 0 are displayed in red and points in homological dimension 1 are displayed in green.

Kernel-based Statistical Tests. In order to formally assess the statistical power of persistence diagrams with respect to the groups of interest, such as survived vs. not-survived, or triple-negative cancer subtype vs. other subtype, we need to be able to run statistical tests on distributions of persistence diagrams. Several recent works have looked at this question from a theoretical point of view [35, 52, 58]. In this article, we focus on *Kernel Mean Embeddings* [26], that is, we characterize a sample of a distribution $\mathcal{D}$ of persistence diagrams $\hat{\mathcal{D}}_n = \{D_1, \cdots, D_n\}$ by embedding the diagrams in a Hilbert space $\mathcal{H}$ with a continuous map Φ, and by taking the mean (in the Hilbert space) of this sample: $\Phi(\hat{\mathcal{D}}_n) := \frac{1}{n}\sum_{i=1}^{n}\Phi(D_i)$.

Now, given two samples $\hat{\mathcal{D}}_n$ and $\hat{\mathcal{D}}'_n$, one can compute the statistic:

$$\mathrm{MMD}(\hat{\mathcal{D}}_n, \hat{\mathcal{D}}'_n) := \|\Phi(\hat{\mathcal{D}}_n) - \Phi(\hat{\mathcal{D}}'_n)\|_{\mathcal{H}},$$

also called the *maximum mean discrepancy*, and use it to perform statistical tests in order to check whether $\mathcal{D}$ and $\mathcal{D}'$ are the same. This statistic has been shown to be a good proxy, with quantified approximation bounds, to its continuous version $\|\Phi(\mathcal{D}) - \Phi(\mathcal{D}')\|_{\mathcal{H}}$ in [26], where $\Phi(\mathcal{D})$ is defined as $\mathbb{E}_{D\sim\mathcal{D}}[\Phi(D)]$.

Choice of the embedding function. It might not be totally clear how to choose such a map Φ for embedding persistence diagrams. This can actually be done quite easily with the use of *kernels*:

▶ **Definition 3.** *Let $\mathcal{D}_{N,L}$ be the space of persistence diagrams with at most N points included in $[-L, L]^2$. A* kernel *is a pairwise function $k : \mathcal{D}_{N,L} \times \mathcal{D}_{N,L} \to \mathbb{R}$ such that the matrix $K = ((k(D_i, D_j)))_{1\leq i,j\leq n}$ is positive semi-definite for any family of persistence diagrams $D_1, \cdots, D_n \in \mathcal{D}_{N,L}$.*

A useful result of kernel methods actually relates kernels to embeddings in Hilbert spaces:

▶ **Proposition 4.** *Let k be a kernel on $\mathcal{D}_{N,L}$. Then, there exists a Hilbert space $\mathcal{H}_k$ and a map Φ_k such that, for any $D, D' \in \mathcal{D}_{N,L}$, one has $k(D, D') = \langle\Phi(D), \Phi(D')\rangle_{\mathcal{H}_k}$.*

In other words, any kernel matrix can be interpreted as a Gram matrix in an implicit (and potentially infinite-dimensional) Hilbert space. Moreover, the statistic MMD can be easily computed from k with:

$$\begin{aligned}\mathrm{MMD}(\hat{\mathcal{D}}_n, \hat{\mathcal{D}}'_n)^2 &= \left\langle \frac{1}{n}\sum_{i=1}^{n}\Phi(D_i) - \frac{1}{m}\sum_{j=1}^{m}\Phi(D'_j),\ \frac{1}{n}\sum_{i=1}^{n}\Phi(D_i) - \frac{1}{m}\sum_{j=1}^{m}\Phi(D'_j)\right\rangle_{\mathcal{H}_k} \\ &= \frac{1}{n^2}\sum_{i=1}^{n}\sum_{u=1}^{n}\langle\Phi(D_i),\Phi(D_u)\rangle_{\mathcal{H}_k} + \frac{1}{m^2}\sum_{j=1}^{m}\sum_{v=1}^{m}\langle\Phi(D'_i),\Phi(D'_v)\rangle_{\mathcal{H}_k} \\ &\qquad - \frac{2}{nm}\sum_{i=1}^{n}\sum_{j=1}^{m}\langle\Phi(D_i),\Phi(D'_j)\rangle_{\mathcal{H}_k} \\ &= \frac{1}{n^2}\|K\|_1 + \frac{1}{m^2}\|K'\|_1 - \frac{2}{nm}\|\tilde{K}\|_1,\end{aligned}$$

where K, K' and $\tilde{K}$ are the kernel matrices computed on $\mathcal{D}\times\mathcal{D}$, $\mathcal{D}'\times\mathcal{D}'$, and $\mathcal{D}\times\mathcal{D}'$ respectively. Note however that it has been shown in [26] that MMD is a biased statistic – in practice, we compute the *unbiased* MMD, defined as:

$$\begin{aligned}\mathrm{MMD}_u(\hat{\mathcal{D}}_n, \hat{\mathcal{D}}'_n)^2 &= \frac{1}{n(n-1)}\sum_{\substack{i=1\\u\neq i}}^{n}\langle\Phi(D_i),\Phi(D_u)\rangle_{\mathcal{H}_k} + \frac{1}{m(m-1)}\sum_{\substack{j=1\\v\neq j}}^{m}\langle\Phi(D'_i),\Phi(D'_v)\rangle_{\mathcal{H}_k} \\ &\qquad - \frac{2}{nm}\sum_{i=1}^{n}\sum_{j=1}^{m}\langle\Phi(D_i),\Phi(D'_j)\rangle_{\mathcal{H}_k}\end{aligned}$$

Now it only remains to pick a kernel for persistence diagrams. Several choices have been proposed in recent works [2, 6, 11, 34, 50], and we will focus on one called the *Sliced Wasserstein kernel* k_{SW} [11] in this work, since it has been shown to be one of the most efficient approach in different statistical tasks [11]. Its definition is based on the *Sliced Wasserstein distance* SW between persistence diagrams, which is defined (informally) as the integral over all possible lines of the 1-Wasserstein distance (see Section 2.3) computed between projections of these diagrams onto a line going through the origin. In practice, one does not compute this integral exactly but rather samples a fixed number of lines, finding the average Wasserstein distance between the corresponding projections. We refer the interested reader to [11] for a precise definition of this distance, and we merely recall the definition of the associated kernel:

▶ **Definition 5** ([11]). *Let $D, D' \in \mathcal{D}_{N,L}$ and $\sigma > 0$. The* Sliced Wasserstein kernel *is:*

$$k_{\mathrm{SW}}(D, D') = \mathrm{e}^{-\frac{\mathrm{SW}(D,D')}{\sigma^2}},$$

where SW *denotes the Sliced Wasserstein distance between persistence diagrams.*

One can easily see that k_{SW} can be interpreted as a Gaussian kernel, with its only parameter σ being the corresponding bandwidth.

Characteristic kernels. There is a specific class of kernels in the literature that is of particular interest when it comes to statistical tests: the so-called *characteristic* kernels [57, 55].

▶ **Definition 6.** *A kernel k is called* characteristic *if its corresponding map Φ_k is injective on distributions, i.e., for any pair of distributions $\mathcal{D}$ and $\mathcal{D}'$, one has:*

$$\|\Phi(\mathcal{D}) - \Phi(\mathcal{D}')\|_{\mathcal{H}_k} = 0 \Longrightarrow \mathcal{D} = \mathcal{D}'$$

Obviously, any statistical test based on a kernel requires it to be characteristic in order to be theoretically backed-up. Even though it is not clear whether the Sliced Wasserstein kernel is characteristic or not, there exists a strategy to build a characteristic kernel out of another one, that was first presented in [35], and that we use again in this work:

▶ **Theorem 7** ([35])**.** *Let k be a kernel on $\mathcal{D}_{N,L}$ whose associated map Φ_k is continuous and injective and whose associated Hilbert space $\mathcal{H}_k$ is separable. Then the kernel $\tilde{k} := \mathrm{e}^k$ is a characteristic kernel.*

Theorem 7 is actually a consequence of a more general theorem that is valid on any compact metric space (the fact that $\mathcal{D}_{N,L}$ is compact, with respect to the first Wasserstein distance between persistence diagrams, was proved in [35]). Moreover, it has been shown in [11] that the map $\Phi_{k_{\mathrm{SW}}}$ associated to k_{SW} is continuous and injective. Finally, since it is also known that $\mathcal{D}_{N,L}$ is separable [43], it follows that the Hilbert space associated to k_{SW} is separable as well, as the completion of the span of a separable space. Hence the following result:

▶ **Proposition 8.** *The kernel $\tilde{k}_{\mathrm{SW}} := \mathrm{e}^{k_{\mathrm{SW}}}$ is characteristic.*

All of the statistical analysis presented in the following section has been performed with the kernel $\tilde{k}_{\mathrm{SW}}$, that we call the *characteristic Sliced Wasserstein kernel*.

Comparison with NN features. Concerning the features given by nearest neighbors analysis, i.e., the means and variances of the distribution of Euclidean distances to the closest neighbors (see Section 2.2), we use kernel-based statistical tests based on the MMD computed with a standard Gaussian kernel (which is known to be characteristic). Moreover, we also test the independence between persistence diagrams and nearest neighbors features in order to check whether these two types of features are complementary or not. Again, kernel methods can be used to define the correlation between features living in different spaces. The so-called *constrained covariance* (COCO for short) [27] is defined as:

$$\mathrm{COCO}(\hat{\mathcal{D}}_n^X, \hat{\mathcal{D}}_n^Y) = \frac{1}{n}\sqrt{\|\tilde{K}_X \tilde{K}_Y\|_2},$$

where $\hat{\mathcal{D}}_n^X$ (resp. $\hat{\mathcal{D}}_n^Y$) is a sample from a distribution in a space X (resp. Y), $\tilde{K}_X$ (resp. $\tilde{K}_Y$) is the centered (i.e., multiplied with $\mathbf{I} - \frac{1}{n}\,\mathbf{1}\mathbf{1}^T$) version of the kernel matrix K_X (resp. K_Y) computed on $\hat{\mathcal{D}}_n^X$ (resp. $\hat{\mathcal{D}}_n^Y$), and $\|\cdot\|_2$ is the largest singular value. It has been shown in [27] that the COCO can be used as a general measure of correlation (for random variables that are not directly comparable), since having a null COCO is equivalent to being independent (see Theorem 6 in [27]) for characteristic kernels[1].

[1] The cited result is actually proved for the so-called *universal kernels* but we leave this subtlety aside in the context of this work since it has no effect on our analysis

3.3 Results

We focus on statistical significance between populations of patients instead of building a classifier. This is due to the lack of tissue area and access to tissue heterogeneity typically available in whole-slide images which are typically used for diagnosis. In an ongoing analysis, we are expanding the analysis for the same patients to whole slide imaging, where the point clouds will be $\approx 400\times$ larger. In this section, we provide the experimental results obtained on our data using the *characteristic Sliced Wasserstein kernel* $\tilde{k}_{\mathrm{SW}}$ presented in Section 3.2 for persistence diagrams and a standard Gaussian kernel for the NN features. For both types of descriptors, the kernel bandwidth was selected manually as the median of all pairwise distances (the distances used being the Sliced Wasserstein distance for persistence diagrams and the Euclidean distance for NN features). Moreover, the p-values were computed with $2 \cdot 10^3$ random permutations. The individual sample labels were shuffled and p-values were calculated from the rank of the true labels.

Triple-negative subtype. In this first experiment, we separate the patients with respect to their cancer subtypes. More specifically, we aim at distinguishing between patients diagnosed with triple-negative breast cancer and those with other subtypes. Triple-negative breast cancer is especially interesting due to its high ability to provoke an immune response, or immunogenecity, among subtypes. However, triple-negative breast cancer patients typically have poor prognosis due to the lack of response to hormonal or receptor-status therapy. By better understanding the immune profiles associated with triple-negative breast cancers and the association with treatment response (i.e. overall survival), it could be possible to design targeted immunotherapies [42].

We show in Figure 7 (left) the p-values obtained with persistence diagrams, and the ones computed with NN features, for each (pair of) phenotypes. It can be seen from this plot that the p-values obtained with persistence diagrams are most of the time comparable to those given by NN features, with the exception of CD8 and the CD8-pancytokeratin pair. We find that the NN metrics are not significant, and this was further verified with the full NN distribution shapes. On the other hand, persistence diagrams demonstrated consistency of the p-values including CD8-involved pairs, indicating they reveal topology beyond that quantified by the NN algorithm. Moreover, 1-dimensional persistence diagrams, which encode higher-order interactions between the phenotypes (that cannot be retrieved from NN analysis), also seem to be statistically more efficient than their 0-dimensional counterparts.

Survival. In this second experiment, we now aim at distinguishing between patients that were alive at the latest follow-up after diagnosis. Although this includes causes unrelated to the breast cancer morbidity and associated treatment, such as dying of natural causes or other disease, this is still a good measure of overall disease-free survival. The corresponding p-values are displayed in Figure 7 (right). It can be seen that the p-values corresponding to persistence diagrams are in general much lower than those corresponding of NN features, especially in PD-L1 involved pairs. PD-L1 combinations are relatively rare and, as explained at the end of Section 3.1, NN features are sensitive to noise and the counting statistics on the number of phenotype pairs. Characterizing the spatial interactions of PD-L1 expression, however, would provide valuable insight into the possible immuno-repressive patterns in the tumor immune microenvironment.

We see, on the other hand, stability of persistence diagrams providing statistically significant measures. This makes diagrams a more robust descriptor than NN alone at the same statistical power. Similarly, it is clear from the distribution of values that persistence diagrams are more stable descriptors than NN features, picking up topology relating to PD-L1.

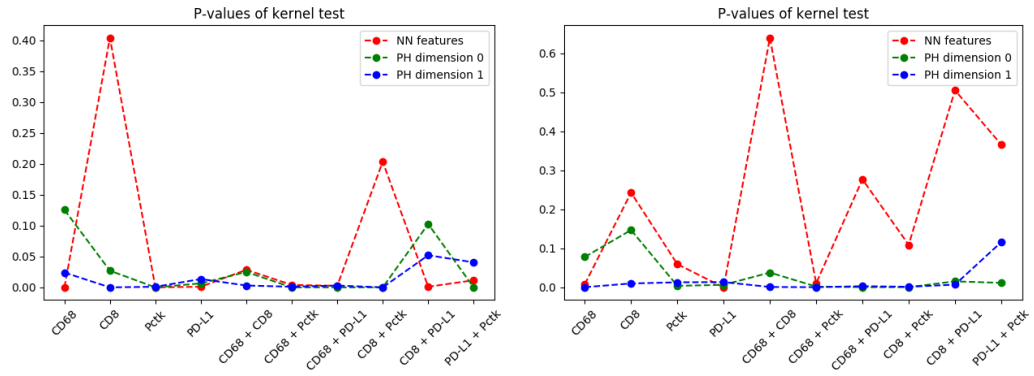


Figure 7 P-values computed by kernel-based statistical tests for NN features (red), 0-dimensional persistence diagrams (green) and 1-dimensional persistence diagrams (blue), for different (pairs of) phenotypes. "Pancytokeratin" has been abbreviated to "Pctk".

Correlation. Finally, we check the correlation values, as measured with COCO, between the NN features and persistence diagrams. We show the computed values in Figure 8. One can see that the correlation is always less than 0.1, which indicates that these features are almost statistically completely independent, and thus complementary. Moreover, these correlations seem to be oblivious to homological dimension since the shape of the curves for 0- and 1-dimensional persistence diagrams is roughly the same.

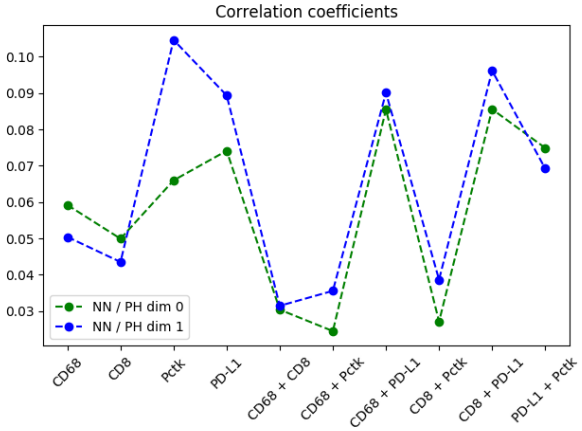


Figure 8 Correlation coefficients, measured with COCO, between NN features and 0-dimensional persistence diagrams (green), and 1-dimensional persistence diagrams (blue).

4 Open Questions and Future Work

We presented a novel approach for cancer research through the analysis of qmIF data using persistent homology, evaluated our method on a unique cohort of 671 patients using high-throughput tumor microarrays with a median 8.5 year follow-up. Our preliminary analyses show that features derived from persistent homology between groups of patients stratified by survival and triple negative status are statistically significant and are complementary to the state-of-the-art nearest neighbor approach. This indicates that the persistent homology features can be used as a complementary biomarker. Although the features do not separate the groups well enough to form a viable classifier, our results indicate feasibility of strong classification results in future work using more tissue area and larger associated point clouds. We are actively pursuing this by performing qmIF on tissue sections that contain 400× more area. If a successful classifier can be built, it could be possible to characterize patient immune profiles to build specific treatments. It could also be possible to use features derived from persistent homology to study functional breast cancer dynamics. For example, the relationship between persistence diagrams and other biological data such as proteomics or genomic sequencing could reveal factors that play a role in cancer initiation or progression.

Open questions. Our preliminary study is by no means comprehensive, and many questions remain open. Here is a list of the future investigations that we plan to work on:

- We only considered single phenotypes and pairs of phenotypes. However, one might be interested in the interactions between more than two phenotypes, although this would greatly increase the number of persistence diagrams computed for each patient. Moreover, there is no single solution on how to combine the different stain intensities. In this work, we merely took the average between normalized stain intensities, even though it would be interesting to weight the filtrations given by stain intensities in order to take the range of stain intensity values into account. The weight coefficients could even be learned so as to avoid a brute force search, using for instance recent works on differentiability of persistence diagrams for learning [5, 15, 31, 48].
- More generally, the question of turning a point cloud with different stain intensity values into one (or more) persistence diagram has many different solutions, the most natural one being to use Alpha or Rips filtrations, even though it would not be satisfactory since stain intensity values would be left aside. In this work, we built images with fixed resolution, that is, number of pixels, on top of the point clouds and used these images to compute persistence. However, other choices of filtrations are possible. For instance, one could think of constructing a graph on top of the point cloud, such as a δ-neighborhood graph, and then filter this graph with the stain intensity values on the nodes. Note that the δ parameter actually plays the role of the resolution of the image.
- Multiple stain intensities actually fits into the multiparameter persistence framework [10, 28], where data is filtered by several filtrations at the same time. Our approach of taking linear combinations of stain intensities actually amounts to draw lines in this multiparameter space and compute usual persistence along this line, which is the approach that is also advocated in recent works [19, 40]. However, multiparameter persistence is a current area of research, and invariants have been obtained in recent works, at least for bifiltrations, that is, filtrations with two parameters [3, 4, 16]. Even though they are harder to encode than persistence diagrams, it might be interesting to apply these results in our context.

- There are many different choices available when it comes to computing statistics on persistence diagrams. In this work, we restricted to kernel-based approaches, even though many other choices are available, see for instance [2, 6, 34, 50]. Overall, statistics and machine learning with persistence diagrams is also a current area of research, with new methods appearing regularly.

References

1 Shahira Abousamra, Danielle Fassler, Le Hou, Yuwei Zhang, Rajarsi Gupta, Tahsin Kurc, Luisa F. Escobar-Hoyos, Dimitris Samaras, Beatrice Knudson, Kenneth Shroyer, Joel Saltz, and Chao Chen. Weakly-supervised deep stain decomposition for multiplex ihc images. In *IEEE International Symposium on Biomedical Imaging (ISBI)*, 2019.

2 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), 2017.

3 Magnus Botnan and William Crawley-Boevey. Decomposition of persistence modules. *arXiv*, November 2018. `arXiv:1811.08946`.

4 Magnus Botnan and Michael Lesnick. Algebraic stability of zigzag persistence modules. *Algebraic and Geometric Topology*, 18(6):3133–3204, October 2018.

5 Rickard Brüel-Gabrielsson, Bradley Nelson, Anjan Dwaraknath, Primoz Skraba, Leonidas Guibas, and Gunnar Carlsson. A topology layer for machine learning. *arXiv*, May 2019. `arXiv:1905.12200`.

6 Peter Bubenik. Statistical topological data analysis using persistence landscapes. *Journal of Machine Learning Research*, 16(77):77–102, 2015.

7 Mickaël Buchet, Frédéric Chazal, Steve Y Oudot, and Donald R Sheehy. Efficient and robust persistent homology for measures. *Computational Geometry*, 58:70–96, 2016.

8 Samantha Burugu, Karama Asleh-Aburaya, and Torsten O Nielsen. Immune infiltrates in the breast cancer microenvironment: detection, characterization and clinical implication. *Breast Cancer*, 24(1):3–15, 2017.

9 Jung Byun, Sandeep Singhal, Samson Park, IK Dae, Ambar Caban, Nasreen Vohra, Eliseo Perez-Stable, Anna Napoles, and Kevin Gardner. Transcription regulatory networks associated with luminal master regulator expression and breast cancer survival, 2019.

10 Gunnar Carlsson and Afra Zomorodian. The theory of multidimensional persistence. *Discrete and Computational Geometry*, 42(1):71–93, July 2009.

11 Mathieu Carrière, Marco Cuturi, and Steve Oudot. Sliced Wasserstein kernel for persistence diagrams. In *International Conference on Machine Learning*, volume 70, pages 664–673, July 2017.

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

13 Frédéric Chazal, David Cohen-Steiner, Marc Glisse, Leonidas Guibas, and Steve Oudot. Proximity of persistence modules and their diagrams. In *International Symposium on Computational Geometry*, page 237, 2009.

14 Frédéric Chazal, Vin de Silva, Marc Glisse, and Steve Oudot. *The structure and stability of persistence modules*. Springer International Publishing, 2016.

15 Chao Chen, Xiuyan Ni, Qinxun Bai, and Yusu Wang. A topological regularizer for classifiers via persistent homology. In *International Conference on Artificial Intelligence and Statistics*, pages 2573–2582, 2019. URL: `http://proceedings.mlr.press/v89/chen19g.html`.

16 Jérémy Cochoy and Steve Oudot. Decomposition of exact pfd persistence bimodules. *arXiv*, May 2016. `arXiv:1605.09726`.

17 David Cohen-Steiner, Herbert Edelsbrunner, and John Harer. Stability of persistence diagrams. *Discrete & Computational Geometry*, 37(1):103–120, January 2007.

18 David Cohen-Steiner, Herbert Edelsbrunner, John Harer, and Yuriy Mileyko. Lipschitz functions have l p-stable persistence. *Foundations of computational mathematics*, 10(2):127–139, 2010.

19 René Corbet, Ulderico Fugacci, Michael Kerber, Claudia Landi, and Bei Wang. A kernel for multi-parameter persistent homology. *Computers & Graphics: X*, 2:100005, December 2019.

20 Xiaofeng Dai, Liangjian Xiang, Ting Li, and Zhonghu Bai. Cancer hallmarks, biomarkers and breast cancer molecular subtypes. *Journal of Cancer*, 7(10):1281, 2016.

21 Herbert Edelsbrunner and John Harer. Persistent homology-a survey. *Contemporary mathematics*, 453:257–282, 2008.

22 Herbert Edelsbrunner and John Harer. *Computational topology: an introduction*. American Mathematical Soc., 2010.

23 Danielle J Fassler, Shahira Abousamra, Rajarsi Gupta, Chao Chen, Maozheng Zhao, David Paredes-Merino, Syeda Areeha Batool, Beatrice Knudsen, Luisa Escobar-Hoyos, Kenneth R Shroyer, Dimitris Samaras, Tahsin Kurc, and Joel Saltz. Deep learning-based image analysis methods for brightfield-acquired multiplex immunohistochemistry images, 2019. under review.

24 Mingchen Gao, Chao Chen, Shaoting Zhang, Zhen Qian, Dimitris Metaxas, and Leon Axel. Segmenting the papillary muscles and the trabeculae from high resolution cardiac ct through restoration of topological handles. In *International Conference on Information Processing in Medical Imaging*, pages 184–195. Springer, 2013.

25 Robyn Gartrell, Douglas Marks, Thomas Hart, Gen Li, Danielle Davari, Alan Wu, Zoe Blake, Yan Lu, Kayleigh Askin, Anthea Monod, et al. Quantitative analysis of immune infiltrates in primary melanoma. *Cancer immunology research*, 6(4):481–493, 2018.

26 Arthur Gretton, Karsten Borgwardt, Malte Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13:723–773, 2012.

27 Arthur Gretton, Ralf Herbrich, Alexander Smola, Olivier Bousquet, and Bernhard Schölkopf. Kernel methods for measuring independence. *Journal of Machine Learning Research*, 6:2075–2129, 2005.

28 Heather Harrington, Nina Otter, Hal Schenck, and Ulrike Tillmann. Stratifying multiparameter persistent homology. *SIAM Journal on Applied Algebra and Geometry*, 3(3):439–471, January 2019.

29 Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning*. Springer-Verlag, 2003.

30 Christoph Hofer, Roland Kwitt, Marc Niethammer, and Andreas Uhl. Deep learning with topological signatures. In *Advances in Neural Information Processing Systems*, pages 1634–1644, 2017.

31 Xiaoling Hu, Li Fuxin, Dimitris Samaras, and Chao Chen. Topology-preserving deep image segmentation. In *the Thirty-third Conference on Neural Information Processing Systems (NeurIPS)*, 2019.

32 Jessica Kalra and Jennifer Baker. Multiplex immunohistochemistry for mapping the tumor microenvironment. In *Signal Transduction Immunohistochemistry*, pages 237–251. Springer, 2017.

33 Lida Kanari, Paweł Dłotko, Martina Scolamiero, Ran Levi, Julian Shillcock, Kathryn Hess, and Henry Markram. A topological representation of branching neuronal morphologies. *Neuroinformatics*, 16(1):3–13, 2018.

34 Genki Kusano, Yasuaki Hiraoka, and Kenji Fukumizu. Persistence weighted Gaussian kernel for topological data analysis. In *International Conference on Machine Learning*, volume 48, pages 2004–2013, June 2016.

35 Roland Kwitt, Stefan Huber, Marc Niethammer, Weili Lin, and Ulrich Bauer. Statistical topological data analysis - a kernel perspective. In *Advances in Neural Information Processing Systems*, pages 3070–3078, 2015.

36 Théo Lacombe, Marco Cuturi, and Steve Oudot. Large scale computation of means and clusters for persistence diagrams using optimal transport. In *Advances in Neural Information Processing Systems*, pages 9770–9780, 2018.

37 Peter Lawson, Andrew B Sholl, J Quincy Brown, Brittany Terese Fasy, and Carola Wenk. persistent homology for the quantitative evaluation of architectural features in prostate cancer histology. *Scientific reports*, 9, 2019.

38 Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

39 Hyekyoung Lee, Hyejin Kang, Moo K Chung, Bung-Nyun Kim, and Dong Soo Lee. Persistent brain network homology from the perspective of dendrogram. *IEEE transactions on medical imaging*, 31(12):2267–2277, 2012.

40 Michael Lesnick and Matthew Wright. Interactive visualization of 2D persistence modules. *arXiv*, December 2015. `arXiv:1512.00180`.

41 Yanjie Li, Dingkang Wang, Giorgio A Ascoli, Partha Mitra, and Yusu Wang. Metrics for comparing neuronal tree shapes based on persistent homology. *PloS one*, 12(8):e0182184, 2017.

42 Zhixian Liu, Mengyuan Li, Zehang Jiang, and Xiaosheng Wang. A comprehensive immunologic portrait of triple-negative breast cancer. *Translational oncology*, 11(2):311–329, 2018.

43 Yuriy Mileyko, Sayan Mukherjee, and John Harer. Probability measures on the space of persistence diagrams. *Inverse Problems*, 27(12):124007, December 2011.

44 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.

45 Steve Oudot. *Persistence theory: from quiver representations to data analysis*. American Mathematical Society, 2015.

46 Deepti Pachauri, Chris Hinrichs, Moo K Chung, Sterling C Johnson, and Vikas Singh. Topology-based kernels with application to inference problems in alzheimer's disease. *IEEE transactions on medical imaging*, 30(10):1760–1770, 2011.

47 Edwin Roger Parra, Alejandro Francisco-Cruz, and Ignacio Ivan Wistuba. State-of-the-art of profiling immune contexture in the era of multiplexed staining and digital analysis to study paraffin tumor tissues. *Cancers*, 11(2):247, 2019.

48 Adrien Poulenard, Primoz Skraba, and Maks Ovsjanikov. Topological function optimization for continuous shape matching. In *Computer Graphics Forum*, volume 37, pages 13–25. Wiley Online Library, 2018.

49 Lajos Pusztai, Thomas Karn, Anton Safonov, Maysa M Abu-Khalaf, and Giampaolo Bianchini. New strategies in breast cancer: immunotherapy. *Clinical Cancer Research*, 22(9):2105–2110, 2016.

50 Jan Reininghaus, Stefan Huber, Ulrich Bauer, and Roland Kwitt. A stable multi-scale kernel for topological machine learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

51 Abbas Rizvi, Pablo Cámara, Elena Kandror, Thomas Roberts, Ira Schieren, Tom Maniatis, and Raul Rabadan. Single-cell topological RNA-seq analysis reveals insights into cellular differentiation and development. *Nature Biotechnology*, 35(6):551–560, May 2017.

52 Andrew Robinson and Katharine Turner. Hypothesis testing for topological data analysis. *Journal of Applied and Computational Topology*, 1(2):241–261, December 2017.

53 Rebecca L Siegel, Kimberly D Miller, and Ahmedin Jemal. Cancer statistics, 2019. *CA: a cancer journal for clinicians*, 69(1):7–34, 2019.

54 Alexandra Signoriello, Marcus Bosenberg, Mark Shattuck, and Corey O'Hern. Modeling the spatiotemporal evolution of the melanoma tumor microenvironment. In *APS Meeting Abstracts*, 2016.

55 Carl-Johann Simon-Gabriel and Bernhard Schölkopf. Kernel distribution embeddings: universal kernels, characteristic kernels and kernel metrics on distributions. *Journal of Machine Learning Research*, 19(44):1–29, 2018.

56 Gurjeet Singh, Facundo Mémoli, and Gunnar Carlsson. Topological methods for the analysis of high dimensional data sets and 3D object recognition. In *Eurographics Symposium on Point-Based Graphics*, pages 91–100, 2007.

57 Bharath K Sriperumbudur, Kenji Fukumizu, and Gert RG Lanckriet. Universality, characteristic kernels and RKHS embedding of measures. *Journal of Machine Learning Research*, 12(Jul):2389–2410, 2011.

58 Mikael Vejdemo-Johansson and Sayan Mukherjee. Multiple testing with persistent homology. *arXiv*, December 2018. `arXiv:1812.06491`.

59 Pengxiang Wu, Chao Chen, Yusu Wang, Shaoting Zhang, Changhe Yuan, Zhen Qian, Dimitris Metaxas, and Leon Axel. Optimal topological cycles and their application in cardiac trabeculae restoration. In *International Conference on Information Processing in Medical Imaging*, pages 80–92. Springer, 2017.

60 Afra Zomorodian and Gunnar Carlsson. Computing persistent homology. *Discrete & Computational Geometry*, 33(2):249–274, 2005.

Homotopic Curve Shortening and the Affine Curve-Shortening Flow

Sergey Avvakumov
Institute of Science and Technology Austria (IST Austria),
Am Campus 1, 3400 Klosterneuburg, Austria
sergey.avvakumov@ist.ac.at

Gabriel Nivasch[1]
Ariel University, Ariel, Israel
gabrieln@ariel.ac.il

Abstract

We define and study a discrete process that generalizes the convex-layer decomposition of a planar point set. Our process, which we call *homotopic curve shortening* (HCS), starts with a closed curve (which might self-intersect) in the presence of a set $P \subset \mathbb{R}^2$ of point obstacles, and evolves in discrete steps, where each step consists of (1) taking shortcuts around the obstacles, and (2) reducing the curve to its shortest homotopic equivalent.

We find experimentally that, if the initial curve is held fixed and P is chosen to be either a very fine regular grid or a uniformly random point set, then HCS behaves at the limit like the affine curve-shortening flow (ACSF). This connection between HCS and ACSF generalizes the link between "grid peeling" and the ACSF observed by Eppstein et al. (2017), which applied only to convex curves, and which was studied only for regular grids.

We prove that HCS satisfies some properties analogous to those of ACSF: HCS is invariant under affine transformations, preserves convexity, and does not increase the total absolute curvature. Furthermore, the number of self-intersections of a curve, or intersections between two curves (appropriately defined), does not increase. Finally, if the initial curve is simple, then the number of inflection points (appropriately defined) does not increase.

2012 ACM Subject Classification Theory of computation → Computational geometry; Mathematics of computing → Geometric topology

Keywords and phrases affine curve-shortening flow, shortest homotopic path, integer grid, convex-layer decomposition

Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.12

Related Version A full version of this paper is available at `https://arxiv.org/abs/1909.00263`.

Supplementary Material Our code is available at `https://github.com/savvakumov/`.

Funding *Sergey Avvakumov*: Supported by the Austrian Science Fund (FWF), Project P31312-N35.

Acknowledgements Thanks to Arseniy Akopyan, Imre Bárány, Jeff Erickson, Radoslav Fulek, Jeremy Schiff, Arkadiy Skopenkov, and Peter Synak for useful discussions. Thanks also to the referees for their useful comments.

[1] Corresponding author

36th International Symposium on Computational Geometry (SoCG 2020).
Editors: Sergio Cabello and Danny Z. Chen; Article No. 12; pp. 12:1–12:15
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

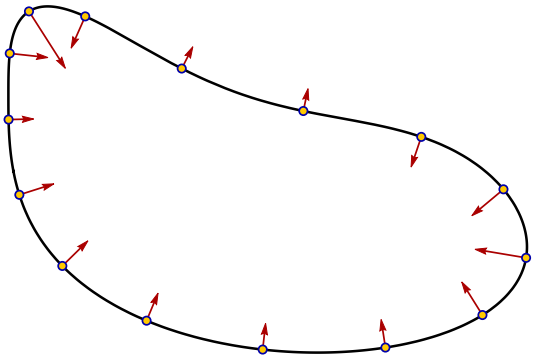


Figure 1 Affine curve-shortening flow. The arrows indicate the instantaneous velocity of different points along the curve at the shown time moment.

1 Introduction

Let $\mathbb{S}^1$ be the unit circle. In this paper we call a piecewise-smooth function $\gamma : [0,1] \to \mathbb{R}^2$ a *path*, and a piecewise-smooth function $\gamma : \mathbb{S}^1 \to \mathbb{R}^2$ a *closed curve*, or simply a *curve*. If γ is injective then the curve or path is said to be *simple*. We say that two paths or curves γ, δ are ε-*close* to each other if their Fréchet distance is at most ε, i.e. if they can be re-parametrized such that for every t, the Euclidean distance between the points $\gamma(t)$, $\delta(t)$ is at most ε.

1.1 Shortest Homotopic Curves

Let P be a finite set of points in the plane, which we regard as obstacles. Two curves γ, δ that avoid P are said to be *homotopic* if there exists a way to continuously transform γ into δ while avoiding P at all times. And two paths γ, δ that avoid P (except possibly at the endpoints) and satisfy $\gamma(0) = \delta(0)$, $\gamma(1) = \delta(1)$ are said to be *homotopic* if there exists a way to continuously transform γ into δ, without moving their endpoints, while avoiding P at all times (except possibly at the endpoints). We extend these definitions to the case where γ avoids obstacles but δ does not, by requiring the continuous transformation of γ into δ to avoid obstacles at all times except possibly at the last moment.

Then, for every curve (resp. path) γ in the presence of obstacles there exists a unique shortest curve (resp. path) δ that is homotopic to γ. The problem of computing the shortest path or curve homotopic to a given piecewise-linear path or curve, under the presence of polygonal or point obstacles, has been studied extensively. A simple and efficient algorithm for this task is the so-called "funnel algorithm" [12, 26, 27]. See also [7, 9, 18].

1.2 The Affine Curve-Shortening Flow

In the *affine curve-shortening flow*, a smooth curve $\gamma \subset \mathbb{R}^2$ varies with time in the following way. At each moment in time, each point of γ moves perpendicularly to the curve, towards its local center of curvature, with instantaneous velocity $r^{-1/3}$, where r is that point's radius of curvature at that time. See Figure 1.

The ACSF was first studied by Alvarez et al. [3] and Sapiro and Tannenbaum [28]. It differs from the more usual *curve-shortening flow* (CSF) [10, 14], in which each point is given instantaneous velocity r^{-1}. Unlike the CSF, the ACSF is invariant under affine transformations: Applying an affine transformation to a curve, and then performing the

ACSF, gives the same results (after rescaling the time parameter appropriately) as performing the ACSF and then applying the affine transformation to the shortened curves. Moreover, if the affine transformation preserves area, then the time scale is unaffected.

The ACSF was originally applied in computer vision, as a way of smoothing object boundaries [10] and of computing shape descriptors that are insensitive to the distortions caused by changes of viewpoint.

Properties of the CSF and ACSF for Simple Curves. Under either the CSF or the ACSF, a simple curve remains simple, and its length decreases strictly with time ([14], [28], resp.). Furthermore, a pair of disjoint curves, run simultaneously, remain disjoint at all times ([29], [5], resp.). More generally, the number of intersections between two curves never increases ([4], [5], resp.). The total absolute curvature[2] of a curve decreases strictly with time and tends to 2π ([21, 22], [5], resp.). The number of inflection points of a simple curve does not increase with time ([4], [5], resp.).

Under the CSF, a simple curve eventually becomes convex and then converges to a circle as it collapses to a point [21, 22]. Correspondingly, under the ACSF, a simple curve becomes convex and then converges to an ellipse as it collapses to a point [5].

Self-Intersecting Curves. When the initial curve is not simple, a self-intersection might collapse and form a cusp with infinite curvature. For the CSF, it has been shown that, as long as the initial curve satisfies some natural conditions, it is possible with some care to continue the flow past the singularity [2, 4]. Angenent [4] generalized these results to a wide range of flows, but unfortunately the ACSF is not included in this range [5]. Hence, no rigorous results have been obtained for self-intersecting curves under the ACSF. Still, ACSF computer simulations can be run on curves that have self-intersections or singularities with little difficulty.

1.3 Relation to Grid Peeling

Let P be a finite set of points in the plane. The *convex-layer decomposition* (also called the *onion decomposition*) of P is the partition of P into sets $P_1, P_2, P_3, \ldots$ obtained as follows: Let $Q_0 = P$. Then, for each $i \geq 1$ for which $Q_{i-1} \neq \emptyset$, let P_i be the set of vertices of the convex hull of Q_{i-1}, and let $Q_i = Q_{i-1} \setminus P_i$. In other words, we repeatedly remove from P the set of vertices of its convex hull. See [6, 13, 16, 17].

Eppstein et al. [19], following Har-Peled and Lidický [24], studied *grid peeling*, which is the convex-layer decomposition of subsets of the integer grid $\mathbb{Z}^2$. Eppstein et al. found an experimental connection between ACSF for convex curves and grid peeling. Specifically, let γ be a fixed convex curve. Let n be large, let $(\mathbb{Z}/n)^2$ be the uniform grid with spacing $1/n$, and let $P_n(\gamma)$ be the set of points of $(\mathbb{Z}/n)^2$ that are contained in the region bounded by γ. Then, as $n \to \infty$, the convex-layer decomposition of $P_n(\gamma)$ seems experimentally to converge to the ACSF evolution of γ, after the time scale is adjusted appropriately. They formulated this connection precisely in the form of a conjecture. They also raised the question whether there is a way to generalize the grid peeling process so as to approximate ACSF for non-convex curves as well.

Dalal [16] studied the convex-layer decomposition of point sets chosen uniformly and independently at random from a fixed convex domain, in the plane as well as in $\mathbb{R}^d$.

[2] Let $\gamma : [0,1] \to \mathbb{R}^2$ be a smooth closed curve, and let $\alpha : [0,1] \to \mathbb{S}^1$ be continuous such that $\alpha(s)$ is tangent to $\gamma(s)$ for all $s \in [0,1]$. Then the *total absolute curvature* of γ is the total distance traversed by $\alpha(s)$ in $\mathbb{S}^1$ as s goes from 0 to 1. If γ is convex then its total absolute curvature is exactly 2π; otherwise, it is larger than 2π.

1.4 Our Contribution

In this paper we describe a generalization of the convex-layer decomposition to non-convex, and even non-simple, curves. We call our process *homotopic curve shortening*, or HCS. Under HCS, an initial curve evolves in discrete steps in the presence of point obstacles. We find that, if the obstacles form a uniform grid, then HCS shares the same experimental connection to ACSF that grid peeling does. Hence, HCS is the desired generalization sought by Eppstein et al. [19]. We also find that the same experimental connection between ACSF and HCS (and in particular, between ACSF and the convex-layer decomposition) holds when the obstacles are distributed uniformly at random, with the sole difference being in the constant of proportionality.

Although the experimental connection between HCS and ACSF seems hard to prove, we do prove that HCS satisfies some simple properties analogous to those of ACSF: HCS is invariant under affine transformations, preserves convexity, and does not increase the total absolute curvature. Furthermore, the number of self-intersections of a curve, or intersections between two curves (appropriately defined), does not increase. Finally, if the initial curve is simple, then the number of inflection points (appropriately defined) does not increase.

Organization of This Paper. In Section 2 we describe homotopic curve shortening (HCS), our generalization of the convex-layer decomposition. In Section 3 we present our conjectured connection between ACSF and HCS, as well as experimental evidence supporting this connection. In Section 4 we state our theoretical results, to the effect that HCS satisfies some properties analogous to those of ACSF. In Section 5 we sketch the proofs the results stated in Section 4. Missing details can be found in the full version in the arXiv.

2 Homotopic Curve Shortening

Let P be a finite set of obstacle points. A *P-curve* (resp. *P-path*) is a curve (resp. path) that is composed of straight-line segments, where each segment starts and ends at obstacle points.

Homotopic curve shortening (HCS) is a discrete process that starts with an initial P-curve γ_0 (which might self-intersect), and at each step, the current P-curve γ_n is turned into a new P-curve $\gamma_{n+1} = \mathrm{HCS}_P(\gamma_n)$.

The definition of $\gamma' = \mathrm{HCS}_P(\gamma)$ for a given P-curve γ is as follows. Let $(p_0, \ldots, p_{m-1})$ be the circular list of obstacle points visited by γ. Call p_i *nailed* if γ goes straight through p_i, i.e. if $\angle p_{i-1}p_ip_{i+1} = \pi$.[3] Let $(q_0, \ldots, q_{k-1})$ be the circular list of nailed vertices of γ. Suppose first that $k \geq 1$. Then γ' is obtained through the following three substeps:

1. *Splitting.* We split γ into k P-paths $\delta_0, \ldots, \delta_{k-1}$ at the nailed vertices, where each δ_i goes from q_i to q_{i+1}.
2. *Shortcutting.* For each non-endpoint vertex p_i of each δ_i, we make the curve avoid p_i by taking a small shortcut. Specifically, let $\varepsilon > 0$ be sufficiently small, and let C_{p_i} be a circle of radius ε centered at p_i. Let e_i be the segment $p_{i-1}p_i$ of δ_i. Let $x_i = e_i \cap C_{p_i}$ and $y_i = e_{i+1} \cap C_{p_i}$. Then we make the path go straight from x_i to y_i instead of through p_i. Call the resulting path ρ_i, and let ρ be the curve obtained by concatenating all the paths ρ_i.
3. *Shortening.* Each ρ_i in ρ is replaced by the shortest P-path homotopic to it. The resulting curve is γ'.

[3] All indices in circular sequences are modulo the length of the sequence.

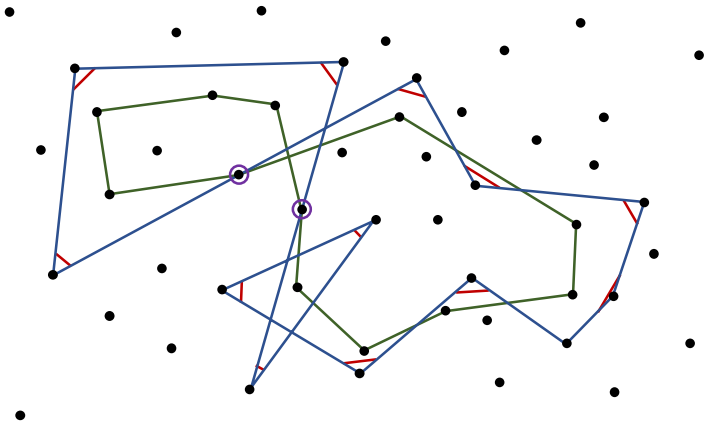


Figure 2 Computation of a single step of homotopic curve shortening: Given a P-curve γ (blue), we first identify its nailed vertices (purple). In this case, the two nailed vertices split γ into two paths δ_0, δ_1. In each δ_i we take a small shortcut around each intermediate vertex (red). Then we replace each δ_i by the shortest path homotopic to it, obtaining the new P-curve $\gamma' = \mathrm{HCS}_P(\gamma)$ (green).

If γ has no nailed vertices ($k = 0$) then γ' is obtained by performing the shortcutting and shortening steps on the single closed curve γ. Figure 2 illustrates one HCS step on a sample curve.

The process terminates when the curve collapses to a point. This will certainly happen after a finite number of steps, since at each step the curve gets strictly shorter, and there is a finite number of distinct P-curves of at most a certain length.

HCS for Convex Curves. If the initial curve γ_0 is the boundary of the convex hull of P, then the HCS evolution of γ_0 is equivalent to the convex-layer decomposition of P. Namely, for every $i \geq 0$, the curve γ_i is the boundary of a convex polygon, and the set of vertices of this polygon equals the $(i+1)$-st convex layer of P. See Section 4 below.

3 Experimental Connection Between ACSF and HCS

Our experiments show that HCS, using $P = (\mathbb{Z}/n)^2$ as the obstacle set, approximates ACSF at the limit as $n \to \infty$, just as grid peeling approximates ACSF for convex curves. The connection between the two processes is formalized in the following conjecture, which generalizes Conjecture 1 of [19].

▶ **Conjecture 1.** *There exists a constant $c_g \approx 1.6$ such that the following is true: Let δ be a piecewise-smooth initial curve. Fix a time $t > 0$, and let $\delta' = \delta(t)$ under ACSF. For a fixed n, let γ_0 be the shortest curve homotopic to δ under obstacle set $P_n = (\mathbb{Z}/n)^2$. Let $m = c_g t n^{4/3}$, and let $\gamma_m = \mathrm{HCS}_P^{(m)}(\gamma_0)$ be the result of m iterations of HCS starting with γ_0. Then, as $n \to \infty$, the Fréchet distance between γ_m and δ' tends to 0.*

Furthermore, we find that the connection between ACSF and HCS also holds if the uniform grid $(\mathbb{Z}/n)^2$ is replaced by a random point set, though with a different constant of time proportionality.

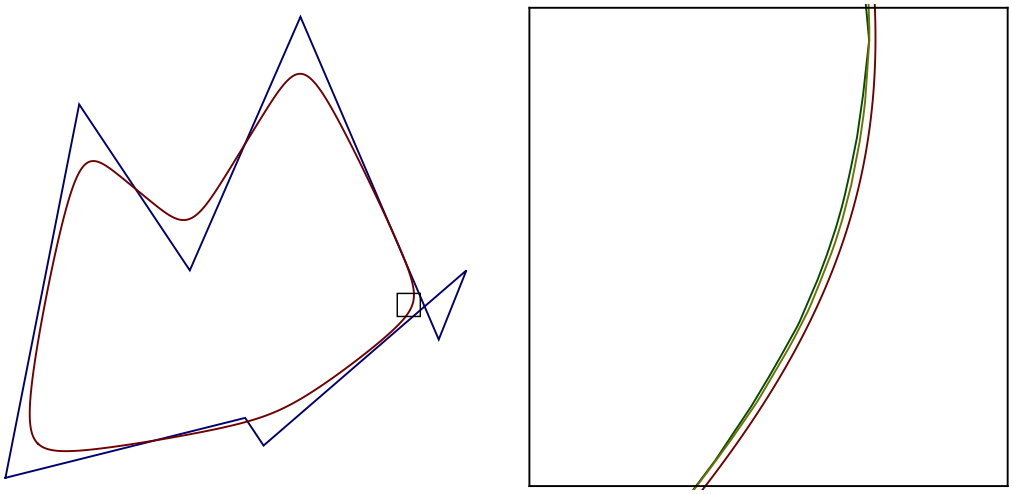


Figure 3 Left: Initial curve Δ (blue) and simulated ACSF result after the curve's length reduced to 70% of its original length (red). Right: Comparison between ACSF approximation (red), HCS with $n = 10^7$ uniform-grid obstacles (green), and HCS with $n = 10^7$ random obstacles (yellow) on a small portion of the curve.

▶ **Conjecture 2.** *There exists a constant $c_r \approx 1.3$ such that the following is true: Let δ be a piecewise-smooth initial curve, contained in a convex region R of area A. Fix a time $t > 0$, and let $\delta' = \delta(t)$ under ACSF. For a fixed n, let P be a set of An^2 obstacle points chosen uniformly and independently at random from R. Let γ_0 be the shortest curve homotopic to δ under obstacle set P. Let $m = c_r t n^{4/3}$, and let $\gamma_m = \mathrm{HCS}_P^{(m)}(\gamma_0)$ be the result of m iterations of HCS starting with γ_0. Then, as $n \to \infty$, the Fréchet distance between γ_m and δ' is almost surely smaller than ε, for some $\varepsilon = \varepsilon(n)$ that tends to 0 with n.*

3.1 Experiments

We tested Conjectures 1 and 2 on a variety of test curves. We found that for all our test curves, the result of HCS does seem to converge to the result of ACSF as $n \to \infty$, both for grid and for random obstacle sets.

We illustrate our experiments on the piecewise-liner curve Δ having vertices $(0, 0)$, $(0.16, 0.81)$, $(0.4, 0.45)$, $(0.64, 1)$, $(0.94, 0.3)$, $(1, 0.45)$, $(0.56, 0.07)$, $(0.52, 0.13)$. We approximated ACSF using an approach similar to the one in [19]. We ran our ACSF simulation on Δ until we obtained a curve Δ' whose length equals 70% of the original length of Δ. See Figure 3 (left). This happened at $t^* \approx 0.0266$. By this time, the self-intersection and an inflection point of the curve have disappeared.

Then we introduced in the unit square $[0, 1]^2 \supset \Delta$ a set P of n obstacle points, where P is either a uniform grid (i.e. a $\sqrt{n} \times \sqrt{n}$ grid) G_n, or a random set R_n. For each case, we initially snapped each vertex of Δ to its closest point in P, obtaining a P-curve, and then we ran HCS until the length of the curve shrank to 70% of its original length, obtaining a new curve $\Delta'' = \Delta''(P)$. We did this for several values of n. For each case, we computed $h(\Delta', \Delta'')$, where $h(\gamma_1, \gamma_2)$ for piecewise-linear curves γ_1, γ_2 is defined as the maximum distance between a vertex of one curve and the closest point on the other curve. (For "nice" curves as ours, there is no significant difference, if at all, between this distance h and either the Hausdorff or the Fréchet distance between the two curves.)

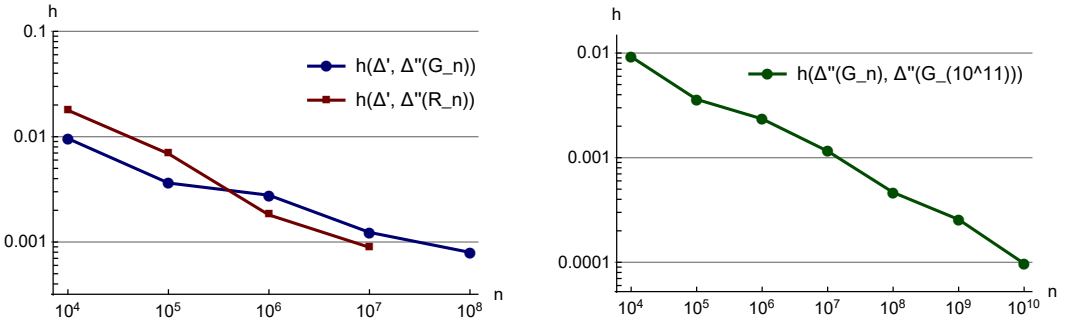


Figure 4 Left: Distance between ACSF approximation and HCS with uniform-grid obstacles (blue curve) or random obstacles (red curve, average of 5 trials), for increasing values of n, the number of obstacles. Right: Distance between HCS with uniform-grid obstacles for $n = 10^4, \ldots, 10^{10}$ and with $n = 10^{11}$.

Table 1 Approximations of the constants c_g and c_r given by the experiments.

n	iterations with G_n	c_g	avg. iterations with R_n	c_r
10^4	20	1.616	15.6	1.261
10^5	93	1.619	75.2	1.309
10^6	434	1.628	351.2	1.317
10^7	2006	1.621	1628.6	1.316
10^8	9266	1.613		

For random obstacles, we conducted this experiment for $n = 10^4, 10^5, 10^6, 10^7$, taking the average of 5 samples for each value of n. Our random-obstacle program is limited by memory rather than by time, since it stores all the obstacle points in memory. For uniform-grid obstacles, we conducted this experiment also for $n = 10^8$. After this point, our ACSF approximation Δ' does not seem to be accurate enough for reliable comparisons. The results are shown in Figure 4 (left).

We also checked whether the relation between the ACSF time t^* and the number of HCS iterations m behaves as predicted by Conjectures 1 and 2. For this purpose, we computed $c = m/(t^* n^{2/3})$ for each case, and checked whether c is roughly constant. The results are shown in Table 1.

As we can see, Conjectures 1 and 2 are well supported by the experiments.

Finally, we measured the rate of convergence of the uniform-grid HCS to its limit shape as $n \to \infty$. To this end, we computed $h(\Delta''(G_n), \Delta''(G_m))$ for $n \in \{10^4, 10^5, \ldots, 10^{10}\}$ and $m = 10^{11}$. See Figure 4 (right). As we can see, increasing n by a factor of 10 has the effect of multiplying the distance by roughly a factor of 0.47.

See the full version in the arXiv for some implementation details of our ACSF and HCS simulations.

4 Properties of Homotopic Curve Shortening

We now prove that HCS satisfies some properties analogous to those of ACSF.

▶ **Theorem 3.** *HCS is invariant under affine transformations. Namely, if P is a set of obstacle points, γ is a P-curve, and T is a non-degenerate affine transformation, then* $T(\mathrm{HCS}_P(\gamma)) = \mathrm{HCS}_{T(P)}(T(\gamma))$.

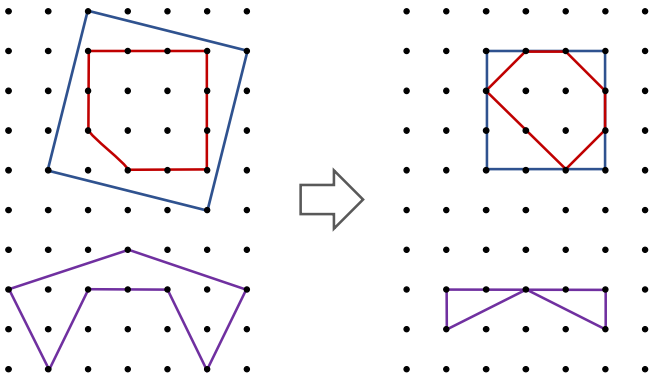


Figure 5 HCS might cause disjoint curves to intersect, or a simple curve to self-intersect.

In particular, if T is a grid-preserving affine transformation, meaning that T maps $(\mathbb{Z}/n)^2$ injectively to itself, then the HCS evolution using $P = (\mathbb{Z}/n)^2$ (as in Conjecture 1) is unaffected by T. Hence, HCS on uniform-grid obstacles is invariant under a certain subset of the area-preserving affine transformations, just as in grid peeling [19].

Also, if T is an area-preserving affine transformation, then the probability distribution of random sets P in the convex region R of Conjecture 2 stays unaffected after applying T to R.

▶ **Theorem 4.** *Let γ be a simple P-curve, and let $\gamma' = \mathrm{HCS}_P(\gamma)$. If γ is the boundary of a convex polygon, then so is γ'. Hence, under HCS, once a curve becomes the boundary of a convex polygon, it stays that way.*

The *total absolute curvature* of a piecewise-linear curve γ with vertices $(p_0, \ldots, p_{m-1})$ is the sum of the exterior angles $\sum_{i=0}^{m-1}(\pi - |\angle p_{i-1}p_ip_{i+1}|)$. It equals 2π if γ is the boundary of a convex polygon, and it is larger than 2π otherwise.

▶ **Theorem 5.** *Let γ be a P-curve, and let $\gamma' = \mathrm{HCS}_P(\gamma)$. Let α, α' be the total absolute curvature of γ, γ', respectively. Then $\alpha \geq \alpha'$. Hence, under HCS, the total absolute curvature of a curve never increases.*

If γ, δ are disjoint P-curves, then $\mathrm{HCS}_P(\gamma), \mathrm{HCS}_P(\delta)$ are not necessarily disjoint. Similarly, if γ is a simple P-curve, then $\mathrm{HCS}_P(\gamma)$ is not necessarily simple. See Figure 5.

Curves γ, δ are called *disjoinable* if they can be made into disjoint curves by peforming on them an arbitrarily small perturbation. Similarly, a curve γ is called *self-disjoinable* if it can be turned into a simple curve by an arbitrarily small perturbation. Note that if γ is self-disjoinable then γ, γ are disjoinable, though the reverse is not necessarily true: Consider for example a curve γ that makes two complete clockwise turns around the unit circle.

Akitaya et al. [1] recently found an $O(n \log n)$-time algorithm for deciding whether a given mapping of a graph into the plane is a so-called *weak embedding*. This algorithm can decide, in particular, whether a given curve is self-disjoinable.

An intersection between two curves, or between two portions of one curve, is called *transversal*, if at the point of intersection both curves are differentiable and their normal vectors are not parallel at that point. If all intersections between curves γ_1 and γ_2 are transversal, then we say that γ_1, γ_2 are themselves *transversal*. Similarly, if all self-intersections of γ are transversal, then we say that γ is *self-transversal*. (Transversal and self-transversal curves are sometimes called *generic*, see e.g. [11].)

If γ is self-transversal, we denote by $\chi(\gamma)$ the number of self-intersections of γ.[4] If γ is not self-transversal, then we define $\chi(\gamma)$ as the minimum of $\chi(\widehat{\gamma})$ among all self-transversal curves $\widehat{\gamma}$ that are ε-close to γ, for all small enough $\varepsilon > 0$. Hence, $\chi(\gamma) = 0$ if and only if γ is self-disjoinable. We define similarly the number of intersections $\chi(\gamma_1, \gamma_2)$ between two curves. Then, γ_1 and γ_2 are disjoinable if and only if $\chi(\gamma_1, \gamma_2) = 0$. Fulek and Tóth recently proved that the problem of computing $\chi(\gamma)$ is NP-hard [20].

▶ **Theorem 6.** *Let γ be a P-curve, and let $\gamma' = \mathrm{HCS}_P(\gamma)$. Then their self-intersection numbers satisfy $\chi(\gamma') \leq \chi(\gamma)$. Let δ be another P-curve, and let $\delta' = \mathrm{HCS}_P(\delta)$. Then their intersection numbers satisfy $\chi(\gamma', \delta') \leq \chi(\gamma, \delta)$. In particular, if γ is self-disjoinable, so is γ', and if γ, δ are disjoinable, then so are γ', δ'. Hence, under HCS, the intersection and self-intersection numbers never increase.*

With the technique of Theorem 6 we can obtain an upper bound on the number of iterations of HCS:

▶ **Theorem 7.** *If $|P| = n$ then the HCS process starting with any P-curve ends in at most $n/2$ iterations. If $P = \{1, 2, \ldots, \sqrt{n}\}^2$ then the process ends in at most $O(n^{2/3})$ iterations. If P is uniformly and independently chosen at random inside a fixed convex domain, then the expected number of iterations is $O(n^{2/3})$.*

We say that an obstacle set P is in *general position* if no three points of P lie on a line. Note that if P is in general position then there are no nailed vertices in HCS.

▶ **Theorem 8.** *Let P be an obstacle set in general position. Let γ be a simple P-curve. Then $\mathrm{HCS}_P(\gamma)$ is also simple. Let γ_1, γ_2 be disjoint P-curves. Then $\mathrm{HCS}_P(\gamma_1), \mathrm{HCS}_P(\gamma_2)$ are also disjoint. Hence, under HCS with obstacles in general position, a simple curve stays simple, and a pair of disjoint curves stay disjoint.*

Let γ be a simple piecewise-linear curve with vertices $(v_0, \ldots, v_{n-1})$. Assume that the sequence of vertices is minimal, meaning no v_{i-1}, v_i, v_{i+1} lie on a straight line. An *inflection edge* of γ is an edge $v_i v_{i+1}$ such that the previous and next vertices v_{i-1}, v_{i+2} lie on opposite sides of the line through v_i, v_{i+1}. Let $\varphi(\gamma)$ be the number of inflection edges of γ. Note that $\varphi(\gamma)$ is always even, since every inflection edge lies either after a sequence of clockwise vertices and before a sequence of counterclockwise vertices, or vice versa.

If γ is not simple but self-disjoinable, then we define $\varphi(\gamma)$ as the minimum of $\varphi(\gamma')$ over all simple piecewise-linear curves γ' that are ε-close to γ, for all sufficiently small $\varepsilon > 0$. (Note that for a given γ there might exist different curves γ' with different values of $\varphi(\gamma')$. For example, if γ goes from a point p to a point q and back n times, then γ' could be a spiral with just two inflection edges, or a double zig-zag with $2n - 2$ inflection edges.)

▶ **Theorem 9.** *Let γ be self-disjoinable, and let $\gamma' = \mathrm{HCS}_P(\gamma)$. Then their inflection-edge numbers satisfy $\varphi(\gamma') \leq \varphi(\gamma)$. Hence, under HCS on a self-disjoinable curve, the curve's number of inflection edges never increases.*

5 Proofs

In order to prove Theorems 3–9, we rely on two different approaches for computing shortest homotopic curves. The first approach uses a triangulation of the ambient space, while the second aproach consists of repeatedly releasing unstable vertices. We start by describing these two approaches in detail.

[4] A self-intersection in a curve $\gamma : \mathbb{S}^1 \to \mathbb{R}^2$ is a pair $s \neq t$ such that $\gamma(s) = \gamma(t)$. Hence, if γ passes k times through a certain point, that counts as $\binom{k}{2}$ self-intersections.

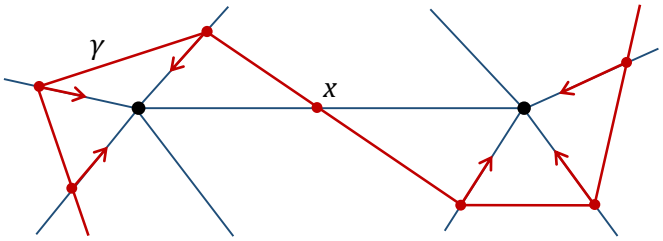


Figure 6 In the shortest curve homotopic to γ, the position of the point x is not uniquely defined.

5.1 Triangulations

Let P be a finite set of point obstacles, and let γ be a piecewise-smooth curve avoiding P. Assume without loss of generality that γ is contained in the convex hull of P (by adding points outside the convex hull of γ if necessary). Let $\mathcal{T}$ be a triangulation of the convex hull of P using the points of P as vertices.

We can assume without loss of generality that the curve γ intersects each triangle edge transversally. Let $\mathcal{E} = \mathcal{E}(\gamma)$ be the circular sequence of triangle edges intersected by γ. Then a piecewise-smooth homotopic change of γ can only have two possible types of effects on $\mathcal{E}$: Either an adjacent pair ee is inserted somewhere in the sequence, or an existing such pair is deleted. Hence, two curves γ, γ' are homotopic if and only if their corresponding edge sequences $\mathcal{E}(\gamma)$, $\mathcal{E}(\gamma')$ are *equivalent*, in the sense that they can be transformed into one another by a sequence of operations of these two types.

Call an edge sequence $\mathcal{E}(\gamma)$ *reduced* if it contains no adjacent pair ee. Then every edge sequence is equivalent to a unique reduced sequence. (Proof sketch: Supposing for a contradiction that there exist two distinct equivalent reduced sequences S_1, S_2, consider a transformation of S_1 into S_2 that uses the minimum possible number of deletions, and among those, consider one in which the first deletion is done as early as possible. Then it is easy to arrive at a contradiction.)

Hence, in order to compute the shortest curve homotopic to γ, we first compute $\mathcal{E}(\gamma)$, then we reduce this sequence by repeatedly removing adjacent pairs, obtaining a reduced sequence $\mathcal{E}'$, then we place a point $x(e)$ on each $e \in \mathcal{E}'$, and then we slide the points $x(e)$ along their edges so as to minimize the length of the curve. This last step can be done by the above-mentioned "funnel algorithm", the details of which we omit.

See the full version of this paper for a proof that there is always a unique shortest curve. Note that, even though the shortest curve is always unique, the final positions of the points $x(e)$ are not necessarily unique. This can happen if a triangulation edge is an edge of the final curve. See Figure 6.

5.2 The Vertex Release Algorithm

We now present another simple algorithm for the shortest homotopic curve problem. This algorithm is not mentioned in any previous publication that we are aware of, but it is similar in spirit to well-known algorithms, in particular to the funnel algorithm.

As a warm-up, let us first consider the case in which the obstacles are are not single points but rather polygons. Let γ be a curve that avoids all the obstacles. Call a vertex v of γ *unstable* if v does not lie on any obstacle, or if v lies on the boundary of an obstacle T, but T lies locally on the side of γ at which the angle is larger than π. If v is unstable, then the process of *releasing* v is as follows: Let u and w be the previous and next vertices

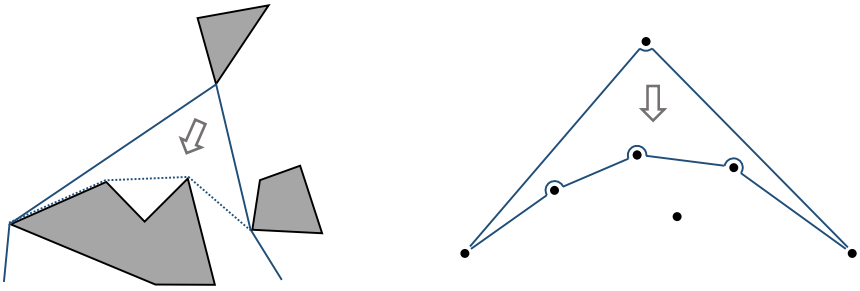


Figure 7 Releasing an unstable vertex in the presence of polygonal obstacles (left) or point obstacles (right).

of γ. Suppose first that $u \neq w$. Let Δ be the triangle uvw, and let S be the set of obstacle vertices that lie inside Δ. Let $u, z_1, \ldots, z_k, w$ be the vertices of the convex hull of $S \setminus \{v\}$ in order. Then we replace v by $z_1, \ldots, z_k$ in γ. The new vertices $z_1, \ldots, z_k$ are necessarily stable, but u and w might change from stable to unstable or vice versa. If $u = w$ then we simply remove v and w from γ. See Figure 7 (left).

Then the algorithm consists of releasing unstable vertices one by one, in an arbitrary order, until no more unstable vertices remain.

If there are also point obstacles, then the algorithm becomes slightly more complicated. For each curve vertex v that lies on a point obstacle, we need to remember the corresponding signed angle α_v that the curve turns around the obstacle, since this angle could be larger than 2π in absolute value. The angle α_v is always congruent modulo 2π to $\angle uvw$, where u and w are the previous and next vertices. A vertex v is unstable if and only if $|\alpha_v| < \pi$.

Whenever we release an unstable vertex v preceded by u and followed by w, we proceed as described above, and we update the angles as follows (see Figure 7, right):

- If $u \neq w$ then we give to each new vertex z_i the unique appropriate angle that has the opposite sign of α_v and satisfies $\pi \leq |\alpha_{z_i}| < 2\pi$. We then update the angles α_u and α_w as follows: Denote $z_0 = u$ and $z_{k+1} = w$ (in order to handle properly the case $k = 0$). We add to α_u the angle $\angle vuz_1$, and we add to α_w the angle $\angle z_k wv$.
- If $u = w$ then we update α_u by adding to it the angle α_w.

For the proof of correctness of the vertex release algorithm, see the full version of this paper.

5.3 Proof of Theorems 3–5

Theorems 3–5 follow easily from the vertex-release algorithm.

Proof of Theorem 3. The claim follows from the fact that shortest homotopic curves and paths are invariant under affine transformations. Namely, let γ be a curve or path in the presence of obstacle points P, let δ be the shortest curve or path homotopic to γ, and let $T : \mathbb{R}^2 \to \mathbb{R}^2$ be a non-degenerate affine transformation. Then the shortest curve or path homotopic to $T(\gamma)$ in the presence of $T(P)$ is $T(\delta)$. This, in turn, follows from the fact that T does not affect whether a vertex is stable or unstable, and furthermore, if a vertex is unstable, then it does not matter whether we first release the vertex and then apply T, or do these operations in the opposite order. ◀

Theorem 4 is also trivial, since the property of being the boundary of a convex polygon is preserved by each vertex release.

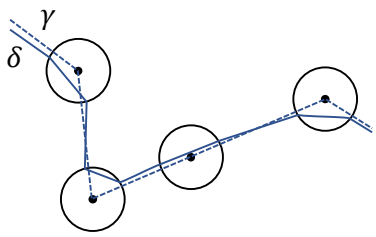


Figure 8 A P-curve γ and a corresponding type-2 curve δ.

Proof of Theorem 5. Given a curve γ with vertices $(p_0, \ldots, p_{m-1})$, let $v_i \in \mathbb{S}^1$ be the unit vector parallel to $\overrightarrow{p_i p_{i+1}}$ for each i . Call a tour of $\mathbb{S}^1$ *valid* if it visits the vectors $v_0, v_1, \ldots, v_{m-1}, v_0$ in this order. Then the total absolute curvature of γ equals the length of the shortest valid tour of $\mathbb{S}^1$.

Now let γ be a given P-curve, and let $\gamma' = \mathrm{HCS}_P(\gamma)$. Recall that γ' is obtained from γ by a series of vertex releases. Each vertex release replaces two adjacent vectors $v_i, v_{i+1} \in \mathbb{S}^1$ by a certain number $k \geq 1$ of vectors $w_1, \ldots, w_k$ lying between them, in this order. Hence, the shortest valid tour of $\mathbb{S}^1$ for the old vector sequence goes from v_i to v_{i+1} through $w_1, \ldots, w_k$, and hence this tour is also valid for the new vector sequence. ◀

5.4 Proof sketch of Theorems 6–8

The proof of Theorems 6–8 is based on the triangulation technique. Let γ be a P-curve, let $\varepsilon > 0$ be small enough, and let $\widehat{\gamma}$ be a self-transversal curve that is ε-close to γ and has the minimum possible number of self-intersections.

In order to prove Theorem 6, we proceed as follows:

1. We show that, without loss of generality, we can assume that $\widehat{\gamma}$ passes through the "correct side" of each non-nailed obstacle, as in the "shortcutting" step of HCS.
2. We modify $\widehat{\gamma}$ homotopically, by first eliminating repetitions in its edge sequence $\mathcal{E}$ and then sliding its vertices along the triangulation edges, until each vertex comes within ε of its final position as given by $\gamma' = \mathrm{HCS}_P(\gamma)$. We show that the number of self-intersections never increases in the process.

The case of two curves is similar.

In order to do the first step, we define a type of curves that are ε-close to P-curves and pass through the "correct side" of non-nailed obstacles. We call them *type-2 curves*. We also define a "snapping" operation, which transforms $\widehat{\gamma}$ into a type-2 curve without increasing its number of self-intersections.

Type-2 Curves. Let γ be a P-curve, let $(p_0, \ldots, p_{k-1})$ be the circular list of obstacles visited by γ, and let $\varepsilon > 0$ be small enough. For each $p \in P$, let C_p be a circle of radius ε centered at p. For each i, let $x_i \in C_{p_i}$ be a point at distance at most ε^2 from the segment $p_{i-1}p_i$, and let $y_i \in C_{p_i}$ be a point at distance at most ε^2 from the segment $p_i p_{i+1}$. Then a type-2 curve δ corresponding to γ travels in a straight line from y_{i-1} to x_i and then in a straight line from x_i to y_i for each i. See Figure 8. We call each segment $y_{i-1}x_i$ a *long part* and each segment $x_i y_i$ a *short part*. If $\angle p_{i-1}p_i p_{i+1} \neq \pi$ and ε is chosen small enough, then p_i lies on the side of the curve at which the angle is larger than π. If $\angle p_{i-1}p_i p_{i+1} = \pi$ then the corresponding short part passes within distance ε^2 of v_i.

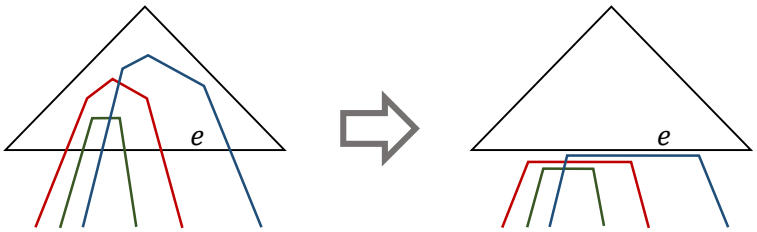


Figure 9 Reducing a curve's edge sequence without increasing its number of self-intersections. Different portions of the curve are shown in different colors.

The Snapping Operation. Let γ be a P-curve, and let $\widehat{\gamma}$ be a curve (ε^2)-close to γ. We define the type-2 curve $\text{snap}(\widehat{\gamma})$ as follows. For each p_i visited by $\widehat{\gamma}$ there exists a point z_i in $\widehat{\gamma}$ that is within distance ε^2 of p_i. Let y_i be the first intersection of $\widehat{\gamma}$ with C_{p_i} that comes after z_i, and let x_i be the last intersection of $\widehat{\gamma}$ with C_{p_i} that comes before z_i. (Thus, the part of $\widehat{\gamma}$ between x_i and y_i is entirely contained in the disk bounded by C_{p_i}.) Then we let $\text{snap}(\widehat{\gamma})$ be the type-2 curve that uses these points x_i, y_i for all i as vertices.

The curve $\delta = \text{snap}(\widehat{\gamma})$ also is also self-transversal, and it satisfies $\chi(\delta) \leq \chi(\widehat{\gamma})$. Similarly, if γ_1, γ_2 are two P-curves, and $\widehat{\gamma}_1$, $\widehat{\gamma}_2$ are transversal curves (ε^2)-close to them, respectively, such that no intersection between $\widehat{\gamma}_1$ and $\widehat{\gamma}_2$ occurs on any circle C_p, then the curves $\delta_1 = \text{snap}(\widehat{\gamma}_1), \delta_2 = \text{snap}(\widehat{\gamma}_2)$ are transversal and satisfy $\chi(\delta_1, \delta_2) \leq \chi(\widehat{\gamma}_1, \widehat{\gamma}_2)$. See the full version of this paper.

Proof of Theorem 6. Let γ be a P-curve, let $\varepsilon > 0$ be small enough, and let $\widehat{\gamma}$ be a self-transversal curve that is ε-close to γ and has the minimum possible number of self-intersections. Fix a triangulation $\mathcal{T}$ of P. Assume without loss of generality that $\widehat{\gamma}$ does not pass through any obstacle, and that no self-intersection of γ lies on any edge of $\mathcal{T}$. Let $\eta = \text{snap}(\widehat{\gamma})$. Partition η into paths $\eta_0, \ldots, \eta_{k-1}$ that are ε-close to the corresponding paths $\delta_0, \ldots, \delta_{k-1}$ of the HCS "splitting" step, by introducing split points as follows: For each nailed visit to an obstacle $p \in P$, we choose a split point that is within distance $O(\varepsilon)$ of p and lies on a triangle edge (where the implicit constant depends only on P).

Then we modify each η_i into a homotopic path η_i' whose edge sequence $\mathcal{E}(\eta_i')$ is reduced. We do this without increasing the number of intersections, by repeatedly doing the following: Let e be triangulation edge such that ee appears one or more times in the sequences $\mathcal{E}(\eta_i')$. We shortcut the corresponding paths η_i' so as to not cross e at all, instead keeping a small distance from e. We make the distance to e inversely related to the distance between the two crossing points of η_i' with e. See Figure 9.

Next, we modify each η_i' into η_i'' by straightening out each part within each triangle of $\mathcal{T}$. Hence, each η_i'' is determined by the position of its vertices $x(e)$ along the triangle edges e.

Finally, we slide the vertices $x(e)$ along the edges to within ε of their final positions, as given by γ'. We do this without changing the order of any pair of vertices along the same edge, unless necessary. Call the resulting paths η_i'''. Meaning, if in γ' there are several vertices along an edge that coincide, then in the paths η_i''' we place those vertices within ε of each other, conserving the order they had in η_i''. Let η'', η''' be the curves formed by concatenating the paths η_i'', η_i''' for all i, respectively. Hence, by construction, η''' is ε-close to γ'.

The number of self-intersections of η''' is not larger than that of η''. See the full version of this paper. This concludes the proof of Theorem 6 for the case of the number of self-intersections of a single curve. The case of the number of intersections of two curves is similar. ◀

Theorem 7 follows by running HCS simultaneously on the given curve γ_0 and on the boundary δ_0 of the convex hull of P. The HCS process starting with δ_0 is just the convex-layer decomposition of P, so we can apply the known bounds on the number of convex layers. Denote $\gamma_{i+1} = \mathrm{HCS}_P(\gamma_i)$ and $\delta_{i+1} = \mathrm{HCS}_P(\delta_i)$ for all i. By the proof of Theorem 6, the two curves stay disjoinable throughout the HCS process, with δ_i bounding γ_i for all i. See the full version for more details, as well as for the proof of Theorem 8.

5.5 Proof sketch of Theorem 9

The proof of Theorem 9 (regarding the number of inflection edges) is based of the vertex-release algorithm. The basic idea is that, given a self-disjoinable curve, if the vertex releases are performed in an appropriate order, then the curve stays self-disjoinable at all times. Moreover, no vertex release increases the number of inflection edges. Along the way, we develop enough machinery to re-prove Theorem 6. The proof appears in the full version of this paper.

6 Discussion

One of the reasons continuous curve-shortening flows were introduced and studied was to overcome the shortcomings of the *Birkhoff curve-shortening process* ([8], see also e.g. [15]), specifically the fact that it might cause the number of curve intersections to increase [23, 25]. As we have shown, HCS is a discrete process that overcomes this flaw without introducing analytical difficulties, at least in the plane. It would be interesting to check whether HCS can be applied on more general surfaces.

References

1 Hugo A. Akitaya, Radoslav Fulek, and Csaba D. Tóth. Recognizing weak embeddings of graphs. In *Proc. 29th Symp. on Discrete Algorithms*, pages 274–292, 2018. doi:10.1137/1.9781611975031.20.

2 Steven J. Altschuler and Matthew A. Grayson. Shortening space curves and flow through singularities. *J. Differential Geom.*, 35(2):283–298, 1992. doi:10.4310/jdg/1214448076.

3 Luis Alvarez, Frédéric Guichard, Pierre-Luis Lions, and Jean-Michel Morel. Axioms and fundamental equations of image processing. *Arch. Rational Mech. Anal.*, 123(3):199–257, 1993. doi:10.1007/BF00375127.

4 Sigurd Angenent. Parabolic equations for curves on surfaces: Part II. Intersections, blow-up and generalized solutions. *Annals of Mathematics*, 133(1):171–215, 1991. doi:10.2307/2944327.

5 Sigurd Angenent, Guillermo Sapiro, and Allen Tannenbaum. On the affine heat equation for non-convex curves. *J. Amer. Math. Soc.*, 11(3):601–634, 1998. doi:10.1090/S0894-0347-98-00262-8.

6 Vic Barnett. The ordering of multivariate data. *J. Roy. Statist. Soc. Ser. A*, 139(3):318–355, 1976. doi:10.2307/2344839.

7 Sergei Bespamyatnikh. Computing homotopic shortest paths in the plane. *Journal of Algorithms*, 49(2):284–303, 2003. doi:10.1016/S0196-6774(03)00090-7.

8 George D. Birkhoff. Dynamical systems with two degrees of freedom. *Trans. Amer. Math. Soc.*, 18:199–300, 1917.

9 Sergio Cabello, Yuanxin Liu, Andrea Mantler, and Jack Snoeyink. Testing homotopy for paths in the plane. *Discrete & Computational Geometry*, 31(1):61–81, 2004. doi:10.1007/s00454-003-2949-y.

10 Frédéric Cao. *Geometric Curve Evolution and Image Processing*, volume 1805 of *Lecture Notes in Mathematics*. Springer-Verlag, Berlin, 2003. doi:10.1007/b10404.

11 Hsien-Chih Chang and Jeff Erickson. Untangling planar curves. *Discrete & Computational Geometry*, 58:889–920, 2017. doi:10.1007/s00454-017-9907-6.

12 Bernard Chazelle. A theorem on polygon cutting with applications. In *Proc. 23rd Annual Symposium on Foundations of Computer Science (FOCS 1982)*, pages 339–349, 1982. doi:10.1109/SFCS.1982.58.

13 Bernard Chazelle. On the convex layers of a planar set. *IEEE Trans. Inform. Theory*, 31(4):509–517, 1985. doi:10.1109/TIT.1985.1057060.

14 Kai-Seng Chou and Xi-Ping Zhu. *The Curve Shortening Problem*. Chapman & Hall/CRC, Boca Raton, FL, 2001. doi:10.1201/9781420035704.

15 Cristopher B. Croke. Area and the length of the shortest closed geodesic. *J. Differential Geometry*, 27:1–21, 1988.

16 Ketan Dalal. Counting the onion. *Random Struct. Algor.*, 24(2):155–165, 2004. doi:10.1002/rsa.10114.

17 William F. Eddy. Convex Hull Peeling. In *COMPSTAT 1982 5th Symposium held at Toulouse 1982*, pages 42–47. Physica-Verlag, 1982. doi:10.1007/978-3-642-51461-6_4.

18 Alon Efrat, Stephen G. Kobourov, and Anna Lubiw. Computing homotopic shortest paths efficiently. *Computational Geometry*, 35(3):162–172, 2006. doi:10.1016/j.comgeo.2006.03.003.

19 David Eppstein, Sariel Har-Peled, and Gabriel Nivasch. Grid peeling and the affine curve-shortening flow. *Experimental Mathematics*, page to appear, 2018. doi:10.1080/10586458.2018.1466379.

20 Radoslav Fulek and Csaba D. Tóth. Crossing minimization in perturbed drawings. In T. Biedl and A. Kerren, editors, *Proc. 26th Symp. Graph Drawing and Network Visualization*, pages 229–241. Springer, 2018. doi:10.1007/978-3-030-04414-5_16.

21 Michael Gage and Richard S. Hamilton. The heat equation shrinking convex plane curves. *J. Differential Geom.*, 23(1):69–96, 1986. doi:10.4310/jdg/1214439902.

22 Matthew A. Grayson. The heat equation shrinks embedded plane curves to round points. *J. Differential Geom.*, 26(2):285–314, 1987. doi:10.4310/jdg/1214441371.

23 Matthew A. Grayson. Shortening embedded curves. *Annals of Mathematics*, 129(1):79–111, 1989.

24 Sariel Har-Peled and Bernard Lidický. Peeling the grid. *SIAM J. Discrete Math.*, 27(2):650–655, 2013. doi:10.1137/120892660.

25 Joel Hass and Peter Scott. Shortening curves on surfaces. *Topology*, 33:25–43, 1994. doi:10.1016/0040-9383(94)90033-7.

26 John Hershberger and Jack Snoeyink. Computing minimum length paths of a given homotopy class. *Computational Geometry*, 4(2):63–97, 1994. doi:10.1016/0925-7721(94)90010-8.

27 Der-Tsai Lee and Franco P. Preparata. Euclidean shortest paths in the presence of rectilinear barriers. *Networks*, 14(3):393–410, 1984. doi:10.1002/net.3230140304.

28 Guillermo Sapiro and Allen Tannenbaum. Affine invariant scale-space. *Int. J. Comput. Vision*, 11(1):25–44, 1993. doi:10.1007/bf01420591.

29 Brian White. Evolution of curves and surfaces by mean curvature. In *Proceedings of the International Congress of Mathematicians, Vol. I (Beijing, 2002)*, pages 525–538, 2002. URL: https://www.mathunion.org/fileadmin/ICM/Proceedings/ICM2002.1/ICM2002.1.ocr.pdf.

Empty Squares in Arbitrary Orientation Among Points

Sang Won Bae
Division of Computer Science and Engineering, Kyonggi University, Suwon, South Korea
swbae@kgu.ac.kr

Sang Duk Yoon
Department of Service and Design Engineering, Sungshin Women's University, Seoul, South Korea
sangduk.yoon@sungshin.ac.kr

Abstract

This paper studies empty squares in arbitrary orientation among a set P of n points in the plane. We prove that the number of empty squares with four contact pairs is between $\Omega(n)$ and $O(n^2)$, and that these bounds are tight, provided P is in a certain general position. A contact pair of a square is a pair of a point $p \in P$ and a side ℓ of the square with $p \in \ell$. The upper bound $O(n^2)$ also applies to the number of empty squares with four contact points, while we construct a point set among which there is no square of four contact points. We then present an algorithm that maintains a combinatorial structure of the L_∞ Voronoi diagram of P, while the axes of the plane continuously rotate by 90 degrees, and simultaneously reports all empty squares with four contact pairs among P in an output-sensitive way within $O(s \log n)$ time and $O(n)$ space, where s denotes the number of reported squares. Several new algorithmic results are also obtained: a largest empty square among P and a square annulus of minimum width or minimum area that encloses P over all orientations can be computed in worst-case $O(n^2 \log n)$ time.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases empty square, arbitrary orientation, Erdős–Szekeres problem, L_∞ Voronoi diagram, largest empty square problem, square annulus

Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.13

Related Version A full version [8] of the paper is available at `https://arxiv.org/abs/1911.12988`.

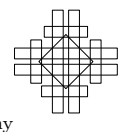
Funding *Sang Won Bae*: Supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (2018R1D1A1B07042755). *Sang Duk Yoon*: Supported by "Cooperative Research Program for Agriculture Science & Technology Development (Project No. PJ01526903)" Rural Development Administration, Republic of Korea.

1 Introduction

We start by posing the following combinatorial question:

> *Given a set P of n points in a proper general position in $\mathbb{R}^2$, how many empty squares in arbitrary orientation whose boundary contains four points in P can there be?*

For a square, its contact pair is a pair of a point $p \in P$ and a side ℓ of it such that $p \in \ell$, regarding ℓ as a segment including its endpoints. An analogous question asks the number of empty squares with four contact pairs. These questions can be seen as a new variant of the Erdős–Szekeres problem [22, 23] for empty squares. Let $s = s(P)$ and $s^* = s^*(P)$ be the number of empty squares with four contact points and with four contact pairs, respectively. The difference between contact points and contact pairs is that the first counts the number of points, while the second counts the number of incidences; in particular contact points that are corners count as two contact pairs. In this paper, we prove that $0 \le s < c_1 n^2$ and $c_2 n < s^* < c_3 n^2$ for some constants $c_1, c_2, c_3 > 0$. These lower and upper bounds are tight



by the existence of point sets with the asymptotically same number of such squares. These questions and results are shown to be intrinsic to several computational problems on empty squares, implying new algorithmic results.

For the purpose, we provide a solid understanding of empty squares in arbitrary orientation among P by establishing a geometric and topological relation among those squares. The family of axis-parallel empty squares is well understood by the L_∞ Voronoi diagram of P. We extend this knowledge to those in arbitrary orientation by investigating the L_∞ Voronoi diagrams of P with the axes rotated. Our proof for the above combinatorial results is based on new observations made upon the consideration of the Voronoi diagram in this way. This also motivates the problem of maintaining the L_∞ diagram of P while the axes continuously rotate. A noteworthy observation states that every combinatorial change of the diagram during the rotation of the axes corresponds to an empty square with four contact pairs. Hence, the total amount of changes of the diagram is bounded by $\Theta(s^*) = O(n^2)$.

Based on the observations, we then present an output-sensitive algorithm that finds all empty squares with four contact pairs in $O(s^* \log n)$ time using $O(n)$ space. Our algorithm indeed maintains a combinatorial description of the L_∞ Voronoi diagram as the axes continuously rotates by 90 degrees by capturing every occurrence of such special squares and handling them so that the diagram is correctly maintained as the invariants. To our best knowledge, there was no such algorithmic result in the literature. Our algorithm also applies to two other geometric problems, achieving new algorithmic results:

(1) A *largest empty square* in arbitrary orientation can be found in $O(n^2 \log n)$ time. This improves the previous $O(n^3)$-time algorithm by Bae [6]. We also solve some query versions of this problem.

(2) A *square annulus* is the closed region between a pair of concentric and parallel squares. A square annulus in arbitrary orientation with minimum width or area can be computed in $O(n^2 \log n)$ time, improving the previous $O(n^3 \log n)$ and $O(n^3)$-time algorithms [5,6].

Our combinatorial problem on the number of empty squares in arbitrary orientation can be viewed as a new variant of the *Erdős–Szekeres problem*, which has a long history since 1935 [22] with consistent effort [23, 41] on its original version and many other variants and generalizations [9–11, 18, 21, 24, 26, 28, 36, 38, 42]. Among them the most relevant to us is the problem of bounding the number of empty convex k-gons whose corners are chosen from P, called *k-holes*. The maximum number of k-holes among n points is proven to be $\Theta(n^k)$ for $k \geq 3$ and sufficiently large n by Bárány and Valtr [10]. Its minimum is known to be $\Theta(n^2)$ for $3 \leq k \leq 4$; $\Omega(n)$ and $O(n^2)$ for $5 \leq k \leq 6$ [9]; and zero for $k \geq 7$ [28]. For more details on this subject, see survey papers by Morris and Soltan [33] and [34].

Since any four points in P lying on the boundary of an empty square form a 4-hole, s cannot exceed the number of 4-holes among P. This, however, gives only trivial upper and lower bounds on s and s^*. In this paper, we give asymptotically tight bounds on s and s^* since we rather focus on algorithmic applications of the bounds.

Dobkin, Edelsbrunner, and Overmars [16] presented an algorithm that enumerates all convex k-holes for $3 \leq k \leq 6$. Their algorithm in particular for $k = 3, 4$ is indeed output-sensitive in time proportional to the number of reported k-holes, which is comparable to our algorithm that enumerates all squares with four contact pairs. Rote et al. [39], Rote and Woeginger [40], and Mitchell et al. [32] considered the problem of exactly counting convex k-gons and convex k-holes faster than enumerating them. Some minimization and maximization problems have also been considered in the literature [2, 13, 17, 19, 20].

The problem of maintaining the L_∞ (or, equivalently, L_1) Voronoi diagram while the axes rotate cannot be found in the literature. A similar paradigm about rotating axes can be seen in Bae et al. [7] in which the authors show how to maintain the *orthogonal convex*

hull of P in $O(n^2)$ time. Alegría-Galicia et al. [4] recently improved it into $O(n \log n)$ time using $O(n)$ space. As will be seen in the following, the orthogonal convex hull of P indeed describes the unbounded edges of the L_∞ Voronoi diagram of P.

The problem of finding a largest empty square is a square variant of the well-known *largest empty rectangle problem*. The largest empty rectangle problem is one of the most intensively studied problems in early time of computational geometry. After early results [15, 31, 35, 37], the currently fastest algorithm that finds a largest empty axis-parallel rectangle among P runs in $O(n \log^2 n)$ time by Aggarwal and Suri [3]. Mckenna et al. [31] proved a lower bound of $\Omega(n \log n)$ for this problem. Chauduri et al. [14] showed that there are $O(n^3)$ combinatorially different classes of maximal empty rectangles over all orientations and presented an $O(n^3)$-time algorithm that computes a largest empty rectangle in arbitrary orientation by enumerating all those classes.

The largest empty square problem, however, has attained relatively less interest. It is obvious that a largest empty axis-parallel square can be found in $O(n \log n)$ time by computing the L_∞ Voronoi diagram of P [29, 30], as also remarked in early papers, including Naamad et al. [35] and Chazelle et al. [15]. It is rather surprising, however, that no further result about the largest empty square problem in arbitrary orientation has been come up with for over three decades, to our best knowledge. From an easy observation that any maximal empty square in arbitrary orientation is contained in a maximal empty rectangle, Bae [6] recently showed how to compute a largest empty square in arbitrary orientation in $O(n^3)$ time. In this paper, we improve this to $O(n^2 \log n)$ time.

The square annulus problem asks to find a square annulus of minimum width or area that encloses a given set P of points. Its fixed-orientation version is solved in $O(n \log n)$ time [1, 25]. Bae [5] presented the first $O(n^3 \log n)$-time algorithm for the problem in arbitrary orientation, and improved it to $O(n^3)$ time [6]. In this paper, we further improve it to $O(n^2 \log n)$ time.

2 Preliminaries

We consider the standard coordinate system with the (horizontal) x-axis and the (vertical) y-axis in the plane $\mathbb{R}^2$. We mean by the *orientation* of any line, half-line, or line segment ℓ a unique real number $\theta \in [0, \pi)$ such that ℓ is parallel to a counter-clockwise rotated copy of the x-axis by θ.

For any square S in $\mathbb{R}^2$, the *orientation* of S is a real number $\theta \in [0, \pi/2)$ such that the orientation of each side of S is either θ or $\theta + \pi/2$. We regard the set $\mathbb{O} := [0, \pi/2)$ of all orientations of squares as a topological space homeomorphic to a circle.

Each side of a square, as a subset of $\mathbb{R}^2$, is assumed to include its incident corners. We identify the four sides of a square S by the *top*, *bottom*, *left*, and *right* sides, denoted by $\square(S)$, $\square(S)$, $\square(S)$, and $\square(S)$, respectively. This identification is clear, regardless of the orientation of S since it is chosen from $\mathbb{O} = [0, \pi/2)$. The *center* of a square is the intersection point of its two diagonals, and its *radius* is half its side length.

Let P be a set of n points in $\mathbb{R}^2$. A square is called *empty* if no point in P lies in its interior. An empty square may contain some points in P on its boundary. A pair $(p, \square)$ of a point $p \in P$ and a side identifier $\square \in \{\square, \square, \square, \square\}$ is called a *contact pair* of S if $p \in \square(S)$. A set of contact pairs is called a *contact type*. If κ is the set of all contact pairs of S, then we say that κ is the *contact type* of S. A *contact point* $p \in P$ of S is a point on a side of S, that is, one belonging to a contact pair of S. Each contact point $p \in P$ of S may lie either on the relative interior of a side of S or at a corner of S. In the former case, the contact point p contributes to one contact pair of S, while in the latter case, it contributes to two contact pairs.

Throughout the paper, we assume that P is in *general position* in the following sense:

There is no square in arbitrary orientation with five or more contact pairs among P.

Note that the assumption implies the number of empty squares with four contact pairs is finite.

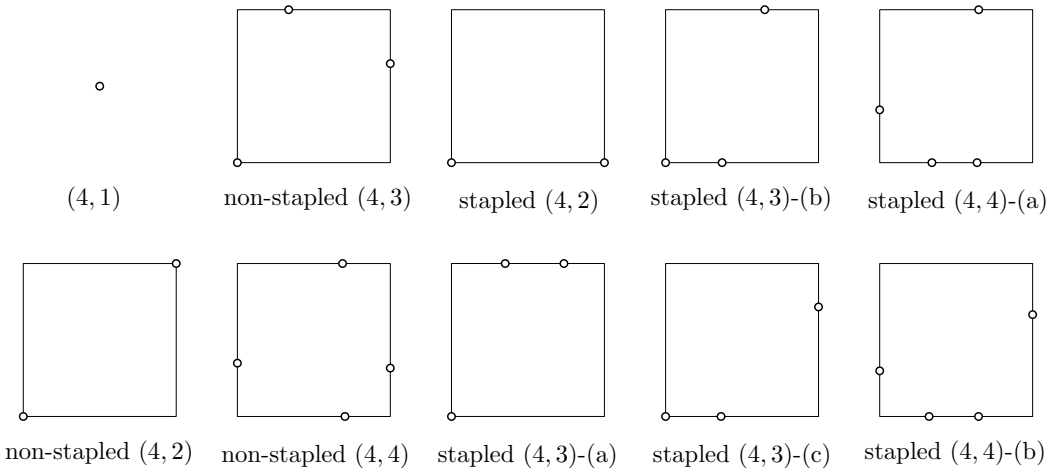


Figure 1 Illustration of 10 types of 4-squares and the type names. Small circles on the boundary of each square depict its contact points in P. The four left ones are non-stapled types and the six right ones are stapled types.

Consider any empty square S of contact type κ. We call S an *m-square* or *(m, k)-square* if $m = |\kappa|$ and k is the number of contact points in κ. A side of S is called *pinned* if it contains a point in P, so it is involved in some contact pair in κ; or *stapled* if it contains two distinct points in P. If S has a stapled side, then S is called *stapled*. From the general position assumption, there are no three or more contact pairs in κ involving a common side of S, and there is at most one stapled side of S.

We then classify the 4-squares into 10 types under the symmetry group of the square. See Figure 1 for an illustration to the 10 types of 4-squares with type names.

Throughout the paper, we are less interested in trivial $(4, 1)$-squares in most cases. Hereafter, we thus mean by a 4-square a nontrivial 4-square, that is, a $(4, k)$-square with $k \geq 2$, unless stated otherwise.

3 Empty Squares and the Voronoi Diagram

The empty squares among P are closely related to the Voronoi diagram of P under the L_∞ distance. In this section, we define the Voronoi diagram for every $\theta \in \mathbb{O}$ based on empty squares and collect several essential properties of empty squares in terms of the Voronoi diagram, based on which we will be able to bound the number of 4-squares and to present an efficient algorithm that computes all 4-squares.

3.1 Definition of Voronoi diagrams

Let P be a given set of n points in general position as discussed in Section 2. For each $\theta \in \mathbb{O}$, we define $\mathsf{VD}(\theta)$ to be the L_∞ Voronoi diagram of P with the axes rotated by θ, or equivalently, the Voronoi diagram of P under the symmetric convex distance function d_θ based on a unit square whose orientation is θ. The *Voronoi region* of $p \in P$ in θ is

$$VR_p(\theta) := \{x \in \mathbb{R}^2 \mid d_\theta(x, p) < d_\theta(x, q), q \in P \setminus \{p\}\}.$$

The diagram $\mathsf{VD}(\theta)$ can also be defined in terms of empty squares. More precisely, we view $\mathsf{VD}(\theta)$ as a plane graph whose vertices $\hat{V}(\theta)$ and edges $\hat{E}(\theta)$ are determined as follows:

- The vertex set $\hat{V}(\theta)$ consists of the centers of all empty squares in orientation θ with three or four pinned sides, and a point *at infinity*, denoted by $\hat{\infty}$.
- An edge is contained in $\hat{E}(\theta)$ if and only if it is a maximal set of centers of all empty squares in orientation θ having a common contact type with two pinned sides. Each edge in $\hat{E}(\theta)$ is either a half-line or a line segment, called *unbounded* or *bounded*, respectively.

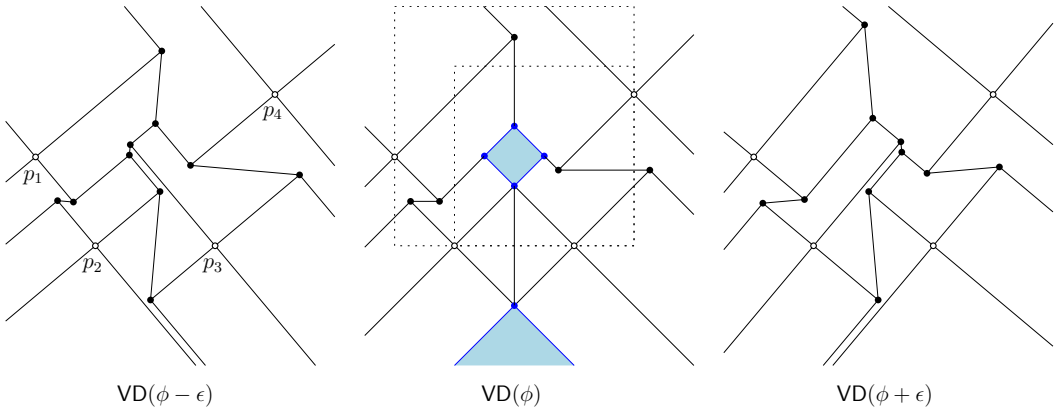


Figure 2 Illustration of $\mathsf{VD}(\phi-\epsilon)$, $\mathsf{VD}(\phi)$, and $\mathsf{VD}(\phi+\epsilon)$ for $P=\{p_1,\dots,p_4\}$ and some $\phi\in\mathbb{O}$.

See Figure 2. From our definition, there are four more edges incident to each $p\in P$ such that each of them corresponds to 2-squares with one corner anchored at p. In this way, each point $p\in P$ is also a vertex in $\hat{V}(\theta)$ since p is the center of a trivial $(4,1)$-square with its four sides pinned. The diagram $\mathsf{VD}(\theta)$ as a plane graph divides the plane into its *faces*. Each face of $\mathsf{VD}(\theta)$ is the locus of centers of empty squares having a common contact type with *one pinned side*; hence, for every contact pair $(p,\square)$, there exists a unique face of $\mathsf{VD}(\theta)$ consisting of the centers of 1-squares with contact type $\{(p,\square)\}$. Therefore, the Voronoi region $VR_p(\theta)$ of each $p\in P$ includes exactly four faces of $\mathsf{VD}(\theta)$. On the other hand, there may exist some *neutral* faces of $\mathsf{VD}(\theta)$ that do not belong to any Voronoi region $VR_p(\theta)$, if it corresponds to 2-squares with a stapled side (see the two shaded faces of $\mathsf{VD}(\phi)$ in color lightblue in Figure 2). This is obviously a degenerate case which has been avoided from most discussions about Voronoi diagrams in the literature. In this way, our definition of $\mathsf{VD}(\theta)$ completely represents all cases of point set P, even if there are four equidistant points in P under d_θ or there are two points in P such that pq is in orientation θ or $\theta+\pi/2$.

The *combinatorial structure* of $\mathsf{VD}(\theta)$ is represented by its underlying graph $VG(\theta)=(V(\theta),E(\theta))$, called the *Voronoi graph*; conversely, $\mathsf{VD}(\theta)$ is a plane embedding of $VG(\theta)$. More precisely, the vertices and edges of $VG(\theta)$ are described and identified as follows:

- Each vertex $v\in V(\theta)$ corresponds to $\hat{v}\in\hat{V}(\theta)$. In particular, the vertex at infinity, denoted by $\infty\in V(\theta)$, corresponds to $\hat{\infty}\in\hat{V}(\theta)$. Each $v\in V(\theta)\setminus\{\infty\}$ is identified by the contact type κ_v of the square defining $\hat{v}\in\hat{V}(\theta)$. For completeness, we define $\kappa_\infty:=\emptyset$.
- Each edge $e\in E(\theta)$ corresponds to $\hat{e}\in\hat{E}(\theta)$. Each edge $e=uv\in E(\theta)$ for $u,v\in V(\theta)$ is identified by a triple $(\kappa_u,\kappa_v;\kappa_e)$, where κ_e is the contact type of the squares defining $\hat{e}\in\hat{E}(\theta)$. If e is bounded, then we have $\kappa_e=\kappa_u\cap\kappa_v$.

Hence, for $\theta,\theta'\in\mathbb{O}$, two vertices $v\in V(\theta)$ and $v'\in V(\theta')$ are the same if $\kappa_v=\kappa_{v'}$; two edges $uv\in E(\theta)$ and $u'v'\in E(\theta')$ are the same if $(\kappa_u,\kappa_v;\kappa_{uv})=(\kappa_{u'},\kappa_{v'};\kappa_{u'v'})$. We say that $\mathsf{VD}(\theta)$ and $\mathsf{VD}(\theta')$ are *combinatorially equivalent* if $VG(\theta)=VG(\theta')$.

3.2 Basic properties

For each vertex $v \in V(\theta)$, we call v and its embedding $\hat{v} \in \hat{V}(\theta)$ *regular* if $|\kappa_v| = 3$, that is, its corresponding empty square is a 3-square. For each edge $e \in E(\theta)$, we call e and its embedding $\hat{e} \in \hat{E}(\theta)$ *regular* if $|\kappa_e| = 2$.

Each edge $e \in E(\theta)$ is called *sliding* if the two pinned sides in κ_e are parallel, or *growing*, otherwise. From the properties of the L_∞ Voronoi diagram, we observe that any sliding edge is in orientation θ or $\theta + \pi/2$, while any growing edge is in orientation $\theta + \pi/4$ or $\theta + 3\pi/4$ (modulo π). For $e \in E(\theta)$, we regard each growing edge to be directed in which its corresponding square is growing.

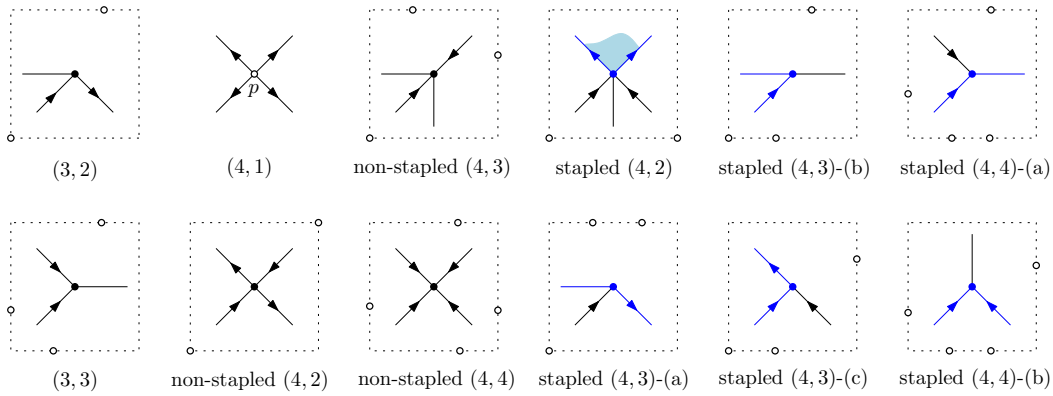


Figure 3 Illustration of the 12 vertex types of $\mathsf{VD}(\theta)$ labeled with their type names. Dotted squares and small circles on them depict an empty square corresponding to each vertex type and its contact points. Dots and line segments depict vertices and all incident edges to each vertex; in black if regular, or in blue, otherwise. The arrows on edges depict the direction in which the corresponding square grows. The shaded area in the stapled $(4, 2)$-type depicts a neutral face which does not belong to any Voronoi region $VR_p(\theta)$ for $p \in P$. (We keep the above convention on every figure in this paper.) Note that the right six types are stapled, while the left six are not.

For each vertex $v \in V(\theta)$ with $v \neq \infty$, the local structure of the diagram $\mathsf{VD}(\theta)$ around $\hat{v}$ is completely determined by its contact type κ_v. From the possible contact types of 3- and 4-squares, we classify all vertices in $V(\theta) \setminus \{\infty\}$ into 12 vertex types.

▶ **Lemma 3.1.** *There are* 12 *types for the vertices of* $\mathsf{VD}(\theta)$ *as illustrated in Figure 3. For any vertex* $v \in V(\theta) \setminus \{\infty\}$*, the contact types* κ_e *of all edges* $e \in E(\theta)$ *incident to* v *can be obtained just from the contact type* κ_v *without knowing the other incident vertex of* e*.*

Consider all squares with contact type κ_e defining $\hat{e} \in \hat{E}(\theta)$. If e is sliding, then all these squares have the same radius; if e is growing, then their radius grows along $\hat{e}$. The following lemma is an immediate observation.

▶ **Lemma 3.2** (Boissonnat et al. [12])**.** *Let* $e = uv \in E(\theta)$ *be any bounded edge for any* $\theta \in \mathbb{O}$*, and* S_u *and* S_v *be the empty squares in* θ *with contact types* κ_u *and* κ_v*, respectively. Then, the union* R *of all squares in* θ *with contact type* κ_e *is equal to* $S_u \cup S_v$*. More specifically, if* e *is sliding, then* $R = S_u \cup S_v$ *forms a rectangle; if* e *is growing, then one of* S_u *and* S_v *completely contains the other, so* R *is a square.*

The above lemma indeed extends to the case of unbounded edges. Consider any unbounded edge $e \in E(\theta)$ and the union R of all squares as declared in Lemma 3.2. Observe that R forms an empty unbounded quadrant rotated by θ and the empty quadrant R has two or three

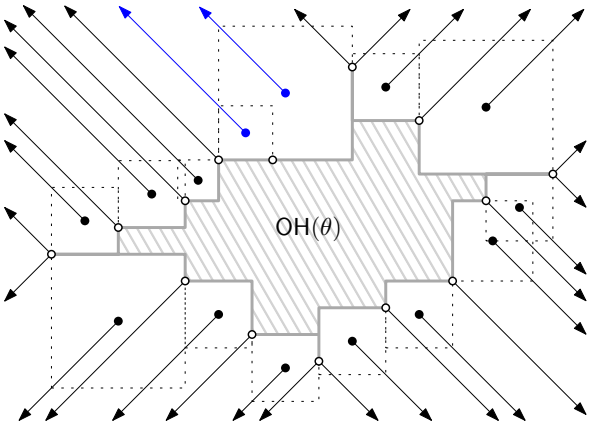


Figure 4 Illustration to Lemma 3.3. The orthogonal convex hull $\mathsf{OH}(\theta)$ of P (shaded region) and the four staircases (gray thick lines) describe all unbounded edges (arrows) and their incident vertices of $\mathsf{VD}(\theta)$ (small circles and dots).

contact points in P. This tells us a relation between unbounded edges and the orthogonal convex hull of P. The *orthogonal convex hull* of point set P is defined to be the minimal subset of $\mathbb{R}^2$ such that any vertical or horizontal line intersects it in at most one connected component. It is known that the orthogonal convex hull is obtained by subtracting all empty quadrants (of four directions) from the whole plane $\mathbb{R}^2$ and its boundary is represented by four monotone chains, called the *staircases*. For $\theta \in \mathbb{O}$, let $\mathsf{OH}(\theta)$ denote the orthogonal convex hull of P with the axes rotated by θ. See Figure 4 and Bae et al. [7] for more details on $\mathsf{OH}(\theta)$, including the precise definition of $\mathsf{OH}(\theta)$ and the staircases.

▶ **Lemma 3.3.** *For any $\theta \in \mathbb{O}$, all the unbounded edges and their incident vertices of $\mathsf{VD}(\theta)$ are explicitly described by $\mathsf{OH}(\theta)$, in the sense that if $v \in V(\theta) \setminus \{\infty\}$ is a vertex incident to an unbounded edge, then either*

- **(i)** *we have $\hat{v} = p \in P$ and p coincides with a vertex of $\mathsf{OH}(\theta)$, or*
- **(ii)** *its contact points in κ_v appear consecutively in a staircase of $\mathsf{OH}(\theta)$.*

Another implication of Lemma 3.1 is that the degree of every vertex, except $\infty \in V(\theta)$, is at least three and at most five. This implies the linear complexity of $\mathsf{VD}(\theta)$ for any $\theta \in \mathbb{O}$.

▶ **Lemma 3.4.** *For any $\theta \in \mathbb{O}$, the number of vertices, edges, and faces of $\mathsf{VD}(\theta)$ is $\Theta(n)$.*

3.3 Combinatorial changes of $\mathsf{VD}(\theta)$ and 4-squares

Let $\mathfrak{S}_4$ be the set of all nontrivial 4-squares among P. By our general position assumption on P, we know that $\mathfrak{S}_4$ is finite. Let $s_4 := |\mathfrak{S}_4|$ be the number of 4-squares among P. For any orientation $\theta \in \mathbb{O}$, we call θ *regular* if there is no nontrivial 4-square in orientation θ, or *degenerate*, otherwise. Since there are only finitely many (exactly s_4) 4-squares, all orientations $\theta \in \mathbb{O}$ but at most s_4 of them are regular.

In a regular orientation θ, the diagram $\mathsf{VD}(\theta)$ has the following properties.

▶ **Lemma 3.5.** *For any regular orientation $\theta \in \mathbb{O}$, every vertex in $V(\theta)$, except those in P and ∞, is regular, and every edge in $E(\theta)$ is regular. There are five types of bounded edges, as shown in Figure 5, and two types of unbounded edges.*

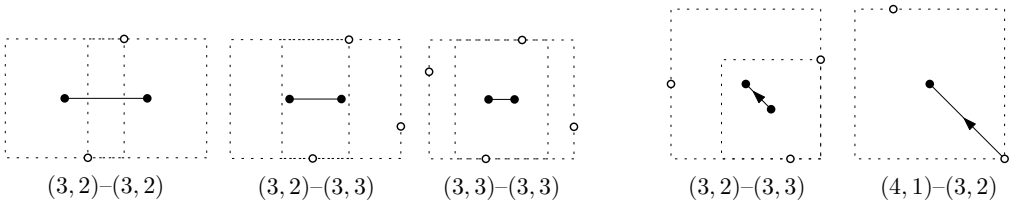


Figure 5 Illustration of five types of bounded edges of $\mathsf{VD}(\theta)$ for a regular orientation $\theta \in \mathbb{O}$, depending on the types of incident vertices. The first three are sliding and the last two are growing.

Consider any contact type κ with three contact pairs and three pinned sides. For any $\theta \in \mathbb{O}$, let $S_\kappa(\theta)$ be the square in orientation θ whose contact type includes the three contact pairs in κ, regardless of its emptiness. Note that $S_\kappa(\theta)$ is well defined only in a closed interval of $\mathbb{O}$, or is never defined for any $\theta \in \mathbb{O}$. For each θ for which $S_\kappa(\theta)$ is well defined, we call θ *valid for* κ if $S_\kappa(\theta)$ is empty and its contact type is exactly κ. Note that if θ is valid for κ, then there exists a vertex $v \in V(\theta)$ with $\kappa_v = \kappa$.

▶ **Lemma 3.6.** *For any contact type κ with three contact pairs and three pinned sides, the set of all valid orientations for κ forms zero or more open intervals $I \subset \mathbb{O}$ such that $S_\kappa(\phi)$ is a 4-square for any endpoint ϕ of I.*

We call each of these open intervals described in Lemma 3.6 a *valid interval* for κ. Lemma 3.6 also implies that any endpoint of a valid interval for κ is a degenerate orientation.

For any degenerate orientation $\phi \in \mathbb{O}$, let $\mathfrak{S}_4(\phi) \subseteq \mathfrak{S}_4$ be the set of 4-squares whose orientation is ϕ. Note that $\mathfrak{S}_4(\phi)$ is nonempty and consists of at most $O(n)$ squares by Lemma 3.4. We then observe the following.

▶ **Lemma 3.7.** *Let $S \in \mathfrak{S}_4(\phi)$ be any 4-square in orientation ϕ and κ be its contact type. Then, there are exactly two or four contact types κ' with three contact pairs and three pinned sides such that $S_{\kappa'}(\phi) = S$ and either $\phi - \epsilon$ or $\phi + \epsilon$ is valid for κ' for sufficiently small $\epsilon > 0$. Specifically, the number of such κ' is four if S is non-stapled, or two if S is stapled.*

Figure 6 illustrates transitions of a non-regular vertex, which corresponds to a 4-square in $\mathfrak{S}_4(\phi)$, from and to regular vertices, locally at $\theta = \phi$, so almost describes our proof for Lemma 3.7. Observe that any non-stapled 4-square is relevant to an *edge flip* of $\mathsf{VD}(\theta)$, and that only stapled $(4, 3)$-(b)- or (c)-type squares show a bit different behavior; such a non-regular vertex suddenly appears on a regular edge and splits to two regular vertices, or, reversely, two regular vertices are merged into such a non-regular one and soon disappear.

The above discussions provide us a thorough view on the 3-squares and the 4-squares. Consider the graph $\mathfrak{G}$ whose vertex set is the set $\mathfrak{S}_4$ of all 4-squares and whose edge set consists of edges (S, S') for $S, S' \in \mathfrak{S}_4$ such that there is a valid interval $I = (\phi, \phi')$ for some contact type κ such that $S_\kappa(\phi) = S$ and $S_\kappa(\phi') = S'$. The graph $\mathfrak{G}$ is well defined by Lemma 3.6 and the degree of every vertex in $\mathfrak{G}$ is exactly two or four by Lemma 3.7.

4 Number of 4-Squares

In this section, we prove asymptotically tight upper and lower bounds on the number of 4-squares and $(4, k)$-squares for each $2 \le k \le 4$. Following are two main theorems.

▶ **Theorem 4.1.** *The number of 4-squares among n points in general position is always between $\Omega(n)$ and $O(n^2)$. These lower and upper bounds are asymptotically tight.*

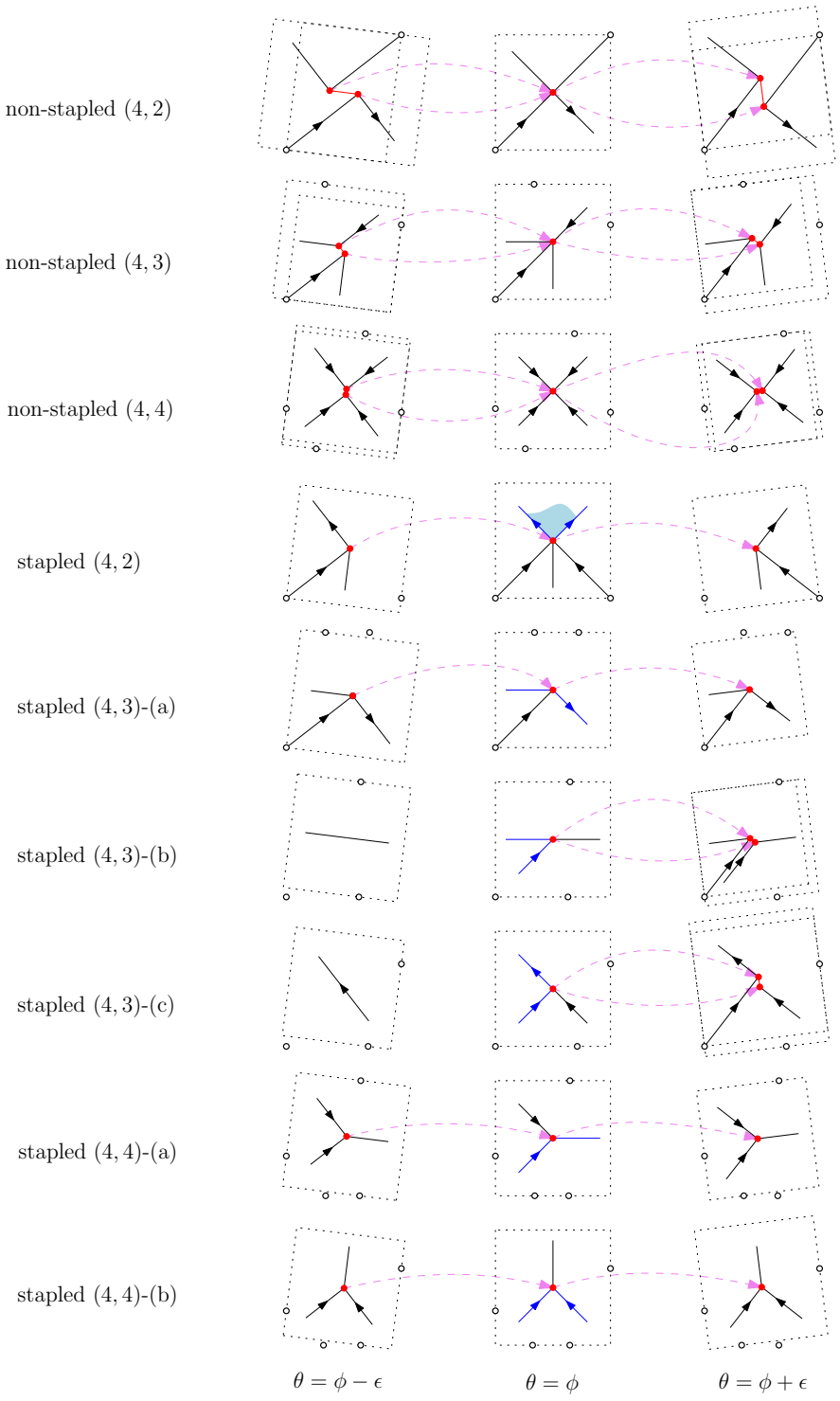


Figure 6 Transitions between regular (3-squares) and non-regular vertices (4-squares).

▶ **Theorem 4.2.** *Among n points in general position, the number of empty squares whose boundary contains four points of P is between 0 and $O(n^2)$. These lower and upper bounds are asymptotically tight.*

For any positive integers m and $k \leq m$, let $s_m(P)$ and $s_{m,k}(P)$ be the number of m-squares and (m,k)-squares, respectively, among P. We then define

$$\sigma_m(n) := \min_{|P|=n} s_m(P), \qquad \sigma_{m,k}(n) := \min_{|P|=n} s_{m,k}(P),$$
$$\Sigma_m(n) := \max_{|P|=n} s_m(P), \quad \text{and} \quad \Sigma_{m,k}(n) := \max_{|P|=n} s_{m,k}(P).$$

In the following, we show the asymptotically tight bounds on these quantities for $m = 4$. It is obvious that $\sigma_{4,1}(n) = \Sigma_{4,1}(n) = n$ and $\Sigma_{4,k}(n) \leq \binom{n}{k}$ for $2 \leq k \leq 4$. So, we have $\sigma_4(n) = \Omega(n)$ and $\Sigma_4(n) = O(n^4)$.

4.1 Upper bounds

Here, we prove the upper bounds $\Sigma_{4,k}(n)$ for $2 \leq k \leq 4$ and $\Sigma_4(n)$ are quadratic in n. We first show that there exists a point set P with $n = |P|$ having $\Omega(n^2)$ many 4-squares.

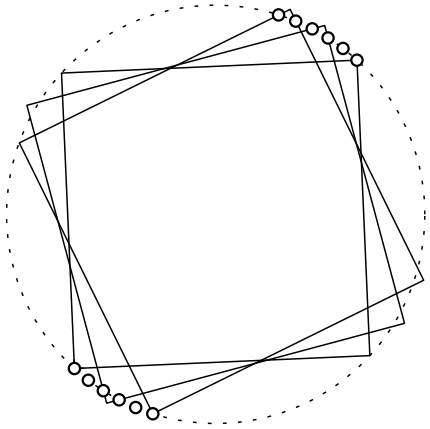


Figure 7 Illustration of a point set P_n with $\Omega(n^2)$ many 4-squares.

▶ **Lemma 4.3.** *For any $n \geq 4$, there exists a set P_n of n points such that $s_{4,k}(P_n) = \Omega(n^2)$ for each $2 \leq k \leq 4$ and thus $s_4(P_n) = \Omega(n^2)$.*

This already proves that $\Sigma_{4,2}(n) = \Theta(n^2)$. In the following, we prove the matching upper bounds of $\Sigma_{4,3}(n)$ and $\Sigma_{4,4}(n)$.

Upper bound of $\Sigma_{4,3}(n)$

Any $(4,3)$-square is of one of the four types: non-stapled $(4,3)$ and stapled $(4,3)$-(a–c) types, see Figure 1. We bound the number of $(4,3)$-squares of each type, separately.

First, consider the stapled $(4,3)$-(b–c) types. Note that if S is in this case with two contact points $p, q \in P$ on its stapled side, then one of p and q also lie on a corner of S.

▶ **Lemma 4.4.** *Let $S \in \mathfrak{S}_4(\phi)$ be a square of stapled $(4,3)$-(b)- or (c)-type with contact type κ, and $v \in V(\phi)$ be the vertex with $\kappa_v = \kappa$. Then, there exists a stapled $(4,2)$-square $S' \in \mathfrak{S}_4(\phi)$ with contact type κ' such that v is adjacent to $v' \in V(\phi)$ with $\kappa_{v'} = \kappa'$ in $VG(\phi)$.*

From Lemma 3.1, we know that each vertex $v' \in V(\phi)$ of stapled $(4,2)$-type has two growing edges whose growing direction is outwards from v'. Hence, for each vertex of stapled $(4,2)$-type, there can be at most two adjacent vertices whose corresponding square is larger and has three contact points. This implies that the number of stapled $(4,3)$-(b)- and (c)-type squares is at most twice the number of stapled $(4,2)$-squares, which is $O(n^2)$, by Lemma 4.4.

Next, we consider the other two types of $(4,3)$-squares: non-stapled $(4,3)$-type and stapled $(4,3)$-(a)-type. Consider any $(4,3)$-square $S \in \mathfrak{S}_4(\phi)$ whose type is one of the above two. Without loss of generality, assume that a contact point $p \in P$ lies on the bottom-left corner of S. Then, regardless of its specific type, the other two contact points $q_1, q_2 \in P$ lie on either the top or the right side of S. So, $\kappa = \{(p,\square), (p,\square), (q_1,\square_1), (q_2,\square_2)\}$ for $\square_1,\square_2 \in \{\square,\square\}$. See Figure 1. Notice that q_1 and q_2 are two equidistantly closest points from p under the distance function d_ϕ among those points in the quadrant with apex p in orientation ϕ.

In the following, we count the number of those pairs (q_1, q_2) with the discussed property for each fixed $p \in P$. This bounds the number of those $(4,3)$-squares with p on the bottom-left corner. More precisely, let ℓ_θ be the upward half-line from p in orientation $\theta \in [0,\pi)$, and $\angle_\theta$ be the cone with apex p and angle span $\pi/4$ defined by two half-lines ℓ_θ and $\ell_{\theta+\pi/4}$. Then, for any $\theta \in \mathbb{O}$, consider the two subsets

$$Q_{\square}(\theta) := P \setminus \{p\} \cap \angle_\theta \quad \text{and} \quad Q_{\square}(\theta) := P \setminus \{p\} \cap \angle_{\theta+\pi/4}.$$

For any $q \in P \setminus \{p\}$, define a function $f_q \colon \mathbb{O} \to \mathbb{R}$ to be

$$f_q(\theta) := \begin{cases} d_\theta(p,q) & \text{if } q \in Q_{\square}(\theta) \cup Q_{\square}(\theta) \\ \infty & \text{otherwise} \end{cases}.$$

Then, our task is reduced to find the complexity of the lower envelope F of the functions f_q for all $q \in P \setminus \{p\}$. By a trick similar to that used in Hershberger [27], we show the following.

▶ **Lemma 4.5.** *The complexity of the lower envelope F of f_q for all $q \in P \setminus \{p\}$ is $O(n)$.*

This proves that the number of non-stapled $(4,3)$-squares and stapled $(4,3)$-(a)-squares among P is at most $O(n^2)$. Combining the above discussions about stapled $(4,3)$-(b–c) squares and Lemma 4.3, we conclude that $\Sigma_{4,3}(n) = \Theta(n^2)$.

Upper bound of $\Sigma_{4,4}(n)$

Now, we prove the matching upper bound on the number of $(4,4)$-squares. Any $(4,4)$-square is one of the three types: non-stapled $(4,4)$ and stapled $(4,4)$-(a–b) types. See Figure 1.

▶ **Lemma 4.6.** *Let $S \in \mathfrak{S}_4(\phi)$ be a stapled $(4,4)$-square with contact type κ, and $v \in V(\phi)$ be the vertex with $\kappa_v = \kappa$. Then, there exists a stapled $(4,3)$-square $S' \in \mathfrak{S}_4(\phi)$ with contact type κ' such that v is adjacent to $v' \in V(\phi)$ in $VG(\phi)$ with $\kappa_{v'} = \kappa'$.*

From Lemma 3.1, we know that each vertex $v' \in V(\phi)$ of stapled $(4,3)$-type has at most one growing edge whose growing direction is outwards from v'. Hence, for each vertex of stapled $(4,3)$-type, there can be at most one adjacent vertex whose corresponding square is larger and has four contact points. This implies that the number of stapled $(4,4)$-squares is at most the number of stapled $(4,3)$-squares, which is $O(n^2)$ by Lemmas 4.4 and 4.6.

Lastly, we bound the number of non-stapled $(4,4)$-squares. Recall that we have so far proved that the number of 4-squares whose type is not non-stapled $(4,4)$-type is $O(n^2)$.

▶ **Lemma 4.7.** *In the graph $\mathfrak{G}$ defined in Section 3, any non-stapled $(4,4)$-square is adjacent to at least one 4-square that is not of non-stapled $(4,4)$-type.*

Lemma 3.7 states that the degree of each 4-square in $\mathfrak{S}_4$ is of degree at most four, and hence is adjacent to at most four non-stapled $(4,4)$-squares. Since the number of 4-squares that is not of non-stapled $(4,4)$-type is $O(n^2)$ and each non-stapled $(4,4)$-square is adjacent to one of them by Lemma 4.7, we conclude that the number of non-stapled $(4,4)$-squares is also $O(n^2)$. This proves the upper bound of Theorem 4.2.

Since $\Sigma_4(n) \leq \sum_{1\leq k\leq 4} \Sigma_{4,k}(n)$, we have that $\Sigma_4(n) = O(n^2)$. By Lemma 4.3, we have $\Sigma_4(n) = \Omega(n^2)$, completing our proof for the claimed upper bound of Theorem 4.1.

4.2 Lower bounds

We then turn into proving the lower bounds. As discussed above, we already have $\sigma_4(n) = \Omega(n)$, which matches the claimed lower bound in Theorem 4.1. Here, we show $\Omega(n)$ lower bounds for $\sigma_{4,2}(n)$ and $\sigma_{4,3}(n)$, and then construct a point set with $O(n)$ 4-squares.

▶ **Lemma 4.8.** *For any integer $n \geq 3$, $\sigma_{4,2}(n) = \Omega(n)$ and $\sigma_{4,3}(n) = \Omega(n)$.*

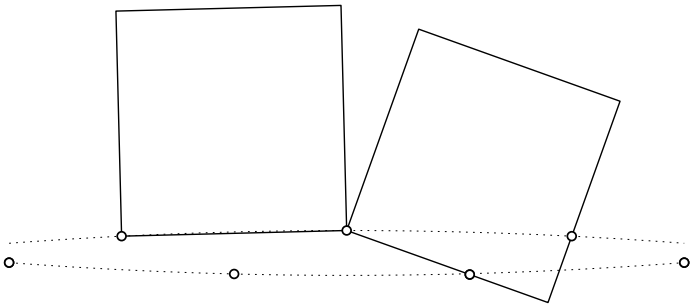


Figure 8 Illustration of a point set P'_n with $O(n)$ many 4-squares.

We finally construct a point set having a small number of 4-squares.

▶ **Lemma 4.9.** *For any integer $n \geq 1$, there exists a set P'_n of n points such that $s_{4,2}(P'_n) = O(n)$, $s_{4,3}(P'_n) = O(n)$, $s_{4,4}(P'_n) = 0$, and thus $s_4(P'_n) = O(n)$.*

Consequently, by Lemma 4.9, we have $\sigma_{4,2}(n) = \Theta(n)$, $\sigma_{4,3}(n) = \Theta(n)$, and thus $\sigma_4(n) = \Theta(n)$, while $\sigma_{4,4}(n) = 0$. This proves the lower bounds in Theorems 4.1 and 4.2.

5 Maintaining the L_∞ Voronoi Diagram under Rotation

Lemma 3.6 implies that for any two consecutive degenerate orientations $\phi_1, \phi_2 \in \mathbb{O}$ the vertex set $V(\theta)$ stays the same for all $\phi_1 < \theta < \phi_2$. By Lemma 3.2, a change in the edge set $E(\theta)$ happens when and only when its incident vertices change, so $\mathsf{VD}(\theta)$ for all $\phi_1 < \theta < \phi_2$ are combinatorially equivalent. So, the combinatorial change of $\mathsf{VD}(\theta)$ happens at every degenerate orientation $\theta = \phi$ only.

In the following, $\epsilon \in \mathbb{R}$ denotes arbitrarily small positive real. For any degenerate orientation $\phi \in \mathbb{O}$ and a 4-square $S \in \mathfrak{S}_4(\phi)$ with contact type κ, let $V_S^- \subseteq V(\phi - \epsilon)$ and $V_S^+ \subseteq V(\phi + \epsilon)$ be the sets of regular vertices v in $V(\phi - \epsilon)$ and $V(\phi + \epsilon)$, respectively, such that $\kappa_v \subset \kappa$. For each degenerate orientation $\phi \in \mathbb{O}$, define

$$V^-(\phi) := V(\phi - \epsilon) \setminus V(\phi + \epsilon) \quad \text{and} \quad V^+(\phi) := V(\phi + \epsilon) \setminus V(\phi - \epsilon)$$

to be the sets of vertices to be deleted and inserted, respectively, as θ goes through ϕ. Similarly, define

$$E^-(\phi) := E(\phi - \epsilon) \setminus E(\phi + \epsilon) \quad \text{and} \quad E^+(\phi) := E(\phi + \epsilon) \setminus E(\phi - \epsilon).$$

Lemmas 3.6 and 3.7 imply the following.

▶ **Lemma 5.1.** *For any degenerate orientation $\phi \in \mathbb{O}$, the following hold:*
(i) $V^-(\phi) = \bigcup_{S \in \mathfrak{S}_4(\phi)} V_S^-$ *and* $V^+(\phi) = \bigcup_{S \in \mathfrak{S}_4(\phi)} V_S^+$.
(ii) $E^-(\phi)$ *and* $E^+(\phi)$ *consist of edges incident to a vertex in* $V^-(\phi)$ *and* $V^+(\phi)$*, respectively.*
(iii) $|V^-(\phi)| + |V^+(\phi)| + |E^-(\phi)| + |E^+(\phi)| = \Theta(|\mathfrak{S}_4(\phi)|)$.

5.1 Events

Thus, every combinatorial change of $\mathsf{VD}(\theta)$ can be specified by finding all degenerate orientations and all 4-squares. For the purpose, our algorithm handles *events*.

- An *edge event* is a pair (e, ϕ) for a bounded edge $e \in E(\phi - \epsilon)$ and an orientation $\phi \in \mathbb{O}$ such that the embedding $\hat{e} \in \hat{E}(\phi - \epsilon)$ of e is about to collapse into a point in orientation ϕ. As a result, the two vertices u, v incident to e are merged into one with contact type $\kappa_u \cup \kappa_v$, and there is a unique 4-square $S \in \mathfrak{S}_4(\phi)$ such that $S = S_{\kappa_u}(\phi) = S_{\kappa_v}(\phi)$ and its contact type is $\kappa_u \cup \kappa_v$. We call S the *relevant* square to this edge event.
- An *align event* is a triple (p, q, ϕ) for distinct $p, q \in P$ and $\phi \in \mathbb{O}$ such that the orientation of segment pq is either ϕ or $\phi + \pi/2$ and there is a 4-square in $\mathfrak{S}_4(\phi)$ one of whose sides contains both p and q. Each such 4-square is called *relevant* to this align event.

We then observe the following lemmas.

▶ **Lemma 5.2.** *For any degenerate orientation $\phi \in \mathbb{O}$, an event occurs at ϕ. More precisely, every stapled 4-square in $\mathfrak{S}_4(\phi)$ is relevant to an align event at ϕ and every non-stapled 4-square in $\mathfrak{S}_4(\phi)$ is relevant to an edge event at ϕ.*

We call an align event (p, q, ϕ) *an outer align event* if both p and q appear consecutively on the boundary of $\mathsf{OH}(\phi)$ or, otherwise, *an inner align event*. By Lemma 3.3, we can precompute all outer align events using the algorithm by Alegría-Galicia et al. [4].

▶ **Lemma 5.3.** *There are $O(n)$ outer align events and we can compute them in $O(n \log n)$ time.*

We then observe the following for inner align events. See also Figure 9.

▶ **Lemma 5.4.** *If an inner align event occurs at $\phi \in \mathbb{O}$, then there is a stapled $(4, 3)$-square in $\mathfrak{S}_4(\phi)$ that is relevant to an edge event that occurs at ϕ.*

Lemma 5.4 implies that every inner align event can be noticed by handling an edge event whose relevant 4-square is of stapled $(4, 3)$-type. This, together with Lemma 5.3, allows us to maintain the diagram $\mathsf{VD}(\theta)$ in an efficient and output-sensitive way, as we do not need to test all pairs of points $p, q \in P$ for potential align events.

In order to catch every edge event, we define the potential edge event as follows: for any regular orientation $\theta \in \mathbb{O}$ and any bounded edge $e = uv \in E(\theta)$, the *potential edge event* $w(e, \theta)$ for e and θ is a pair (e, ϕ) such that $\theta < \phi < \pi/2$ and $S_{\kappa_u}(\phi) = S_{\kappa_v}(\phi)$, regardless of its emptiness. If such ϕ does not exist, then $w(e, \theta)$ is undefined.

▶ **Lemma 5.5.** *The potential edge event $w(e, \theta)$ is uniquely defined, unless undefined. Given e and θ, one can decide if $w(e, \theta)$ is defined and compute it, if defined, in $O(1)$ time.*

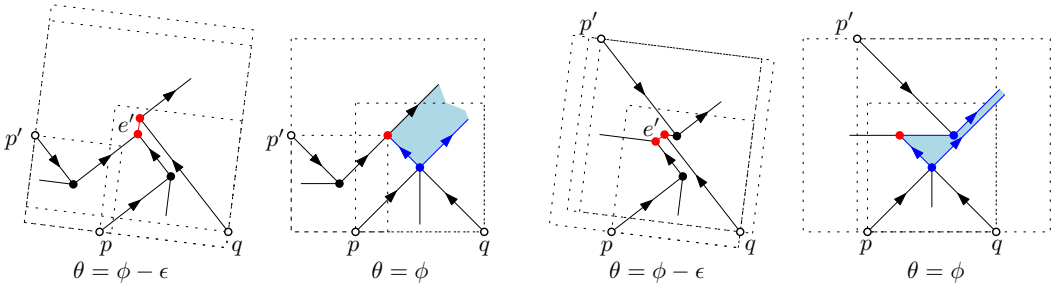


Figure 9 Illustration to the proof of Lemma 5.4. This figure shows the combinatorial changes around the vertex corresponding to a stapled $(4, 2)$-square and incident edges (a) when p' lies on the left side of S' and (b) when p' lies on the top side of S', at $\theta = \phi - \epsilon$ and $\theta = \phi$. The red edge $\hat{e}$ in $\phi - \epsilon$ is collapsed into the red vertex in ϕ, and the blue edges are those that are non-regular.

5.2 Algorithm

Our algorithm maintains the combinatorial structure $VG(\theta)$ of the Voronoi diagrams $\mathsf{VD}(\theta)$ as $\theta \in \mathbb{O}$ continuously increases from 0 to $\pi/2$. For the purpose, we increase θ and stop at every degenerate orientation ϕ to find all 4-squares in $\mathfrak{S}_4(\phi)$ and update $VG(\theta)$ according to the corresponding changes. For the purpose, we maintain data structures, keeping the invariants at the current orientation $\theta \in \mathbb{O}$ as follows.

- The *graph* $G = (V, E)$ stores the current Voronoi graph $VG(\theta) = (V(\theta), E(\theta))$.
- The *event queue* $\mathcal{Q}$ is a priority queue that stores potential edge events $w(e, \theta)$ for all $e \in E$ and all outer align events that occur after θ, ordered by their associated orientations.
- The *search tree* $\mathcal{T}$ is a balanced binary search tree on the set $K := P \times \{\square, \diamond, \square, \diamond\}$ of all contact pairs indexed by any total order on K. Each node labeled by $(p, \square) \in K$ stores the set of all regular vertices $v \in V$ such that $(p, \square) \in \kappa_v$, denoted by $\mathcal{T}(p, \square)$, into a sorted list $L(p, \square)$ by the order along the boundary of the face of $\mathsf{VD}(\theta)$ for $(p, \square)$.

Note that the structures we maintain stay the same between any two consecutive degenerate orientations. Also, by Lemmas 3.4 and 5.3, the space used by the data structures is bounded by $O(n)$ at any time by the invariants.

Our algorithm runs in two phases: the initialization and the main loop. Without loss of generality, we assume that $0 \in \mathbb{O}$ is a regular orientation. In the initialization phase, we initialize the data structures for $\theta = 0$. We first compute $VG(0)$ by any optimal algorithm computing the L_∞ Voronoi diagram [29, 30]. Then, for any regular vertex $v \in V(0)$, we insert v into V and $\mathcal{T}$; for any bounded edge $e \in E(0)$, we insert e into E, we compute the potential edge event $w(e, 0)$, if defined, and insert it into $\mathcal{Q}$. Compute all outer align events by Lemma 5.3 and insert them into $\mathcal{Q}$.

We are then ready to run the main loop of our algorithm from the current orientation $\theta = 0$. In the main loop, we repeatedly recognize the next degenerate orientation $\phi > \theta$ by finding an event with a smallest orientation from $\mathcal{Q}$, collect all events that occur at ϕ by extracting them from $\mathcal{Q}$, and handle them by performing the following two steps: (1) computing all 4-squares in $\mathfrak{S}_4(\phi)$ and (2) updating our structures as θ proceeds over ϕ.

For the first step, let W be the set of all events in $\mathcal{Q}$ whose associated orientation is ϕ. The set W can be obtained by repeatedly performing operations on the event queue $\mathcal{Q}$; check if the associated orientation of the minimum element in $\mathcal{Q}$ is exactly ϕ and extract it, if so.

For each event $w \in W$, we find all squares relevant to w, according to the type of w. We initialize $\mathfrak{S}_4(w)$ to be an empty set as a variable, and will finally consist of all 4-squares relevant to w. Consequently, we have the following.

▶ **Lemma 5.6.** *The first step of the main loop can be done in $O(|\mathfrak{S}_4(\phi)| \log n)$ time and we have $\mathfrak{S}_4(\phi) = \bigcup_{w \in W} \mathfrak{S}_4(w)$ in the end.*

In the second step, we first specify the sets $V^-(\phi)$, $V^+(\phi)$, $E^-(\phi)$, and $E^+(\phi)$, and then update our data structures to keep the invariants accordingly. For each $S \in \mathfrak{S}_4(\phi)$, we compute V_S^- and V_S^+ by Lemma 3.7 and its proof as illustrated in Figure 6. By Lemma 5.1(i), we obtain $V^-(\phi)$ and $V^+(\phi)$. By Lemma 5.1(ii), we can compute the edge sets $E^-(\phi)$ and $E^+(\phi)$ by searching the neighbors of $V^-(\phi)$ in $VG = VG(\theta)$.

▶ **Lemma 5.7.** *The sets $E^-(\phi)$ and $E^+(\phi)$ can be found in $O(|\mathfrak{S}_4(\phi)| \log |\mathfrak{S}_4(\phi)|)$ time.*

We are ready to update our structures for $\phi + \epsilon$ for arbitrarily small $\epsilon > 0$. Note that we currently have $V = V(\theta) = V(\phi - \epsilon)$ and $E = E(\theta) = E(\phi - \epsilon)$. We update V and E as follows: delete all vertices in $V^-(\phi)$ from V and all edges in $E^-(\phi)$ from E, and then insert all vertices in $V^+(\phi)$ into V and all edges in $E^+(\phi)$ into E. Then, update $\mathcal{T}$ and $\mathcal{Q}$ as follows: We delete each $v \in V^-(\phi)$ from $\mathcal{T}$ and insert each $v \in V^+(\phi)$ into $\mathcal{T}$. For each $e \in E^-(\phi)$, we delete the potential edge event for e from $\mathcal{Q}$. For each $e \in E^+(\phi)$, we compute the potential edge event $w(e, \phi + \epsilon)$ by Lemma 5.5 and insert it into $\mathcal{Q}$, if defined. Lastly, set θ to be $\phi + \epsilon$.

We finally conclude the following.

▶ **Theorem 5.8.** *Given a set P of n points in general position, the total amount of combinatorial changes of the L_∞ Voronoi diagram of P while the axes rotate by $\pi/2$ is bounded by $\Theta(s_4)$, where s_4 denotes the number of 4-squares among P. The combinatorial structure of the Voronoi diagram can be maintained explicitly in total $O(s_4 \log n)$ time using $O(n)$ space.*

▶ **Corollary 5.9.** *Given a set P of n points in general position, we can compute all 4-squares among points in P in $O(s_4 \log n)$ time and $O(n)$ space.*

6 Maximal Empty Squares

It is now clear that our algorithm in the previous section collects a full description of all maximal empty squares in $O(n^2 \log n)$ time and its complexity is $O(n^2)$. Hence, it is not difficult to derive an algorithm that finds a largest empty square over all orientations.

▶ **Theorem 6.1.** *Given a set P of n points in the plane, a largest empty square among P in arbitrary orientation can be computed in worst-case $O(n^2 \log n)$ time.*

Some query versions of the problem can also be considered.

▶ **Theorem 6.2.** *Given a set P of n points, in $O(n^2 \log n)$ time, we can preprocess P into a data structure of size $O(n^2 \alpha(n))$ that answers the following query in $O(\log n)$ time: given an orientation $\beta \in \mathbb{O}$, find a largest empty square in orientation β.*

▶ **Theorem 6.3.** *Given a set P of n points, in $O(s_4 \log n)$ time, we can preprocess P into a data structure of size $O(s_4)$ that answers the following query in $O(\log n)$ time: given a point $c \in \mathbb{R}^2$ and $\beta \in \mathbb{O}$, find a largest empty square centered at c in orientation β.*

By a similar approach to that used in Bae [6], we reduce the problem of computing a square annulus of minimum width or area to that of finding all maximal empty squares.

▶ **Theorem 6.4.** *Given a set P of n points, a square annulus of minimum width or minimum area in arbitrary orientation that encloses P can be computed in $O(n^2 \log n)$ time.*

References

1 M. Abellanas, Ferran Hurtado, C. Icking, L. Ma, B. Palop, and P.A. Ramos. Best fitting rectangles. In *Proc. Euro. Workshop Comput. Geom. (EuroCG 2003)*, 2003.

2 A. Aggarwal, M.M. Klawe, S. Moran, P. Shor, and R. Wilber. Geometric applications of a matrix-searching algorithm. *Alogorithmica*, 2:195–208, 1987.

3 A. Aggarwal and S. Suri. Fast algorithm for computing the largest empty rectangle. In *Proc. 3rd ACM Sympos. Comput. Geom. (SoCG 1987)*, pages 278–290, 1987.

4 Carlos Alegría-Galicia, David Orden, Carlos Seara, and Jorge Urrutia. On the $O\beta$-hull of a planar point set. *Comput. Geom.: Theory Appl.*, 68:277–291, 2018.

5 Sang Won Bae. Computing a minimum-width square annulus in arbitrary orientation. *Theoret. Comput. Sci.*, 718:2–13, 2018.

6 Sang Won Bae. On the minimum-area rectangular and square annulus problem, 2019. submitted to CGTA. `arXiv:1904.06832`.

7 Sang Won Bae, Chunseok Lee, Hee-Kap Ahn, Sunghee Choi, and Kyung-Yong Chwa. Computing minimum-area rectilinear convex hull and L-shape. *Comput. Geom.: Theory Appl.*, 42(9):903–912, 2009.

8 Sang Won Bae and Sang Duk Yoon. Empty squares in arbitrary orientation among points, 2019. `arXiv:1911.12988`.

9 I. Bárány and Z. Füredi. Empty simplices in Euclidean space. *Canad. Math. Bull.*, 30:436–445, 1987.

10 I. Bárány and P. Valtr. A positive fraction Erdős–Szekeres theorem. *Discrete Comput. Geom.*, 19(3):335–342, 1998.

11 I. Bárány and P. Valtr. Planar point sets with a small number of empty convex polygons. *Studia Sci. Math. Hungar.*, 41:243–269, 2005.

12 J.-D. Boissonnat, M. Sharir, B. Tagansky, and M. Yvinec. Voronoi diagrams in higher dimensions under certain polyhedral distance functions. *Discrete Comput. Geom.*, 19(4):485–519, 1998.

13 J.E. Boyce, D.P. Dobkin, R.L. Drysdale, and L.J. Guibas. Finding extreman polygons. *SIAM J. Comput.*, 14:134–147, 1985.

14 Jeet Chaudhuri, Subhas C. Nandy, and Sandip Das. Largest empty rectangle among a point set. *J. Algo.*, 46:54–78, 2003.

15 B. Chazelle, R.L. Drysdale, and D.T. Lee. Computing the largest empty rectangle. *SIAM J. Comput.*, 15:300–315, 1986.

16 David P. Dobkin, Herbert Edelsbrunner, and Mark H. Overmars. Searching for empty convex polygons. *Algorithmica*, 5(1):561–571, 1990.

17 R.L. Drysdale and J.W. Jaromczyk. A note on lower bounds for the maximum area and maximum perimeter k-gon problems. *Inform. Proc. Lett.*, 32(6):301–303, 1989.

18 A. Dumistrescu. Planar sets with few empty convex polygons. *Studia Sci. Math. Hungar.*, 36:93–109, 2000.

19 David Eppstein. New algorithms for minimum area k-gons. In *Proc. 3rd Annu. ACM-SIAM Sympos. Discrete Algo. (SODA'92)*, pages 83–88, 1992.

20 David Eppstein, Mark Overmars, Günter Rote, and Gerhard Woeginger. Finding minimum area k-gons. *Discrete Comput. Geom.*, 7(1):45–58, 1992.

21 Pual Erdős. Some more problems on elementary geometry. *Austral. Math. Soc. Gaz.*, 5:52–54, 1978.

22 Pual Erdős and Georege Szekeres. A combinatorial problem in geometry. *Compositio Math.*, 2:463–470, 1935.

23 Pual Erdős and Georege Szekeres. On some extremum problems in elementary geometry. *Ann. Univ. Sci. Budapest Eötvös Sect. Math.*, 3–4:53–62, 1961.

24 Tobias Gerken. Empty convex hexagons in planar point sets. *Discrete Comput. Geom.*, 39(1):239–272, 2008.

25 Olga N. Gluchshenko, Horst W. Hamacher, and Arie Tamir. An optimal $O(n \log n)$ algorithm for finding an enclosing planar rectilinear annulus of minimum width. *Operations Research Lett.*, 37(3):168–170, 2009.

26 H. Harborth. Kovexe Fünfecke in ebenen Punktmengen. *Elem. Math.*, 33:116–118, 1978.

27 J. Hershberger. Finding the upper envelope of n line segments in $O(n \log n)$ time. *Inform. Proc. Lett.*, 33:169–174, 1989.

28 J.D. Horton. Sets with no empty convex 7-gons. *Canad. Math. Bull.*, 26:482–484, 1983.

29 D. T. Lee. Two-dimensional Voronoi diagrams in the L_p-metric. *J. ACM*, 27:604–618, 1980.

30 D.T. Lee and C.K. Wong. Voronol diagrams in $L_1(L_\infty)$ metrics with 2-dimensional storage applications. *SIAM J. Comput.*, 9:200–211, 1980.

31 M. Mckenna, J. O'Rourke, and S. Suri. Finding the largest rectangle in an orthogonal polygon. In *Proc. 23rd Annual Allerton Conf. Comm. Control Comput.*, 1985.

32 J.S.B. Mitchell, G. Rote, G. Sundaram, and G. Woeginger. Counting convex polygons in planar point sets. *Inform. Proc. Lett.*, 56(1):45–49, 1995.

33 W. Morris and V. Soltan. The Erdős–Szekeres problem on points in convex position—a survey. *Bull. Amer. Math. Soc.*, 33:437–458, 2000.

34 Walter Morris and Valeriu Soltan. The Erdős–Szekeres problem. In John Forbes Nash, Jr. and Michael Th. Rassias, editors, *Open Problems in Mathematics*, pages 351–375. Springer International Publishing, Cham, 2016.

35 A. Naamad, D.T. Lee, and W.L. Hsu. On the maximum empty rectangle problem. *Discrete Appl. Math.*, 8:267–277, 1984.

36 C. M. Nicolás. The empty hexagon theorem. *Discrete Comput. Geom.*, 38(2):389–397, 2007.

37 M. Orlowski. A new algorithm for largest empty rectangle problem. *Algorithmica*, 5:65–73, 1990.

38 Rom Pinchasi, Radoš Radoičić, and Micha Sharir. On empty convex polygons in a planar point set. *J. Combinat. Theory, Series A*, 113(3):385–419, 2006.

39 G. Rote, Z. Wang, G. Woeginger, and B. Zhi. Counting k-subsets and convex k-gons in the plane. *Inform. Proc. Lett.*, 38(4):149–151, 1991.

40 G. Rote and G. Woeginger. Counting convex k-gons in planar point sets. *Inform. Proc. Lett.*, 41(4):191–194, 1992.

41 George Szekeres and Lindsay Peters. Computer solution to the 17-point Erdős–Szekeres problem. *ANZIAM J.*, 48(2):151–164, 2006.

42 P. Valtr. On the minimum number of empty polygons in planar point sets. *Studia Sci. Math. Hungar.*, 30:155–163, 1995.

Holes and Islands in Random Point Sets

Martin Balko
Department of Applied Mathematics, Faculty of Mathematics and Physics,
Charles University, Prague, Czech Republic
balko@kam.mff.cuni.cz

Manfred Scheucher
Institut für Mathematik, Technische Universität Berlin, Germany
scheucher@math.tu-berlin.de

Pavel Valtr
Department of Applied Mathematics, Faculty of Mathematics and Physics,
Charles University, Prague, Czech Republic
Department of Computer Science, ETH Zürich, Switzerland

Abstract

For $d \in \mathbb{N}$, let S be a finite set of points in $\mathbb{R}^d$ in general position. A set H of k points from S is a *k-hole* in S if all points from H lie on the boundary of the convex hull $\mathrm{conv}(H)$ of H and the interior of $\mathrm{conv}(H)$ does not contain any point from S. A set I of k points from S is a *k-island* in S if $\mathrm{conv}(I) \cap S = I$. Note that each k-hole in S is a k-island in S.

For fixed positive integers d, k and a convex body K in $\mathbb{R}^d$ with d-dimensional Lebesgue measure 1, let S be a set of n points chosen uniformly and independently at random from K. We show that the expected number of k-islands in S is in $O(n^d)$. In the case $k = d+1$, we prove that the expected number of empty simplices (that is, $(d+1)$-holes) in S is at most $2^{d-1} \cdot d! \cdot \binom{n}{d}$. Our results improve and generalize previous bounds by Bárány and Füredi [4], Valtr [19], Fabila-Monroy and Huemer [8], and Fabila-Monroy, Huemer, and Mitsche [9].

2012 ACM Subject Classification Mathematics of computing → Combinatoric problems; Theory of computation → Computational geometry

Keywords and phrases stochastic geometry, random point set, Erdős-Szekeres type problem, k-hole, k-island, empty polytope, convex position, Horton set

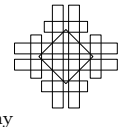
Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.14

Related Version A full version of this paper is available at `https://arxiv.org/abs/2003.00909`.

Funding *Martin Balko*: was supported by the grant no. 18-19158S of the Czech Science Foundation (GAČR), by the Center for Foundations of Modern Computer Science (Charles University project UNCE/SCI/004), and by the PRIMUS/17/SCI/3 project of Charles University. This article is part of a project that has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 810115).
Manfred Scheucher: was supported by DFG Grant FE 340/12-1.
Pavel Valtr: was supported by the grant no. 18-19158S of the Czech Science Foundation (GAČR) and by the PRIMUS/17/SCI/3 project of Charles University.

1 Introduction

For $d \in \mathbb{N}$, let S be a finite set of points in $\mathbb{R}^d$. The set S is in *general position* if, for every $k = 1, \ldots, d-1$, no $k+2$ points of S lie in an affine k-dimensional subspace. A set H of k points from S is a *k-hole* in S if H is in convex position and the interior of the convex hull $\mathrm{conv}(H)$ of H does not contain any point from S; see Figure 1 for an illustration in the plane. We say that a subset of S is a *hole* in S if it is a k-hole in S for some integer k.



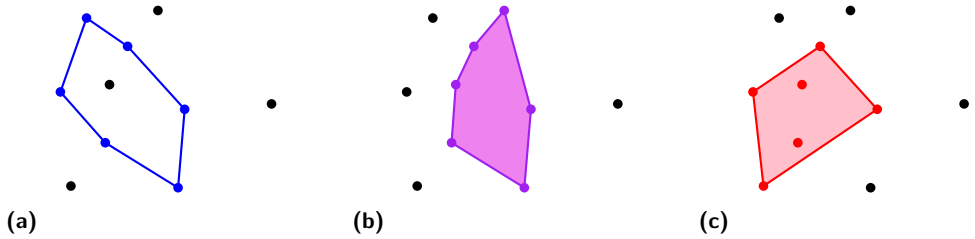


Figure 1 (a) A 6-tuple of points in convex position in a planar set S of 10 points. (b) A 6-hole in S. (c) A 6-island in S whose points are not in convex position.

Let $h(k)$ be the smallest positive integer N such that every set of N points in general position in the plane contains a k-hole. In the 1970s, Erdős [6] asked whether the number $h(k)$ exists for every $k \in \mathbb{N}$. It was shown in the 1970s and 1980s that $h(4) = 5$, $h(5) = 10$ [11], and that $h(k)$ does not exist for every $k \geq 7$ [12]. That is, while every sufficiently large set contains a 4-hole and a 5-hole, Horton constructed arbitrarily large sets with no 7-holes. His construction was generalized to so-called *Horton sets* by Valtr [18]. The existence of 6-holes in every sufficiently large point set remained open until 2007, when Gerken [10] and Nicolas [15] independently showed that $h(6)$ exists; see also [20].

These problems were also considered in higher dimensions. For $d \geq 2$, let $h_d(k)$ be the smallest positive integer N such that every set of N points in general position in $\mathbb{R}^d$ contains a k-hole. In particular, $h_2(k) = h(k)$ for every k. Valtr [18] showed that $h_d(k)$ exists for $k \leq 2d + 1$ but it does not exist for $k > 2^{d-1}(P(d-1) + 1)$, where $P(d-1)$ denotes the product of the first $d-1$ prime numbers. The latter result was obtained by constructing multidimensional analogues of the Horton sets.

After the existence of k-holes was settled, counting the minimum number $H_k(n)$ of k-holes in any set of n points in the plane in general position attracted a lot of attention. It is known, and not difficult to show, that $H_3(n)$ and $H_4(n)$ are in $\Omega(n^2)$. The currently best known lower bounds on $H_3(n)$ and $H_4(n)$ were proved in [1]. The best known upper bounds are due to Bárány and Valtr [5]. Altogether, these estimates are

$$n^2 + \Omega(n \log^{2/3} n) \leq H_3(n) \leq 1.6196n^2 + o(n^2)$$

and

$$\tfrac{n^2}{2} + \Omega(n \log^{3/4} n) \leq H_4(n) \leq 1.9397n^2 + o(n^2).$$

For $H_5(n)$ and $H_6(n)$, the best quadratic upper bounds can be found in [5]. The best lower bounds, however, are only $H_5(n) \geq \Omega(n \log^{4/5} n)$ [1] and $H_6(n) \geq \Omega(n)$ [21]. For more details, we also refer to the second author's dissertation [17].

The quadratic upper bound on $H_3(n)$ can be also obtained using random point sets. For $d \in \mathbb{N}$, a *convex body* in $\mathbb{R}^d$ is a compact convex set in $\mathbb{R}^d$ with a nonempty interior. Let k be a positive integer and let $K \subseteq \mathbb{R}^d$ be a convex body with d-dimensional Lebesgue measure $\lambda_d(K) = 1$. We use $EH^K_{d,k}(n)$ to denote the expected number of k-holes in sets of n points chosen independently and uniformly at random from K. The quadratic upper bound on $H_3(n)$ then also follows from the following bound of Bárány and Füredi [4] on the expected number of $(d+1)$-holes:

$$EH^K_{d,d+1}(n) \leq (2d)^{2d^2} \cdot \binom{n}{d} \tag{1}$$

for any d and K. In the plane, Bárány and Füredi [4] proved $EH^K_{2,3}(n) \leq 2n^2 + O(n \log n)$ for every K. This bound was later slightly improved by Valtr [19], who showed $EH^K_{2,3}(n) \leq 4\binom{n}{2}$ for any K. In the other direction, every set of n points in $\mathbb{R}^d$ in general position contains at least $\binom{n-1}{d}$ $(d+1)$-holes [4, 13].

The expected number $EH^K_{2,4}(n)$ of 4-holes in random sets of n points in the plane was considered by Fabila-Monroy, Huemer, and Mitsche [9], who showed

$$EH^K_{2,4}(n) \leq 18\pi D^2 n^2 + o(n^2) \tag{2}$$

for any K, where $D = D(K)$ is the diameter of K. Since we have $D \geq 2/\sqrt{\pi}$, by the Isodiametric inequality [7], the leading constant in (2) is at least 72 for any K.

In this paper, we study the number of k-holes in random point sets in $\mathbb{R}^d$. In particular, we obtain results that imply quadratic upper bounds on $H_k(n)$ for any fixed k and that both strengthen and generalize the bounds by Bárány and Füredi [4], Valtr [19], and Fabila-Monroy, Huemer, and Mitsche [9].

2 Our results

Throughout the whole paper we only consider point sets in $\mathbb{R}^d$ that are finite and in general position.

2.1 Islands and holes in random point sets

First, we prove a result that gives the estimate $O(n^d)$ on the minimum number of k-holes in a set of n points in $\mathbb{R}^d$ for any fixed d and k. In fact, we prove the upper bound $O(n^d)$ even for so-called k-islands, which are also frequently studied in discrete geometry. A set I of k points from a point set $S \subseteq \mathbb{R}^d$ is a *k-island* in S if $\mathrm{conv}(I) \cap S = I$; see part (c) of Figure 1. Note that k-holes in S are exactly those k-islands in S that are in convex position. A subset of S is an *island* in S if it is a k-island in S for some integer k.

▶ **Theorem 1.** *Let $d \geq 2$ and $k \geq d+1$ be integers and let K be a convex body in $\mathbb{R}^d$ with $\lambda_d(K) = 1$. If S is a set of $n \geq k$ points chosen uniformly and independently at random from K, then the expected number of k-islands in S is at most*

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

which is in $O(n^d)$ for any fixed d and k.

The bound in Theorem 1 is tight up to a constant multiplicative factor that depends on d and k, as, for any fixed $k \geq d$, every set S of n points in $\mathbb{R}^d$ in general position contains at least $\Omega(n^d)$ k-islands. To see this, observe that any d-tuple T of points from S determines a k-island with $k-d$ closest points to the hyperplane spanned by T (ties can be broken by, for example, taking points with lexicographically smallest coordinates), as S is in general position and thus T is a d-hole in S. Any such k-tuple of points from S contains $\binom{k}{d}$ d-tuples of points from S and thus we have at least $\binom{n}{d}/\binom{k}{d} \in \Omega(n^d)$ k-islands in S.

Thus, by Theorem 1, random point sets in $\mathbb{R}^d$ asymptotically achieve the minimum number of k-islands. This is in contrast with the fact that, unlike Horton sets, they contain arbitrarily large holes. Quite recently, Balogh, González-Aguilar, and Salazar [3] showed that the expected number of vertices of the largest hole in a set of n random points chosen independently and uniformly over a convex body in the plane is in $\Theta(\log n/(\log \log n))$.

For k-holes, we modify the proof of Theorem 1 to obtain a slightly better estimate.

▶ **Theorem 2.** *Let $d \geq 2$ and $k \geq d+1$ be integers and let K be a convex body in $\mathbb{R}^d$ with $\lambda_d(K) = 1$. If S is a set of $n \geq k$ points chosen uniformly and independently at random from K, then the expected number $EH^K_{d,k}(n)$ of k-holes in S is in $O(n^d)$ for any fixed d and k. More precisely,*

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

For $d = 2$ and $k = 4$, Theorem 2 implies $EH^K_{2,4}(n) \leq 128 \cdot n^2 + o(n^2)$ for any K, which is a worse estimate than (2) if the diameter of K is at most $8/(3\sqrt{\pi}) \simeq 1.5$. However, the proof of Theorem 2 can be modified to give $EH^K_{2,4}(n) \leq 12n^2 + o(n^2)$ for any K, which is always better than (2); see the final remarks in Section 3. We believe that the leading constant in $EH^K_{2,4}(n)$ can be estimated even more precisely and we hope to discuss this direction in future work.

In the case $k = d+1$, the bound in Theorem 2 simplifies to the following estimate on the expected number of $(d+1)$-holes (also called *empty simplices*) in random sets of n points in $\mathbb{R}^d$.

▶ **Corollary 3.** *Let $d \geq 2$ be an integer and let K be a convex body in $\mathbb{R}^d$ with $\lambda_d(K) = 1$. If S is a set of n points chosen uniformly and independently at random from K, then the expected number of $(d+1)$-holes in S satisfies*

$$EH^K_{d,d+1}(n) \leq 2^{d-1} \cdot d! \cdot \binom{n}{d}.$$

Corollary 3 is stronger than the bound (1) by Bárány and Füredi [4] and, in the planar case, coincides with the bound $EH^K_{2,3}(n) \leq 4\binom{n}{2}$ by Valtr [19]. Very recently, Reitzner and Temesvari [16] proved an upper bound on $EH^K_{d,d+1}(n)$ that is asymptotically tight if $d = 2$ or if $d \geq 3$ and K is an ellipsoid. In the planar case, their result shows that the bound $4\binom{n}{2}$ on $EH^K_{2,3}(n)$ is best possible, up to a smaller order error term. No tight bounds on $EH^K_{d,d+1}(n)$ are known if $d \geq 3$ and K is not an ellipsoid.

We also consider islands of all possible sizes and show that their expected number is in $2^{\Theta\left(n^{(d-1)/(d+1)}\right)}$.

▶ **Theorem 4.** *Let $d \geq 2$ be an integer and let K be a convex body in $\mathbb{R}^d$ with $\lambda_d(K) = 1$. Then there are constants $C_1 = C_1(d)$, $C_2 = C_2(d)$, and $n_0 = n_0(d)$ such that for every set S of $n \geq n_0$ points chosen uniformly and independently at random from K the expected number E^K_d of islands in S satisfies*

$$2^{C_1 \cdot n^{(d-1)/(d+1)}} \leq E^K_d \leq 2^{C_2 \cdot n^{(d-1)/(d+1)}}.$$

Since each island in S has at most n points, there is a $k \in \{1, \ldots, n\}$ such that the expected number of k-islands in S is at least $(1/n)$-fraction of the expected number of all islands, which is still in $2^{\Omega\left(n^{(d-1)/(d+1)}\right)}$. This shows that the expected number of k-islands can become asymptotically much larger than $O(n^d)$ if k is not fixed. Due to space limitations, the proof of Theorem 4 is omitted.

2.2 Islands and holes in d-Horton sets

To our knowledge, Theorem 1 is the first nontrivial upper bound on the minimum number of k-islands a point set in $\mathbb{R}^d$ with $d > 2$ can have. For $d = 2$, Fabila-Monroy and Huemer [8] showed that, for every fixed $k \in \mathbb{N}$, the Horton sets with n points contain only $O(n^2)$

k-islands. For $d > 2$, Valtr [18] introduced a d-dimensional analogue of Horton sets. Perhaps surprisingly, these sets contain asymptotically more than $O(n^d)$ k-islands for $k \geq d+1$. For each k with $d+1 \leq k \leq 3 \cdot 2^{d-1}$, they even contain asymptotically more than $O(n^d)$ k-holes.

▶ **Theorem 5.** *Let $d \geq 2$ and k be fixed positive integers. Then every d-dimensional Horton set H with n points contains at least $\Omega(n^{\min\{2^{d-1},k\}})$ k-islands in H. If $k \leq 3 \cdot 2^{d-1}$, then H even contains at least $\Omega(n^{\min\{2^{d-1},k\}})$ k-holes in H.*

3 Proofs of Theorem 1 and Theorem 2

Let d and k be positive integers and let K be a convex body in $\mathbb{R}^d$ with $\lambda_d(K) = 1$. Let S be a set of n points chosen uniformly and independently at random from K. Note that S is in general position with probability 1. We assume $k \geq d+1$, as otherwise the number of k-islands in S is trivially $\binom{n}{k}$ in every set of n points in $\mathbb{R}^d$ in general position. We also assume $d \geq 2$ and $n \geq k$, as otherwise the number of k-islands is trivially $n-k+1$ and 0, respectively, in every set of n points in $\mathbb{R}^d$.

First, we prove Theorem 1 by showing that the expected number of k-islands in S is at most

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

which is in $O(n^d)$ for any fixed d and k. At the end of this section, we improve the bound for k-holes, which will prove Theorem 2.

Let Q be a set of k points from S. We first introduce a suitable unique ordering $q_1, \ldots, q_k$ of points from Q. First, we take a set D of $d+1$ points from Q that determine a simplex Δ with largest volume among all $(d+1)$-tuples of points from Q. Let q_1q_2 be the longest edge of Δ with q_1 lexicographically smaller than q_2 and let a be the number of points from Q inside Δ. For every $i = 2, \ldots, d$, let q_{i+1} be the furthest point from $D \setminus \{q_1, \ldots, q_i\}$ to $\mathrm{aff}(q_1, \ldots, q_i)$. Next, we let $q_{d+2}, \ldots, q_{d+a+1}$ be the a points of Q inside Δ ordered lexicographically. The remaining $k-d-a-1$ points $q_{d+a+2}, \ldots, q_k$ from Q lie outside of Δ and we order them so that, for every $i = 1, \ldots, k-a-d-1$, the point $q_{d+a+i+1}$ is closest to $\mathrm{conv}(\{q_1, \ldots, q_{d+a+i}\})$ among the points $q_{d+a+i+1}, \ldots, q_k$. In case of a tie in any of the conditions, we choose the point with lexicographically smallest coordinates. Note, however, that a tie occurs with probability 0.

Clearly, there is a unique such ordering $q_1, \ldots, q_k$ of Q. We call this ordering the *canonical (k,a)-ordering* of Q. To reformulate, an ordering $q_1, \ldots, q_k$ of Q is the canonical (k,a)-ordering of Q if and only if the following five conditions are satisfied:

- **(L1)** The d-dimensional simplex Δ, with vertices $q_1, \ldots, q_{d+1}$ has the largest d-dimensional Lebesgue measure among all d-dimensional simplices spanned by points from Q.
- **(L2)** For every $i = 1, \ldots, d-1$, the point q_{i+1} has the largest distance among all points from $\{q_{i+1}, \ldots, q_d\}$ to the $(i-1)$-dimensional affine subspace $\mathrm{aff}(q_1, \ldots, q_i)$ spanned by $q_1, \ldots, q_i$. Moreover, q_1 is lexicographically smaller than q_2.
- **(L3)** For every $i = 1, \ldots, d-1$, the distance between q_{i+1} and $\mathrm{aff}(q_1, \ldots, q_i)$ is at least as large as the distance between q_{d+1} and $\mathrm{aff}(q_1, \ldots, q_i)$. Also, the distance between q_1 and q_2 is at least as large as the distance between q_{d+1} and any q_i with $i \in \{1, \ldots, d\}$.
- **(L4)** The points $q_{d+2}, \ldots, q_{d+a+1}$ lie inside Δ and are ordered lexicographically.
- **(L5)** The points $q_{d+a+2}, \ldots, q_k$ lie outside of Δ. For every $i = 1, \ldots, k-a-d-1$, the point $q_{d+a+i+1}$ is closest to $\mathrm{conv}(\{q_1, \ldots, q_{d+a+i}\})$ among the points $q_{d+a+i+1}, \ldots, q_k$.

Figure 2 gives an illustration in $\mathbb{R}^2$. We note that the conditions (L2) and (L3) can be merged together. However, later in the proof, we use the fact that the probability that the points from Q satisfy the condition (L2) equals $1/d!$, so we stated the two conditions separately.

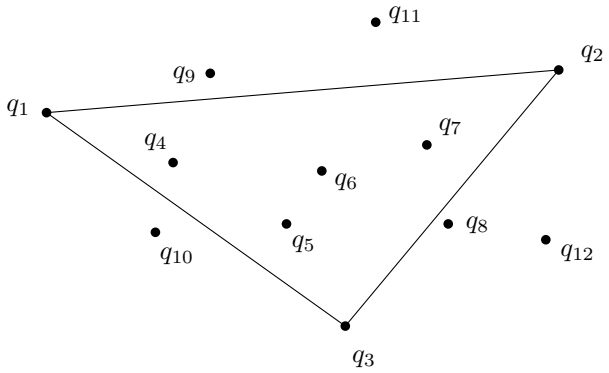


Figure 2 An illustration of the canonical (k, a)-ordering of a planar point set Q. Here we have $k = 12$ points and $a = 4$ of the points lie inside the largest area triangle $\triangle$ with vertices q_1, q_2, q_3.

Before going into details, we first give a high-level overview of the proof of Theorem 1. First, we prove an $O(1/n^{a+1})$ bound on the probability that $\triangle$ contains precisely the points $p_{d+2}, \ldots, p_{d+1+a}$ from S (Lemma 9), which means that the points $p_1, \ldots, p_{d+1+a}$ determine an island in S. Next, for $i = d + 2 + a, \ldots, k$, we show that, conditioned on the fact that the $(i-1)$-tuple $(p_1, \ldots, p_{i-1})$ determines an island in S in the canonical (k, a)-ordering, the i-tuple $(p_1, \ldots, p_i)$ determines an island in S in the canonical (k, a)-ordering with probability $O(1/n)$ (Lemma 10). Then it immediately follows that the probability that I determines a k-island in S with the desired properties is at most $O\left(1/n^{a+1} \cdot (1/n)^{k-(d+1+a)}\right) = O(n^{d-k})$. Since there are $n \cdot (n-1) \cdots (n-k+1) = O(n^k)$ possibilities to select such an ordered subset I and each k-island in S is counted at most $k!$ times, we obtain the desired bound $O\left(n^k \cdot n^{d-k} \cdot k!\right) = O(n^d)$ on the expected number of k-islands in S.

We now proceed with the proof. Let $p_1, \ldots, p_k$ be points from S in the order in which they are drawn from K. We use Δ to denote the d-dimensional simplex with vertices $p_1, \ldots, p_{d+1}$. We eventually show that the probability that $p_1, \ldots, p_k$ is the canonical (k, a)-ordering of a k-island in S for some a is at most $O(1/n^{k-d})$. First, however, we need to state some notation and prove some auxiliary results.

Consider the points $p_1, \ldots, p_d$. Without loss of generality, we can assume that, for each $i = 1, \ldots, d$, the point p_i has the last $d - i + 1$ coordinates equal to zero. Otherwise we apply a suitable isometry to S. Then, for every $i = 1, \ldots, d$, the distance between p_{i+1} and the $(i-1)$-dimensional affine subspace spanned by $p_1, \ldots, p_i$ is equal to the absolute value of the ith coordinate of p_{i+1}. Moreover, after applying a suitable rotation, we can also assume that the first coordinate of each of the points $p_1, \ldots, p_d$ is nonnegative.

Let Δ_0 be the $(d-1)$-dimensional simplex with vertices $p_1, \ldots, p_d$ and let H be the hyperplane containing Δ_0. Note that, according to our assumptions about $p_1, \ldots, p_d$, we have $H = \{(x_1, \ldots, x_d) \in \mathbb{R}^d \colon x_d = 0\}$. Let B be the set of points $(x_1, \ldots, x_d) \in \mathbb{R}^d$ that satisfy the following three conditions:

- **(i)** $x_1 \geq 0$,
- **(ii)** $|x_i|$ is at most as large as the absolute value of the ith coordinate of p_{i+1} for every $i \in \{1, \ldots, d-1\}$, and
- **(iii)** $|x_d| \leq d/\lambda_{d-1}(\Delta_0)$.

See Figures 3a and 3b for illustrations in $\mathbb{R}^2$ and $\mathbb{R}^3$, respectively. Observe that B is a d-dimensional axis-parallel box. For $h \in \mathbb{R}$, we use I_h to denote the intersection of B with the hyperplane $x_d = h$.

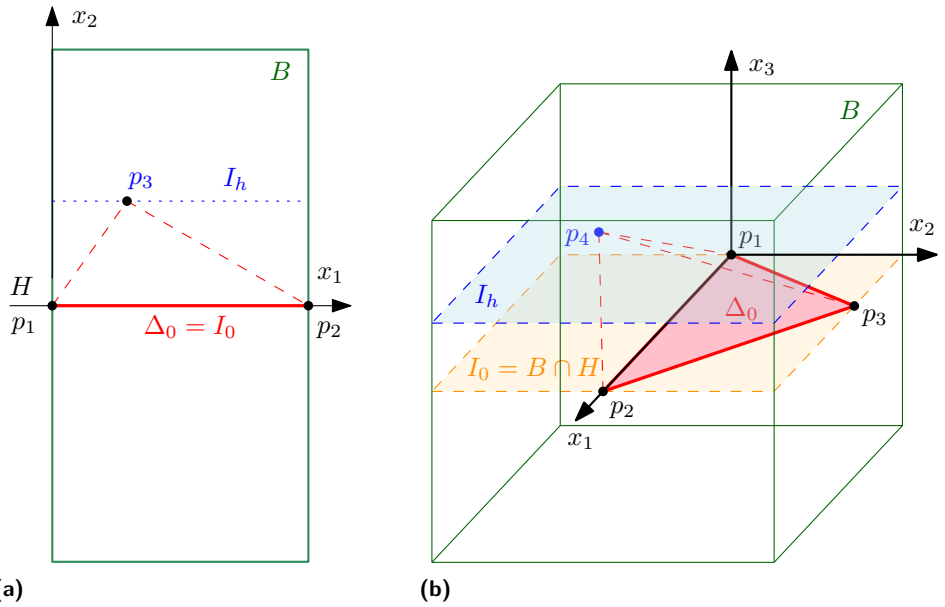


Figure 3 An illustration of the proof of Theorem 1 in (a) $\mathbb{R}^2$ and (b) $\mathbb{R}^3$.

Having fixed $p_1, \dots, p_d$, we now try to restrict possible locations of the points $p_{d+1}, \dots, p_k$, one by one, so that $p_1, \dots, p_k$ is the canonical (k, a)-ordering of a k-island in S for some a. First, we observe that the position of the point p_{d+1} is restricted to B.

▶ **Lemma 6.** *If $p_1, \dots, p_{d+1}$ satisfy condition (L3), then p_{d+1} lies in the box B.*

Proof. Let $p_{d+1} = (x_1, \dots, x_d)$. According to our choice of points $p_1, \dots, p_d$ and from the assumption that $p_1, \dots, p_d$ satisfy (L3), we get $x_1 \geq 0$ and also that $|x_i|$ is at most as large as the absolute value of the ith coordinate of p_{i+1} for every $i \in \{1, \dots, d-1\}$.

It remains to show that $|x_d| \leq d/\lambda_{d-1}(\Delta_0)$. The simplex Δ spanned by $p_1, \dots, p_{d+1}$ is contained in the convex body K, as $p_1, \dots, p_{d+1} \in K$ and K is convex. Thus $\lambda_d(\Delta) \leq \lambda_d(K) = 1$. On the other hand, the volume $\lambda_d(\Delta)$ equals $\lambda_{d-1}(\Delta_0) \cdot h/d$, where h is the distance between p_{d+1} and the hyperplane H containing Δ_0. According to our assumptions about $p_1, \dots, p_d$, the distance h equals $|x_d|$. Since $\lambda_d(\Delta) \leq 1$, it follows that $|x_d| = h \leq d/\lambda_{d-1}(\Delta_0)$ and thus $p_{d+1} \in B$. ◀

The following auxiliary lemma gives an identity that is needed later. We omit the proof, which can be found, for example, in [2, Section 1].

▶ **Lemma 7** ([2])**.** *For all nonnegative integers a and b, we have*

$$\int_0^1 x^a (1-x)^b \, \mathrm{d}x = \frac{a!\, b!}{(a+b+1)!}.$$

We will also use the following result, called the *Asymptotic Upper Bound Theorem* [14], that estimates the maximum number of facets in a polytope.

▶ **Theorem 8** (Asymptotic Upper Bound Theorem [14]). *For every integer $d \geq 2$, a d-dimensional convex polytope with N vertices has at most $2\binom{N}{\lfloor d/2 \rfloor}$ facets.*

Let a be an integer satisfying $0 \leq a \leq k-d-1$ and let E_a be the event that $p_1, \ldots, p_k$ is the canonical (k,a)-ordering such that $\{p_1, \ldots, p_{d+a+1}\}$ is an island in S. To estimate the probability that $p_1, \ldots, p_k$ is the canonical (k,a)-ordering of a k-island in S, we first find an upper bound on the conditional probability of E_a, conditioned on the event L_2 that $p_1, \ldots, p_d$ satisfy (L2).

▶ **Lemma 9.** *For every $a \in \{0, \ldots, k-d-1\}$, the probability $\Pr[E_a \mid L_2]$ is at most*

$$\frac{2^{d-1} \cdot d!}{(k-a-d-1)! \cdot (n-k+1)^{a+1}}.$$

Proof. It follows from Lemma 6 that, in order to satisfy (L3), the point p_{d+1} must lie in the box B. In particular, p_{d+1} is contained in $I_h \cap K$ for some real number $h \in [-d/\lambda_{d-1}(\Delta_0), d/\lambda_{d-1}(\Delta_0)]$. If $p_{d+1} \in I_h$, then the simplex $\Delta = \operatorname{conv}(\{p_1, \ldots, p_{d+1}\})$ has volume $\lambda_d(\Delta) = \lambda_{d-1}(\Delta_0) \cdot |h|/d$ and the a points $p_{d+2}, \ldots, p_{d+a+1}$ satisfy (L4) with probability

$$\frac{1}{a!} \cdot (\lambda_d(\Delta))^a = \frac{1}{a!} \cdot \left(\frac{\lambda_{d-1}(\Delta_0) \cdot |h|}{d} \right)^a,$$

as they all lie in $\Delta \subseteq K$ in the unique order.

In order to satisfy the condition (L5), the $k-a-d-1$ points $p_{d+a+i+1}$, for $i \in \{1, \ldots, k-a-d-1\}$, must have increasing distance to $\operatorname{conv}(\{p_1, \ldots, p_{d+a+i}\})$ as the index i increases, which happens with probability at most $\frac{1}{(k-a-d-1)!}$. Since $\{p_1, \ldots, p_{d+a+1}\}$ must be an island in S, the $n-d-a-1$ points from $S \setminus \{p_1, \ldots, p_{d+a+1}\}$ must lie outside Δ. If $p_{d+1} \in I_h$, then this happens with probability

$$(\lambda_d(K \setminus \Delta))^{n-d-a-1} = (\lambda_d(K) - \lambda_d(\Delta))^{n-d-a-1} = \left(1 - \frac{\lambda_{d-1}(\Delta_0) \cdot |h|}{d}\right)^{n-d-a-1},$$

as they all lie in $K \setminus \Delta$ and we have $\Delta \subseteq K$ and $\lambda_d(K) = 1$.

Altogether, we get that $\Pr[E_a \mid L_2]$ is at most

$$\int_{-d/\lambda_{d-1}(\Delta_0)}^{d/\lambda_{d-1}(\Delta_0)} \frac{\lambda_{d-1}(I_h \cap K)}{a! \cdot (k-a-d-1)!} \cdot \left(\frac{\lambda_{d-1}(\Delta_0) \cdot |h|}{d} \right)^a \cdot \left(1 - \frac{\lambda_{d-1}(\Delta_0) \cdot |h|}{d}\right)^{n-d-a-1} \mathrm{d}h.$$

Since we have $\lambda_{d-1}(I_0) = \lambda_{d-1}(I_h)$ for every $h \in [-d/\lambda_{d-1}(\Delta_0), d/\lambda_{d-1}(\Delta_0)]$, we obtain $\lambda_{d-1}(I_h \cap K) \leq \lambda_{d-1}(I_0)$ and thus $\Pr[E_a \mid L_2]$ is at most

$$\frac{2 \cdot \lambda_{d-1}(I_0)}{a! \cdot (k-a-d-1)!} \cdot \int_0^{d/\lambda_{d-1}(\Delta_0)} \left(\frac{\lambda_{d-1}(\Delta_0) \cdot h}{d} \right)^a \cdot \left(1 - \frac{\lambda_{d-1}(\Delta_0) \cdot h}{d}\right)^{n-d-a-1} \mathrm{d}h.$$

By substituting $t = \frac{\lambda_{d-1}(\Delta_0) \cdot h}{d}$, we obtain

$$\Pr[E_a \mid L_2] \leq \frac{2d \cdot \lambda_{d-1}(I_0)}{a! \cdot (k-a-d-1)! \cdot \lambda_{d-1}(\Delta_0)} \cdot \int_0^1 t^a (1-t)^{n-d-a-1} \mathrm{d}t.$$

By Lemma 7, the right side in the above inequality equals

$$\frac{2d \cdot \lambda_{d-1}(I_0)}{a! \cdot (k-a-d-1)! \cdot \lambda_{d-1}(\Delta_0)} \cdot \frac{a! \cdot (n-d-a-1)!}{(n-d)!} = \frac{2d \cdot \lambda_{d-1}(I_0)}{(k-a-d-1)! \cdot \lambda_{d-1}(\Delta_0)} \cdot \frac{(n-d-a-1)!}{(n-d)!}.$$

For every $i = 1, \dots, d-1$, let h_i be the distance between the point p_{i+1} and the $(i-1)$-dimensional affine subspace spanned by $p_1, \dots, p_i$. Since the volume of the box I_0 satisfies

$$\lambda_{d-1}(I_0) = h_1(2h_2)\cdots(2h_{d-1}) = 2^{d-2} \cdot h_1 \cdots h_{d-1}$$

and the volume of the $(d-1)$-dimensional simplex Δ_0 is

$$\lambda_{d-1}(\Delta_0) = \frac{h_1}{1} \cdot \frac{h_2}{2} \cdot \cdots \cdot \frac{h_{d-1}}{d-1} = \frac{h_1 \cdots h_{d-1}}{(d-1)!},$$

we obtain $\lambda_{d-1}(I_0)/\lambda_{d-1}(\Delta_0) = 2^{d-2} \cdot (d-1)!$. Thus

$$\begin{aligned}\Pr[E_a \mid L_2] &\leq \frac{2^{d-1} \cdot d!}{(k-a-d-1)!} \cdot \frac{(n-d-a-1)!}{(n-d)!} \\ &= \frac{2^{d-1} \cdot d!}{(k-a-d-1)! \cdot (n-d)\cdots(n-d-a)} \\ &\leq \frac{2^{d-1} \cdot d!}{(k-a-d-1)! \cdot (n-k+1)^{a+1}},\end{aligned}$$

where the last inequality follows from $a \leq k-d-1$. ◀

For every $i \in \{d+a+1, \dots, k\}$, let $E_{a,i}$ be the event that $p_1, \dots, p_k$ is the canonical (k,a)-ordering such that $\{p_1, \dots, p_i\}$ is an island in S. Note that in the event $E_{a,i}$ the condition (L5) implies that $\{p_1, \dots, p_j\}$ is an island in S for every $j \in \{d+a+1, \dots, i\}$. Thus we have

$$L_2 \supseteq E_a = E_{a,d+a+1} \supseteq E_{a,d+a+2} \supseteq \cdots \supseteq E_{a,k}.$$

Moreover, the event $E_{a,k}$ says that $p_1, \dots, p_k$ is the canonical (k,a)-ordering of a k-island in S. For $i \in \{d+a+2, \dots, k\}$, we now estimate the conditional probability of $E_{a,i}$, conditioned on $E_{a,i-1}$.

▶ **Lemma 10.** *For every $i \in \{d+a+2, \dots, k\}$, we have*

$$\Pr[E_{a,i} \mid E_{a,i-1}] \leq \frac{2d^{2d-1} \cdot \binom{k}{\lfloor d/2 \rfloor}}{n-i+1}.$$

Proof. Let $i \in \{d+a+2, \dots, k\}$ and assume that the event $E_{a,i-1}$ holds. That is, $p_1, \dots, p_k$ is the canonical (k,a)-ordering such that $\{p_1, \dots, p_{i-1}\}$ is an $(i-1)$-island in S.

First, we assume that Δ is a regular simplex with height $\eta > 0$. At the end of the proof we show that the case when Δ is an arbitrary simplex follows by applying a suitable affine transformation.

For every $j \in \{1, \dots, d+1\}$, let F_j be the facet $\mathrm{conv}(\{p_1, \dots, p_{d+1}\} \setminus \{p_j\})$ of Δ and let H_j be the hyperplane parallel to F_j that contains p_j. We use H_j^+ to denote the halfspace determined by H_j such that $\Delta \subseteq H_j^+$. We set $\Delta^* = \cap_{j=1}^{d+1} H_j^+$; see Figures 4a and 4b for illustrations in $\mathbb{R}^2$ and $\mathbb{R}^3$, respectively. Note that Δ^* is a d-dimensional simplex containing Δ. Also, notice that if $x \notin \Delta^*$, then $x \notin H_j^+$ for some j and the distance between x and the hyperplane containing F_j is larger than η.

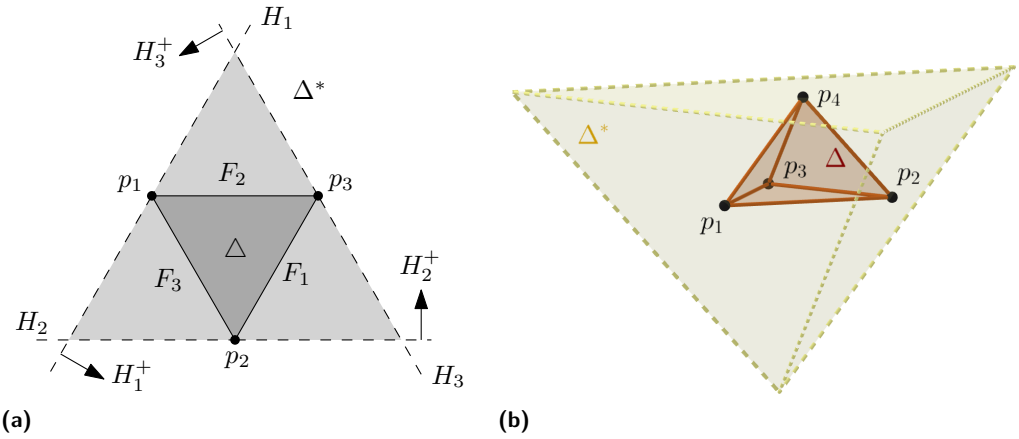


Figure 4 An illustration of (a) the simplex Δ^* in $\mathbb{R}^2$ and (b) in $\mathbb{R}^3$.

We show that the fact that $p_1, \ldots, p_k$ is the canonical (k, a)-ordering implies that every point from $\{p_1, \ldots, p_k\}$ is contained in Δ^*. Suppose for contradiction that some point $p \in \{p_1, \ldots, p_k\}$ does not lie inside Δ^*. Then there is a facet F_j of Δ such that the distance η' between p and the hyperplane containing F_j is larger than η. Then, however, the simplex Δ' spanned by vertices of F_j and by p has volume larger than Δ, because

$$\lambda_d(\Delta') = \frac{1}{d} \cdot \lambda_{d-1}(F_j) \cdot \eta' > \frac{1}{d} \cdot \lambda_{d-1}(F_j) \cdot \eta = \lambda_d(\Delta).$$

This contradicts the fact that $p_1, \ldots, p_k$ is the canonical (k, a)-ordering, as, according to (L1), Δ has the largest d-dimensional Lebesgue measure among all d-dimensional simplices spanned by points from $\{p_1, \ldots, p_k\}$.

Let σ be the barycenter of Δ. For every point $p \in \Delta^* \setminus \Delta$, the line segment σp intersects at least one facet of Δ. For every $j \in \{1, \ldots, d+1\}$, we use R_j to denote the set of points $p \in \Delta^* \setminus \Delta$ for which the line segment σp intersects the facet F_j of Δ. Observe that each set R_j is convex and the sets $R_1, \ldots, R_{d+1}$ partition $\Delta^* \setminus \Delta$ (up to their intersection of d-dimensional Lebesgue measure 0); see Figure 5 for an illustration in the plane.

Consider the point p_i. Since $p_1, \ldots, p_k$ is the canonical (k, a)-ordering, the condition (L5) implies that p_i lies outside of the polytope $\mathrm{conv}(\{p_1, \ldots, p_{i-1}\})$. To bound the probability $\Pr[E_{a,i} \mid E_{a,i-1}]$, we need to estimate the probability that $\mathrm{conv}(\{p_1, \ldots, p_i\}) \setminus \mathrm{conv}(\{p_1, \ldots, p_{i-1}\})$ does not contain any point from $S \setminus \{p_1, \ldots, p_i\}$, conditioned on $E_{a,i-1}$. We know that p_i lies in $\Delta^* \setminus \Delta$ and that $p_i \in R_j$ for some $j \in \{1, \ldots, d+1\}$.

Since $p_i \notin \mathrm{conv}(\{p_1, \ldots, p_{i-1}\})$, there is a facet φ of the polytope $\mathrm{conv}(\{p_1, \ldots, p_{i-1}\})$ contained in the closure of R_j such that σp_i intersects φ. Since S is in general position with probability 1, we can assume that φ is a $(d-1)$-dimensional simplex. The point p_i is contained in the convex set C_φ that contains all points $c \in \mathbb{R}^d$ such that the line segment σc intersects φ. We use $H(0)$ to denote the hyperplane containing φ. For a positive $r \in \mathbb{R}$, let $H(r)$ be the hyperplane parallel to $H(0)$ at distance r from $H(0)$ such that $H(r)$ is contained in the halfspace determined by $H(0)$ that does not contain $\mathrm{conv}(\{p_1, \ldots, p_{i-1}\})$. Then we have $p_i \in H(h)$ for some positive $h \in \mathbb{R}$.

Since $p_i \in K$ and $\varphi \subseteq K$, the convexity of K implies that the simplex $\mathrm{conv}(\varphi \cup \{p_i\})$ has volume $\lambda_d(\mathrm{conv}(\varphi \cup \{p_i\})) \leq \lambda_d(K) = 1$. Since $\lambda_d(\mathrm{conv}(\varphi \cup \{p_i\})) = \lambda_{d-1}(\varphi) \cdot h/d$, we obtain $h \leq d/\lambda_{d-1}(\varphi)$.

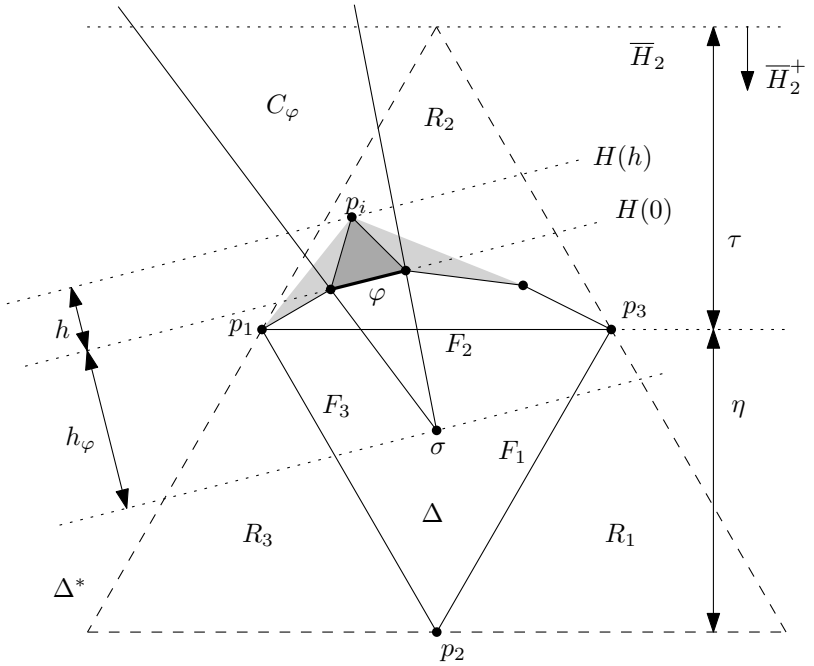


Figure 5 An illustration of the proof of Lemma 10. In order for $\{p_1, \dots, p_i\}$ to be an i-island in S, the light gray part cannot contain points from S. We estimate the probability of this event from above by the probability that the dark gray simplex $\mathrm{conv}(\varphi \cup \{p_i\})$ contains no point of S. Note that the parameters η and τ coincide for $d = 2$, as then $\tau = \frac{d^2-1}{d+1}\eta = \eta$.

The point p_i lies in the $(d-1)$-dimensional simplex $C_\varphi \cap H(h)$, which is a scaled copy of φ. We show that

$$\lambda_{d-1}(C_\varphi \cap H(h)) \leq d^{2d-2} \cdot \lambda_{d-1}(\varphi). \tag{3}$$

Let h_φ be the distance between $H(0)$ and σ and, for every $j \in \{1, \dots, d+1\}$, let $\overline{H}_j$ be the hyperplane parallel to F_j containing the vertex $H_1 \cap \cdots \cap H_{j-1} \cap H_{j+1} \cap \cdots \cap H_{d+1}$ of Δ^*. We denote by $\overline{H}_j^+$ the halfspace determined by $\overline{H}_j$ containing Δ^*. Since Δ lies on the same side of $H(0)$ as σ, we see that h_φ is at least as large as the distance between σ and F_j, which is $\eta/(d+1)$. Since p_i lies in $\Delta^* \subseteq \overline{H}_j^+$, we see that h is at most as large as the distance τ between $\overline{H}_j$ and the hyperplane containing the facet F_j of Δ. Note that $\tau + \eta/(d+1)$ is the distance of the barycenter of Δ^* and a vertex of Δ^* and $d\eta/(d+1)$ is the distance of the barycenter of Δ^* and a facet of Δ^*. Thus we get $\tau = \frac{d^2\eta}{d+1} - \frac{\eta}{d+1} = \frac{d^2-1}{d+1}\eta$ from the fact that the distance between the barycenter of a d-dimensional simplex and any of its vertices is d-times as large as the distance between the barycenter and a facet. Consequently, $h \leq \frac{d^2-1}{d+1}\eta$ and $\frac{\eta}{d+1} \leq h_\varphi$, which implies $h \leq (d^2-1)h_\varphi$. Thus $C_\varphi \cap H(h)$ is a scaled copy of φ by a factor of size at most d^2. This gives $\lambda_{d-1}(C_\varphi \cap H(h)) \leq d^{2d-2} \cdot \lambda_{d-1}(\varphi)$.

Since the simplex $\mathrm{conv}(\varphi \cup \{p_i\})$ is a subset of the closure of $\mathrm{conv}(\{p_1, \dots, p_i\}) \setminus \mathrm{conv}(\{p_1, \dots, p_{i-1}\})$, the probability $\Pr[E_{a,i} \mid E_{a,i-1}]$ can be bounded from above by the conditional probability of the event $A_{i,\varphi}$ that $p_i \in C_\varphi \cap K$ and that no point from $S \setminus \{p_1, \dots, p_i\}$ lies in $\mathrm{conv}(\varphi \cup \{p_i\})$, conditioned on $E_{a,i-1}$. All points from $S \setminus \{p_1, \dots, p_i\}$ lie outside of $\mathrm{conv}(\varphi \cup \{p_i\})$ with probability

$$\left(1 - \frac{\lambda_d(\mathrm{conv}(\varphi \cup \{p_i\}))}{\lambda_d(K \setminus \mathrm{conv}(\{p_1, \dots, p_{i-1}\}))}\right)^{n-i}.$$

Since $\lambda_d(K \setminus \operatorname{conv}(\{p_1, \dots, p_{i-1}\})) \leq \lambda_d(K) = 1$, this is bounded from above by

$$(1 - \lambda_d(\operatorname{conv}(\varphi \cup \{p_i\})))^{n-i} = \left(1 - \frac{\lambda_{d-1}(\varphi) \cdot h}{d}\right)^{n-i}.$$

Since the sets C_φ partition $K \setminus \operatorname{conv}(\{p_1, \dots, p_{i-1}\})$ (up to intersections of d-dimensional Lebesgue measure 0) and since $h \leq d/\lambda_{d-1}(\varphi)$, we have, by the law of total probability,

$$\begin{aligned}\Pr[E_{a,i} \mid E_{a,i-1}] &\leq \sum_\varphi \Pr[A_{i,\varphi} \mid E_{a,i-1}] \\ &\leq \sum_\varphi \int_0^{d/\lambda_{d-1}(\varphi)} \lambda_{d-1}(C_\varphi \cap H(h)) \cdot \left(1 - \frac{\lambda_{d-1}(\varphi) \cdot h}{d}\right)^{n-i} \mathrm{d}h.\end{aligned}$$

The sums in the above expression are taken over all facets φ of the convex polytope $\operatorname{conv}(\{p_1, \dots, p_{i-1}\})$. Using (3), we can estimate $\Pr[E_{a,i} \mid E_{a,i-1}]$ from above by

$$d^{2d-2} \cdot \sum_\varphi \lambda_{d-1}(\varphi) \cdot \int_0^{d/\lambda_{d-1}(\varphi)} \left(1 - \frac{\lambda_{d-1}(\varphi) \cdot h}{d}\right)^{n-i} \mathrm{d}h.$$

By substituting $t = \frac{\lambda_{d-1}(\varphi) \cdot h}{d}$, we can rewrite this expression as

$$d^{2d-2} \cdot \sum_\varphi \frac{d \cdot \lambda_{d-1}(\varphi)}{\lambda_{d-1}(\varphi)} \cdot \int_0^1 (1-t)^{n-i} \,\mathrm{d}t = d^{2d-1} \cdot \sum_\varphi \int_0^1 1 \cdot (1-t)^{n-i} \,\mathrm{d}t.$$

By Lemma 7, this equals

$$d^{2d-1} \cdot \sum_\varphi \frac{0! \cdot (n-i)!}{(n-i+1)!} = \frac{d^{2d-1}}{n-i+1} \sum_\varphi 1.$$

Since $\operatorname{conv}(\{p_1, \dots, p_{i-1}\})$ is a convex polytope in $\mathbb{R}^d$ with at most $i - 1 \leq k$ vertices, Theorem 8 implies that the number of facets φ of $\operatorname{conv}(\{p_1, \dots, p_{i-1}\})$ is at most $2\binom{k}{\lfloor d/2 \rfloor}$. Altogether, we have derived the desired bound

$$\Pr[E_{a,i} \mid E_{a,i-1}] \leq \frac{2d^{2d-1} \cdot \binom{k}{\lfloor d/2 \rfloor}}{n-i+1}$$

in the case when Δ is a regular simplex.

If Δ is not regular, we first apply a volume-preserving affine transformation F that maps Δ to a regular simplex $F(\Delta)$. The simplex $F(\Delta)$ is then contained in the convex body $F(K)$ of volume 1. Since F translates the uniform distribution on $F(K)$ to the uniform distribution on K and preserves holes and islands, we obtain the required upper bound also in the general case. ◀

Now, we finish the proof of Theorem 1.

Proof of Theorem 1. We estimate the expected value of the number X of k-islands in S. The number of ordered k-tuples of points from S is $n(n-1)\cdots(n-k-1)$. Since every subset of S of size k admits a unique labeling that satisfies the conditions (L1), (L2), (L3), (L4), and (L5), we have

$$\begin{aligned}\mathbb{E}[X] &= n(n-1)\cdots(n-k+1)\cdot \Pr\left[\cup_{a=0}^{k-d-1} E_{a,k}\right] \\ &= n(n-1)\cdots(n-k+1)\cdot \sum_{a=0}^{k-d-1} \Pr\left[E_{a,k}\right],\end{aligned}$$

as the events $E_{0,k},\dots E_{k-d-1,k}$ are pairwise disjoint.

The probability of the event L_2, which says that the points $p_1,\dots,p_d$ satisfy the condition (L2), is $1/d!$. Let $P = \sum_{a=0}^{k-d-1} \Pr\left[E_{a,k} \mid L_2\right]$. For any two events E, E' with $E \supseteq E'$ and $\Pr[E] > 0$, we have $\Pr[E'] = \Pr[E \cap E'] = \Pr[E' \mid E]\cdot\Pr[E]$. Thus, using $L_2 \supseteq E_a = E_{a,d+a+1} \supseteq E_{a,d+a+2} \supseteq \cdots \supseteq E_{a,k}$, we get

$$\mathbb{E}[X] = n(n-1)\cdots(n-k+1)\cdot\Pr[L_2]\cdot P = \frac{n(n-1)\cdots(n-k+1)}{d!}\cdot P$$

and

$$P = \sum_{a=0}^{k-d-1} \Pr[E_a \mid L_2]\cdot \prod_{i=d+a+2}^{k} \Pr[E_{a,i} \mid E_{a,i-1}].$$

For every $a \in \{d+2,\dots,k-d-1\}$, Lemma 9 gives

$$\Pr[E_a \mid L_2] \le \frac{2^{d-1}\cdot d!}{(k-a-d-1)!\cdot(n-k+1)^{a+1}} \le \frac{2^{d-1}\cdot d!}{(n-k+1)^{a+1}}$$

and, due to Lemma 10,

$$\Pr[E_{a,i} \mid E_{a,i-1}] \le \frac{2d^{2d-1}\cdot\binom{k}{\lfloor d/2\rfloor}}{n-i+1}$$

for every $i \in \{d+a+2,\dots,k\}$.

Using these estimates we derive

$$\begin{aligned}P &\le 2^{d-1}\cdot d!\cdot\left(2d^{2d-1}\binom{k}{\lfloor d/2\rfloor}\right)^{k-d-1}\cdot\sum_{a=0}^{k-d-1}\frac{1}{(n-k+1)^{a+1}}\cdot\prod_{i=d+a+2}^{k}\frac{1}{n-i+1} \\ &\le 2^{d-1}\cdot d!\cdot\left(2d^{2d-1}\binom{k}{\lfloor d/2\rfloor}\right)^{k-d-1}\cdot\sum_{a=0}^{k-d-1}\frac{1}{(n-k+1)^{a+1}}\cdot\frac{1}{(n-k+1)^{k-d-a-1}} \\ &= 2^{d-1}\cdot d!\cdot\left(2d^{2d-1}\binom{k}{\lfloor d/2\rfloor}\right)^{k-d-1}\cdot(k-d)\cdot\frac{1}{(n-k+1)^{k-d}}.\end{aligned}$$

Thus the expected number of k-islands in S satisfies

$$\begin{aligned}\mathbb{E}[X] &= \frac{n(n-1)\cdots(n-k+1)}{d!}\cdot P \\ &\le \frac{2^{d-1}\cdot d!\cdot\left(2d^{2d-1}\binom{k}{\lfloor d/2\rfloor}\right)^{k-d-1}\cdot(k-d)}{d!}\cdot\frac{n(n-1)\cdots(n-k+1)}{(n-k+1)^{k-d}} \\ &= 2^{d-1}\cdot\left(2d^{2d-1}\binom{k}{\lfloor d/2\rfloor}\right)^{k-d-1}\cdot(k-d)\cdot\frac{n(n-1)\cdots(n-k+2)}{(n-k+1)^{k-d-1}}.\end{aligned}$$

This finishes the proof of Theorem 1. ◀

In the rest of the section, we sketch the proof of Theorem 2 by showing that a slight modification of the above proof yields an improved bound on the expected number $EH^K_{d,k}(n)$ of k-holes in S.

Sketch of the proof of Theorem 2. If k points from S determine a k-hole in S, then, in particular, the simplex Δ contains no points of S in its interior. Therefore

$$EH^K_{d,k}(n) \leq n(n-1)\cdots(n-k+1)\cdot \Pr[E_{0,k}].$$

Then we proceed exactly as in the proof of Theorem 1, but we only consider the case $a = 0$. This gives the same bounds as before with the term $(k-d)$ missing and with an additional factor $\frac{1}{(k-d-1)!}$ from Lemma 9, which proves Theorem 2. ◀

For $d = 2$ and $k = 4$, Theorem 2 gives $EH^K_{2,4}(n) \leq 128n^2 + o(n^2)$. We can obtain an even better estimate $EH^K_{2,4}(n) \leq 12n^2 + o(n^2)$ in this case. First, we have only three facets φ, as they correspond to the sides of the triangle Δ. Thus the term $\left(2\binom{k}{\lfloor d/2\rfloor}\right)^{k-d-1} = 8$ is replaced by 3. Moreover, the inequality (3) can be replaced by

$$\lambda_1(C_\varphi \cap H(h) \cap \Delta^*) \leq \lambda_1(\varphi),$$

since every line $H(h)$ intersects $R_j \subseteq \Delta^*$ in a line segment of length at most $\lambda_1(F_j) = \lambda(\varphi)$. This then removes the factor $d^{(2d-2)(k-d-1)} = 4$.

4 Proof of Theorem 5

Here, for every d, we state the definition of a d-dimensional analogue of Horton sets on n points from [18] and show that, for all fixed integers d and k, every d-dimensional Horton set H with n points contains at least $\Omega(n^{\min\{2^{d-1},k\}})$ k-islands in H. If $k \leq 3\cdot 2^{d-1}$, then we show that H contains at least $\Omega(n^k)$ k-holes in H.

First, we need to introduce some notation. A set Q of points in $\mathbb{R}^d$ is in *strongly general position* if Q is in general position and, for every $i = 1,\ldots,d-1$, no $(i+1)$-tuple of points from Q determines an i-dimensional affine subspace of $\mathbb{R}^d$ that is parallel to the $(d-i)$-dimensional linear subspace of $\mathbb{R}^d$ that contains the last $d-i$ axes. Let $\pi\colon \mathbb{R}^d \to \mathbb{R}^{d-1}$ be the projection defined by $\pi(x_1,\ldots,x_d) = (x_1,\ldots,x_{d-1})$. For $Q \subseteq \mathbb{R}^d$, we use $\pi(Q)$ to denote the set $\{\pi(q) \in \mathbb{R}^{d-1}\colon q \in Q\}$. If Q is a set of n points $q_0,\ldots,q_{n-1}$ from $\mathbb{R}^d$ in strongly general position that are ordered so that their first coordinates increase, then, for all $m \in \mathbb{N}$ and $i \in \{0,1,\ldots,m-1\}$, we define $Q_{i,m} = \{q_j \in Q\colon j \equiv i \pmod m\}$.

For two sets A and B of points from $\mathbb{R}^d$ with $|A|, |B| \geq d$, we say that B is *deep below* A and A is *high above* B if B lies entirely below any hyperplane determined by d points of A and A lies entirely above any hyperplane determined by d points of A. For point sets A' and B' in $\mathbb{R}^d$ of arbitrarily size, we say that B' is *deep below* A' and A' is *high above* B' if there are sets $A \supseteq A'$ and $B \supseteq B'$ such that $|A|, |B| \geq d$, B is deep below A, and A is high above B.

Let $p_2 < p_3 < p_4 < \cdots$ be the sequence of all prime numbers. That is, $p_2 = 2$, $p_3 = 3$, $p_4 = 5$ and so on.

We can now state the definition of the d-dimensional Horton sets from [18]. Every finite set of n points in $\mathbb{R}$ is 1-*Horton*. For $d \geq 2$, finite set H of points from $\mathbb{R}^d$ in strongly general position is a *d-Horton set* if it satisfies the following conditions:

(a) the set H is empty or it consists of a single point, or

(b) H satisfies the following three conditions:

- **(i)** if $d > 2$, then $\pi(H)$ is $(d-1)$-Horton,
- **(ii)** for every $i \in \{0, 1, \dots, p_d - 1\}$, the set H_{i,p_d} is d-Horton,
- **(iii)** every $I \subseteq \{0, 1, \dots, p_d - 1\}$ with $|I| \geq 2$ can be partitioned into two nonempty subsets J and $I \setminus J$ such that $\cup_{j \in J} H_{j,p_d}$ lies deep below $\cup_{i \in I \setminus J} H_{i,p_d}$.

Valtr [18] showed that such sets indeed exist and that they contain no k-hole with $k > 2^{d-1}(p_2 p_3 \cdots p_d + 1)$. The 2-Horton sets are known as *Horton sets*. We show that d-Horton sets with $d \geq 3$ contain many k-islands for $k \geq d+1$ and thus cannot provide the upper bound $O(n^d)$ that follows from Theorem 1. This contrasts with the situation in the plane, as 2-Horton sets of n points contain only $O(n^2)$ k-islands for any fixed k [8].

Let d and k be fixed positive integers. Assume first that $k \geq 2^{d-1}$. We want to prove that there are $\Omega(n^{2^{d-1}})$ k-islands in every d-Horton set H with n points. We proceed by induction on d. For $d = 1$ there are $n - k + 1 = \Omega(n)$ k-islands in every 1-Horton set.

Assume now that $d > 1$ and that the statement holds for $d-1$. The d-Horton set H consists of $p_d \in O(1)$ subsets H_{i,p_d}, each of size at least $\lfloor n/p_d \rfloor \in \Omega(n)$. The set $\{0, \dots, p_d - 1\}$ is ordered by a linear ordering $\prec$ such that, for all i and j with $i \prec j$, the set H_{i,p_d} is deep below H_{j,p_d}; see [18]. Take two of sets $X = H_{a,p_d}$ and $Y = H_{b,p_d}$ such that $a \prec b$ are consecutive in $\prec$. Since $k \geq 2^{d-1}$, we have $\lceil k/2 \rceil \geq \lfloor k/2 \rfloor \geq 2^{d-2}$. Thus by the inductive hypothesis, the $(d-1)$-Horton set $\pi(X)$ of size at least $\Omega(n)$ contains at least $\Omega(n^{2^{d-2}})$ $\lfloor k/2 \rfloor$-islands. Similarly, the $(d-1)$-Horton set $\pi(Y)$ of size at least $\Omega(n)$ contains at least $\Omega(n^{2^{d-2}})$ $\lceil k/2 \rceil$-islands.

Let $\pi(A)$ be any of the $\Omega(n^{2^{d-2}})$ $\lfloor k/2 \rfloor$-islands in $\pi(X)$, where $A \subseteq X$. Similarly, let $\pi(B)$ be any of the $\Omega(n^{2^{d-2}})$ $\lceil k/2 \rceil$-islands in $\pi(Y)$, where $B \subseteq Y$. We show that $A \cup B$ is a k-island in H. Suppose for contradiction that there is a point $x \in H \setminus (A \cup B)$ that lies in $\mathrm{conv}(A \cup B)$. Since a and b are consecutive in $\prec$, the point x lies in $X \cup Y = H_{a,p_d} \cup H_{b,p_d}$. By symmetry, we may assume without loss of generality that $x \in X$. Since $x \notin A$ and H is in strongly general position, we have $\pi(x) \in \pi(X) \setminus \pi(A)$. Using the fact that $\pi(A)$ is a $\lfloor k/2 \rfloor$-island in $\pi(X)$, we obtain $\pi(x) \notin \mathrm{conv}(\pi(A))$ and thus $x \notin \mathrm{conv}(A)$. Since X is deep below Y, we have $x \notin \mathrm{conv}(B)$. Thus, by Carathédory's theorem, x lies in the convex hull of a $(d+1)$-tuple $T \subseteq A \cup B$ that contains a point from A and also a point from B.

Note that, for $U = (T \cup \{x\})$, we have $|U \cap A| \geq 2$, as $x \in A$ and $|T \cap A| \geq 1$. We also have $|U \cap B| \geq 2$, as X is deep below Y and $\pi(x) \notin \mathrm{conv}(\pi(A))$. Thus the affine hull of $U \cap A$ intersects the convex hull of $U \cap B$. Then, however, the set $U \cap A$ is not deep below the set $U \cap B$, which contradicts the fact that X is deep below Y.

Altogether, there are at least $\Omega(n^{2^{d-2}}) \cdot \Omega(n^{2^{d-2}}) = \Omega(n^{2^{d-1}})$ such k-islands $A \cup B$, which finishes the proof if k is at least 2^{d-1}. For $k < 2^{d-1}$, we use an analogous argument that gives at least $\Omega(n^{\lfloor k/2 \rfloor}) \cdot \Omega(n^{\lceil k/2 \rceil}) = \Omega(n^k)$ k-islands in the inductive step.

If $d \geq 2$ and $k \leq 3 \cdot 2^{d-1}$ then a slight modification of the above proof gives $\Omega(n^{\min\{2^{d-1}, k\}})$ k-islands which are actually k-holes in H. We just use the simple fact that every 2-Horton set with n points contains $\Omega(n^2)$ k-holes for every $k \in \{2, \dots, 6\}$ as our inductive hypothesis. This is trivial for $k = 2$ and it follows for $k \in \{3, 4\}$ from the well-known fact that every set of n points in $\mathbb{R}^2$ in general position contains at least $\Omega(n^2)$ k-holes. For $k \in \{5, 6\}$, this fact can be proved using basic properties of 2-Horton sets (we omit the details). Then we use the inductive assumption, which says that every d-Horton set of n points contains at least $\Omega(n^{\min\{2^{d-1}, k\}})$ k-holes if $d \geq 2$ and $1 \leq k \leq 3 \cdot 2^{d-1}$. This finishes the proof of Theorem 5.

References

1 O. Aichholzer, M. Balko, T. Hackl, J. Kynčl, I. Parada, M. Scheucher, P. Valtr, and B. Vogtenhuber. A Superlinear Lower Bound on the Number of 5-Holes. In *33rd International Symposium on Computational Geometry (SoCG 2017)*, volume 77 of *Leibniz International Proceedings in Informatics*, pages 8:1–8:16, 2017. Full version: arXiv:1703.05253. doi:10.4230/LIPIcs.SoCG.2017.8.

2 G. E. Andrews, R. Askey, and R. Roy. *Special functions*, volume 71 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, Cambridge, 1999. doi:10.1017/CBO9781107325937.

3 J. Balogh, H. González-Aguilar, and G. Salazar. Large convex holes in random point sets. *Computational Geometry*, 46(6):725–733, 2013. doi:10.1016/j.comgeo.2012.11.004.

4 I. Bárány and Z. Füredi. Empty simplices in Euclidean space. *Canadian Mathematical Bulletin*, 30(4):436–445, 1987. doi:10.4153/cmb-1987-064-1.

5 I. Bárány and P. Valtr. Planar point sets with a small number of empty convex polygons. *Studia Scientiarum Mathematicarum Hungarica*, 41(2):243–266, 2004. doi:10.1556/sscmath.41.2004.2.4.

6 P. Erdős. Some more problems on elementary geometry. *Australian Mathematical Society Gazette*, 5:52–54, 1978. URL: http://www.renyi.hu/~p_erdos/1978-44.pdf.

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

8 R. Fabila-Monroy and C. Huemer. Covering Islands in Plane Point Sets. In *Computational Geometry: XIV Spanish Meeting on Computational Geometry, EGC 2011*, volume 7579 of *Lecture Notes in Computer Science*, pages 220–225. Springer, 2012. doi:10.1007/978-3-642-34191-5_21.

9 R. Fabila-Monroy, C. Huemer, and D. Mitsche. Empty non-convex and convex four-gons in random point sets. *Studia Scientiarum Mathematicarum Hungarica. A Quarterly of the Hungarian Academy of Sciences*, 52(1):52–64, 2015. doi:10.1556/SScMath.52.2015.1.1301.

10 T. Gerken. Empty Convex Hexagons in Planar Point Sets. *Discrete & Computational Geometry*, 39(1):239–272, 2008. doi:10.1007/s00454-007-9018-x.

11 H. Harborth. Konvexe Fünfecke in ebenen Punktmengen. *Elemente der Mathematik*, 33:116–118, 1978. In German. URL: http://www.digizeitschriften.de/dms/img/?PID=GDZPPN002079801.

12 J. D. Horton. Sets with no empty convex 7-gons. *Canadian Mathematical Bulletin*, 26:482–484, 1983. doi:10.4153/CMB-1983-077-8.

13 M. Katchalski and A. Meir. On empty triangles determined by points in the plane. *Acta Mathematica Hungarica*, 51(3-4):323–328, 1988. doi:10.1007/BF01903339.

14 J. Matoušek. *Lectures on discrete geometry*, volume 212 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 2002. doi:10.1007/978-1-4613-0039-7.

15 M. C. Nicolas. The Empty Hexagon Theorem. *Discrete & Computational Geometry*, 38(2):389–397, 2007. doi:10.1007/s00454-007-1343-6.

16 Matthias Reitzner and Daniel Temesvari. Stars of empty simplices, 2019. arXiv:1808.08734.

17 M. Scheucher. *Points, Lines, and Circles: Some Contributions to Combinatorial Geometry*. PhD thesis, Technische Universität Berlin, Institut für Mathematik, 2019.

18 P. Valtr. Sets in $\mathbb{R}^d$ with no large empty convex subsets. *Discrete Mathematics*, 108(1):115–124, 1992. doi:10.1016/0012-365X(92)90665-3.

19 P. Valtr. On the minimum number of empty polygons in planar point sets. *Studia Scientiarum Mathematicarum Hungarica*, pages 155–163, 1995. URL: https://refubium.fu-berlin.de/handle/fub188/18741.

20 P. Valtr. On empty hexagons. In *Surveys on Discrete and Computational Geometry: Twenty Years Later*, volume 453 of *Contemporary Mathematics*, pages 433–441. American Mathematical Society, 2008. URL: http://bookstore.ams.org/conm-453.

21 P. Valtr. On empty pentagons and hexagons in planar point sets. In *Proceedings of Computing: The Eighteenth Australasian Theory Symposium (CATS 2012)*, pages 47–48, Melbourne, Australia, 2012. URL: http://crpit.com/confpapers/CRPITV128Valtr.pdf.

The Reeb Graph Edit Distance Is Universal

Ulrich Bauer
Department of Mathematics, Technical University of Munich (TUM), Germany
ulrich.bauer@tum.de

Claudia Landi
Dipartimento di Scienze e Metodi dell'Ingegneria, Università degli Studi di Modena e Reggio Emilia, Reggio Emilia, Italy
clandi@unimore.it

Facundo Mémoli
Department of Mathematics, The Ohio State University, Columbus, OH, USA
memoli@math.osu.edu

Abstract

We consider the setting of Reeb graphs of piecewise linear functions and study distances between them that are stable, meaning that functions which are similar in the supremum norm ought to have similar Reeb graphs. We define an edit distance for Reeb graphs and prove that it is stable and universal, meaning that it provides an upper bound to any other stable distance. In contrast, via a specific construction, we show that the interleaving distance and the functional distortion distance on Reeb graphs are not universal.

2012 ACM Subject Classification Theory of computation → Computational geometry; Mathematics of computing → Algebraic topology

Keywords and phrases Reeb graphs, topological descriptors, edit distance, interleaving distance

Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.15

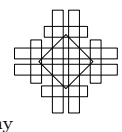
Funding This research has been partially supported by FAR 2014 (UniMORE), ARCES (University of Bologna), and the DFG Collaborative Research Center SFB/TRR 109 "Discretization in Geometry and Dynamics".

Acknowledgements We thank Barbara Di Fabio and Yusu Wang for valuable discussions.

1 Introduction

The concept of a Reeb graph of a Morse function first appeared in [13] and has subsequently been applied to problems in shape analysis in [14, 10]. The literature on Reeb graphs in the computational geometry and computational topology is ever growing (see, e.g., [2, 3] for a discussion and references). The Reeb graph plays a central role in topological data analysis, not least because of the success of Mapper [15], a data analysis method providing a discretization of the Reeb graph for a function defined on a point cloud.

A recent line of work has concentrated on questions about identifying suitable notions of distance between Reeb graphs. These include the so called *functional distortion distance* [2], the *interleaving distance* [6], and various *graph edit distances* [9, 7, 1]. Naturally, there is a strong interest in understanding the connection between different existing distances. In this regard, it has been shown in [3] that the functional distortion and the interleaving distances are bi-Lipschitz equivalent. The edit distances defined in [9, 7] for Reeb graphs of curves and surfaces, respectively, are shown to be universal in their respective settings, so the functional distortion and interleaving distances restricted to the same settings are a lower bound for those distances. Moreover, an example in [7] shows that the functional distortion distance can be strictly smaller than the edit distance considered in that paper.



In this paper, we consider the setting of piecewise linear (PL) functions on compact triangulable spaces, and in this realm we study the properties of *stability* and *universality* of distances between Reeb graphs. The notion of stability has been introduced by Cohen-Steiner et al. [4] in the context of persistence diagrams, and is a key property for topological descriptors [12]. Stability means that two objects at a given distance are assigned descriptors at no more than that distance. This requires a notion of distance on both the collection of objects as well as on the collection of descriptors. The practical relevance of stability lies in the guaranteed robustness of the method with respect to bounded imprecision, caused by noise, coarse sampling, or other sources of uncertainty. However, the stability of a descriptor is not sufficient to warrant *discriminativeness*, i.e., the ability to distinguish different objects: a construction that assigns to every object the same descriptor is certainly stable, but contains no information. For that reason, given a fixed distance on the objects and a construction for a descriptor, it is desirable to assign to the descriptors a distance that is as large as possible while still satisfying the stability property. In that sense, such a distance is then the most discriminative stable distance. Following Lesnick [11], we call such a distance *universal*, noting that the concept already appears in [5] in the context of topological descriptors.

Inspired by a construction of distance between filtered spaces [12], we first construct a novel distance δ_U based on considering joint pullbacks of two given Reeb graphs and prove that this distance satisfies both stability and universality. Via analyzing a specific construction we then prove that neither the functional distortion nor the interleaving distances are universal. Finally, we define two edit-like additional distances between Reeb graphs that reinterpret those appearing in [9, 7, 1] and prove that both are stable and universal. As a consequence, both distances agree with δ_U.

2 Topological aspects of Reeb graphs

We start by exploring some topological ideas behind the definition of Reeb graphs. All maps and functions considered in this paper will be assumed to be continuous. Otherwise, we call them set maps and set functions.

2.1 Reeb graphs as quotient spaces

The classical construction of a Reeb graph [13] is given via an equivalence relation as follows:

▶ **Definition 2.1.** *For $f : X \to \mathbb{R}$ a Morse function on a compact smooth manifold, the* Reeb graph of f *is the quotient space $X/\sim_f$, with $x \sim_f y$ if and only if x and y belong to the same connected component of some level set $f^{-1}(t)$ (implying $t = f(x) = f(y)$).*

While this definition was originally considered in the setting of Morse theory, it does not make explicit use of the smooth structure, and so it can be applied quite broadly. However, some additional assumptions on the space X and the function f are justified in order to maintain some of the characteristic properties of Reeb graphs in a generalized setting. With this motivation in mind, we revisit the definition in terms of quotient maps and functions with discrete fibers.

A *quotient map* $p : X \to Y$ is a surjection such that a set U is open in Y if and only if $p^{-1}(U)$ is open in X. In particular, a surjection between compact Hausdorff spaces is a quotient map by the closed map lemma. A quotient map $p : X \to Y$ is characterized by the universal property that a set map $\Phi : Y \to Z$ into any topological space Z is continuous if and only if $\Phi \circ p$ is continuous.

The motivation for considering quotient maps and functions with discrete fibers is explained by the following fact.

▶ **Proposition 2.2.** *Let $f : X \to \mathbb{R}$ be a function with locally connected fibers, and let $q : X \to X/\!\sim_f$ be the canonical quotient map. Then the induced function $\tilde{f} : X/\!\sim_f \to \mathbb{R}$ with $f = \tilde{f} \circ q$ has discrete fibers.*

Proof. To see that the fibers of $\tilde{f}$ are discrete, we show that any subset S of $\tilde{f}^{-1}(t)$ is closed. Let $T = \tilde{f}^{-1}(t) \setminus S$. Then $q^{-1}(T)$ is a disjoint union of connected components of $f^{-1}(t)$. Since $f^{-1}(t)$ is locally connected, each of its connected components is open in the fiber, and so $q^{-1}(T)$ is open in $f^{-1}(t)$, implying that $q^{-1}(S)$ is closed in $f^{-1}(t)$ and hence in X. Since q is a quotient map, $q^{-1}(S)$ is closed if and only if S is closed, yielding the claim. ◀

2.2 Reeb quotient maps and Reeb graphs of piecewise linear functions

We now define a class of quotient maps that leave Reeb graphs invariant up to isomorphism. The main goal is to provide a natural construction for lifting a function $f : X \to \mathbb{R}$ to a space Y through a quotient map $Y \to X$ in a way that yields isomorphic Reeb graphs. To this end, we will define a general notion of Reeb quotient maps and Reeb graphs.

▶ **Definition 2.3.** *A* Reeb domain *is a connected compact triangulable space. A* Reeb quotient map *is a surjective piecewise linear map of Reeb domains with connected fibers.*

We remark that connectedness of Reeb domains is assumed only for the sake of simplicity (see Remark 3.4).

As shown in Corollary 2.8, Reeb domains and Reeb quotient maps constitute a subcategory of the category of triangulable spaces and piecewise linear maps.

▶ **Definition 2.4.** *A* Reeb graph *is a pair $(R_f, \tilde{f})$ where R_f is a Reeb domain endowed with a PL function $\tilde{f} : R_f \to \mathbb{R}$ with discrete fibers, called a* Reeb function*.*

In particular, the isomorphisms between Reeb graphs are PL homeomorphisms that preserve the function values of the associated Reeb functions. While the definition does not assume this explicitly, a Reeb graph is indeed a *finite topological graph* (a compact triangulable space of dimension at most 1).

▶ **Proposition 2.5.** *For any Reeb graph $(R_f, \tilde{f})$, the space R_f is a finite topological graph.*

Proof. By definition, $\tilde{f}$ is (simplexwise) linear for some triangulation of R_f. If there were a simplex σ of dimension at least 2 in the triangulation of R_f, then for any x in the interior of σ, the intersection $\sigma \cap \tilde{f}^{-1}(\tilde{f}(x))$ would have to be of dimension at least 1. But this would contradict the assumption that $\tilde{f}$ has discrete fibers. ◀

▶ **Definition 2.6.** *Generalizing the classical definition (Definition 2.1), we say that a Reeb graph $(R_f, \tilde{f})$ is a* Reeb graph of $f : X \to \mathbb{R}$ *if there is a Reeb quotient map $p : X \to R_f$ such that $f = \tilde{f} \circ p$.*

We now proceed to prove that Reeb quotient maps are closed under composition. We start by showing that not only the fibers, but more generally all preimages of closed connected sets are connected.

▶ **Proposition 2.7.** *If $p\colon X \to Y$ is a Reeb quotient map, then the preimage $p^{-1}(K)$ of a closed connected set $K \subseteq Y$ is connected.*

Proof. Assume that K is nonempty; otherwise, the claim holds trivially. Let $p^{-1}(K) = U \cup V$, with U, V nonempty and closed in $p^{-1}(K)$. To show that $p^{-1}(K)$ is connected, it suffices to show that $U \cap V$ is necessarily nonempty.

Because $p^{-1}(K)$ is closed in X, the sets U and V are also closed in X. The images $p(U)$ and $p(V)$ are closed by the closed map lemma, and their union is K. By connectedness of K, their intersection is nonempty. Let $y \in p(U) \cap p(V)$. We have

$$p^{-1}(y) = (p^{-1}(y) \cap U) \cup (p^{-1}(y) \cap V).$$

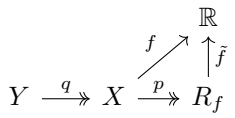
The subspaces $(p^{-1}(y) \cap U)$ and $(p^{-1}(y) \cap V)$ are closed in $p^{-1}(y)$, and by connectedness of the fiber $p^{-1}(y)$, their intersection must be nonempty. In particular, $U \cap V$ is nonempty. ◀

▶ **Corollary 2.8.** *If $p\colon X \to Y$ and $q\colon Y \to Z$ are Reeb quotient maps, then the composition $q \circ p : X \to Z$ is a Reeb quotient map too.*

As mentioned before, the main purpose of Reeb quotient maps is to lift Reeb functions to larger domains while maintaining the same Reeb graph. The following property is a consequence of the above statement:

▶ **Corollary 2.9.** *Let $(R_f, \tilde{f})$ be a Reeb graph of a function $f : X \to \mathbb{R}$, and let $q : Y \to X$ be a Reeb quotient map. Then $(R_f, \tilde{f})$ is also a Reeb graph of $f \circ q : Y \to \mathbb{R}$.*

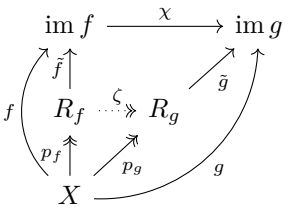
Proof. Let $p : X \to R_f$ be the Reeb quotient map factoring $f = \tilde{f} \circ p$, as in the following diagram:



Then by Corollary 2.8, $(R_f, \tilde{f})$ is also a Reeb graph for $f \circ q = \tilde{f} \circ (p \circ q) : Y \to \mathbb{R}$ via the Reeb quotient map $p \circ q : Y \to R_f$. ◀

The following lemma shows how a transformation $g = \xi \circ f$ of a function f lifts to a Reeb quotient map ζ between the corresponding Reeb graphs.

▶ **Lemma 2.10.** *Consider a commutative diagram*



where $(R_f, \tilde{f}), (R_g, \tilde{g})$ are Reeb graphs, $p_f : X \to R_f, p_g : X \to R_g$ are Reeb quotient maps, and $\chi : \operatorname{im} f \to \operatorname{im} g$ is a PL function such that $g = \chi \circ f$. Then $\zeta = p_g \circ p_f^{-1}$ is a Reeb quotient map from R_f to R_g.

In particular, if χ is a PL homeomorphism, then so is ζ. Note that the definition of ζ does not involve the function χ; the existence of χ already ensures that ζ is a Reeb quotient map.

Proof. Let $x \in R_f$, and let $t = \tilde{f}(x)$. Then $C = p_f^{-1}(x)$ is a connected component of $f^{-1}(t)$ by the assumption that p_f is a Reeb quotient map. By commutativity, we have

$$f^{-1} \subseteq f^{-1} \circ \chi^{-1} \circ \chi = g^{-1} \circ \chi,$$

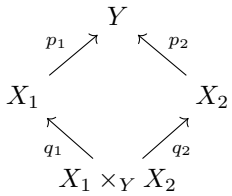
and since C is connected, there must be a single $y \in R_g$ with $p_g(C) = \{y\}$. Hence, $\zeta = p_g \circ p_f^{-1}$ is a set map. Moreover, since p_g is continuous and p_f is closed, the map ζ is continuous; since p_g and p_f are PL, the map ζ is PL as well.

Now let $y \in R_g$ and let $s = \tilde{g}(y)$. Similarly to above, $C = p_g^{-1}(y)$ is a connected component of $g^{-1}(s)$. We have $p_f(C) = p_f \circ p_g^{-1}(y) = \zeta^{-1}(y) \neq \emptyset$, so ζ is surjective, and the fiber $\zeta^{-1}(y) = p_f(C)$ is connected as the image of a connected set. ◀

▶ Remark 2.11. By Proposition 2.2 and Lemma 2.10, given a Reeb graph $(R_f, \tilde{f})$ of $f : X \to \mathbb{R}$ with Reeb quotient map $p\colon X \to R_f$, there is a canonical isomorphism $R_f \cong X/\!\sim_f$. As a consequence, the Reeb graph $(R_f, \tilde{f})$ together with the Reeb quotient map p is unique up to a unique isomorphism, defining the Reeb graph as a universal property.

We now show that Reeb quotient maps are stable under pullbacks.

▶ **Proposition 2.12.** *Consider a pullback diagram of PL maps $p_1\colon X_1 \to Y$, $p_2\colon X_2 \to Y$:*



If the map p_1 (resp. p_2) is a Reeb quotient map, then so is the map q_2 (resp. q_1). Hence, the class of Reeb quotient maps is stable under pullbacks.

Proof. First note that the category of compact triangulable spaces has all pullbacks [16]. For $x_2 \in X_2$, by surjectivity of p_1 there is some $x_1 \in X_1$ such that $p_1(x_1) = p_2(x_2)$. Thus $(x_1, x_2) \in X_1 \times_Y X_2$ and $q_2(x_1, x_2) = x_2$, proving that q_2 is surjective. Moreover, for $x_2 \in X_2$, we have $q_2^{-1}(x_2) = p_1^{-1}(p_2(x_2)) \times \{x_2\}$. By assumption, $p_1^{-1}(p_2(x_2))$ is connected as a fiber of p_1, implying that $p_1^{-1}(p_2(x_2)) \times \{x_2\}$ is connected. Finally, applying Proposition 2.7 to q_2, we obtain that the pullback space $X_1 \times_Y X_2$ is connected. The proof for q_1 is analogous. ◀

3 Stable and universal distances

Throughout this paper, we will use the term *distance* to describe an extended pseudo-metric $d : X \times X \to [0, \infty]$ on some collection X. Our main goal is the introduction of a distance between Reeb graphs that is stable and universal in the following sense.

▶ **Definition 3.1.** *We say that a distance d_S between Reeb graphs is* stable *if and only if given any two Reeb graphs $(R_f, \tilde{f})$ and $(R_g, \tilde{g})$, for any Reeb domain X with Reeb quotient maps $p_f : X \to R_f$ and $p_g : X \to R_g$ we have*

$$d_S((R_f, \tilde{f}), (R_g, \tilde{g})) \leq \|\tilde{f} \circ p_f - \tilde{g} \circ p_g\|_\infty. \tag{S}$$

Note that stability implies that isomorphic Reeb graphs have distance 0. Indeed, an isomorphism of Reeb graphs $\gamma : R_f \to R_g$ yields $d_S((R_f, \tilde{f}), (R_g, \tilde{g})) \leq \|\tilde{f} \circ \mathrm{id} - \tilde{g} \circ \gamma\|_\infty = 0$.

Moreover, we say that a stable distance d_U between Reeb graphs is universal *if and only if for any other stable distance d_S between Reeb graphs, we have*

$$d_S((R_f, \tilde{f}), (R_g, \tilde{g})) \leq d_U((R_f, \tilde{f}), (R_g, \tilde{g})), \tag{U}$$

for all $(R_f, \tilde{f})$ and $(R_g, \tilde{g})$.

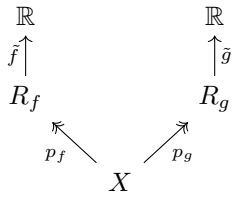
▶ Remark 3.2. By connectedness of R_f and R_g, there is at least one space X with maps p_f, p_g as needed to define the stability property: $X = R_f \times R_g$, with p_f, p_g the canonical projections. The resulting functions $f = \tilde{f} \circ p_f, g = \tilde{g} \circ p_g : R_f \times R_g \to \mathbb{R}$ then satisfy $\|f - g\|_\infty = \max(\sup \tilde{f} - \inf \tilde{g}, \sup \tilde{g} - \inf \tilde{f})$. In particular, by compactness a stable distance for Reeb graphs is always finite.

The definition of stability yields the following universal distance.

▶ **Definition 3.3.** *For any two Reeb graphs $(R_f, \tilde{f}), (R_g, \tilde{g})$, let*

$$\delta_U((R_f, \tilde{f}), (R_g, \tilde{g})) := \inf_{p_f \colon R_f \leftarrow X \to R_g \colon p_g} \|\tilde{f} \circ p_f - \tilde{g} \circ p_g\|_\infty,$$

where the infimum is taken over all possible Reeb domains X and Reeb quotient maps $p_f \colon X \to R_f$ and $p_g \colon X \to R_g$, as in the following diagram.



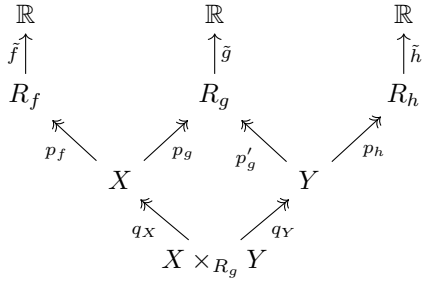
▶ Remark 3.4. The connectedness assumption for Reeb domains can be dropped by adapting the definition of the universal distance as follows. If R_f and R_g have a different number of connected components, then $\delta_U(R_f, R_g) := \infty$. If both R_f and R_g have n connected components so that $R_f = \coprod_{i \in [n]} F_i$ and $R_g = \coprod_{i \in [n]} G_i$ with each F_i and G_i connected, then

$$\delta_U(R_f, R_g) := \min_{\gamma} \inf_{p : F_i \leftarrow X \to G_{\gamma(i)} : q} \|\tilde{f} \circ p - \tilde{g} \circ q\|_\infty$$

where γ varies among all permutations on n objects, $i \in [n]$, and the infimum is taken over all possible Reeb domains X and Reeb quotient maps $p \colon X \to F_i$ and $q \colon X \to G_i$.

▶ **Proposition 3.5.** *The distance δ_U is the largest stable distance on Reeb graphs. Hence, δ_U is universal.*

Proof. To see that δ_U is a distance, the only non-trivial part is showing the triangle inequality. To this end, given diagrams $p_f : R_f \leftarrow X \to R_g : p_g$ and $p'_g : R_g \leftarrow Y \to R_h : p_h$, we can form a pullback of the diagram $p_g : X \to R_g \leftarrow Y : p'_g$ to obtain the diagram



where $X \times_{R_g} Y$ is a Reeb domain and q_X, q_Y are Reeb quotient maps by Proposition 2.12. Defining $f = \tilde{f} \circ p_f \circ q_X$, $g = \tilde{g} \circ p_g \circ q_X = \tilde{g} \circ p'_g \circ q_Y$, and $h = \tilde{h} \circ p_h \circ q_Y$, we have

$$\begin{aligned}\delta_U((R_f, \tilde{f}), (R_h, \tilde{h})) \leq \|f - h\|_\infty &\leq \|f - g\|_\infty + \|g - h\|_\infty \\ &= \|\tilde{f} \circ p_f - \tilde{g} \circ p_g\|_\infty + \|\tilde{g} \circ p'_g - \tilde{h} \circ p_h\|_\infty,\end{aligned}$$

where the last equality holds because q_X and q_Y are surjective. Hence

$$\delta_U((R_f, \tilde{f}), (R_h, \tilde{h})) \leq \delta_U((R_f, \tilde{f}), (R_g, \tilde{g})) + \delta_U((R_g, \tilde{g}), (R_h, \tilde{h})).$$

The stability of δ_U is immediate from its definition. Moreover, for any stable distance d_S between Reeb graphs, combining the stability of d_S and the definition of δ_U, we obtain $d_S \leq \delta_U$, implying that δ_U is universal. ◀

▶ **Corollary 3.6.** *The universal distance δ_U is a metric on isomorphism classes of Reeb graphs.*

Proof. According to Remark 3.2, by stability, δ_U is always finite. Moreover, we recall from [6] that there exists a stable distance d_I, the *interleaving distance*, which is a metric on isomorphism classes of Reeb graphs; in particular, $d_I((R_f, \tilde{f}), (R_g, \tilde{g})) = 0$ if and only if $(R_f, \tilde{f}) \cong (R_g, \tilde{g})$. By stability of d_I and universality of δ_U, we have $d_I((R_f, \tilde{f}), (R_g, \tilde{g})) \leq \delta_U((R_f, \tilde{f}), (R_g, \tilde{g}))$. Thus, $\delta_U((R_f, \tilde{f}), (R_g, \tilde{g})) = 0$ implies $d_I(R_f, R_g) = 0$ and hence $(R_f, \tilde{f}) \cong (R_g, \tilde{g})$. ◀

▶ **Example 3.7.** Consider the one point Reeb graph $(*, c)$ endowed with the function identical to $c \in \mathbb{R}$. Then, for any Reeb graph $(R_f, \tilde{f})$, we have $\delta_U((R_f, \tilde{f}), (*, c)) = \|\tilde{f} - c\|_\infty$.

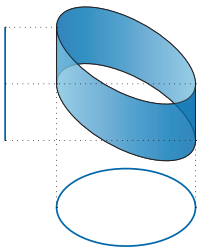
We now consider an example where we can explicitly determine the value of the distance $\delta_U((R_f, \tilde{f}), (R_g, \tilde{g}))$ between two specific simple Reeb graphs $R_f = \mathbb{S}^1 = \{(x, y) \in \mathbb{R}^2 : x^2 + y^2 = 1\}$ with $\tilde{f}(x, y) = x$ and $R_g = [-1, 1]$ with $\tilde{g}(t) = t$. The example demonstrates the non-universality of certain distances proposed in the literature. We prove:

▶ **Proposition 3.8.** $\delta_U((R_f, \tilde{f}), (R_g, \tilde{g})) = 1$.

The proof of this proposition will be obtained from the two claims below.

▷ Claim 3.9. $\delta_U(R_f, R_g) \leq 1$.

Proof. Consider the cylinder $C = \{(x, y, z) \in \mathbb{R}^3 : x^2 + y^2 = 1, |2z - x| \leq 1\}$ together with functions $f(x, y, z) = x$ and $g(x, y, z) = z$ defined on C. Then $(R_f, \tilde{f})$ is a Reeb graph of f



via the Reeb quotient map $(x, y, z) \mapsto (x, y)$, and $(R_g, \tilde{g})$ is a Reeb graph of g via the Reeb quotient map $(x, y, z) \mapsto z$. Since we have $|f(c) - g(c)| \leq 1$ for all $c \in C$, this implies that $\delta_U((R_f, \tilde{f}), (R_g, \tilde{g})) \leq 1$. ◁

▷ Claim 3.10. $\delta_U((R_f, \tilde{f}), (R_g, \tilde{g})) \geq 1$.

Proof. Assume for a contradiction that there is a diagram $p_f : R_f \leftarrow Z \rightarrow R_g : p_g$ of Reeb quotient maps such that, letting $\hat{f} = \tilde{f} \circ p_f$ and $\hat{g} = \tilde{g} \circ p_g$, we have $\|\hat{f} - \hat{g}\|_\infty = \delta < 1$. We then observe the following:

- $\hat{g}^{-1}(0) \subseteq \hat{f}^{-1}([-\delta, +\delta])$.
- $\tilde{f}^{-1}([-\delta, +\delta])$ consists of two circular arcs homeomorphic by $\tilde{f}$ to $[-\delta, +\delta]$, and thus, by Proposition 2.7, $\hat{f}^{-1}([-\delta, +\delta])$ consists of two connected components C_+ and C_- as well.
- For both components we have $\hat{f}(C_\pm) = [-\delta, \delta]$, and so $\|\hat{f} - \hat{g}\|_\infty = \delta$ implies that $0 \in \hat{g}(C_\pm)$. Thus $\hat{g}^{-1}(0) \cap C_- \neq \emptyset$ and $\hat{g}^{-1}(0) \cap C_+ \neq \emptyset$.

But since $\hat{g}^{-1}(0) \subseteq C_- \sqcup C_+$, this would contradict the assumption that the fiber $\hat{g}^{-1}(0)$ is connected. ◁

The current example illustrates that the *functional distortion distance* introduced in [2] and the *interleaving distance* introduced in [6] are both stable but fail to be universal. We first recall the definition of the former. For any Reeb graph $(R_f, \tilde{f}), (R_g, \tilde{g})$, consider the metric on R_f given by

$$d_f(x, y) = \inf\{b - a \mid x, y \text{ are in the same connected component of } \tilde{f}^{-1}([a, b])\}.$$

Given maps $\phi : R_f \to R_g$ and $\psi : R_g \to R_f$, we write

$$G(\phi, \psi) = \big\{(p, \phi(p)) : p \in R_f\big\} \cup \big\{(\psi(q), q) : q \in R_g\big\}$$

for the correspondences induced by the two maps, and

$$D(\phi, \psi) = \sup_{(p,q),(p',q') \in G(\phi,\psi)} \frac{1}{2} \left| d_f(p, p') - d_g(q, q') \right|$$

for the metric distortion induced by (ϕ, ψ). The functional distortion distance is then defined as

$$d_{FD}(R_f, R_g) = \inf_{\phi,\psi} (\max\big\{D(\phi, \psi), \|f - g \circ \phi\|_\infty, \|f \circ \psi - g\|_\infty\big\}).$$

To see that neither the functional distortion distance nor the interleaving distance are universal, we establish:

▶ **Proposition 3.11.** $d_I((R_f, \tilde{f}), (R_g, \tilde{g})) \leq d_{FD}((R_f, \tilde{f}), (R_g, \tilde{g})) \leq \frac{1}{2}$.

Proof. By [3, Lemma 8], the functional distortion distance is an upper bound on the interleaving distance on Reeb graphs [6], and so it is enough to prove that $d_{FD}((R_f, \tilde{f}), (R_g, \tilde{g})) \leq \frac{1}{2}$. To this end, consider the maps

$$\phi : R_f \to R_g,\ (x, y) \mapsto x \quad \text{and} \quad \psi : R_g \to R_f,\ t \mapsto \left(t, \sqrt{1 - t^2}\right).$$

For every pair $p, p' \in R_f$ one can verify that

$$|\tilde{f}(p) - \tilde{f}(p')| \leq d_f(p, p') \leq |\tilde{f}(p) - f(p')| + 1,$$

while for every pair $q, q' \in R_g$, we have

$$d_g(q, q') = |\tilde{g}(q) - \tilde{g}(q')|.$$

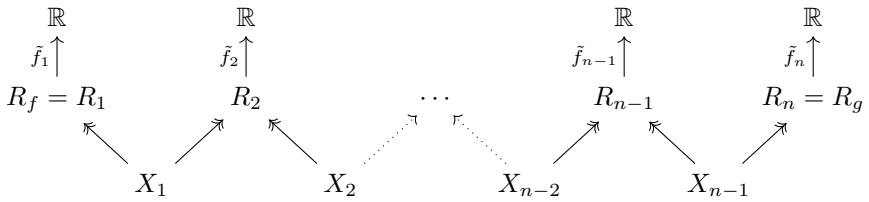
This implies that for any two corresponding pairs $(p, q), (p', q') \in G(\phi, \psi)$, we have

$$|d_f(p, p') - d_g(q, q')| \leq 1,$$

and thus $D(\phi, \psi) \leq \frac{1}{2}$. Both maps preserve function values, so $d_{FD}(R_f, R_g) \leq \frac{1}{2}$. ◀

4 Edit distances

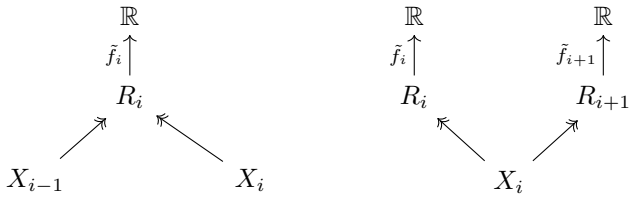
Given a pair of Reeb graphs R_f, R_g, consider a diagram of the form



(1)

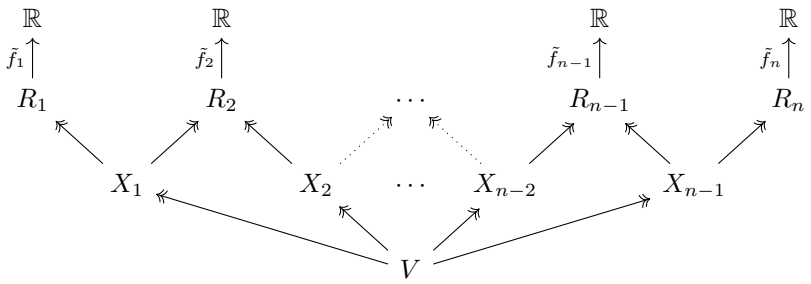
where for $n \in \mathbb{N}$ $\tilde{f}_1, \ldots, \tilde{f}_n$ are Reeb functions with $\tilde{f}_1 = \tilde{f}$ and $\tilde{f}_n = \tilde{g}$, and the maps $X_i \to R_i, R_{i+1}$ for $i = 1, \ldots, n-1$, are Reeb quotient maps. We call the diagram a *Reeb zigzag diagram* between R_f and R_g. Observe that, by Remark 3.2, between any two Reeb graphs R_f and R_g there exists a Reeb zigzag diagram.

A Reeb zigzag diagram can be regarded as being composed of the following elementary diagrams:



This way, we may think of a Reeb zigzag diagram as a sequence of operations transforming the R_f into R_g. The elementary diagram on the left corresponds to an *edit* operation: the space X_{i-1}, together with a function $X_{i-1} \to \mathbb{R}$ with Reeb graph R_i, is transformed to another space X_i, with a function $X_i \to \mathbb{R}$ having the same Reeb graph R_i. The elementary diagram on the right corresponds to a *relabel* operation: the function on X_i with Reeb graph R_i is transformed to another function with Reeb graph R_{i+1}. The idea of edit and relabel operations is inspired by previous work on edit distances for Reeb graphs [7, 1].

In order to define an edit distance using Reeb zigzag diagrams, we need to assign a cost to a given Reeb zigzag diagram between R_f and R_g. To that end, we can consider a cone from a space V by Reeb quotient maps $V \to R_i$:



(2)

We call this diagram a *Reeb cone*. Any Reeb zigzag diagram admits such a cone. Indeed, the limit over the lower part of the diagram (1) can be constructed from iterated pullbacks, and since Reeb quotient maps are stable under pullbacks, the maps in the resulting limit diagram are Reeb quotient maps as well. In a Reeb cone, by commutativity, each of the Reeb functions $\tilde{f}_i$ induces a unique function $f_i : V \to \mathbb{R}$. By Corollary 2.9, the Reeb graph

of f_i is isomorphic to R_i. This way, we pull back the individual functions $\tilde{f}_i$ to functions f_i on a common space with the same Reeb graphs, where they can be compared using the supremum norm.

Using these ideas, we can now introduce distances on Reeb graphs, and proceed to prove that they are stable and universal.

▶ **Definition 4.1.** *Given a Reeb cone from a space V as in* (2)*, we define the* spread *of the functions $(f_i)_{i=1,\dots,n} : V \to \mathbb{R}$, as the function*

$$s^V : V \to \mathbb{R},\ x \mapsto \max_{i=1,\dots,n} f_i(x) - \min_{j=1,\dots,n} f_j(x).$$

Moreover, for a Reeb zigzag diagram Z between R_f and R_g as in (1)*, consider the limit of Z, denoted by L. The* cost *of the Reeb zigzag diagram Z is the supremum norm of the spread s^L,*

$$c_Z := \|s^L\|_\infty = \sup_{x \in L} \left(\max_i f_i(x) - \min_j f_j(x) \right).$$

▶ **Definition 4.2.** *We define the* (PL) edit distance δ_e *between Reeb graphs $(R_f, \tilde{f})$ and $(R_g, \tilde{g})$ as the infimum cost of all Reeb zigzag diagrams Z between R_f and R_g:*

$$\delta_e(R_f, R_g) = \inf_Z c_Z.$$

Moreover, we define the graph edit distance δ_{eGraph} *between Reeb graphs $(R_f, \tilde{f})$ and $(R_g, \tilde{g})$ analogously by restricting the infimum to Reeb zigzag diagrams Z where all the spaces X_i and R_i are finite topological graphs.*

Thus, on Reeb graphs we have two edit distances, satisfying

$$\delta_e \leq \delta_{eGraph}\,. \tag{3}$$

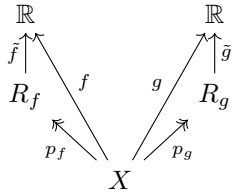
The Reeb graph edit distance δ_{eGraph} is a categorical reformulation of the definition given in [1]. The main goal is to prove that these distances have the stability and universality properties (Propositions 4.4 and 4.5, Theorem 5.6, and Corollary 5.7). As a consequence, whenever applicable, they actually coincide with the canonical universal distance δ_U defined in Definition 3.3:

▶ **Corollary 4.3.** $\delta_U = \delta_e = \delta_{eGraph}\,.$

The proofs of stability and universality for δ_e are straightforward and are given next. The verification of stability and universality for δ_{eGraph} follows in Section 5.

▶ **Proposition 4.4.** *δ_e is a stable distance.*

Proof. Let $(R_f, \tilde{f}), (R_g, \tilde{g})$ be Reeb graphs. For any space X such that there exist two Reeb quotient maps $p_f : X \to R_f$ and $p_g : X \to R_g$, the diagram



is a Reeb zigzag diagram with limit object X. The cost of this Reeb zigzag diagram is exactly $\|f - g\|_\infty$. Hence, $\delta_e((R_f, \tilde{f}), (R_g, \tilde{g})) \leq \|f - g\|_\infty$. ◀

Our proof of universality of the edit distance is similar to previous universality proofs for the bottleneck distance [5] and for the interleaving distance [11].

▶ **Proposition 4.5.** *δ_e is a universal distance.*

Proof. Let $(R_f, \tilde{f}), (R_g, \tilde{g})$ be Reeb graphs with $\delta_e((R_f, \tilde{f}), (R_g, \tilde{g})) = d$. Hence, for any $\varepsilon > 0$, there is a Reeb zigzag diagram Z between $R_f = R_1$ and $R_g = R_n$, with limit L and functions f_i as in Definition 4.1, having cost

$$c_Z = \|s^L\|_\infty = \|\max_i f_i - \min_j f_j\|_\infty \leq d + \varepsilon.$$

Let $p_f : L \to R_f$ and $p_g : L \to R_g$ be the induced Reeb quotient maps. If d_S is any other stable distance (cf. Definition 3.1) between R_f and R_g, we have

$$d_S((R_f, \tilde{f}), (R_g, \tilde{g})) \leq \|\tilde{f} \circ p_f - \tilde{g} \circ p_g\|_\infty \leq \|\max_i f_i - \min_j f_j\|_\infty \leq d + \varepsilon.$$

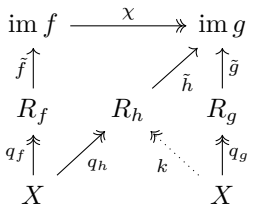
Since the above holds for all $\varepsilon > 0$, we have $d_S((R_f, \tilde{f}), (R_g, \tilde{g})) \leq d = \delta_e((R_f, \tilde{f}), (R_g, \tilde{g}))$. ◀

5 Stability and universality of the Reeb graph edit distance

We now turn to the proof of stability and universality for the Reeb graph edit distance. Recall that, in the case of δ_{eGraph}, the admissible Reeb zigzag diagrams are PL zigzags of finite topological graphs. As mentioned above, the distance δ_{eGraph} is applicable to Reeb graphs of compact triangulable spaces.

▶ **Lemma 5.1.** *Let X be a compact triangulable space, with PL functions $f, g : X \to \mathbb{R}$, simplexwise linear on a triangulation $|K| \cong X$ of X by some simplicial complex K. Let $\chi : \operatorname{im} f \to \operatorname{im} g$ be a weakly monotonic PL surjection such that $\chi \circ f(v) = g(v)$ for every vertex $v \in V$ of K. Then there is a Reeb quotient map $X/\!\sim_f \to X/\!\sim_g$.*

Proof. Without loss of generality, assume $X = |K|$. For simplicity, we write $R_f = X/\!\sim_f$, $R_g = X/\!\sim_g$, and $R_h = X/\!\sim_h$, where $h = \chi \circ f$. Applying Proposition 2.2, f can be factorized as $f = \tilde{f} \circ q_f$, where $q_f : X \to R_f$ is the canonical projection and $\tilde{f} : R_f \to \mathbb{R}$ is a Reeb function. Analogously, we obtain $g = \tilde{g} \circ q_g$ and $h = \tilde{h} \circ q_h$. We show that there is a Reeb quotient map $k : X \to R_h$ making the following diagram commute:



The claim then follows by applying Lemma 2.10 to obtain Reeb quotient maps $R_f \to R_h$ and $R_h \to R_g$, which compose to the desired map $R_f \to R_g$.

In order to prove the existence of such a Reeb quotient map k, we define the relation

$$k = q_h \circ ((h^{-1} \circ g) \cap \mathrm{st}_K)$$

on $X \times R_h$. Here st_K denotes the open star on $X = |K|$, defined as

$$\mathrm{st}_K(x) = \{y \in X \mid \sigma \in K, y \in \sigma^\circ, x \in \sigma\},$$

where σ° is the interior of the simplex σ. Note that the converse relation to the open star is the *(closed) carrier*, $\mathrm{st}_K^{-1} = \mathrm{carr}_K$, where $\mathrm{carr}_K(A)$ is the underlying space of the smallest subcomplex of K containing $A \subseteq X$. We will also use the *open carrier* relation carr_K°, where $\mathrm{carr}_K^\circ(A)$ is the smallest union of open simplices of K covering A. Note that the open carrier relation is symmetric, i.e., $(\mathrm{carr}_K^\circ)^{-1} = \mathrm{carr}_K^\circ$. Moreover, we have $\mathrm{carr}_K^\circ \subseteq \mathrm{st}_K$.

The remainder of the proof is split into several lemmas. Lemma 5.2 describes the behaviour of the functions h and g on the simplices of K. Lemma 5.3 shows that k is a continuous surjection, and Lemma 5.4 shows that k has connected fibers. Since $\tilde{h} \circ k = g$, we conclude that k is PL. Thus, k is a Reeb quotient map, and the claim follows from Lemma 2.10. ◀

▶ **Lemma 5.2.** *For every simplex σ in K, $g(\sigma) = h(\sigma)$ and $g(\sigma^\circ) \subseteq h(\sigma^\circ)$.*

Proof. We have $h(\sigma) = g(\sigma)$ because h is equal to g on the vertices of K, and $h = \chi \circ f$ with f linear on σ and χ a weakly monotonic surjection.

To show that $g(\sigma^\circ) \subseteq h(\sigma^\circ)$, note that since g is linear on σ, either g is constant on σ and so $g(\sigma^\circ) = g(\sigma) = h(\sigma)$, or $g(\sigma^\circ) = (g(v), g(w))$ for some vertices v, w of σ. In the latter case, since h and g coincide on the vertices, we have $g(\sigma^\circ) = g(\sigma)^\circ = h(\sigma)^\circ$. Finally, since $h(\sigma^\circ) \subseteq h(\sigma) \subseteq \overline{h(\sigma^\circ)}$ are nested intervals, we have $h(\sigma)^\circ \subseteq h(\sigma^\circ)$ and the claim follows. ◀

▶ **Lemma 5.3.** *k is a continuous surjection.*

Proof. Recall that the relation $k \subseteq X \times R_h$ is a *partial set map* if for any $x \in X$ and $y, y' \in k(x)$, we have $y = y'$. Moreover, a partial set map k is a (total) set map if for every $x \in X$, $k(x) \neq \emptyset$. Finally, a set map k is a surjection if for every $y \in R_h$, there is some $x \in k^{-1}(y)$.

We first show that k is a partial set map, i.e., for any $x \in X$ and $y, y' \in k(x)$, we have $y = y'$. To see this, let $t = g(x)$ and note that $\tilde{h}(y) = \tilde{h}(y') = t$. Let $\sigma \in K$ be such that $x \in \sigma^\circ$. By Lemma 5.2 there is a point $\zeta \in \sigma^\circ$ with $h(\zeta) = g(x) = t$; in particular,

$$\zeta \in h^{-1}(t) \cap \mathrm{st}_K(x).$$

Furthermore, there are points $\xi, \xi' \in h^{-1}(t) \cap \mathrm{st}_K(x)$ with $\xi \in q_h^{-1}(y)$ and $\xi' \in q_h^{-1}(y')$. But since $h^{-1}(t) \cap \tau$ is necessarily connected for every simplex τ, we know that ζ lies in the same connected component of $h^{-1}(t) \cap \mathrm{st}_K(x)$ as both ξ and ξ', and so we have $y = q_h(\xi) = q_h(\xi') = y'$ as claimed.

To show that k is a set map, we need to show that for every $x \in X$, $k(x) \neq \emptyset$. It suffices to show that for every $x \in X$, $\mathrm{st}_K(x)$ contains a point x' with $h(x') = g(x)$. This follows by considering the simplex $\sigma \in K$ with $x \in \sigma^\circ$. Now by Lemma 5.2, there is a point $x' \in \sigma^\circ \subseteq \mathrm{st}_K(x)$ with $h(x') = g(x)$ as claimed.

To show that k is surjective, we show that for every $y \in R_h$, there is some

$$x \in k^{-1}(y) = (\mathrm{carr}_K \circ q_h^{-1})(y) \cap (g^{-1} \circ \tilde{h})(y),$$

or equivalently, there is some $x \in \mathrm{carr}_K \circ q_h^{-1}(y)$ such that $g(x) = \tilde{h}(y)$. If $q_h^{-1}(y)$ contains some vertex v of K, choose $x = v$. Otherwise, let $\xi \in q_h^{-1}(y)$, and let $\sigma \in K$ be such that $\xi \in \sigma^\circ$. Now by Lemma 5.2 there is a point $x \in \sigma \subseteq \mathrm{carr}_K \circ q_h^{-1}(y)$ with $g(x) = h(\xi) = \tilde{h}(y)$.

Finally, to show that k is continuous, we show that for every closed subset L of R_h, the preimage $k^{-1}(L)$ is closed. Since $k^{-1} = (\mathrm{carr}_K \circ q_h^{-1}) \cap (g^{-1} \circ \tilde{h})$, it is sufficient to show that both $\mathrm{carr}_K \circ q_h^{-1}(L)$ and $g^{-1} \circ \tilde{h}(L)$ are closed in X. First note that $\mathrm{carr}_K \circ q_h^{-1}(L)$ is closed as a subcomplex of K. Furthermore, the image $\tilde{h}(L)$ is closed by the closed map lemma. By continuity of g it follows that $g^{-1} \circ \tilde{h}(L)$ is closed in X. ◀

▶ **Lemma 5.4.** *The fibers of k are connected.*

Proof. Let $y \in R_h$ be a point in the Reeb graph with value $t = \tilde{h}(y)$, and $C = q_h^{-1}(y) \subseteq h^{-1}(t)$ the corresponding component of the level set of h. Let $U = \mathrm{carr}_K(C)$, and let L be the corresponding subcomplex of K. Writing $D = k^{-1}(y)$, we have $C = U \cap h^{-1}(t)$ and $D = U \cap g^{-1}(t)$. To prove that D is connected, it is sufficient to show that C and D have finite closed covers with isomorphic nerves; since C is connected, both nerves and hence also D are then connected too.

The cover of C is given by $\{\sigma \cap C \mid \sigma \in L\}$, and similarly the cover of D is $\{\sigma \cap D \mid \sigma \in L\}$. Observe that any two cover elements of C, say $\sigma \cap C$ and $\tau \cap C$, have a nonempty intersection $(\sigma \cap C) \cap (\tau \cap C) = (\sigma \cap \tau) \cap C$ if and only if $t \in h(\sigma \cap \tau)$. Similarly, $\sigma \cap D$ and $\tau \cap D$ have nonempty intersection if and only if $t \in g(\sigma \cap \tau)$. But $g(\sigma \cap \tau) = h(\sigma \cap \tau)$ by Lemma 5.2, and so the nerves of both covers are isomorphic as claimed. ◀

We thus have shown the existence of the Reeb quotient map k. This completes the proof of Lemma 5.1. We will now apply Lemma 5.1 to construct Reeb graph edit zigzags from straight line homotopies.

▶ **Lemma 5.5.** *Let X be a compact triangulable space, with PL functions $f, g : X \to \mathbb{R}$, simplexwise linear on a triangulation $|K| \cong X$. Consider the straight line homotopy $f_\lambda = (1-\lambda)f + \lambda g$, with $0 \le \lambda \le 1$. Then there exists a partition $0 = \lambda_1 < \cdots < \lambda_n = 1$ such that for every $1 \le i < n$ and $\rho \in (\lambda_i, \lambda_{i+1})$, there exist weakly monotonic PL surjections $\chi_i : \mathrm{im}\, f_\rho \to \mathrm{im}\, f_{\lambda_i}$ and $\xi_{i+1} : \mathrm{im}\, f_\rho \to \mathrm{im}\, f_{\lambda_{i+1}}$ with*

$$\chi_i \circ f_\rho(v) = f_{\lambda_i}(v) \quad \text{and} \quad \xi_{i+1} \circ f_\rho(v) = f_{\lambda_{i+1}}(v)$$

for every vertex v of K.

Proof. Consider the set of values $0 < \lambda < 1$ such that there exist vertices $v, w \in K$ with

$$f_\lambda(v) = f_\lambda(w), \quad \text{but} \quad f_\rho(v) \neq f_\rho(w) \quad \text{for every} \quad \rho \neq \lambda.$$

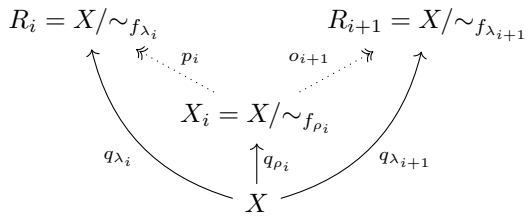
This set is finite because the function $\lambda \mapsto f_\lambda(v) - f_\lambda(w)$ is linear and K has a finite number of vertices. Let $\{\lambda_i\}_{1 \le i \le n}$ be this set together with 0 and 1, indexed in ascending order. By the linearity of f_λ with respect to the parameter λ, we also see that the order induced by f_ρ on the vertices is the same for every $\rho \in (\lambda_i, \lambda_{i+1})$. Indeed, if there exist two distinct vertices v, w of K such that $f_\rho(v) = f_\rho(w)$ for some $\rho \in (\lambda_i, \lambda_{i+1})$, then $f_\lambda(v) = f_\lambda(w)$ for every $\lambda \in [0, 1]$. By continuity, the order is still weakly preserved along $[\lambda_i, \lambda_{i+1}]$.

Therefore, the function $f_\rho(v) \mapsto f_{\lambda_i}(v)$ is well-defined and can be extended to a piecewise linear function χ_i satisfying the claim. The function ξ_{i+1} can be defined similarly. ◀

▶ **Theorem 5.6.** *δ_{eGraph} is a stable distance.*

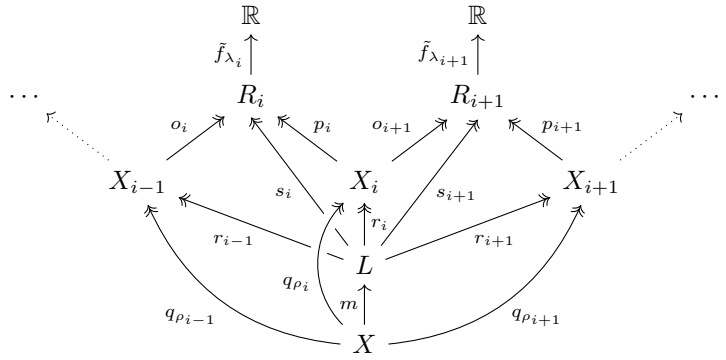
Proof. Let $X \cong |K|$ be a compact triangulable space with $f, g : X \to \mathbb{R}$ be PL functions, simplexwise linear on K; without loss of generality, assume $X = |K|$. Consider the straight line homotopy $f_\lambda = (1-\lambda)f + \lambda g$, with $0 \le \lambda \le 1$, and take values $\lambda_i \in [0, 1]$, $1 \le i \le n$, as in Lemma 5.5. Set $\rho_i = (\lambda_i + \lambda_{i+1})/2$.

We first define a Reeb cone of the form (2), with $V = X$, $R_i = X/\!\sim_{f_{\lambda_i}}$, $i = 1, \ldots, n$, and $X_i = X/\!\sim_{f_{\rho_i}}$, $i = 1, \ldots, n-1$. The canonical projections $q_{\rho_i} : X \to X_i$ and $q_{\lambda_i} : X \to R_i$ are Reeb quotient maps, and the Reeb functions $R_i \to \mathbb{R}$ are induced by f_{λ_i} as in Proposition 2.2. To complete the construction, we show that there are Reeb quotient maps $p_i : X/\!\sim_{f_{\rho_i}} \to X/\!\sim_{f_{\lambda_i}}$ and $o_{i+1} : X/\!\sim_{f_{\rho_i}} \to X/\!\sim_{f_{\lambda_{i+1}}}$ that make the following diagram commute:



We prove the existence of p_i, that of o_{i+1} being analogous. By Lemma 5.5, there is a weakly monotonic PL surjection $\chi_i : \operatorname{im} f_{\rho_i} \to \operatorname{im} f_{\lambda_i}$ such that $\chi_i \circ f_{\rho_i} = f_{\lambda_i}$. Hence, Lemma 5.1 provides the desired Reeb quotient map $p_i : X/\!\sim_{f_{\rho_i}} \to X/\!\sim_{f_{\lambda_i}}$.

Now consider the limit L over the resulting Reeb zigzag diagram Z consisting of the maps p_i and o_i, with maps $r_i : L \to X_i$ and $s_i : L \to R_i$. Since the maps from X in the above Reeb cone factor through a unique map $m\colon X \to L$ by the universal property of the limit, we obtain the commutative diagram



We have $f_{\lambda_i} = f^L_{\lambda_i} \circ m$ for $1 \le i \le n$, with $f^L_{\lambda_i} = \tilde{f}_{\lambda_i} \circ s_i$. Hence, for every $\ell \in L$,

$$s^L(\ell) = \max_j f^L_{\lambda_j}(\ell) - \min_k f^L_{\lambda_k}(\ell) \le \sum_{i=1}^{n-1} |f^L_{\lambda_{i+1}}(\ell) - f^L_{\lambda_i}(\ell)|.$$

By the surjectivity of q_{ρ_i}, for every i there is $x_{\ell,i} \in X$ such that $q_{\rho_i}(x_{\ell,i}) = r_i(\ell)$. Thus,

$$|f^L_{\lambda_{i+1}}(\ell) - f^L_{\lambda_i}(\ell)| = |f_{\lambda_{i+1}}(x_{\ell,i}) - f_{\lambda_i}(x_{\ell,i})| \le (\lambda_{i+1} - \lambda_i) \cdot \|f - g\|_\infty.$$

Together, for every $\ell \in L$ we have

$$s^L(\ell) \le \sum_{i=1}^{n-1} (\lambda_{i+1} - \lambda_i) \cdot \|f - g\|_\infty = \|f - g\|_\infty.$$

We conclude that

$$\delta_e(R_f, R_g) \le c_Z = \|s^L\|_\infty \le \|f - g\|_\infty,$$

showing that δ_e is a stable distance. ◀

▶ **Corollary 5.7.** *$\delta_{eGraph} = \delta_U$ is the universal distance.*

Proof. The claim is a direct consequence of inequality (3) together with Theorem 5.6 and Propositions 4.4 and 4.5. ◀

6 Discussion

We believe that the following questions are of interest and could motivate further research:

- *Do minimizers in the definition of the universal distance always exist?* This would have algorithmic implications. See below.
- *Is the interleaving distance [6] bi-Lipschitz equivalent to the universal distance?* If the answer to this question is affirmative, then by results of [3], one would obtain the bi-Lipschitz equivalence between the universal distance and the functional distortion distance from [2].
- *What is the computational complexity of the universal distance?* This problem is at least graph-isomorphism hard, which can be seen as follows. First note that bipartite graphs form a graph-isomorphism complete class of graphs. Any bipartite simple graph can be interpreted as a Reeb graph with function values in $\{0, 1\}$ corresponding to the partition of the vertex set. Using Corollary 3.6, these Reeb graphs are at universal distance 0 if and only if the bipartite graphs are isomorphic, so both of these decision problems are graph-isomorphism complete. A similar observation has been made for the interleaving distance [6].

  These considerations motivate the following two ancillary questions:
  - Is the universal distance a minimum over a certain finite set, possibly of cardinality polynomial in the size of the input Reeb graphs?
  - More generally, are the possible values of the universal distance always contained in some canonical set of values, constructed from the sets of vertex function values of the two Reeb graphs? Related results in the context of manifolds endowed with Morse functions are in the work of Donatini and Frosini [8]. This work carries over to the setting of Reeb graphs by the results of [7].
- *How do the theoretical properties of the universal distance extend to more genreal settings?*
  - The definition of the universal distance also makes sense in a more general topological setting, where we consider locally compact Hausdorff spaces as Reeb domains and proper quotient maps with connected fibers as Reeb quotient maps. The distance one obtains in this larger category can still be applied to finite Reeb graphs, in which case it will be smaller or equal to the PL universal distance that we described in this paper. However, we conjecture that in this case the two distances actually coincide.
  - *Reeb spaces:* Generalizing our definitions and results up to Section 5 to Reeb spaces of piecewise linear maps $X \to \mathbb{R}^n$ is straightforward. Do our results of Section 5 generalize as well?

References

1 Ulrich Bauer, Barbara Di Fabio, and Claudia Landi. An Edit Distance for Reeb Graphs. In *Eurographics Workshop on 3D Object Retrieval*. The Eurographics Association, 2016. doi:10.2312/3dor.20161084.

2 Ulrich Bauer, Xiaoyin Ge, and Yusu Wang. Measuring distance between Reeb graphs. In *Proceedings of the Thirtieth Annual Symposium on Computational Geometry*, SoCG'14, New York, NY, USA, 2014. ACM. doi:10.1145/2582112.2582169.

3 Ulrich Bauer, Elizabeth Munch, and Yusu Wang. Strong equivalence of the interleaving and functional distortion metrics for Reeb graphs. In *31st International Symposium on Computational Geometry (SoCG 2015)*, Leibniz International Proceedings in Informatics (LIPIcs), pages 461–475, Dagstuhl, Germany, 2015. doi:10.4230/LIPIcs.SOCG.2015.461.

4 David Cohen-Steiner, Herbert Edelsbrunner, and John Harer. Stability of persistence diagrams. *Discrete Comput. Geom.*, 37(1):103–120, 2007. doi:10.1007/s00454-006-1276-5.

5 Michele d'Amico, Patrizio Frosini, and Claudia Landi. Natural pseudo-distance and optimal matching between reduced size functions. *Acta Applicandae Mathematicae*, 109(2):527–554, 2010. doi:10.1007/s10440-008-9332-1.

6 Vin de Silva, Elizabeth Munch, and Amit Patel. Categorified Reeb graphs. *Discrete & Computational Geometry*, 55(4):854–906, 2016. doi:10.1007/s00454-016-9763-9.

7 Barbara Di Fabio and Claudia Landi. The edit distance for Reeb graphs of surfaces. *Discrete & Computational Geometry*, 55(2):423–461, 2016. doi:10.1007/s00454-016-9758-6.

8 Pietro Donatini and Patrizio Frosini. Natural pseudodistances between closed manifolds. *Forum Math.*, 16(5):695–715, 2004. doi:10.1515/form.2004.032.

9 Barbara Di Fabio and Claudia Landi. Reeb graphs of curves are stable under function perturbations. *Mathematical Methods in the Applied Sciences*, 35(12):1456–1471, 2012. doi:10.1002/mma.2533.

10 Masaki Hilaga, Yoshihisa Shinagawa, Taku Kohmura, and Tosiyasu L. Kunii. Topology matching for fully automatic similarity estimation of 3D shapes. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '01, pages 203–212. ACM Press, 2001. doi:10.1145/383259.383282.

11 Michael Lesnick. The theory of the interleaving distance on multidimensional persistence modules. *Foundations of Computational Mathematics*, 15(3):613–650, 2015. doi:10.1007/s10208-015-9255-y.

12 Facundo Mémoli. A distance between filtered spaces via tripods. Preprint, 2017. arXiv:1704.03965.

13 Georges Reeb. Sur les points singuliers d'une forme de Pfaff complétement intégrable ou d'une fonction numérique. *Comptes Rendus de L'Académie des Sciences*, 222:847–849, 1946.

14 Yoshihisa Shinagawa and Tosiyasu L. Kunii. Constructing a Reeb graph automatically from cross sections. *IEEE Computer Graphics and Applications*, 11(6):44–51, 1991. doi:10.1109/38.103393.

15 Gurjeet Singh, Facundo Mémoli, and Gunnar Carlsson. Topological Methods for the Analysis of High Dimensional Data Sets and 3D Object Recognition. In *Eurographics Symposium on Point-Based Graphics*. The Eurographics Association, 2007. doi:10.2312/SPBG/SPBG07/091-100.

16 John R. Stallings. *Brick's Quasi Simple Filtrations and 3-Manifolds*, volume 188 of *London Mathematical Society Lecture Note Series*, page 188–203. Cambridge University Press, 1993. doi:10.1017/CBO9780511661860.017.

Book Embeddings of Nonplanar Graphs with Small Faces in Few Pages

Michael A. Bekos
Department of Computer Science, University of Tübingen, Germany
bekos@informatik.uni-tuebingen.de

Giordano Da Lozzo
Department of Engineering, Roma Tre University, Rome, Italy
giordano.dalozzo@uniroma3.it

Svenja M. Griesbach
Department of Mathematics and Computer Science, University of Cologne, Germany
sgriesba@smail.uni-koeln.de

Martin Gronemann
Department of Mathematics and Computer Science, University of Cologne, Germany
gronemann@informatik.uni-koeln.de

Fabrizio Montecchiani
Department of Engineering, University of Perugia, Italy
fabrizio.montecchiani@unipg.it

Chrysanthi Raftopoulou
School of Applied Mathematical & Physical Sciences, NTUA, Athens, Greece
crisraft@mail.ntua.gr

Abstract

An embedding of a graph in a book, called *book embedding*, consists of a linear ordering of its vertices along the spine of the book and an assignment of its edges to the pages of the book, so that no two edges on the same page cross. The *book thickness* of a graph is the minimum number of pages over all its book embeddings. For planar graphs, a fundamental result is due to Yannakakis, who proposed an algorithm to compute embeddings of planar graphs in books with four pages. Our main contribution is a technique that generalizes this result to a much wider family of nonplanar graphs, which is characterized by a biconnected skeleton of crossing-free edges whose faces have bounded degree. Notably, this family includes all 1-planar and all optimal 2-planar graphs as subgraphs. We prove that this family of graphs has bounded book thickness, and as a corollary, we obtain the first constant upper bound for the book thickness of optimal 2-planar graphs.

2012 ACM Subject Classification Theory of computation → Computational geometry; Mathematics of computing → Graph algorithms

Keywords and phrases Book embeddings, Book thickness, Nonplanar graphs, Planar skeleton

Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.16

Related Version A full version of this paper is available at `https://arxiv.org/abs/2003.07655`.

Funding *Michael A. Bekos*: Partially supported by DFG grant KA812/18-1.
Giordano Da Lozzo: Partially supported by MSCA-RISE project "CONNECT", N° 734922, and by MIUR, grant 20174LF3T8 "AHeAD: efficient Algorithms for HArnessing networked Data".
Fabrizio Montecchiani: Partially supported by MIUR, grant 20174LF3T8 "AHeAD: efficient Algorithms for HArnessing networked Data", and by Dipartimento di Ingegneria, Università degli studi di Perugia, grant RICBA19FM: "Modelli, algoritmi e sistemi per la visualizzazione di grafi e reti".

Acknowledgements This work began at the Dagstuhl Seminar 19092 "Beyond-Planar Graphs: Combinatorics, Models and Algorithms" (February 24 - March 1, 2019).

36th International Symposium on Computational Geometry (SoCG 2020).
Editors: Sergio Cabello and Danny Z. Chen; Article No. 16; pp. 16:1–16:17

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

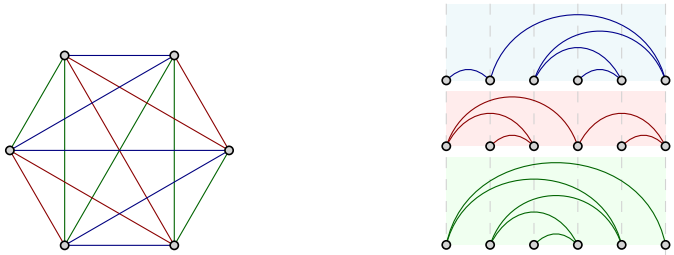


Figure 1 Graph K_6 and a book embedding of it with the minimum of three pages.

1 Introduction

Book embeddings of graphs form a well-known topic in topological graph theory that has been a fruitful subject of intense research over the years, with seminal results dating back to the 70s [38]. In a *book embedding* of a graph G, the vertices of G are restricted to a line, called the *spine* of the book, and the edges of G are assigned to different half-planes delimited by the spine, called *pages* of the book. From a combinatorial point of view, computing a book embedding of a graph corresponds to finding a *linear ordering* of its vertices and a partition of its edges, such that no two edges in the same part cross; see Fig. 1. The *book thickness* (also known as *stack number* or *page number*) of a graph is the minimum number of pages required by any of its book embeddings, while the *book thickness* of a family of graphs $\mathcal{G}$ is the maximum book thickness of any graph $G \in \mathcal{G}$.

Book embeddings were originally motivated by the design of VLSI circuits [14, 42], but they also find applications, among others, in sorting permutations [39, 43], compact graph encodings [30, 35], graph drawing [9, 10, 45], and computational origami [1]; for a more complete list, we point the reader to [22]. Unfortunately, determining the book thickness of a graph turns out to be an NP-complete problem even for maximal planar graphs [44]. This negative result has motivated a large body of research devoted to the study of upper bounds on the book thickness of meaningful graph families.

In this direction, there is a very rich literature concerning planar graphs. The most notable result is due to Yannakakis, who back in 1986 exploited a peeling-into-levels technique (a flavor of it is given in Section 2) to prove that the book thickness of any planar graph is at most 4 [46, 47], improving uppon a series of previous results [13, 27, 29]. Even though it is not yet known whether the book thickness of planar graphs is 3 or 4, there exist several improved bounds for particular subfamilies of planar graphs. Bernhart and Kainen [8] showed that the book thickness of a graph G is 1 if and only if G is outerplanar, while its book thickness is at most 2 if and only if G is *subhamiltonian*, that is, G is a subgraph of a Hamiltonian planar graph. In particular, several subfamilies of planar graphs are known to be subhamiltonian, e.g., 4-connected planar graphs [37], planar graphs without separating triangles [31], Halin graphs [15], series-parallel graphs [40], bipartite planar graphs [17], planar graphs of maximum degree 4 [6], triconnected planar graphs of maximum degree 5 [28], and maximal planar graphs of maximum degree 6 [24]. In this plethora of results, we should also mention that planar 3-trees have book thickness 3 [27] and that general (i.e., not necessarily triconnected) planar graphs of maximum degree 5 have book thickness at most 3 [26].

In contrast to the planar case, there exist far fewer results for non-planar graphs. Bernhart and Kainen first observed that the book thickness of a graph can be linear in the number of its vertices; for instance, the book thickness of the complete graph K_n is $\lceil n/2 \rceil$ [8]. Improved bounds are usually obtained by meta-theorems exploiting standard parameters of

the graph. In particular, Malitz proved that if a graph has m edges, then its book thickness is $O(\sqrt{m})$ [34], while if its genus is g, then its book thickness is $O(\sqrt{g})$ [33]. Also, Dujmovic and Wood [23] showed that if a graph has treewidth w, then its book thickness is at most $w + 1$, improving an earlier linear bound by Ganley and Heath [25]. It is also known that all graphs belonging to a minor-closed family have bounded book thickness [11], while the other direction is not necessarily true. As a matter of fact, the family of 1-planar graphs is not closed under taking minors [36], but it has bounded book thickness [3, 4]. We recall that a graph is *h-planar* (with $h \geq 0$), if it can be drawn in the plane such that each edge is crossed at most h times; see, e.g., [19, 32] for recent surveys.

Notably, the approaches presented in [3, 4] form the first non-trivial extensions of the above mentioned peeling-into-levels technique by Yannakakis [46, 47] to graphs that are not planar. Both approaches exploit an important property of 3-connected 1-planar graphs, namely, they can be augmented and drawn so that all pairs of crossing edges are "caged" in the interior of degree-4 faces of a *planar skeleton*, i.e., the graph consisting of all vertices and of all crossing-free edges of the drawing [41]. A similar property also holds for the optimal 2-planar graphs. Each graph in this family admits a drawing whose planar skeleton is simple, biconnected, and has only degree 5 faces, each containing five crossing edges [7]. The book thickness of these graphs, however, has not been studied yet; the best-known upper bound of $O(\log n)$ is derived from the corresponding one for general h-planar graphs [21].

Our contribution. We present a technique that further generalizes the result by Yannakakis to a much wider family of non-planar graphs, called partial k-framed graphs, which is general enough to include all 1-planar graphs and all optimal 2-planar graphs. A graph is *k-framed*, if it admits a drawing having a simple biconnected planar skeleton, whose faces have degree at most $k \geq 3$, and whose crossing edges are in the interiors of these faces. A *partial k-framed* graph is a subgraph of a k-framed graph. Clearly, the book thickness of partial k-framed graphs is lower bounded by $\lceil k/2 \rceil$, as they may contain cliques of size k [8]. In this work, we present an upper bound on the book thickness of partial k-framed graphs that depends linearly *only* on k (but not on n). Our main result is as follows.

▶ **Theorem 1.** *The book thickness of a partial k-framed graph is at most $6\lceil \frac{k}{2} \rceil + 5$.*

Note that the partial 3-framed graphs are exactly the (simple) planar graphs. Also, it is known that 3-connected 1-planar graphs are partial 4-framed [2], while general 1-planar graphs can be augmented to 8-framed. Hence, Theorem 1 implies constant upper bounds for the book thickness of these families of graphs. Since optimal 2-planar graphs are 5-framed, the next corollary guarantees the first constant upper bound on the book thickness of this family.

▶ **Corollary 2.** *The book thickness of an optimal 2-planar graph is at most 23.*

More in general, each partial k-framed graph is h-planar for $h = (\frac{k-2}{2})^2$, and hence for this family of h-planar graphs we prove that the book thickness is $O(\sqrt{h})$, while the best-known upper bound for general h-planar graphs is $O(h \log n)$ [21].

Preliminaries. We assume familiarity with basic graph-theoretic [20] and graph-drawing [18] concepts. Let Γ be a drawing of a graph G. The *planar skeleton* $\sigma(G)$ of G in Γ is the plane subgraph of G induced by the crossing-free edges of G in Γ (where the embedding of $\sigma(G)$ is the one induced by Γ). The edges of $\sigma(G)$ are *crossing-free*, while the edges that belong to G but not to $\sigma(G)$ are *crossing* edges. A *k-framed drawing* of a graph is one such that its crossing-free edges determine a planar skeleton, which is simple, biconnected, spans all

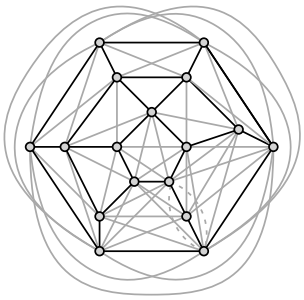


Figure 2 A drawing of a 6-framed graph, whose crossing-free (crossing) edges are black (gray).

the vertices, and has faces of degree at most $k \geq 3$. A graph is *k-framed*, if it admits a k-framed drawing; see Fig. 2. A *partial k-framed* graph is a subgraph of a k-framed graph. Clearly, if a k-framed graph has book thickness at most b, then the book thickness of any of its subgraphs is at most b. Thus, we will only consider k-framed graphs. Further, w.l.o.g., we will also assume that each pair of vertices that belongs to a face f of $\sigma(G)$ is connected either by a crossing-free edge (on the boundary of f) or by a crossing edge (drawn inside f). In other words, the vertices on the boundary of f induce a clique of size at most k. Under this assumption, graph G may contain parallel crossing edges connecting the same pair of vertices, but drawn in the interior of different faces of $\sigma(G)$; see, e.g., the dashed edges of Fig. 2.

2 Proof of Theorem 1

Our approach adopts some ideas from the seminal work by Yannakakis on embeddings of planar graphs in books with five pages [47], not four. The main challenges of our generalization are posed by the crossing edges and by the fact that we cannot augment the input graph so that its underlying planar skeleton is internally-triangulated. Our technique is based on the so-called *peeling-into-levels* decomposition. Let G be an n-vertex k-framed graph with a k-framed drawing Γ. We classify the vertices of G as follows: (i) vertices on the unbounded face of $\sigma(G)$ are at level 0, and (ii) vertices that are on the unbounded face of the subgraph of $\sigma(G)$ obtained by deleting all vertices of levels $\leq i-1$ are at level i $(0 < i < n)$; see, e.g., Fig. 3. Denote by $\sigma_i(G)$ the subgraph of $\sigma(G)$ induced by the vertices of L_i. Observe that $\sigma_i(G)$ is outerplane, but not necessarily connected. Next, we consider $\sigma_i(G)$ and delete any edge that is not incident to the unbounded face. The resulting spanning subgraph of $\sigma_i(G)$ is denoted by $C_i(G)$. By definition, each connected component of $C_i(G)$ is a cactus. Also, the only edges that belong to $\sigma_i(G)$ but not to $C_i(G)$ are the chords of $\sigma_i(G)$. Finally, we denote by G_i the subgraph of G induced by the vertices of $L_0 \cup \ldots \cup L_i$ containing neither chords of $\sigma_i(G)$ nor the crossing edges that are in the interior of the unbounded face of $\sigma(G)$.

Consider an edge e of $\sigma(G)$. If the endpoints of e are assigned to the same level, e is a *level* edge; otherwise, e connects vertices of consecutive levels and is called a *binding* edge; see Fig. 3. By the definition of the level-partition, there is no edge $e \in E$, that connects two vertices of levels i and j, such that $|i-j| > 1$. Another consequence of the level-partition is that any vertex of level $i+1$ lies in the interior of a cycle of level i. Next, we give a characterization for bounded faces of $\sigma(G)$. A bounded face of $\sigma(G)$ is an *intra-level face* of $\sigma_i(G)$ if it is incident to at least one vertex of L_{i-1} but to no vertex of L_{i-2}. We denote by $\mathcal{F}_i$ the set of all the intra-level faces of $\sigma_i(G)$. By definition, the unbounded face of $\sigma_i(G)$ is not an intra-level face. Each intra-level face of $\sigma_i(G)$ has either at least one binding edge between L_{i-1} and L_i on its boundary, or it consists exclusively of L_{i-1}-level edges.

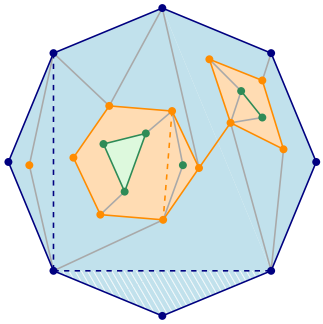


Figure 3 The peeling-into-levels decomposition of an 8-framed graph without its crossing edges. The vertices and level-edges of level L_0 (L_1; L_2, resp.) are blue (orange; green, resp.) and induce $\sigma_0(G)$ ($\sigma_1(G)$; $\sigma_2(G)$, resp.). Chords are drawn dashed; binding edges are drawn gray. The blue (orange; green, resp.) faces are the intra-level faces of $\sigma_1(G)$ ($\sigma_2(G)$; $\sigma_3(G)$, resp.). Graph $\sigma_0(G)$ ($\sigma_1(G)$; $\sigma_2(G)$, resp.) without the dashed chords forms $C_0(G)$ ($C_1(G)$; $C_2(G)$, resp.). The striped blue face is an intra-level face of $\sigma_1(G)$, whose boundary exists exclusively of L_0-level edges.

At a high level, we will inductively compute a book embedding of G_{i+1}, assuming that we have already computed a book embedding of G_i. For this inductive strategy to work, the computed book embeddings satisfy particular invariants, which we define subsequently. We first focus on the base case, in which G consists of only two levels L_0 and L_1 under some additional assumptions (see Section 2.1). Afterwards, we consider the inductive case, in which G consists of more than two levels (see Section 2.2).

2.1 Base case: two-level instances

A *two-level instance* is a k-framed graph G consisting of two levels L_0 and L_1, such that there is no crossing edge in the unbounded face of $\sigma_0(G)$, and either $L_1 = \emptyset$ or $\sigma_1(G) = C_1(G)$, i.e., $\sigma_1(G)$ is chord-less; refer to Fig. 4 of a two-level instance (see Fig. 4). Since $\sigma(G)$ is biconnected, $C_0(G)$ is a simple cycle. Let $u_0, u_1, \ldots, u_{s-1}$ with $s \geq 3$ be the vertices of L_0 in the order they appear in a clockwise traversal of $C_0(G)$ starting from u_0. An edge (u_i, u_j) of $\sigma_0(G)$ is *short* if $i - j = \pm 1$; otherwise it is *long*. By definition, (u_0, u_{s-1}) is long. In the following we will refer to the intra-level faces of $\sigma_1(G)$ simply as intra-level faces, and we will further denote $\mathcal{F}_1$ as $\mathcal{F}$. Consider now the graph $C_1(G)$. Each of its connected components is a cactus; thus, its biconnected components, called *blocks*, are either single edges or simple cycles (that are chordless, as $\sigma_1(G) = C_1(G)$). A connected component of $C_1(G)$ may degenerate into a single vertex, and this vertex itself is a *degenerate* block. A block that consists of more than one vertex is called *non-degenerate*. We equip $\mathcal{F}$ with a linear ordering $\lambda(\mathcal{F})$ as follows. For $i = 0, \ldots, s-1$, the intra-level faces incident to vertex u_i are appended to $\lambda(\mathcal{F})$ as they appear in counterclockwise order around u_i starting from the one incident to (u_{i-1}, u_i) and ending at the one incident to (u_i, u_{i+1}) (indices taken modulo s), unless already present. For a pair of intra-level faces f and f', we write $f \prec_\lambda f'$ if f precedes f' in $\lambda(\mathcal{F})$; similarly, we write $f \preceq_\lambda f'$ if $f = f'$ or $f \prec_\lambda f'$.

Let $C_1, \ldots, C_\gamma$ be the connected components of $C_1(G)$ and let $C \in \{C_1, \ldots, C_\gamma\}$. In general, several intra-level faces in $\mathcal{F}$ may contain vertices of C on their boundary. Let f_C be the first face in the ordering $\lambda(\mathcal{F})$ that contains a vertex of C. Consider now a counterclockwise traversal of the boundary of f_C starting from the vertex of L_0 with the smallest subscript that belongs to f_C. We refer to the vertex, say v_C, of C that is encountered first in this traversal as the *first vertex* of C. Observe that, by definition, v_C is incident

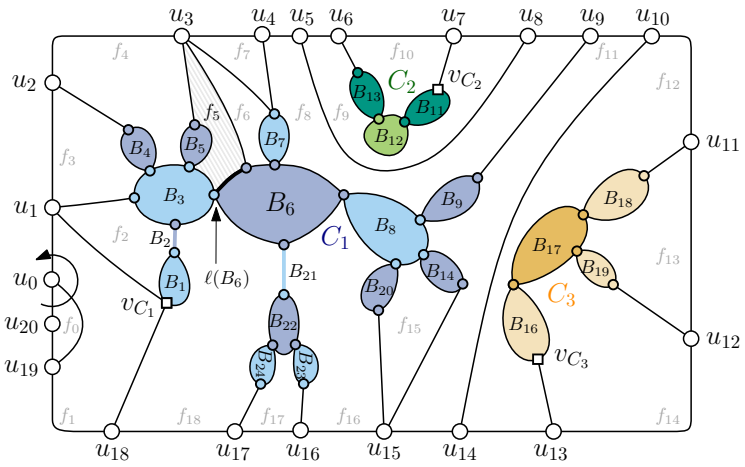


Figure 4 Illustration of the graph $\sigma_1(G)$ of a two-level instance G: $u_0, \ldots, u_{20}$ are the vertices of L_0; $C_1(G)$ consists of three connected components C_1, C_2 and C_3, whose first vertices are denoted by v_{C_1}, v_{C_2} and v_{C_3}, resp.; vertices assigned to each block have the same color as the block; C_1 contains two blocks B_2 and B_{21} that are simple edges; the two level edges (u_5, u_6) and (u_5, u_8) are short and long, resp.; (u_1, v_{C_1}) is a binding edge; the intra-level faces of $\mathcal{F}$ are numbered according to $\lambda(\mathcal{F})$; the intra-level face $d(B_6)$ that discovers B_6 is the face f_5 tilled gray; hence, $\mathrm{dom}(B_6) = u_3$; f_1, f_9 and f_{12} discover the degenerate blocks.

to a binding edge that is on the boundary of f_C. We will further assume that v_C forms a degenerate block r_C of C. The *leader* of a block B of C, denoted by $\ell(B)$, is the first vertex of B that is encountered in any path of C from v_C to B; note that $\ell(B)$ is uniquely defined.

Consider a vertex v of C. If v belongs to only one block of C, then v is *assigned* to that block. Otherwise v is assigned to the block B of C such that v belongs to B and the graph-theoretic distance in C between $\ell(B)$ and v_C is the smallest. It follows that v_C is assigned to the degenerate block r_C, and that for any non-degenerate block B the leader $\ell(B)$ is not assigned to B. We denote by $B(v)$ the block of C that a vertex v is assigned to. Let B be a block of C. Assume first that B is non-degenerate. We refer to the first face in the ordering $\lambda(\mathcal{F})$ containing an edge of B as the face that *discovers* B. Assume now that B is degenerate, i.e., it consists of a single vertex v. We refer to the first face in the ordering $\lambda(\mathcal{F})$ that has v on its boundary as the face that discovers B. In both cases, we denote by $d(B)$ the face in $\mathcal{F}$ that discovers block B. We extend the notion of discovery to the vertices of G. To this end, let v be a vertex of G (which can be incident to several intra-level faces in $\mathcal{F}$). We distinguish whether v belongs to L_0 or L_1. In the former case, face f of $\mathcal{F}$ *discovers* vertex v if f is the first intra-level face in the ordering $\lambda(\mathcal{F})$ that contains v on its boundary. In the latter case, face f in $\mathcal{F}$ *discovers* vertex v if f is the face that discovers the block vertex v is assigned to. In both cases we denote by $d(v)$ the face in $\mathcal{F}$ that discovers vertex v. This yields $d(v) = d(B(v))$ for any $v \in L_1$. The *dominator* $\mathrm{dom}(B)$ of block B is the vertex of L_0 with the smallest subscript that is on the boundary of $d(B)$. Several blocks of C can be discovered by the same face, and by definition, these blocks have the same dominator. Analogously, we define the dominator $\mathrm{dom}(f)$ of an intra-level face f as the vertex of L_0 with the smallest subscript that is on the boundary of f. This yields $\mathrm{dom}(B) = \mathrm{dom}(d(B))$.

▶ **Property 3.** *The face $d(B)$ that discovers block B is the first face in $\lambda(\mathcal{F})$ that has a vertex assigned to block B on its boundary.*

Proof. If B is a degenerate block, the property follows by definition. Otherwise, B contains at least one edge on its boundary. The face $d(B)$ is the first intra-level face in $\lambda(\mathcal{F})$ that contains an edge (v, w) of B on its boundary. Since only $\ell(B)$ is not assigned to B and since (v, w) is a boundary edge of B, at least one of v and w is assigned to B. The property follows from the fact that at most one of the endpoints of (v, w) is not assigned to B. ◀

Let B and B' be two blocks of $C_1(G)$. We say that B *precedes* B' and write $B \prec B'$ if (i) $d(B) \prec_\lambda d(B')$, or (ii) $d(B) = d(B')$ and in a counterclockwise traversal of $d(B)$ starting from $\mathrm{dom}(d(B))$ block B is encountered before block B'. Since $\lambda(\mathcal{F})$ is a well-defined ordering, the relationship "precedes" defines a total ordering of the blocks of $C_1(G)$.

▶ **Property 4.** *Let v be a vertex of G and let $f_v \in \mathcal{F}$ be an intra-level face that contains v on its boundary. Then, $d(v) \preceq_\lambda f_v$ holds.*

Proof. If v belongs to L_0, then the property follows by definition. Otherwise, v belongs to L_1, and $d(v)$ is the intra-level face that discovers the block $B(v)$, that is, $d(v) = d(B(v))$. If $B(v)$ is degenerate, then $d(v)$ is the first intra-level face in $\lambda(\mathcal{F})$ that has v on its boundary. Hence, $d(v) \preceq_\lambda f_v$. Otherwise, by Property 3, $d(B(v))$ is the first intra-level face in $\lambda(\mathcal{F})$ that contains a vertex assigned to block B on its boundary. Since $d(v) = d(B(v))$ and since v is assigned to block B, it follows that $d(v) \preceq_\lambda f_v$. ◀

A vertex v of L_0 belonging to the boundary of an intra-level face f is *prime* with respect to f if no vertex of L_1 and no long level edge is encountered in the clockwise traversal of f from $\mathrm{dom}(f)$ to v. By definition, $\mathrm{dom}(f)$ is prime with respect to f. We say that a vertex v is *f-prime* if either v is prime with respect to face f or v belongs to L_1. By definition, any vertex of L_1 is g-prime with respect to any intra-level face g. Let u_j be a vertex on L_0 that is not $d(u_j)$-prime with $j \in \{1, \ldots, s-1\}$. Let $f_0^{u_j}, \ldots, f_t^{u_j}$ be the faces that have u_j on their boundary in a counterclockwise traversal of u_j starting from (u_{j-1}, u_j) and ending at (u_j, u_{j+1}) (indices taken modulo s). Let d be smallest index such that $f_d^{u_j} = d(u_j)$. The faces $f_0^{u_j}, \ldots, f_{d-1}^{u_j}$ that have u_j as their dominator are called *small*.

2.1.1 Linear ordering

We compute the *linear ordering* ρ of the vertices by first embedding the vertices of L_0 in the order $u_0, u_1, \ldots, u_{s-1}$, and by embedding the remaining vertices of L_1 based on the blocks that they have been assigned to and according to the following rules:

- **R.1** For $j = 0, \ldots, s-1$, let $B_0^j, \ldots, B_{t-1}^j$ be the blocks with u_j as dominator such that the faces that discover them are not small (are small, resp.), and $B_i^j \prec B_{i+1}^j$ for $i = 0, 1, \ldots, t-2$. The vertices assigned to these blocks are placed right after (before, resp.) u_j in ρ.
- **R.2** The vertices assigned to B_i^j are right before those assigned to B_{i+1}^j, for each $i = 0, \ldots, t-2$.
- **R.3** The vertices assigned to the same block B_i^j are in the order they appear in a counterclockwise traversal of the boundary of B_i^j starting from the leader of B_i^j, for $i = 0, \ldots, t-1$.

For a pair of distinct vertices v and w, we write $v \prec_\rho w$ if v precedes w in ρ. By Rule R.1, the vertices of L_1 discovered by f and the f-prime vertices of L_0 are right next to each other in ρ. The next property is consequence of Rules R.1–R.3.

▶ **Property 5.** *The vertices assigned to a block B of L_1 appear consecutively in ρ.*

Properties 6 to 8 will be useful in Section 2.2; for the proofs of Properties 7 and 8 refer to [5].

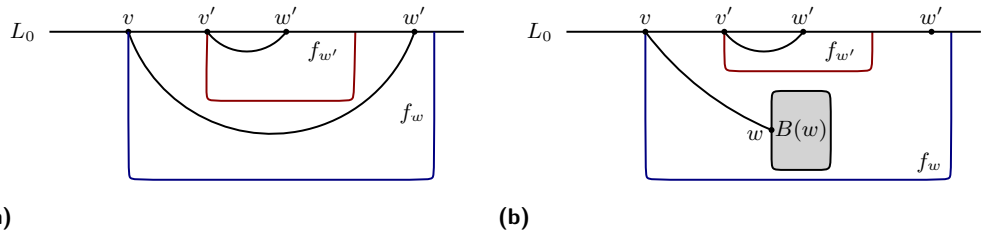


Figure 5 Illustration for the proof of Lemma 9.

▶ **Property 6.** *Let C_1 and C_2 be two connected components of $C_1(G)$ rooted at their first vertices, and let B_1 and B_2 be two non-degenerate blocks of C_1 and C_2, respectively. If there exists a vertex v assigned to B_2 between $\ell(B_1)$ and the vertices assigned to B_1 in ρ, then all vertices assigned to B_2 appear in ρ between $\ell(B_1)$ and the vertices assigned to B_1.*

Proof. Let B_1' be the block that $\ell(B_1)$ is assigned to. Then B_1' is a block of C_1 and $B_1' \neq B_1$. Let w be a vertex assigned to block B_1. Then we have $\ell(B_1) \prec_\rho v \prec_\rho w$ with $\ell(B_1)$ assigned to B_1', v assigned to B_2, and w assigned to B_1. By Property 5, all vertices assigned to the same block are consecutive in ρ, and the claim follows. ◀

▶ **Property 7.** *Let C be a connected component of $C_1(G)$ rooted at its first vertex, and let B be a non-degenerate block of C with two children B_1 and B_2. If $\ell(B_1) \preceq_\rho \ell(B_2)$ and $B_2 \prec B_1$, then all vertices assigned to descendant blocks of B_2 (including B_2) precede in ρ all vertices assigned to descendant blocks of B_1 (including B_1).*

▶ **Property 8.** *Let C be a connected component of $C_1(G)$, and let B_1 and B_2 be two distinct non-degenerate blocks of C. If there is a vertex v assigned to a block B_1 between $\ell(B_2)$ and the remaining vertices of B_2 such that $\ell(B_1) \prec_\rho \ell(B_2)$, then $\ell(B_2)$ is assigned to B_1.*

2.1.2 Edge-to-page assignment

An edge (v, w) is a *dominator* edge if v is the dominator of an intra-level face f_w containing w on its boundary. A dominator edge (v, w) is *backward* if $v \prec_\rho w$ or *forward* otherwise. Next, we prove that all backward edges of G can be assigned to a single page. The proof is reminiscent of a corresponding one by Yannakakis [47] for similarly-defined backward edges.

▶ **Lemma 9.** *Let (v, w) and (v', w') be two backward edges of G, such that v, w, v' and w' are four distinct vertices of G with $v \prec_\rho w$, $v' \prec_\rho w'$ and $v \prec_\rho v'$. Then, $v \prec_\rho w \prec_\rho v' \prec_\rho w'$ or $v \prec_\rho v' \prec_\rho w' \prec_\rho w$ holds.*

Proof. By definition, v and v' are the dominators of two intra-level faces f_w and $f_{w'}$ containing w and w' on their boundaries. If $w \prec_\rho v'$, we have $v \prec_\rho w \prec_\rho v' \prec_\rho w'$. Thus, assume $v' \prec_\rho w$. If w belongs to L_0, then w is not f_w-prime; see Fig. 5a. Since $v \prec_\rho v'$, and v and v' are the dominators of f_w and $f_{w'}$, respectively, it follows that $f_w \prec_\lambda f_w'$. Since vertex w is not f_w-prime, we have $w' \prec_\rho w$. Hence, $v \prec_\rho v' \prec_\rho w' \prec_\rho w$. Assume now that w belongs to L_1; see Fig. 5b. Since v is the dominator of f_w, and $v \prec_\rho w$, the vertex w belongs to a block $B(w)$ discovered by v. By Rule R.1, there is no vertex of L_0 between v and the vertices assigned to $B(w)$ in ρ. Hence, v' cannot appear between v and w in ρ. ◀

Similarly, we can prove that all forward edges can be assigned to a single page.

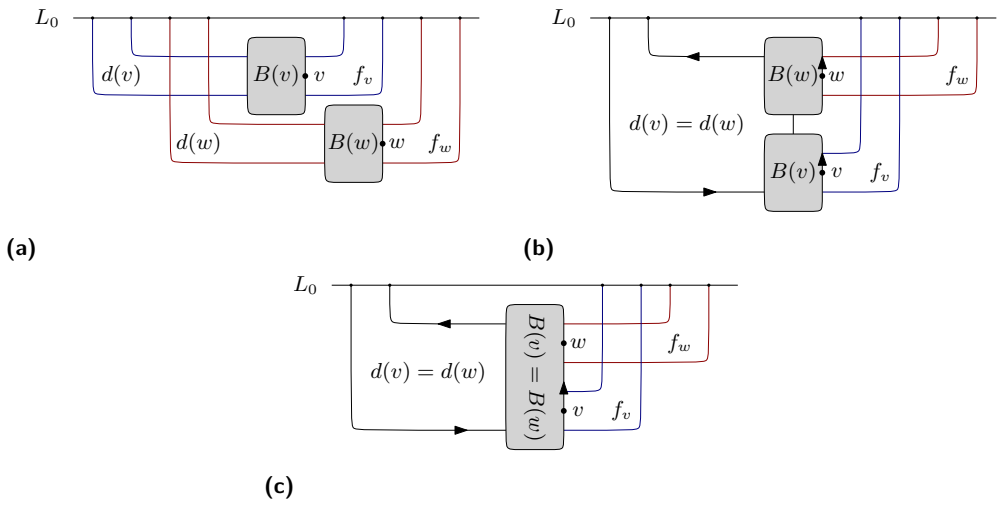


Figure 6 Illustration for the proof of Lemma 13.

▶ **Lemma 10.** *Let (v, w) and (v', w') be two forward edges of G, such that v, w, v' and w' are four distinct vertices of G with $w \prec_\rho v$, $w' \prec_\rho v'$ and $v' \prec_\rho v$. Then, $w' \prec_\rho v' \prec_\rho w \prec_\rho v$ or $w \prec_\rho w' \prec_\rho v' \prec_\rho v$ holds.*

We now present properties helpful for the page assignment of the non-dominator edges.

▶ **Property 11.** *Let v be a $d(v)$-prime vertex of L_0. Then v is f-prime for any intra-level face f that has v on its boundary. Also, $v = dom(f)$, except possibly for $f = d(v)$.*

Proof. Let f be an intra-level face that is different from $d(v)$ such that f has v on its boundary. By planarity, vertex v is the dominator of face f. Thus, v is f-prime. ◀

▶ **Property 12.** *Let w be a $d(w)$-prime vertex. For any vertex v with $v \prec_\rho w$, $d(v) \preceq_\lambda d(w)$.*

Proof. Since w is $d(w)$-prime, w precedes any vertex discovered by a face f with $d(w) \prec_\lambda f$. Assuming to the contrary that $d(w) \prec_\lambda d(v)$, we get $w \prec_\rho v$; a contradiction. ◀

▶ **Lemma 13.** *Let v and w be two vertices of G, such that $v \prec_\rho w$. Also, let f_v and f_w be two intra-level faces containing v and w on their boundaries, respectively, such that $f_v \prec_\lambda f_w$. If the following conditions hold (i) v is $d(v)$-prime, (ii) w is $d(w)$-prime, and (iii) v and w are not the dominators of f_v and f_w, respectively, then $f_v \preceq_\lambda d(w)$ holds.*

Proof. First, observe that by Property 12, we have $d(v) \preceq_\lambda d(w)$. We split the proof into four cases based on whether v and w belong to L_0 or to L_1. **(a)** v and w belong to L_0. Since v is $d(v)$-prime, it follows by Property 11 that v is also f_v-prime. However, since v is not the dominator of f_v, it follows that $d(v) = f_v$. The same holds for vertex w and the faces $d(w)$ and f_w. Now the claim $f_v \preceq_\lambda d(w)$ is an immediate consequence of the assumption $f_v \prec_\lambda f_w$. **(b)** v belongs to L_0 and w belongs to L_1. By Property 11 and Condition (i), we know that v is f_v-prime. By Property 4, we have $d(v) \preceq_\lambda f_v$. If $d(v) \prec_\lambda f_v$, Property 11 implies $v = \text{dom}(f_v)$ which contradicts Condition (iii). However, if $d(v) = f_v$, the claim follows from $d(v) \preceq_\lambda d(w)$. **(c)** v belongs to L_1 and w belongs to L_0. Consider vertex w. As above, by Property 11 and Condition (ii), it follows that w is f_w-prime and therefore, by Condition (iii), $d(w) = f_w$ holds. Recalling the assumption $f_v \prec_\lambda f_w$, the claim $f_v \preceq_\lambda d(w)$

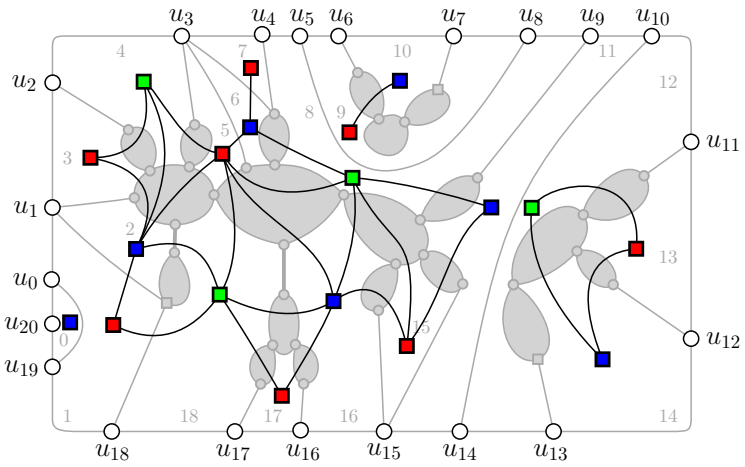


Figure 7 The conflict graph of Fig. 4.

is a direct consequence of $f_v \prec_\lambda f_w$. **(d)** v and w belong to L_1. Assume to the contrary that $d(w) \prec_\lambda f_v$. This implies $d(v) \preceq_\lambda d(w) \prec_\lambda f_v \prec_\lambda f_w$. We consider the two subcases, namely, $d(v) \prec_\lambda d(w)$ and $d(v) = d(w)$. In the former, since v belongs to L_1, vertex v belongs to the boundary of block $B(v)$ discovered by $d(v)$. Similarly, vertex w belongs to the boundary of block $B(w)$ discovered by $d(w)$. Hence, we have $B(v) \neq B(w)$, as $d(v) \prec_\lambda d(w)$; see Fig. 6a. The order $f_v \prec_\lambda f_w$ violates the planarity of $\sigma(G)$; a contradiction. In the latter, since v belongs to L_1, vertex v belongs to the boundary of block $B(v)$ discovered by $d(v) = d(w)$. Similarly, vertex w belongs to the boundary of block $B(w)$ discovered by $d(v) = d(w)$. For the two blocks $B(v)$ and $B(w)$ either $B(v) \neq B(w)$ or $B(v) = B(w)$ holds. First, assume that $B(v) \neq B(w)$; see Fig. 6b. $B(v)$ and $B(w)$ are discovered by the same face, and $v \prec_\rho w$. By Rule R.2 it follows $B(v)$ precedes $B(w)$ in the counterclockwise traversal of $d(v) = d(w)$. With $f_v \prec_\lambda f_w$, the planarity of $\sigma(G)$ is violated; a contradiction. Next, assume $B(v) = B(w)$. Since $v \prec_\rho w$, by Rule R.3, in the counterclockwise traversal of $B(v) = B(w)$ starting from its leader, vertex v precedes w; see Fig. 6c. The order $f_v \prec_\lambda f_w$ violates the planarity of $\sigma(G)$; a contradiction. ◀

▶ **Lemma 14.** *Let v, w, x and z be four vertices of G, such that (v, w) and (x, z) are two non-dominator edges of G, and $v \prec_\rho x \prec_\rho w \prec_\rho z$. Let f_{vw} be a face with v and w on its boundary, and let f_{xz} be a face with x and z on its boundary such that f_{vw} and f_{xz} are two distinct faces. Moreover, v and w are f_{vw}-prime, whereas x and z are f_{xz}-prime. Then $d(x) = f_{vw}$ or $d(w) = f_{xz}$ holds.*

Observe that in Lemma 14 the edges (v, w) and (x, z) form two non-dominator edges that cannot be assigned to the same page. Lemma 14 translates this conflict into a relationship between the two faces f_{vw} and f_{xz} containing these edges. In the following, we model these conflicts as edges of an auxiliary graph which we call the *conflict graph* and denote by $\mathcal{C}(G)$.

▶ **Definition 15.** *The* conflict graph $\mathcal{C}(G)$ *of G is an undirected graph whose vertices are the faces of $\mathcal{F}$. There exists an edge (f, g) with $f \neq g$ in $\mathcal{C}(G)$ if and only if there exists a vertex w of level L_1 on the boundary of g such that $f = d(w)$; see Fig. 7.*

With this definition, we are able to restate Lemma 14 as follows.

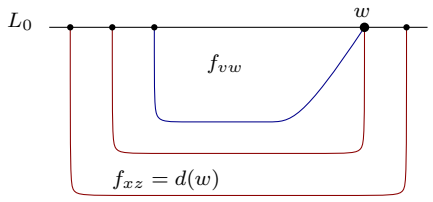


Figure 8 Illustration for the proof of Lemma 16.

▶ **Lemma 16.** *Let (v, w) and (x, z) be two non-dominator edges of G belonging to two distinct faces f_{vw} and f_{xz} such that v and w are f_{vw}-prime, x and z are f_{xz}-prime, $v \prec_\rho w$, and $x \prec_\rho z$. If (v, w) and (x, z) cross in ρ, then there is an edge (f_{vw}, f_{xz}) in $\mathcal{C}(G)$.*

Proof. W.l.o.g. assume $v \prec_\rho x \prec_\rho w \prec_\rho z$. As in Lemma 14, we show $v, x \in L_1$. By Lemma 14, $f_{vw} = d(x)$ or $f_{xz} = d(w)$ holds. Since $x \in L_1$, $(f_{vw}, f_{xz}) \in \mathcal{C}(G)$ if $f_{vw} = d(x)$ holds. Thus, consider $f_{xz} = d(w)$. If $w \in L_1$, $(f_{vw}, f_{xz}) \in \mathcal{C}(G)$. Assume $w \in L_0$. If $f_{vw} \prec_\lambda f_{xz}$, we get $d(w) \preceq_\lambda f_{vw} \prec_\lambda f_{xz} = d(w)$ by Property 4; a contradiction. Otherwise, if w is $d(w)$-prime, we have $d(w) = f_{xz} \prec_\lambda f_{vw}$ and thus, $w = \mathrm{dom}(f_{vw})$ by Property 11; a contradiction. So, w is not $d(w)$-prime. Since w is f_{vw}-prime with $w \neq \mathrm{dom}(f_{vw})$, at least one vertex of L_0 on f_{vw} is right before w in a clockwise traversal of L_0; see Fig. 8. By Property 12 and $v, x \in L_1$, we have $d(v) \preceq_\lambda d(x)$. By Property 4, we get $d(v) \preceq_\lambda d(x) \preceq_\lambda f_{xz}$. In fact, $d(v) = d(x) = f_{xz}$ holds as otherwise $d(v)$ and f_{vw} cannot bound $B(v)$ without violating planarity. Thus, $d(v) = f_{xz}$ and $v \in L_1$ imply $(f_{vw}, f_{xz}) \in \mathcal{C}(G)$. ◀

In the following lemma, we prove an important property of the conflict graph.

▶ **Lemma 17.** *Graph $\mathcal{C}(G)$ is 1-page book embeddable.*

Proof. We order the vertices of $\mathcal{C}(G)$ as in $\lambda(\mathcal{F})$. For a contradiction, suppose $\mathcal{C}(G)$ contains two crossing edges (f, g) and (f', g') such that, w.l.o.g., $f \prec_\lambda f' \prec_\lambda g \prec_\lambda g'$. Then, there is either $v \in L_1$ on f with $g = d(v)$, or $w \in L_1$ on g with $f = d(w)$. In the former, by Property 4, we have $d(v) \preceq_\lambda f$, contradicting $g = d(v) \preceq_\lambda f \prec_\lambda g$. In the latter, we argue analogously for (f', g'). Hence, there exist $w, w' \in L_1$ on g and g', respectively, with $f = d(w)$ and $f' = d(w')$. This yields $d(w) \prec_\lambda d(w') \prec_\lambda g \prec_\lambda g'$. Since $w, w' \in L_1$, they are $d(w)$- and $d(w')$-prime. By Property 12 and since $w \neq w'$, we have $w \prec_\rho w'$. We apply Lemma 13 on w and w' with $f_v = g$ and $f_w = g'$, and obtain $g \preceq_\lambda d(w)$, contradicting $d(w) \prec_\lambda g$. ◀

Since $\mathcal{C}(G)$ is 1-page book embeddable, it is outerplanar [8]. Hence, we have the following.

▶ **Corollary 18.** *Graph $\mathcal{C}(G)$ admits a vertex coloring with three colors.*

We are now ready to give the main result of the section.

▶ **Theorem 19.** *The book thickness of a two-level k-framed graph G is at most $3\lceil \frac{k}{2} \rceil + 2$.*

Sketch. By Lemma 9, we embed all backward edges in page p_0, and all forward edges in page p_1. We next assign the remaining edges of G to three sets R^1, B^1 and G^1, each containing $\lceil \frac{k}{2} \rceil$ pages. We process the intra-level faces of $\mathcal{F}$ according to $\lambda(\mathcal{F})$. Let f be the next face to process. By Corollary 18, face f has a color in $\{r, b, g\}$. The vertices of f induce at most a k-clique C_f in G. We assign the non-dominator edges of C_f to the pages of one of the sets R^1, B^1 and G^1 depending on whether the color of f is r, b, or g, respectively. This is possible since C_f is at most a k-clique [8]. Let (v, w) and (x, z) be two non-dominator edges, and let f_{vw} and f_{xz} be the faces of $\mathcal{F}$ responsible for assigning (v, w) and (x, z) to one

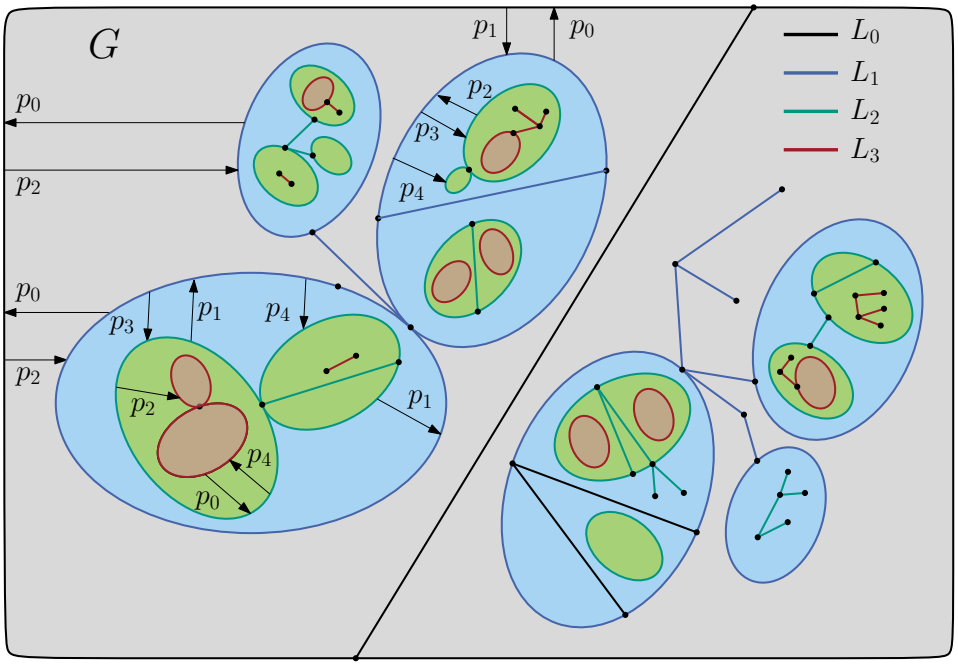


Figure 9 A multi-level instance G with four levels of vertices, such that the bicomponents of $\hat{G}_2$ (which are shaded blue) form two connected components. Incoming edge and the two outgoing edges incident to the components are used to indicate page to which the backward edges and the the two sets of forward edges of each bicomponent are assigned, respectively.

of the pages of $R^1 \cup B^1 \cup G^1$. If v and w are f_{vw}-prime, and x and z are f_{xz}-prime, then by Lemma 16, (v, w) and (x, z) cannot cross. In the full version [5], we prove that no two edges in the same page can cross, even if their endpoints are non-prime vertices of L_0. ◀

2.2 Inductive step: multi-level instances

In this section, we consider the general instances, which we call *multi-level instances*, in which the input k-framed graph G consists of $q \geq 3$ levels $L_0, L_1, \ldots, L_{q-1}$. We refer to Fig. 9 for a schematic representation of a multi-level instance. Initially, we assume that the unbounded face of $\sigma(G)$ contains no crossing edges in its interior; we will eventually drop this assumption. Recall that G_i denotes the subgraph of G induced by the vertices of $L_0 \cup \ldots \cup L_i$ containing neither chords of $\sigma_i(G)$ nor the crossing edges that are in the interior of the unbounded face of $\sigma(G)$. We will further denote by $\hat{G}_i$ the subgraph of G_i that is induced by the vertices of $L_{i-1} \cup L_i$ without the chords of $\sigma_{i+1}(G)$. Observe that $\hat{G}_i$ is not necessarily connected; however, its maximal biconnected components, refered to as *bicomponents* in the following, form two-level instances. To ease the description, we refer to the blocks of all bicomponents of $\hat{G}_i$ simply as the blocks of $\hat{G}_i$. In a book embedding of G_i, we say that two vertices of the level L_j (with $j \leq i$) are *sequential* if there is no other vertex of level L_j between them along the spine. We say that a set U of vertices of level $L_{j'}$ is j-*delimited*, with $j' \neq j$, if either: (a) there exist two sequential vertices of level L_j such that all vertices of U appear between them along the spine, or (b) all vertices of U are preceded or followed along the spine by all vertices of L_j.

A book embedding $\mathcal{E}_i$ of G_i is *good* if it satisfies the following properties[1]:

P.1 The left-to-right order of the vertices on the boundary of each non-degenerate block B of $\hat{G}_i$ in $\mathcal{E}_i$ complies with the order of these vertices in a counterclockwise (clockwise) traversal of the boundary of B, if i is odd (even).

P.2 All vertices of each block B of $\hat{G}_i$, except possibly for its leftmost vertex, are consecutive and $(i-1)$-delimited.

P.3 If between the leftmost vertex $\ell(B)$ of a block B of $\hat{G}_i$ and the remaining vertices of B there is a vertex v of L_i that belongs to a block B' of $\hat{G}_i$ in the same connected component as B, such that the leftmost vertex $\ell(B')$ of B' is to the left of $\ell(B)$, then B and B' share $\ell(B)$.

P.4 Let B and B' be two blocks of $\hat{G}_i$ for which P.3 does not apply, and let $\ell(B)$ and $\ell(B')$ be their leftmost vertices. If $\ell(B)$ precedes $\ell(B')$, then either $\ell(B')$ precedes all remaining vertices of B or all remaining vertices of B' precede all remaining vertices of B.

P.5 For any $j \leq i-2$, all the vertices of each block of $\hat{G}_i$ are j-delimited.

P.6 The edges of G_i are assigned to $6\lceil k/2 \rceil + 5$ pages partitioned as (i) $P = \{p_0, \ldots, p_4\}$, and (ii) $R^j = \{r_1^j, \ldots, r_{\lceil k/2 \rceil}^j\}$, $B^j = \{b_1^j, \ldots, b_{\lceil k/2 \rceil}^j\}$, $G^j = \{g_1^j, \ldots, g_{\lceil k/2 \rceil}^j\}$, $j \in \{0, 1\}$.

P.7 The edges of G_i are classified as backward, forward, or non-dominator such that:

- **a** For $\zeta \leq i$, the non-dominator edges of $\hat{G}_\zeta$ belong to $R^j \cup B^j \cup G^j$ with $j = \zeta \mod 2$.
- **b** The edges that are incident to the leftmost vertex of a bicomponent of $\hat{G}_i$ and that are in its interior are backward.
- **c** Let $\mathcal{B}_i$ be a bicomponent of $\hat{G}_i$. The backward edges of $\hat{G}_i$ in the interior of $\mathcal{B}_i$ are assigned to a single page $b(\mathcal{B}_i)$, while the forward edges are assigned to two pages $f_1(\mathcal{B}_i)$ and $f_2(\mathcal{B}_i)$ of P different from $b(\mathcal{B}_i)$; refer to Fig. 9.
- **d** Let $\mathcal{B}_{i-1}$ be a bicomponent of $\hat{G}_{i-1}$. The blocks $B_{i-1}^1, \ldots, B_{i-1}^\mu$ of $\mathcal{B}_{i-1}$ are the boundaries of several bicomponents of $\hat{G}_i$. Then, the forward edges of $\hat{G}_{i-1}$ incident to B_{i-1}^j, with $j = 1, \ldots, \mu$, are either all assigned to $f_1(\mathcal{B}_{i-1})$ or to $f_2(\mathcal{B}_{i-1})$.
- **e** Let $\langle p'_0, \ldots, p'_4 \rangle$ be a permutation of P. Assume that the backward edges of $\hat{G}_{i-2}$ that are in the interior of a bicomponent $\mathcal{B}_{i-2}$ of $\hat{G}_{i-2}$ have been assigned to p'_0 (in accordance with P.7c), while the forward edges of $\hat{G}_{i-2}$ that are in the interior of $\mathcal{B}_{i-2}$ have been assigned to p'_1 and p'_2 (in accordance to P.7c and P.7d). The blocks of $\mathcal{B}_{i-2}$ are the boundaries of several bicomponents $\mathcal{B}_{i-1}^1, \ldots, \mathcal{B}_{i-1}^\mu$ of $\hat{G}_{i-1}$. Consider now a bicomponent $\mathcal{B}_{i-1}^j$ with $1 \leq j \leq \mu$ of $\hat{G}_{i-1}$. Assume w.l.o.g. that the forward edges of $\mathcal{B}_{i-2}$ incident to $\mathcal{B}_{i-1}^j$ are assigned to p'_1. Then, the backward edges of $\mathcal{B}_{i-1}^j$ (which are incident to its blocks, and thus to the bicomponents of $\hat{G}_i$) are assigned to p'_2, while its forward edges to p'_3 and p'_4.

The book embeddings computed in Section 2.1 can be easily adjusted to become good.

▶ **Lemma 20.** *Any two-level instance admits a good book embedding.*

Finally, the next lemma deals with good book embeddings of multi-level instances.

▶ **Lemma 21.** *Any multi-level instance admits a good book embedding.*

[1] We stress at this point that even though Properties P.7c, P.7d and P.7e might be a bit difficult to be parsed, they formalize the main idea of Yannakakis' algorithm for reusing the same set of pages in a book embedding. Notably, this formalization in the original seminal paper [47] is not present.

Sketch. Assume to have recursively computed a good book embedding $\mathcal{E}_i$ of G_i. We show how to extend $\mathcal{E}_i$ to a good book embedding $\mathcal{E}_{i+1}$ of G_{i+1}. Consider the set $\mathcal{H}$ of bicomponents $\mathcal{B}_1, \ldots, \mathcal{B}_\chi$ of $\hat{G}_{i+1}$, each of which forms a two-level instance. Hence, the vertices delimiting the unbounded faces of $\mathcal{B}_1, \ldots, \mathcal{B}_\chi$ form blocks $B_1, \ldots, B_\chi$ of $\hat{G}_i$, which form a set of cacti in $\sigma_i(G)$. By rooting each connected component in this set at one of its blocks, we associate each bicomponent in $\mathcal{H}$ with a root bicomponent denoted by $r(\mathcal{B}_i)$, $i = 1, \ldots, \chi$, and with a parity bit $\epsilon(\mathcal{B}_i)$ that expresses whether the distance between $\mathcal{B}_i$ and $r(\mathcal{B}_i)$ is odd or even.

Assume to have processed the first $x-1 < \chi$ bicomponents $\mathcal{B}_1, \ldots, \mathcal{B}_{x-1}$ of $\mathcal{H}$ and to have extended $\mathcal{E}_i$ to a good book embedding $\mathcal{E}_i^{x-1}$ of G_i together with $\mathcal{B}_1, \ldots, \mathcal{B}_{x-1}$. Consider the next bicomponent $\mathcal{B}_x$ of $\hat{G}_{i+1}$ in $\mathcal{H}$. The boundary of $\mathcal{B}_x$ is a simple cycle of vertices of L_i. Therefore, the vertices and the edges of this cycle are present in G_i and have been embedded in $\mathcal{E}_i$ and thus in $\mathcal{E}_i^{x-1}$. We show how to extend $\mathcal{E}_i^{x-1}$ to a good book embedding $\mathcal{E}_i^x$ of G_i together with $\mathcal{B}_1, \ldots, \mathcal{B}_x$. Once all blocks in $\mathcal{H}$ have been processed, the obtained book embedding $\mathcal{E}_i^\chi$ is the desired good book embedding $\mathcal{E}_{i+1}$ of G_{i+1}. The vertices that delimit the unbounded face of $\mathcal{B}_x$ form a block B_x of $\hat{G}_i$. Let $u_0, \ldots, u_{s-1}$ be the order of these vertices by Property P.1. We proceed by computing a good book embedding $\mathcal{E}_x$ of $\mathcal{B}_x$ which exists by Lemma 20, such that the left-to-right order of the vertices of $\mathcal{B}_x$ is $u_0, \ldots, u_{s-1}$ in $\mathcal{E}_x$. If i is even, this can be achieved by flipping $\mathcal{B}_x$. Further, note that $\mathcal{E}_x$ is good by Lemma 20. We extend $\mathcal{E}_i^{x-1}$ to a good book embedding $\mathcal{E}_i^x$ in two steps as follows.

In the first step, for $j = 0, 1, \ldots, s-2$, the vertices of $\mathcal{B}_x$ that appear between u_j and u_{j+1} in $\mathcal{E}_x$, if any, are embedded right before u_{j+1} in $\mathcal{E}_i^{x-1}$ in the same left-to-right order as in $\mathcal{E}_x$; also, the vertices of $\mathcal{B}_x$ that appear after u_{s-1} in $\mathcal{E}_x$, if any, are embedded right after u_{s-1} in $\mathcal{E}_i^{x-1}$ in the same left-to-right order as in $\mathcal{E}_x$. In the second step, we assign the internal edges of $\mathcal{B}_x$ to the already existing pages of $\mathcal{E}_i^x$. This step will complete the extension of $\mathcal{E}_i^{x-1}$ to $\mathcal{E}_i^x$. The backward, forward, and non-dominator edges of $\mathcal{E}_x$ that are internal in $\mathcal{B}_x$ will be classified as backward, forward, and non-dominator, respectively, also in $\mathcal{E}_i^x$. The non-dominator edges of $\mathcal{E}_x$ that are internal in $\mathcal{B}_x$ and are assigned to $r_1^1, \ldots, r_{\lceil k/2 \rceil}^1$, $b_1^1, \ldots, b_{\lceil k/2 \rceil}^1$, $g_1^1, \ldots, g_{\lceil k/2 \rceil}^1$ in $\mathcal{E}_x$ are assigned to $r_1^j, \ldots, r_{\lceil k/2 \rceil}^j$, $b_1^j, \ldots, b_{\lceil k/2 \rceil}^j$, $g_1^j, \ldots, g_{\lceil k/2 \rceil}^j$ in $\mathcal{E}_i^x$, respectively, where $j = i + 1 \mod 2$. All backward edges of $\mathcal{E}_x$ have been assigned to page p_0 in $\mathcal{E}_x$, while its forward edges have been assigned to p_1 and p_2; also, recall that no edge of $\mathcal{E}_x$ has been assigned to pages p_3 and p_4. The backward edges of $\mathcal{E}_x$ that are interior to B_x will be assigned to $\mathcal{E}_i^x$ to a common page b of P (i.e., not necessarily to p_0), while the corresponding forward edges assigned to p_1 and p_2 in $\mathcal{E}_x$ will be reassigned to two pages f_1 and f_2, respectively. We determine pages p, f_1 and f_2 as follows. Assuming $i \geq 3$, there is a bicomponent $\mathcal{B}_{i-2}$ of $\hat{G}_{i-2}$, whose boundary vertices form a cycle that, in G_{i+1}, contains the bicomponent $\mathcal{B}_x$ in its interior. Assume w.l.o.g. that the backward edges of $\mathcal{B}_{i-2}$ are assigned to page $p'_0 \in P$, in accordance to P.7c. It follows by P.7e that we may further assume w.l.o.g. that all the backward edges of the bicomponents of $\hat{G}_{i-1}$, whose boundaries are blocks of $\mathcal{B}_{i-2}$, have been assigned to pages p'_1 and p'_2 different from p'_0. Assume also, w.l.o.g., that the forwards edges of $\mathcal{B}_{i-2}$ incident to $\mathcal{B}_x$ have been assigned to p'_1. By Property P.7e, this implies that the backward (forward) edges of bicomponent $\mathcal{B}_x$ must be assigned to page p'_2 (to p'_3 and p'_4, respectively). Note that also of all the previously processed bicomponents of $\hat{G}_{i+1}$ in $\mathcal{H}$ make use of these three pages plus the page p'_1. The choice between the two pages p'_3 and p'_4 is done based on the parity bit $\epsilon(\mathcal{B}_x)$, so that, all forward edges of all bicomponents in $\mathcal{H}$ having the same parity bit will be assigned to the same page in $\{p'_3, p'_4\}$.

We initially assumed that the unbounded face of $\sigma(G)$ contains no crossing edges in its interior, to support the recursive strategy. We drop this assumption as follows. We assign these edges to the pages of $R^0 \cup B^0 \cup G^0$, which results in a good book embedding of G, since the endvertices of the edges already assigned to these pages are 0-delimited. ◀

Altogether, Lemma 21 in conjunction with Lemma 20 completes the proof of Theorem 1.

3 Conclusions and open problems

Our research generalizes a fundamental result by Yannakakis in the area of book embeddings. To achieve $O(k)$ pages for partial k-framed graphs, we exploit the special structure of these graphs which allows us to model the conflicts of the crossing edges by means of a graph with bounded chromatic number (thus keeping the unavoidable relationship with k low).

Even though our result only applies to a subclass of h-planar graphs, it provides useful insights towards a positive answer to the intriguing question of determining whether the book thickness of (general) h-planar graphs is bounded by a function of h only. Another direction for extending our result is to drop the biconnectivity requirement of partial k-framed graphs.

We conclude that the time complexity of our algorithm is $O(k^2n)$, assuming that a k-framed drawing of the considered graph is also provided. It is of interest to investigate whether (partial) k-framed graphs can be recognized in polynomial time. The question remains valid even for the class of optimal 2-planar graphs, which exhibit a quite regular structure. Brandenburg [12] provided a corresponding linear-time recognition algorithm for the class of optimal 1-planar graphs, while Da Lozzo et al. [16] showed that the related question of determining whether a graph admits a planar embedding whose faces have all degree at most k is polynomial-time solvable for $k \leq 4$ and NP-complete for $k \geq 5$.

References

1 Hugo A. Akitaya, Erik D. Demaine, Adam Hesterberg, and Quanquan C. Liu. Upward partitioned book embeddings. In *Graph Drawing*, volume 10692 of *LNCS*, pages 210–223. Springer, 2017.

2 Md. Jawaherul Alam, Franz J. Brandenburg, and Stephen G. Kobourov. Straight-line grid drawings of 3-connected 1-planar graphs. In Stephen K. Wismath and Alexander Wolff, editors, *Graph Drawing*, volume 8242 of *LNCS*, pages 83–94. Springer, 2013. doi:10.1007/978-3-319-03841-4_8.

3 Md. Jawaherul Alam, Franz J. Brandenburg, and Stephen G. Kobourov. On the book thickness of 1-planar graphs. *CoRR*, abs/1510.05891, 2015. arXiv:1510.05891.

4 Michael A. Bekos, Till Bruckdorfer, Michael Kaufmann, and Chrysanthi N. Raftopoulou. The book thickness of 1-planar graphs is constant. *Algorithmica*, 79(2):444–465, 2017. doi:10.1007/s00453-016-0203-2.

5 Michael A. Bekos, Giordano Da Lozzo, Svenja M. Griesbach, Martin Gronemann, Fabrizio Montecchiani, and Chrysanthi Raftopoulou. Book embeddings of nonplanar graphs with small faces in few pages. *CoRR*, abs/2003.07655, 2020. arXiv:2003.07655.

6 Michael A. Bekos, Martin Gronemann, and Chrysanthi N. Raftopoulou. Two-page book embeddings of 4-planar graphs. *Algorithmica*, 75(1):158–185, 2016. doi:10.1007/s00453-015-0016-8.

7 Michael A. Bekos, Michael Kaufmann, and Chrysanthi N. Raftopoulou. On optimal 2- and 3-planar graphs. In Boris Aronov and Matthew J. Katz, editors, *SoCG*, volume 77 of *LIPIcs*, pages 16:1–16:16. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017. doi:10.4230/LIPIcs.SoCG.2017.16.

8 Frank Bernhart and Paul C. Kainen. The book thickness of a graph. *J. Comb. Theory, Ser. B*, 27(3):320–331, 1979. doi:10.1016/0095-8956(79)90021-2.

9 Therese C. Biedl, Thomas C. Shermer, Sue Whitesides, and Stephen K. Wismath. Bounds for orthogonal 3D graph drawing. *J. Graph Algorithms Appl.*, 3(4):63–79, 1999. doi:10.7155/jgaa.00018.

10 Carla Binucci, Giordano Da Lozzo, Emilio Di Giacomo, Walter Didimo, Tamara Mchedlidze, and Maurizio Patrignani. Upward book embeddings of st-graphs. In *SoCG*, volume 129 of *LIPIcs*, pages 13:1–13:22. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2019. doi:10.4230/LIPIcs.SoCG.2019.13.

11 Robin L. Blankenship. *Book Embeddings of Graphs*. PhD thesis, Louisiana State University, 2003.

12 Franz J. Brandenburg. Characterizing and recognizing 4-map graphs. *Algorithmica*, 81(5):1818–1843, 2019. doi:10.1007/s00453-018-0510-x.

13 Jonathan F. Buss and Peter W. Shor. On the pagenumber of planar graphs. In Richard A. DeMillo, editor, *ACM Symposium on Theory of Computing*, pages 98–100. ACM, 1984. doi:10.1145/800057.808670.

14 Fan R. K. Chung, Frank T. Leighton, and Arnold L. Rosenberg. Embedding graphs in books: A layout problem with applications to VLSI design. *SIAM Journal on Algebraic and Discrete Methods*, 8(1):33–58, 1987.

15 Gérard Cornuéjols, Denis Naddef, and William R. Pulleyblank. Halin graphs and the travelling salesman problem. *Math. Program.*, 26(3):287–294, 1983. doi:10.1007/BF02591867.

16 Giordano Da Lozzo, Vít Jelínek, Jan Kratochvíl, and Ignaz Rutter. Planar embeddings with small and uniform faces. In Hee-Kap Ahn and Chan-Su Shin, editors, *ISAAC*, volume 8889 of *LNCS*, pages 633–645. Springer, 2014. doi:10.1007/978-3-319-13075-0_50.

17 Hubert de Fraysseix, Patrice Ossona de Mendez, and János Pach. A left-first search algorithm for planar graphs. *Discrete & Computational Geometry*, 13:459–468, 1995. doi:10.1007/BF02574056.

18 Giuseppe Di Battista, Peter Eades, Roberto Tamassia, and Ioannis G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice-Hall, 1999.

19 Walter Didimo, Giuseppe Liotta, and Fabrizio Montecchiani. A survey on graph drawing beyond planarity. *ACM Comput. Surv.*, 52(1):4:1–4:37, 2019. doi:10.1145/3301281.

20 Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.

21 Vida Dujmović and Fabrizio Frati. Stack and queue layouts via layered separators. *J. Graph Algorithms Appl.*, 22(1):89–99, 2018. doi:10.7155/jgaa.00454.

22 Vida Dujmovic, Pat Morin, and David R. Wood. Layered separators in minor-closed graph classes with applications. *J. Comb. Theory, Ser. B*, 127:111–147, 2017. doi:10.1016/j.jctb.2017.05.006.

23 Vida Dujmović and David R. Wood. Graph treewidth and geometric thickness parameters. *Discrete & Computational Geometry*, 37(4):641–670, 2007. doi:10.1007/s00454-007-1318-7.

24 Günter Ewald. Hamiltonian circuits in simplicial complexes. *Geometriae Dedicata*, 2(1):115–125, 1973. doi:10.1007/BF00149287.

25 Joseph L. Ganley and Lenwood S. Heath. The pagenumber of k-trees is $O(k)$. *Discrete Applied Mathematics*, 109(3):215–221, 2001. doi:10.1016/S0166-218X(00)00178-5.

26 Xiaxia Guan and Weihua Yang. Embedding 5-planar graphs in three pages. *CoRR*, 1801.07097, 2018. arXiv:1801.07097.

27 Lenwood S. Heath. Embedding planar graphs in seven pages. In *FOCS*, pages 74–83. IEEE Computer Society, 1984. doi:10.1109/SFCS.1984.715903.

28 Michael Hoffmann and Boris Klemz. Triconnected planar graphs of maximum degree five are subhamiltonian. In Michael A. Bender, Ola Svensson, and Grzegorz Herman, editors, *ESA*, volume 144 of *LIPIcs*, pages 58:1–58:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.ESA.2019.58.

29 Sorin Istrail. An algorithm for embedding planar graphs in six pages. *Iasi University Annals, Mathematics-Computer Science*, 34(4):329–341, 1988.

30 Guy Jacobson. Space-efficient static trees and graphs. In *Symposium on Foundations of Computer Science*, pages 549–554. IEEE Computer Society, 1989. doi:10.1109/SFCS.1989.63533.

31 Paul C. Kainen and Shannon Overbay. Extension of a theorem of whitney. *Appl. Math. Lett.*, 20(7):835–837, 2007. doi:10.1016/j.aml.2006.08.019.

32 Stephen G. Kobourov, Giuseppe Liotta, and Fabrizio Montecchiani. An annotated bibliography on 1-planarity. *Computer Science Review*, 25:49–67, 2017. doi:10.1016/j.cosrev.2017.06.002.

33 Seth M. Malitz. Genus g graphs have pagenumber $O(\sqrt{g})$. *J. Algorithms*, 17(1):85–109, 1994. doi:10.1006/jagm.1994.1028.

34 Seth M. Malitz. Graphs with E edges have pagenumber $O(\sqrt{E})$. *J. Algorithms*, 17(1):71–84, 1994. doi:10.1006/jagm.1994.1027.

35 J. Ian Munro and Venkatesh Raman. Succinct representation of balanced parentheses and static trees. *SIAM J. Comput.*, 31(3):762–776, 2001. doi:10.1137/S0097539799364092.

36 Jaroslav Nesetril and Patrice Ossona de Mendez. *Sparsity - Graphs, Structures, and Algorithms*, volume 28 of *Algorithms and combinatorics*. Springer, 2012. doi:10.1007/978-3-642-27875-4.

37 Takao Nishizeki and Norishige Chiba. *Planar Graphs: Theory and Algorithms*, chapter 10. Hamiltonian Cycles, pages 171–184. Dover Books on Mathematics. Courier Dover Publications, 2008.

38 Taylor Ollmann. On the book thicknesses of various graphs. In F. Hoffman, R.B. Levow, and R.S.D. Thomas, editors, *Southeastern Conference on Combinatorics, Graph Theory and Computing*, volume VIII of *Congressus Numerantium*, page 459, 1973.

39 Vaughan R. Pratt. Computing permutations with double-ended queues, parallel stacks and parallel queues. In Alfred V. Aho, Allan Borodin, Robert L. Constable, Robert W. Floyd, Michael A. Harrison, Richard M. Karp, and H. Raymond Strong, editors, *ACM Symposium on Theory of Computing*, pages 268–277. ACM, 1973. doi:10.1145/800125.804058.

40 S. Rengarajan and C. E. Veni Madhavan. Stack and queue number of 2-trees. In Ding-Zhu Du and Ming Li, editors, *COCOON*, volume 959 of *LNCS*, pages 203–212. Springer, 1995. doi:10.1007/BFb0030834.

41 Gerhard Ringel. Ein Sechsfarbenproblem auf der kugel. *Abhandlungen aus dem Mathematischen Seminar der Universitaet Hamburg*, 29(1–2):107–117, 1965.

42 Arnold L. Rosenberg. The diogenes approach to testable fault-tolerant arrays of processors. *IEEE Trans. Computers*, 32(10):902–910, 1983. doi:10.1109/TC.1983.1676134.

43 Robert E. Tarjan. Sorting using networks of queues and stacks. *J. ACM*, 19(2):341–346, 1972. doi:10.1145/321694.321704.

44 Avi Wigderson. The complexity of the Hamiltonian circuit problem for maximal planar graphs. Technical Report TR-298, EECS Department, Princeton University, 1982. arXiv:https://www.math.ias.edu/avi/node/820.

45 David R. Wood. Degree constrained book embeddings. *J. Algorithms*, 45(2):144–154, 2002. doi:10.1016/S0196-6774(02)00249-3.

46 Mihalis Yannakakis. Four pages are necessary and sufficient for planar graphs (extended abstract). In Juris Hartmanis, editor, *ACM Symposium on Theory of Computing*, pages 104–108. ACM, 1986. doi:10.1145/12130.12141.

47 Mihalis Yannakakis. Embedding planar graphs in four pages. *J. Comput. Syst. Sci.*, 38(1):36–67, 1989. doi:10.1016/0022-0000(89)90032-9.

Parallel Computation of Alpha Complexes for Biomolecules

Talha Bin Masood[1]
Scientific Visualization Group, Linköping University, Norrköping, Sweden
talha.bin.masood@liu.se

Tathagata Ray
BITS Pilani, Hyderabad Campus, Hyderabad, India
rayt@hyderabad.bits-pilani.ac.in

Vijay Natarajan
Department of Computer Science and Automation, Indian Institute of Science, Bangalore, India
https://www.csa.iisc.ac.in/~vijayn/
vijayn@iisc.ac.in

Abstract

The alpha complex, a subset of the Delaunay triangulation, has been extensively used as the underlying representation for biomolecular structures. We propose a GPU-based parallel algorithm for the computation of the alpha complex, which exploits the knowledge of typical spatial distribution and sizes of atoms in a biomolecule. Unlike existing methods, this algorithm does not require prior construction of the Delaunay triangulation. The algorithm computes the alpha complex in two stages. The first stage proceeds in a bottom-up fashion and computes a superset of the edges, triangles, and tetrahedra belonging to the alpha complex. The false positives from this estimation stage are removed in a subsequent pruning stage to obtain the correct alpha complex. Computational experiments on several biomolecules demonstrate the superior performance of the algorithm, up to a factor of 50 when compared to existing methods that are optimized for biomolecules.

2012 ACM Subject Classification Theory of computation → Parallel algorithms; Computing methodologies → Shape modeling; Applied computing → Molecular structural biology

Keywords and phrases Delaunay triangulation, parallel algorithms, biomolecules, GPU

Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.17

Related Version The full version of this paper is available at [29], https://arxiv.org/abs/1908.05944.

Supplementary Material Source code available at: https://bitbucket.org/vgl_iisc/parallelac/

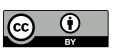
Funding *Vijay Natarajan*: This work is partially supported by a Swarnajayanti Fellowship from the Department of Science and Technology, India (DST/SJF/ETA-02/2015-16); a Mindtree Chair research grant; and the Robert Bosch Centre for Cyber Physical Systems, IISc.

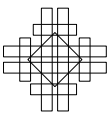
Acknowledgements Part of this work was done when the first author was at Indian Institute of Science, Bangalore. The authors would like to thank Sathish Vadhiyar and Nikhil Ranjanikar for helpful discussions and suggestions during the early phase of this work.

1 Introduction

The alpha complex of a set of points in $\mathbb{R}^3$ is a subset of the Delaunay triangulation. A size parameter α determines the set of simplices (tetrahedra, triangles, edges, and vertices) of the Delaunay triangulation that are included in the alpha complex. It is an elegant representation of the shape of the set of points [16, 18, 14] and has found various applications, particularly in molecular modeling and molecular graphics. The atoms in a biomolecule are represented

[1] Corresponding author





by weighted points in $\mathbb{R}^3$, and the region occupied by the molecule is represented by the union of balls centered at these points. The geometric shape of a biomolecule determines its function, namely how it interacts with other biomolecules. The alpha complex represents the geometric shape of the molecule very efficiently. It has been widely used for computing and studying geometric features such as cavities and channels [25, 26, 10, 30, 33, 28, 23]. Further, an alpha complex based representation is also crucial for accurate computation of geometric properties like volume and surface area [24, 12, 27].

Advances in imaging technology have resulted in a significant increase in the size of molecular structure data. This necessitates the development of efficient methods for storing, processing, and querying these structures. In this paper, we study the problem of efficient construction of the alpha complex with particular focus on point distributions that are typical of biomolecules. In particular, we present a parallel algorithm for computing the alpha complex and an efficient GPU implementation that outperforms existing methods. In contrast to existing algorithms, our algorithm does not require the explicit construction of the Delaunay triangulation.

1.1 Related work

The Delaunay triangulation has been studied within the field of computational geometry for several decades and numerous algorithms have been proposed for its construction [1]. Below, we describe only a few methods that are most relevant to this paper.

A tetrahedron belongs to the Delaunay triangulation of a set of points in $\mathbb{R}^3$ if and only if it satisfies the *empty circumsphere* property, namely no point is contained within the circumsphere of the tetrahedron. The Bowyer-Watson algorithm [4, 34] and the incremental insertion algorithm by Guibas *et al.* [21] are based on the above characterization of the Delaunay triangulation. In both methods, points are inserted incrementally and the triangulation is locally updated to ensure that the Delaunay property is satisfied. The incremental insertion method followed by bi-stellar flipping works in higher dimensions also [20] and can construct the Delaunay triangulation in $O(n \log n + n^{\lceil d/2 \rceil})$ time in the worst case, where n is the number of input points in $\mathbb{R}^d$. A second approach for constructing the Delaunay triangulation is based on its equivalence to the convex hull of the points lifted onto a $(d+1)$-dimensional paraboloid [19].

A third divide-and-conquer approach partitions the inputs points into two or generally multiple subsets, constructs the Delaunay triangulation for each partition, and merges the pieces of the triangulation finally. The merge procedure depends on the ability to order the edges incident on a vertex and hence works only in $\mathbb{R}^2$. The extension to $\mathbb{R}^3$ requires that the merge procedure be executed first [6]. The divide-and-conquer strategy directly extends to a parallel algorithm [31, 5]. The DeWall algorithm [6] partitions the input point set into two halves and first constructs the triangulation of points lying within the boundary region of the two partitions. The Delaunay triangulation of the two halves is then constructed in parallel. The process is repeated recursively resulting in increased parallelism. Cao *et al.* [5] have developed a GPU parallel algorithm, *gDel3D*, that constructs the Delaunay triangulation in two stages. In the first stage, points are inserted in parallel followed by flipping to obtain an approximate Delaunay triangulation. In the second stage, a star splaying procedure works locally to convert non-Delaunay tetrahedra into Delaunay tetrahedra. The algorithm can be extended to construct the weighted Delaunay triangulation for points with weights. Cao *et al.* report a speed up of up to a factor of 10 over a sequential implementation for constructing the weighted Delaunay triangulation of 3 million weighted points.

Existing algorithms for constructing the alpha complex [11, 18, 27, 9] often require that the Delaunay triangulation be computed in a first step, with the exception of a recent method that guarantees output sensitive construction under mild assumptions on weights [32] or a possible construction from Čech complexes [2]. Simplices that belong to the alpha complex are identified using a size filtration in a second step. Simplices that belong to the alpha complex are identified using a size filtration in a second step. In the case of biomolecules, only small values of the size parameter are of interest and the number of simplices in the alpha complex is a small fraction of those contained in the Delaunay triangulation. Hence, the Delaunay triangulation construction is often the bottleneck in the alpha complex computation. The key difficulty lies in the absence of a direct characterization of simplices that belong to the alpha complex.

1.2 Summary of results

We propose an algorithm that avoids the expensive Delaunay triangulation computation and instead directly computes the alpha complex for biomolecules. The key contributions of this paper are summarized below:

- A new characterization of the alpha complex – a set of conditions necessary and sufficient for a simplex to be a part of the alpha complex.
- A new algorithm for computing the alpha complex of a set of weighted points in $\mathbb{R}^3$. The algorithm identifies simplices of the alpha complex in decreasing order of dimension without computing the complete weighted Delaunay triangulation.
- An efficient CUDA-based parallel implementation of this algorithm for biomolecular data that can compute the alpha complex for a 10 million point dataset in approximately 10 seconds.
- A proof of correctness of the algorithm and comprehensive experimental validation to demonstrate that it outperforms existing methods.

While the experimental results presented here focus on biomolecular data, the algorithm is applicable to data from other application domains as well. In particular, the efficient GPU implementation may be used for points that arise in smoothed particle hydrodynamics (SPH) simulations, atomistic simulations in material science, and particle systems that appear in computational fluid dynamics (CFD).

2 Background

In this section, we review the necessary background on Delaunay triangulations required to describe the algorithm and also establish a new characterization of the alpha complex that does not require the Delaunay triangulation. For a detailed description of Delaunay triangulations, alpha complexes, and related structures, we refer the reader to various books on the topic [1, 13, 15].

Let $B = \{b_i\}$ denote a set of balls or weighted points, where $b_i = (p_i, r_i)$ represents a ball centered at p_i with radius r_i. We limit our discussion to balls in $\mathbb{R}^3$, so $p_i = (x_i, y_i, z_i) \in \mathbb{R}^3$. Further, we assume that the points in B are in general position, *i.e.*, no two points have the same location, no three points are collinear, no four points are coplanar, and no subset of five points are equidistant from a point in $\mathbb{R}^3$. Such configurations are called *degeneracies*. In practice, a degenerate input can be handled via symbolic perturbation [17].

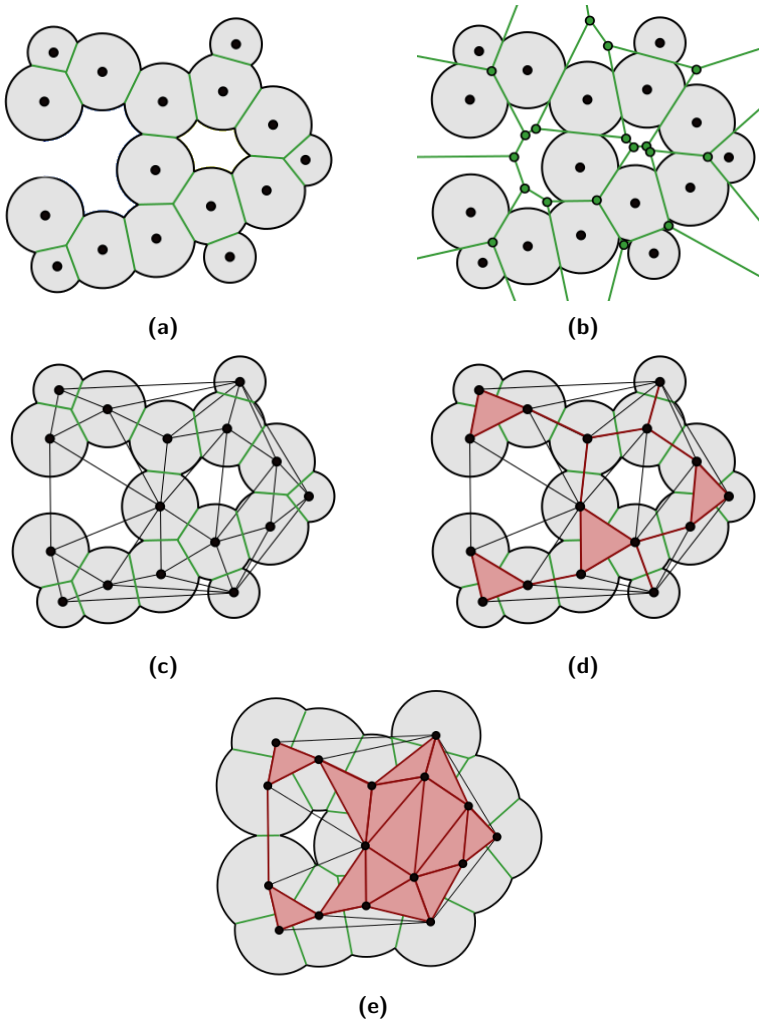


Figure 1 2D weighted Delaunay triangulation and alpha complex. (a) A set of weighted points B in $\mathbb{R}^2$ shown as disks. (b) The weighted Voronoi diagram of B. Voronoi edges and vertices are highlighted in green. (c) The weighted Delaunay complex is the dual of the weighted Voronoi diagram. (d) The alpha complex K_α for $\alpha = 0$ is shown in red. This is the dual of the intersection of the weighted Voronoi diagram and union of balls. (e) The alpha complex shown for an $\alpha > 0$. It is the dual of the intersection of the weighted Voronoi diagram and union of balls after growing them to have radius $\sqrt{r_i^2 + \alpha}$.

2.1 Simplex and simplicial complex

A d-dimensional *simplex* σ^d is defined as the convex hull of $d+1$ affinely independent points. Assuming the centres of balls in B are in general position, all $(d+1)$ sized subsets of B form a simplex $\sigma^d = (p_0^\sigma, p_1^\sigma, \cdots, p_d^\sigma)$. For simplicity, we sometimes use b_i instead of the center p_i to refer to points incident on a simplex. For example, we may write $\sigma^d = (b_0^\sigma, b_1^\sigma, \cdots, b_d^\sigma)$.

A non-empty strict subset of σ^d is also a simplex but with dimension smaller than d. Such a simplex is called a *face* of σ^d. Specifically, a $(d-1)$-dimensional face of σ^d is referred to as a *facet* of σ^d. A set of simplices K is called a *simplicial complex* if: 1) a simplex $\sigma \in K$ implies that all faces of σ also belong to K, and 2) for two simplices σ_1, $\sigma_2 \in K$, either $\sigma_1 \cap \sigma_2 \in K$ or $\sigma_1 \cap \sigma_2 = \emptyset$.

2.2 Power distance and weighted Voronoi diagram

The *power distance* $\pi(p, b_i)$ between a point $p \in \mathbb{R}^3$ and a ball $b_i = (p_i, r_i) \in B$ is defined as

$$\pi(p, b_i) = \|p - p_i\|^2 - r_i^2.$$

The *weighted Voronoi diagram* is an extension of the Voronoi diagram to weighted points. It is a partition of $\mathbb{R}^3$ based on proximity to input balls b_i in terms of the power distance. Points $p \in \mathbb{R}^3$ that are closer to the ball b_i compared to all other balls $b_j \in B$ $(j \neq i)$ constitute the *Voronoi region* of b_i. Points equidistant from two balls b_i, $b_j \in B$ and closer to these two balls compared to other balls constitute a *Voronoi face*. Similarly, points equidistant from three balls and fours balls constitute *Voronoi edges* and *Voronoi vertices* of the weighted Voronoi diagram, respectively. Figure 1b shows the weighted Voronoi diagram for a set of 2D weighted points or disks on the plane. Similar to the unweighted case, the Voronoi regions of the weighted Voronoi diagram are convex and linear. However, the weights may lead to a configuration where the Voronoi region of b_i is disjoint from b_i. This occurs when b_i is contained within another ball b_j. Further, the Voronoi region of b_i may even be empty.

2.3 Weighted Delaunay triangulation

The *weighted Delaunay triangulation* is the dual of the weighted Voronoi diagram, see Figure 1c. It is a simplicial complex consisting of simplices that are dual to the cells of the weighted Voronoi diagram. The following equivalent definition characterizes a simplex σ^d belonging to a Delaunay triangulation D.

▶ **Definition 1** (Weighted Delaunay Triangulation). *A simplex $\sigma^d = (p_0^\sigma, p_1^\sigma, \cdots, p_d^\sigma)$, $0 \leq d \leq 3$, belongs to the weighted Delaunay triangulation D of B if and only if there exists a point $p \in \mathbb{R}^3$ such that*
DT1: *$\pi(p, b_0^\sigma) = \pi(p, b_1^\sigma) = \cdots = \pi(p, b_d^\sigma)$, and*
DT2: *$\pi(p, b_0^\sigma) \leq \pi(p, b_i)$ for $b_i \in B - \sigma^d$.*

A point p that satisfies the above two conditions, DT1 and DT2, is called a *witness* for σ^d. We call a point that minimizes the distance $\pi(p, b_0^\sigma)$ and satisfies both conditions as the *closest witness*, denoted by p^σ_{min}. This minimum distance $\pi(p^\sigma_{\mathsf{min}}, b_0^\sigma)$ is called the Size of the simplex σ^d. A point that minimizes the distance $\pi(p, b_0^\sigma)$ and satisfies DT1 is called the *ortho-center* $p^\sigma_{\mathsf{ortho}}$ of simplex σ^d. The distance $\pi(p^\sigma_{\mathsf{ortho}}, b_0^\sigma)$ is called the $\mathsf{OrthoSize}$ of the simplex σ^d. Clearly, the Size of a simplex is lower bounded by its $\mathsf{OrthoSize}$. Figure 2 shows the two possible scenarios, namely when $\mathsf{OrthoSize} = \mathsf{Size}$ and $\mathsf{OrthoSize} < \mathsf{Size}$.

2.4 Alpha complex

Given a parameter $\alpha \in \mathbb{R}$, we can construct a subset of the weighted Delaunay triangulation by filtering simplices whose Size is less than or equal to α, see Figures 1d and 1e. The resulting subset, called the *alpha complex*, is a subcomplex of the Delaunay complex and is denoted K_α:

$$K_\alpha = \{\sigma^d \in D \quad \text{such that} \quad \mathsf{Size}(\sigma^d) \leq \alpha\}.$$

The following equivalent definition characterizes simplices of the alpha complex without explicitly referring to the Delaunay triangulation.

▶ **Definition 2** (Alpha complex). *A d-dimensional simplex $\sigma^d = (p_0^\sigma, p_1^\sigma, \cdots, p_d^\sigma), 0 \leq d \leq 3$, belongs to the alpha complex K_α of B if and only if there exists a point $p \in \mathbb{R}^3$ such that the following three conditions are satisfied:*
AC1: $\pi(p, b_0^\sigma) = \pi(p, b_1^\sigma) = \cdots = \pi(p, b_d^\sigma)$,
AC2: $\pi(p, b_0^\sigma) \leq \pi(p, b_i)$ *for* $b_i \in B - \sigma^d$, *and*
AC3: $\pi(p, b_0^\sigma) \leq \alpha$ *or equivalently, the* Size *of* σ^d *is at most* α.

3 Algorithm

We now describe an algorithm to compute the alpha complex and prove its correctness. The algorithm utilizes the characterizing conditions introduced above. It first identifies the tetrahedra that belong to the alpha complex, followed by the set of triangles, edges and vertices. Figure 3 illustrates the algorithm as applied to disks on the plane.

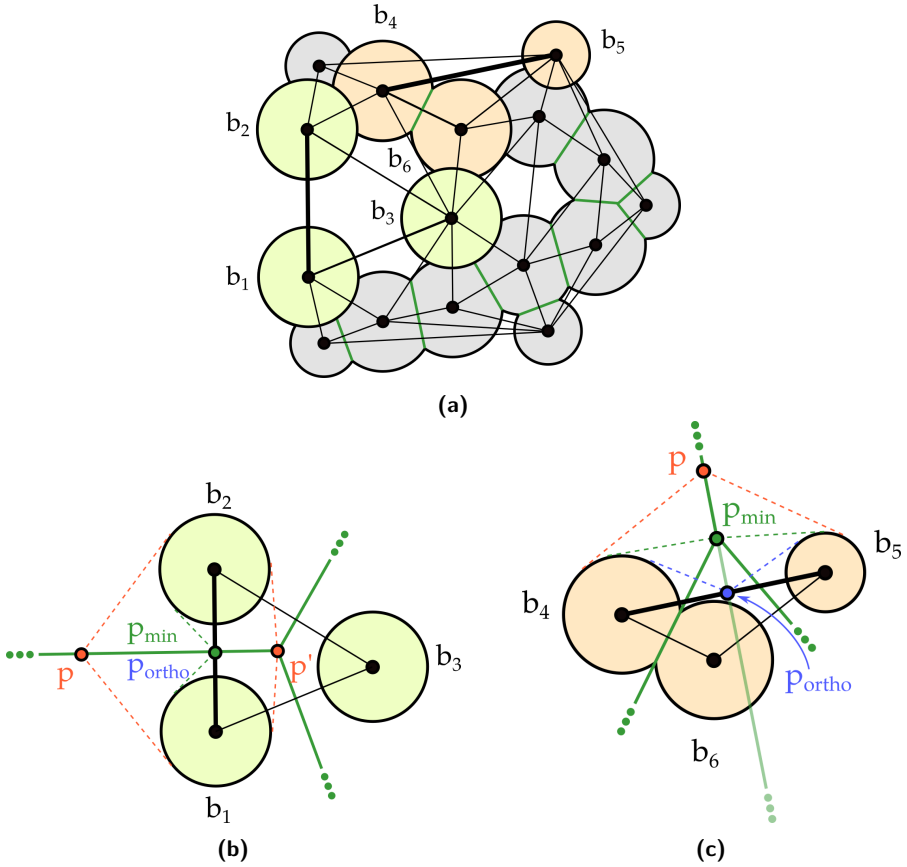


Figure 2 Size and OrthoSize of a simplex. (a) A set B of weighted points. Two edges (bold) belong to the Delaunay triangulation. (b) The Size of edge b_1b_2 is equal to its OrthoSize. Points p, p', p_{min} and p_{ortho} are *witnesses*. Each one is equidistant from b_1 and b_2 and farther away from other disks in B. The distance is proportional to the length of the tangent to the disk that represents the weighted point. The next closest disk from these points is b_3. In this case, p_{min} and p_{ortho} coincide and hence Size = OrthoSize. (c) b_4b_5 is also a Delaunay edge. The location of a neighboring disk b_6 could lead to a different configuration. The point p_{ortho} is closest to b_4 and b_5 among all the points that are equidistant from both. However p_{ortho} is closer to b_6 as compared to b_4 and b_5. The closest point p_{min} that satisfies DT1 and DT2 is farther away, hence for b_4b_5 Size is greater than OrthoSize.

3.1 Outline

The alpha complex of a point set in R^3 consists of simplices of dimensions 0–3, $K_\alpha = K^0_\alpha \cup K^1_\alpha \cup K^2_\alpha \cup K^3_\alpha$, where $K^d_\alpha \subset K_\alpha$ is the set of d-dimensional simplices in K_α. We initialize $K^d_\alpha = \emptyset$ and construct K_α in five steps described below:

Step 1: For $0 \leq d \leq 3$, compute the set of all simplices σ^d such that $\mathsf{OrthoSize}(\sigma^d) \leq \alpha$. Let this set be denoted by $\Sigma_{\mathsf{ortho}} = \Sigma^0_{\mathsf{ortho}} \cup \Sigma^1_{\mathsf{ortho}} \cup \Sigma^2_{\mathsf{ortho}} \cup \Sigma^3_{\mathsf{ortho}}$.

Step 2: For all tetrahedra $\sigma^3 \in \Sigma^3_{\mathsf{ortho}}$, check condition AC2 using $p = p^\sigma_{\mathsf{ortho}}$. If σ^3 satisfies AC2 then insert it into K^3_α.

Step 3: Insert all triangles that are incident on tetrahedra in K^3_α into K^2_α. Let $\Sigma^2_{\mathsf{free}} = \Sigma^2_{\mathsf{ortho}} - \mathsf{Facets}(K^3_\alpha)$, where $\mathsf{Facets}(K^3_\alpha)$ denotes the set of facets of tetrahedra in K^3_α. For all triangles $\sigma^2 \in \Sigma^2_{\mathsf{free}}$, check condition AC2 using $p = p^\sigma_{\mathsf{ortho}}$. If σ^2 satisfies AC2 then insert it into K^2_α.

Step 4: Insert all edges incident on triangles in K^2_α into K^1_α. Let $\Sigma^1_{\mathsf{free}} = \Sigma^1_{\mathsf{ortho}} - \mathsf{Facets}(K^2_\alpha)$, where $\mathsf{Facets}(K^2_\alpha)$ denotes the set of facets of triangles in K^2_α. For all edges $\sigma^1 \in \Sigma^1_{\mathsf{free}}$, check condition AC2 using $p = p^\sigma_{\mathsf{ortho}}$. If σ^1 satisfies AC2 then insert it into K^1_α.

Step 5: Insert all endpoints of edges in K^1_α into K^0_α. Let $\Sigma^0_{\mathsf{free}} = \Sigma^0_{\mathsf{ortho}} - \mathsf{Facets}(K^1_\alpha)$, where $\mathsf{Facets}(K^1_\alpha)$ denotes the set of balls incident on edges in K^1_α. For all balls $b_i = (p_i, r_i) \in \Sigma^0_{\mathsf{free}}$, check condition AC2 using $p = p_i$. If p_i satisfies AC2 then insert it into K^0_α.

Step 1 selects simplices that satisfy AC3. Step 2 recognizes tetrahedra that belong to the alpha complex by checking AC2 using $p = p^\sigma_{\mathsf{ortho}}$. Triangle faces of these tetrahedra also belong to K_α. The other "free" triangles belong to K^2_α if they satisfy AC2. Step 4 identify edges similarly. First all edge faces of triangles in K^2_α are inserted followed by those "free" edges that satisfy AC2. Vertices are identified similarly in Step 5.

A notion related to free simplices, called *unattached* simplices, was introduced by Edelsbrunner [11]. However, the characterization of unattached simplices depends on the fact that they belong to the Delaunay complex.

3.2 Proof of correctness

We now prove that that the algorithm described above correctly computes the alpha complex of the given set of weighted points by proving the following four claims. Each claim states that the set of simplices computed in Steps 2, 3, 4 and 5 are exactly the simplices belonging to the alpha complex. We assume that the input is non-degenerate.

▷ Claim 3. **Step 2 computes K^3_α correctly.**

Proof. For a tetrahedron σ^3, $p^\sigma_{\mathsf{ortho}}$ is the only point that satisfies condition AC1. In Step 2 of the proposed algorithm, we check if AC2 holds for $p^\sigma_{\mathsf{ortho}}$. If yes, then $p^\sigma_{\mathsf{ortho}}$ is a witness for σ^3, *i.e.*, $p^\sigma_{\mathsf{ortho}} = p^\sigma_{\mathsf{min}}$. Further, since $\mathsf{OrthoSize}(\sigma^3) \leq \alpha$ and $p^\sigma_{\mathsf{ortho}} = p^\sigma_{\mathsf{min}}$, we have $\mathsf{Size}(\sigma^3) \leq \alpha$ thereby satisfying AC3. Therefore, σ^3 belongs to K^3_α because it satisfies all three conditions. ◁

We now prove that the algorithm correctly identifies the triangles of the alpha complex.

▶ **Lemma 4.** *A triangle $\sigma^2 \in \Sigma^2_{\mathsf{free}}$ belongs to K^2_α if and only if it satisfies AC2 with $p = p^\sigma_{\mathsf{ortho}}$.*

Proof. We first prove the backward implication, namely if $\sigma^2 \in \Sigma^2_{\mathsf{free}}$ satisfies AC2 with $p = p^\sigma_{\mathsf{ortho}}$, then $\sigma^2 \in K^2_\alpha$. Note that $p^\sigma_{\mathsf{ortho}}$ satisfies AC1 by definition. Further, it satisfies AC2 by assumption and hence $\mathsf{Size}(\sigma^2) = \mathsf{OrthoSize}(\sigma^2)$. We also have $\mathsf{OrthoSize}(\sigma^2) \leq \alpha$ because $\sigma^1 \in \Sigma^2_{\mathsf{free}} \subseteq \Sigma^2_{\mathsf{ortho}}$. So, $\mathsf{Size}(\sigma^2) \leq \alpha$ thereby satisfying AC3. The triangle σ^2 with $p = p^\sigma_{\mathsf{ortho}}$ satisfies all three conditions and hence belongs to K^2_α.

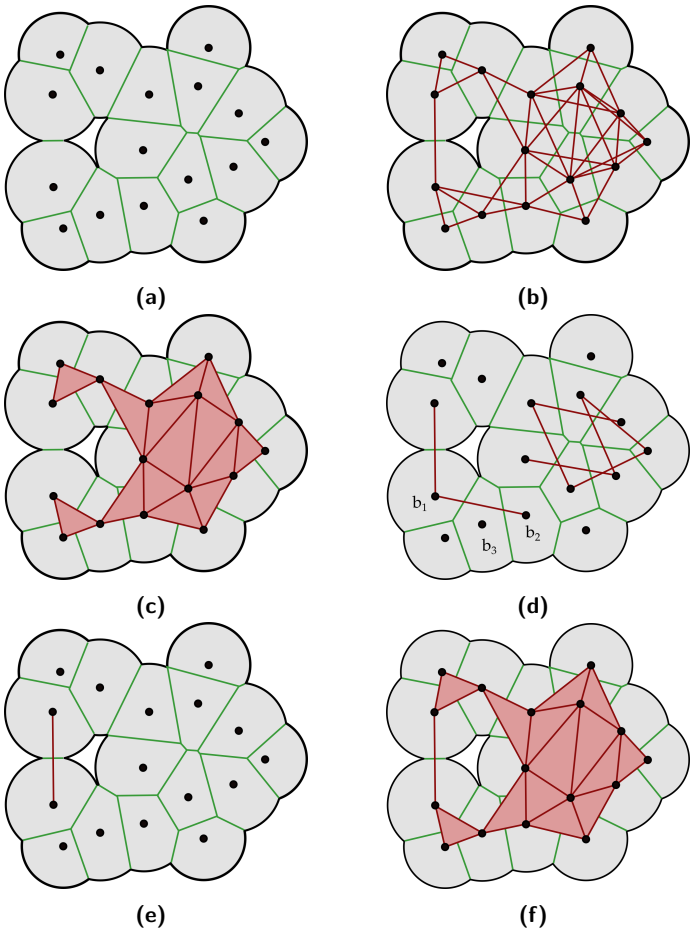


Figure 3 Illustration of the proposed algorithm in 2D. (a) The set of disks B grown by the parameter α. (b) First, compute the set of edges $\Sigma^1_{\mathsf{ortho}}$ whose $\mathsf{OrthoSize} \leq \alpha$ (red). The triangles $\Sigma^2_{\mathsf{ortho}}$ that satisfy this condition are also computed but they are not shown here. (c) Next, identify the triangles that satisfy AC2 (red). (d) Collect edges in $\Sigma^1_{\mathsf{ortho}}$ that are not incident on triangles in K^2_α into Σ^1_{free}. Check if these edges satisfy AC2 with $p = p^\sigma_{\mathsf{ortho}}$. For example, the edge b_1b_2 does not satisfy this condition because b_3 is closer to p_{ortho} than b_1 and b_2. (e) One edge survives the AC2 check and thus belongs to K_α. (f) The alpha complex is obtained as the union of K^2_α, K^1_α and K^0_α.

We will now prove the forward implication via contradiction. Suppose there exists a triangle $\sigma^2 \in \Sigma^2_{\mathsf{free}}$ that belongs to K^2_α but does not satisfy AC2 with $p = p^\sigma_{\mathsf{ortho}}$. In other words, there exists a ball $b_i \in B - \sigma^2$ for which $\pi(p^\sigma_{\mathsf{ortho}}, b_i) < \pi(p^\sigma_{\mathsf{ortho}}, b^\sigma_0)$. Let B_v denote the set of all such balls b_i. The set of points that are equidistant from the three balls $(b^\sigma_0, b^\sigma_1, b^\sigma_2)$ corresponding to σ^2 form a line perpendicular to the plane containing σ^2 called the *radical axis*. Each ball $b_i \in B_v$ partitions the radical axis into two half-intervals based on whether the point on radical axis is closer to b_i or to b^σ_0, see Figure 4. Let $I^+(b_i)$ denote the half interval consisting of points that are closer to b^σ_0 compared to b_i. Let $I^+(B_v)$ denote the intersection of all such half intervals $I^+(b_i)$. We have assumed that $\sigma^2 \in K^2_\alpha$, so there must exist a closest witness p^σ_{min}, and it lies within $I^+(B_v)$. Thus, $I^+(B_v)$ is non-empty. In fact, $I^+(B_v) = I^+(b_j)$ for some $b_j \in B_v$ and p^σ_{min} is exactly the end point of $I^+(b_j)$. This implies that p^σ_{min} is also a closest witness for the tetrahedron $\sigma^3 = (b^\sigma_0, b^\sigma_1, b^\sigma_2, b_j)$. So, σ^3 belongs to K^3_α and its Size is equal to $\mathsf{Size}(\sigma^2)$. However, this means that $\sigma^2 \notin \Sigma^2_{\mathsf{free}}$, a contradiction. So, the forward implication in the lemma is true. ◀

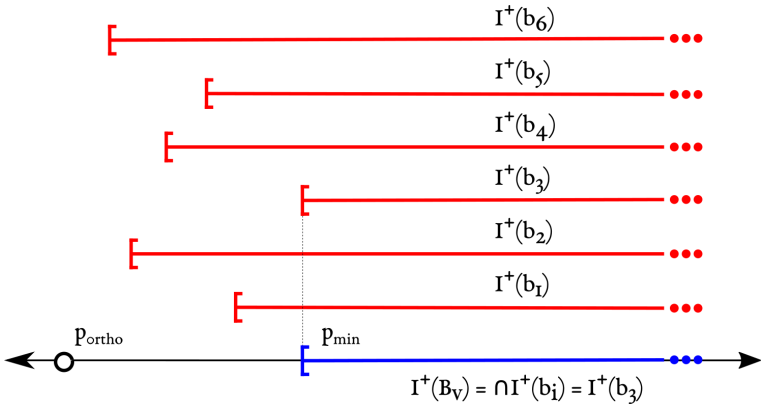


Figure 4 The radical axis of a triangle σ^2 is drawn such that p^σ_{ortho} is at the origin. A ball $b_i \in B - \sigma^2$ divides the radical axis into two half intervals. Points in the half interval $I^+(b_i)$ are closer to b^σ_0 as compared to b_i, *i.e.* for all $p \in I^+(b_i)$, $\pi(p, b^\sigma_0) < \pi(p, b_i)$. Consider the set B_v of balls that are closer to p^σ_{ortho} as compared to b^σ_0 So, $I^+(b_i)$ does not contain p^σ_{ortho}. The intersection of these intervals, denoted by $I^+(B_v)$, is equal to one of the intervals $I^+(b_i)$. Here, $I^+(B_v) = I^+(b_3)$. The end point of the interval $I^+(b_3)$ is the closest witness for the tetrahedron $\sigma^2 \cup b_3$.

▷ Claim 5. **Step 3 computes K^2_α correctly.**

Proof. If a simplex σ^3 belongs to K_α then naturally all of its faces also belong to K_α. The algorithm includes such triangles into K^2_α and remove them from Σ^2_{ortho} to obtain the set of free triangles Σ^2_{free}. It follows directly from Lemma 4 that AC2 is a necessary and sufficient condition for a triangle in Σ^2_{free} to belong to K^2_α. Hence, Step 3 correctly computes the triangles belonging to K^2_α. ◁

The above arguments need to be extended to prove that the edges of the alpha complex are also correctly identified.

▶ **Lemma 6.** *An edge $\sigma^1 \in \Sigma^1_{\text{free}}$ belongs to K^1_α if and only if it satisfies the condition AC2 with $p = p^\sigma_{\text{ortho}}$.*

The proof is similar to that of Lemma 4 and appears in the full version [29]. A general result is likely true for d-dimensional simplices in Σ^d_{free}. However, given the focus on alpha complexes in $\mathbb{R}^3$, we prefer to state and prove these results specific to lower dimensions. We also prefer to provide individual proofs for edges and triangles because it simplifies the exposition and could also potentially help in the design of improved data structures to accelerate computation of different steps of the algorithm.

▷ Claim 7. **Step 4 computes K^1_α correctly.**

Proof. All edge faces of triangles in K^2_α naturally belong to K^1_α. Step 4 inserts all edges incident on triangles in K^2_α into K^1_α as valid edges and removes them from Σ^1_{ortho} to obtain the set of free edges Σ^1_{free}. It follows directly from Lemma 6 that AC2 is a necessary and sufficient condition for an edge $\sigma^1 \in \Sigma^1_{\text{free}}$ to belong to K^1_α. Therefore, Step 4 correctly computes the edges belonging to K^1_α. ◁

▷ Claim 8. **Step 5 computes K^0_α correctly.**

Proof. All vertices incident on K^1_α naturally belong to K^0_α. Step 5 inserts all such vertices in K^0_α as valid vertices and removes them from Σ^0_{ortho} to obtain the set of free vertices Σ^0_{free}. Next, the vertices in Σ^0_{free} for which the center of the ball $b_i = (p_i, r_i)$ satisfies AC2 are also

inserted into K^0_α. Clearly, these vertices also satisfy AC3 because they belong to $\Sigma^0_{\mathsf{ortho}}$. The condition AC1 is not relevant for 0-dimensional simplices. Therefore, these vertices clearly belong to the alpha complex. Similar to Lemmas 4 and 6, it is easy to prove that checking for AC2 for $p = p_i$ is necessary and sufficient condition to decide whether a vertex in Σ^0_{free} belongs to the alpha complex. That is, it is possible to show that vertices in alpha complex that have non-empty Voronoi regions but do not satisfy AC2 for $p = p_i$ would be incident on some edge in K^1_α, and therefore must have been already detected by Step 4 and hence can not belong to Σ^0_{free}. Therefore, Step 5 correctly computes the vertices belonging to K^0_α. ◁

4 Parallel algorithm for biomolecules

Although the algorithm as described above is provably correct, a straightforward implementation will be extremely inefficient with a worst-case running time of $O(n^5)$, where n is the number of weighted points in B. This is because Step 1 requires $O(n^4)$ time to generate all possible tetrahedra. In later steps, we need $O(n)$ effort per simplex to check AC2. However, the input corresponds to atoms in a biomolecule. We show how certain properties of biomolecules can be leveraged to develop a fast parallel implementation.

4.1 Biomolecular data characteristics

Atoms in a biomolecule are well distributed. The following three properties of biomolecules are most relevant:

- The radius of an atom is bounded. The typical radius of an atom in a protein molecule ranges between 1Å to 2Å [3]. Further, a protein molecule contains upwards of thousand atoms. So, the radius is small compared to the total size of the molecule.
- There is a lower bound on the distance between the centres of two atoms. This is called the *van der Waals contact distance*, beyond which the two atoms start repelling each other. In the case of atoms in protein molecules, this distance is at least 1Å. This property together with the upper bound on atomic radii ensures that no atom is completely contained inside another. This means that the weighted Voronoi regions corresponding to the atoms in a biomolecule can be always be assumed to be non-empty.
- Structural biologists are interested in small values of α. The two crucial values are 0Å and 1.4Å. The former corresponds to using van der Waals radius and the latter corresponds to the radius of water molecule, which acts as the solvent.

In the light of the above three properties, we can say that the number of simplices of the alpha complex that are incident on a weighted point (atom) is independent of the total number of input atoms and hence bounded by a constant [22].

4.2 Acceleration data structure

The algorithm will benefit from an efficient method for accessing points of B that belong to a local neighborhood of a given weighted point. We store the weighted points in a grid-based data structure. Let r_{max} denote the radius of the largest atom and assume that the value of the parameter α is available as input. First, we construct a grid with cells of side length $\sqrt{r^2_{max} + \alpha}$ and then bin the input atoms into the grid cells. In our implementation, we do not store the grid explicitly because it may contain several empty cells. Instead, we compute the cell index for each input atom and sort the list of atoms by cell index to ensure that atoms that belong to a particular cell are stored at consecutive locations. The cell index is determined based on a row-major or column-major order. Alternatively, a space-filling curves like the Hilbert curve could also be used to order the cells.

After the atoms are stored in grid cells, the alpha complex is computed in two stages. In the first stage, we employ a bottom-up approach to obtain a conservative estimate of the edges, triangles, and tetrahedra belonging to the alpha complex. The false positives from the first stage are removed in a subsequent pruning stage resulting in the correct alpha complex. We describe these two stages in the following subsections.

4.3 Potential simplices

The first stage essentially corresponds to Step 1 of the algorithm described in the previous section. We compute the set Σ_{ortho} of potential simplices for which $\mathsf{OrthoSize}(\sigma^d) \leq \alpha$. However, for efficiency reasons we process the simplices in the order of increasing dimension. First, we identify edges that satisfy the AC3 condition as described below. Given the size of the grid cell, endpoints of edges that satisfy the condition either lie within the same grid cell or in adjacent cells. So, the grid data structure substantially reduces the time required to compute the list of potential edges $\Sigma^1_{\mathsf{ortho}}$. Beginning from this set of edges, we construct the set of all possible triangles and retain the triangles whose $\mathsf{OrthoSize}$ is no greater than α, resulting in the set $\Sigma^2_{\mathsf{ortho}}$. Finally, we use the triangles in $\Sigma^2_{\mathsf{ortho}}$ to construct the list of tetrahedra that satisfy the $\mathsf{OrthoSize} \leq \alpha$ condition. The above procedure works because the $\mathsf{OrthoSize}$ of a simplex is always greater than or equal to the $\mathsf{OrthoSize}$ of its faces. The set of simplices identified in this stage contains all simplices of the alpha complex. False positives are pruned in the second stage described below.

4.4 Pruning

The second stage corresponds to Steps 2-5 of the algorithm and processes the potential simplices in the decreasing order of dimension. This stage checks the characterizing condition AC2 to prune Σ_{ortho} into K_α. The tetrahedra are processed by checking if any of the input balls are closer to the *ortho-center* than the balls incident on the tetrahedron. If yes, the tetrahedron is pruned away. Else, the tetrahedron is recognized as belonging to the alpha complex and inserted into K^3_α. Triangles incident on these tetrahedra also belong to the alpha complex and are inserted into K^2_α after they are removed from the list of potential triangles $\Sigma^2_{\mathsf{ortho}}$. Next, the triangles in $\Sigma^2_{\mathsf{ortho}}$ are processed by checking if they satisfy AC2. If yes, they are inserted into K^2_α. Otherwise, they are pruned away. All edges incident on triangles belonging to K^2_α are inserted into K^1_α and removed from the set $\Sigma^1_{\mathsf{ortho}}$. Next, the edges in $\Sigma^1_{\mathsf{ortho}}$ are processed by checking if they satisfy AC2. Edges that satisfy AC2 are inserted into K^1_α and the others are pruned away. All the vertices in $\Sigma^0_{\mathsf{ortho}}$ are directly inserted into K^0_α without the AC2 check because for biomolecular data we assume that Voronoi regions of all the atoms are non-empty. The check for condition AC2 for each simplex is again made efficient by the use of the grid data structure. Atoms that may violate AC2 lie within the same cell as that containing the *ortho-center* or within the adjacent cells. Atoms that lie within other cells may be safely ignored.

4.5 CUDA implementation

We use the CUDA framework [7] and the `thrust` library [8] within CUDA to develop a parallel implementation of the algorithm that executes on the many cores of the GPU. The grid computation is implemented as a CUDA kernel where all atoms are processed in parallel. The computation of potential simplices and pruning stages are broken down into multiple CUDA kernels and parallelized differently in order to increase efficiency. We now describe the parallelization strategy in brief.

For computing the set of potential edges, the initial enumeration of possible edges incident on an atom is done using the atoms in the corresponding grid cell and its neighbouring cells. This is done per atom in parallel, the thread corresponding to the atom i being responsible for generating the edges $ij, j > i$ to ensure no duplicate edges are generated. Subsequently, the AC3 condition is checked for the edges in parallel to finally generate the list of potential edges $\Sigma^1_{\mathsf{ortho}}$. For computing potential triangles $\Sigma^2_{\mathsf{ortho}}$, the potential edge list is used as a starting point for the initial enumeration of all possible triangles. This step is also parallelized per atom, the thread i being responsible for generation of triangles of the type $ijk; j, k > i$ if all three edges ij, ik and jk are potential edges. The AC3 condition for the triangle is checked next within a separate kernel and parallelized per triangle to generate the potential triangles $\Sigma^2_{\mathsf{ortho}}$. A similar strategy is used for computing the set of potential tetrahedra $\Sigma^3_{\mathsf{ortho}}$.

The pruning stage is parallelized per tetrahedron, triangle, and edge as required. So, computation of $p^{\sigma}_{\mathsf{ortho}}$ is done in parallel. The grid data structure is again useful in checking for potential violators of the AC2 condition. Only the atoms belonging to the grid cell corresponding to $p^{\sigma}_{\mathsf{ortho}}$ or the those in neighbouring grid cells can violate the AC2 condition.

4.6 Handling large data sizes

Typical protein structures consist of up to 100,000 atoms. Our implementation can handle datasets of this size easily for reasonable values of α. However, the size of datasets is ever increasing. Protein complexes that are available nowadays may consist of millions of atoms, necessitating smart management of GPU memory while handling such data sets.

We propose two strategies and implement one of them. The first strategy is to partition the grid by constructing an octree data structure and choosing an appropriate level in the octree to create partitions. Each partition together with its border cells can be processed independently of other partitions. So, we can copy one partition and its border to the GPU memory, compute its alpha complex, and copy the results back from GPU to CPU memory. After all the partitions are processed, the list of simplices can be concatenated followed by duplicate removal to generate the final alpha complex.

The second strategy is to partition the sorted list of atoms into *chunks* of equal sizes and to process each chunk independently. Here, we assume that the complete list of atoms together with the grid data structure fits in the GPU memory. This is a reasonable assumption considering that datasets containing several million atoms can easily fit on modern GPUs, which typically have at least 2GB video memory. Also, the main difficulty in handling large protein structures is managing the large lists of simplices generated within the intermediate steps of the algorithm, when compared to handling the input list of atoms or the output list of simplices. We compute the alpha complex by executing the algorithm in multiple passes. Each pass computes the alpha complex for a single chunk and copies it back to the CPU memory. We have implemented this second strategy and can handle data sizes of up to 16 million atoms on a GPU with 2GB of memory. Results are reported in the next section.

5 Experimental results

We now present results of computational experiments, which demonstrate that the parallel algorithm is fast in practice and significantly better than the state-of-the-art. We also performed runtime profiling to better understand the bottlenecks and effect of the parameter α on the runtime. We present results for α in the range 0.0 to 2.0. This range is important for structural analysis of biomolecules as it corresponds to *solvent accessible surface* of the biomolecule for typical solvent molecules like water (van der Waals radius = 1.4Å). The value

$\alpha = 0$ corresponds to the *van der Waals* surface of the biomolecule. All experiments, unless stated otherwise, were performed on a Linux system with an nVidia GTX 660 Ti graphics card running CUDA 8.0 and a 2.0GHz Intel Xeon octa core processor with 16 GB of main memory. The default number of threads per block was set at 512 for all the CUDA kernels.

Mach and Koehl describe two techniques for computing alpha complex of biomolecules called *AlphaVol* and *UnionBall* in their paper [27]. Both approaches construct the weighted Delaunay triangulation of input atoms first followed by a filtering step to obtain the alpha complex. *UnionBall* is the state-of-the-art technique for alpha complex computation for biomolecules on multi-core CPU. It uses heuristics and optimizations specific to biomolecular data to improve upon *AlphaVol*. For biomolecules containing 5 million atoms, *AlphaVol* takes approximately 8600 seconds for computing the alpha complex, while *UnionBall* takes approximately 150 seconds. Our method computes the alpha complex in less than 3 seconds for similar sized data, see Table 1.

5.1 Comparison with *gReg3D*

Table 1 Runtime comparison of the proposed algorithm with *gReg3D* on an nVidia GTX 660 Ti graphics card. Timings are reported in milliseconds. %Simplex refers to the size of the alpha complex as a percentage of the size of the weighted Delaunay triangulation. The last column shows the speedup in runtime of our algorithm over *gReg3D*. "*" indicates the data was partitioned and processed in chunks. "–" indicates that the code could not execute due to insufficient memory.

α	PDB id	#Atoms	K_α		*gReg3D*		%Simplex	Speed up
			#Simplices	Time(ms)	#Simplices	Time(ms)		
0.0	1GRM	260	932	13	6295	117	14.8	**9.0**
	1U71	1505	5696	13	40878	115	13.9	**11.1**
	3N0H	1509	5739	14	41244	137	13.9	**10.0**
	4HHB	4384	38796	29	150141	193	25.8	**6.6**
	2J1N	8142	29642	18	227719	229	13.0	**12.7**
	1K4C	16068	62851	27	446383	347	14.1	**12.9**
	2OAU	16647	123175	56	466586	344	26.4	**6.2**
	1AON	58674	262244	65	1650841	879	15.9	**13.5**
	1X9P*	217920	924086	113	6142811	2555	15.0	**22.6**
	1IHM*	677040	2713083	277	–	–	–	–
	4CWU*	5905140	23450403	2709	–	–	–	–
	3IYN*	5975700	24188892	2874	–	–	–	–
1.0	1GRM	260	1598	15	6295	117	25.4	**7.9**
	1U71	1505	10828	17	40878	115	26.5	**8.5**
	3N0H	1509	10965	30	41244	137	26.6	**4.6**
	4HHB	4384	65987	86	150141	193	44.0	**2.2**
	2J1N	8142	58205	30	227719	229	25.6	**7.6**
	1K4C	16068	118467	52	446383	347	26.5	**6.7**
	2OAU	16647	199101	159	466586	344	42.7	**2.2**
	1AON	58674	495683	160	1650841	879	30.0	**5.5**
	1X9P*	217920	1653778	196	6142811	2555	26.9	**13.0**
	1IHM*	677040	5058507	605	–	–	–	–
	4CWU*	5905140	44411353	5118	–	–	–	–
	3IYN*	5975700	45790463	5501	–	–	–	–

We are not aware of any available software that can compute the alpha complex directly without first constructing the complete Delaunay triangulation. In order to compare the performance, we chose the state-of-the-art parallel algorithm for computing the weighted Delaunay triangulation in 3D, *gReg3D* [5]. The CUDA implementation of *gReg3D* is available in the public domain. Table 1 compares the running times of our proposed algorithm with that of *gReg3D* for twelve different biomolecules at $\alpha = 0$ and $\alpha = 1$. As evident from the table, we consistently observe significant speedup over *gReg3D*. The observed speedup is as high as a factor of 22 for the biomolecule 1X9P at $\alpha = 0$, one of the largest molecules in our dataset. Clearly, the speedup goes down for $\alpha = 1$ when compared to $\alpha = 0$ because of the increased number of simplices in the output alpha complex. We also report the number of simplices in the alpha complex compared to the total number of simplices in the Delaunay triangulation under the column "%Simplex". This makes it clear why the speedup decreases as α is increased from 0 to 1. For example, for the protein 1AON, the fraction of alpha complex simplices increases from 15.9% to 30% as α is increased from 0 to 1. Correspondingly, the speedup decreases from a factor of 13.5 to 5.5. We repeated the experiment on a MS Windows system with an nVidia GTX 980 Ti card running CUDA 8.0 and observed similar speedups. However, the individual runtimes both for our algorithm and for *gReg3D* were higher on the GTX 980 Ti.

The starred entries in Table 1 are results for execution using the data partitioning approach. This is necessitated because these four large molecules generate large intermediate simplex lists that can not fit into the GPU memory if all the atoms in the molecule are processed at once. We observe that *gReg3D* is able to successfully compute the Delaunay complex for only one out of these four large molecules and runs out of GPU memory for the remaining three molecules.

We discuss the run time profiling, the effect of the value of α and numerical issues in more detail in the full version of the paper [29].

6 Conclusions

We proposed a novel parallel algorithm to compute the alpha complex for biomolecular data that does not require prior computation of the complete Delaunay triangulation. The useful characterization of simplices that belong to the alpha complex may be of independent interest. The algorithm was implemented using CUDA, which exploits the characteristics of the atom distribution in biomolecules to achieve speedups of up to a factor of 22 compared to the state-of-the-art parallel algorithm for computing the weighted Delaunay triangulation, and up to a factor of 50 speedup over the state-of-the-art implementation that is optimized for biomolecules. In future work, we plan to further improve the runtime efficiency of the parallel implementation and to resolve the numerical issues using real arithmetic.

Applications of alpha complex outside the domain of biomolecular analysis often require the complete filtration of Delaunay complex. The algorithm as presented here is not best suited for such cases. However, the algorithm may be modified to utilize a previously computed alpha complex to efficiently compute the alpha complex for higher values of α. We plan to investigate this extension in future work.

References

1 Franz Aurenhammer, Rolf Klein, and Der-Tsai Lee. *Voronoi Diagrams and Delaunay Triangulations*. World Scientific, 2013. doi:10.1142/8685.

2 Ulrich Bauer and Herbert Edelsbrunner. The Morse theory of Čech and Delaunay filtrations. In Siu-Wing Cheng and Olivier Devillers, editors, *30th Annual Symposium on Computational Geometry, SOCG'14, Kyoto, Japan, June 08 - 11, 2014*, page 484. ACM, 2014. doi:10.1145/2582112.2582167.

3 A Bondi. van der Waals volumes and radii. *The Journal of Physical Chemistry*, 68(3):441–451, 1964.

4 Adrian Bowyer. Computing Dirichlet tessellations. *Comput. J.*, 24(2):162–166, 1981. doi:10.1093/comjnl/24.2.162.

5 Thanh-Tung Cao, Ashwin Nanjappa, Mingcen Gao, and Tiow Seng Tan. A GPU accelerated algorithm for 3d Delaunay triangulation. In John Keyser and Pedro V. Sander, editors, *Symposium on Interactive 3D Graphics and Games, I3D '14, San Francisco, CA, USA - March 14-16, 2014*, pages 47–54. ACM, 2014. doi:10.1145/2556700.2556710.

6 Paolo Cignoni, Claudio Montani, and Roberto Scopigno. DeWall: A fast divide and conquer Delaunay triangulation algorithm in E^d. *Comput. Aided Des.*, 30(5):333–341, 1998. doi:10.1016/S0010-4485(97)00082-1.

7 Nvidia Corporation. CUDA Zone. https://developer.nvidia.com/cuda-zone, 2020. [Online; accessed 17-March-2020].

8 Nvidia Corporation. Thrust. https://developer.nvidia.com/thrust, 2020. [Online; accessed 17-March-2020].

9 Tran Kai Frank Da, Sébastien Loriot, and Mariette Yvinec. 3D alpha shapes. In *CGAL User and Reference Manual*. CGAL Editorial Board, 4.11 edition, 2017. URL: http://doc.cgal.org/4.11/Manual/packages.html#PkgAlphaShapes3Summary.

10 Joe Dundas, Zheng Ouyang, Jeffery Tseng, T. Andrew Binkowski, Yaron Turpaz, and Jie Liang. CASTp: computed atlas of surface topography of proteins with structural and topographical mapping of functionally annotated residues. *Nucleic Acids Research*, 34(Web-Server-Issue):116–118, 2006. doi:10.1093/nar/gkl282.

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

12 H. Edelsbrunner and P. Koehl. The geometry of biomolecular solvation. In *Discrete and Computational Geometry*, pages 243–275. MSRI Publications, 2005.

13 Herbert Edelsbrunner. *Geometry and Topology for Mesh Generation*, volume 7 of *Cambridge monographs on applied and computational mathematics*. Cambridge University Press, 2001.

14 Herbert Edelsbrunner. Alpha shapes – a survey. In R. van de Weygaert, G. Vegter, J. Ritzerveld, and V. Icke, editors, *Tesellations in the Sciences: Virtues, Techniques and Applications of Geometric Tilings*, 2011.

15 Herbert Edelsbrunner and John Harer. *Computational Topology - an Introduction*. American Mathematical Society, 2010. URL: http://www.ams.org/bookstore-getitem/item=MBK-69.

16 Herbert Edelsbrunner, David G. Kirkpatrick, and Raimund Seidel. On the shape of a set of points in the plane. *IEEE Trans. Inf. Theory*, 29(4):551–558, 1983. doi:10.1109/TIT.1983.1056714.

17 Herbert Edelsbrunner and Ernst P. Mücke. Simulation of simplicity: a technique to cope with degenerate cases in geometric algorithms. *ACM Trans. Graph.*, 9(1):66–104, 1990. doi:10.1145/77635.77639.

18 Herbert Edelsbrunner and Ernst P. Mücke. Three-dimensional alpha shapes. *ACM Trans. Graph.*, 13(1):43–72, 1994. doi:10.1145/174462.156635.

19 Herbert Edelsbrunner and Raimund Seidel. Voronoi diagrams and arrangements. *Discret. Comput. Geom.*, 1:25–44, 1986. doi:10.1007/BF02187681.

20 Herbert Edelsbrunner and Nimish R. Shah. Incremental topological flipping works for regular triangulations. *Algorithmica*, 15(3):223–241, 1996. doi:10.1007/BF01975867.

21 Leonidas J. Guibas, Donald E. Knuth, and Micha Sharir. Randomized incremental construction of delaunay and voronoi diagrams. *Algorithmica*, 7(4):381–413, 1992. doi:10.1007/BF01758770.

22 Dan Halperin and Mark H. Overmars. Spheres, molecules, and hidden surface removal. *Comput. Geom.*, 11(2):83–102, 1998. doi:10.1016/S0925-7721(98)00023-6.

23 Michael Krone, Barbora Kozlíková, Norbert Lindow, Marc Baaden, Daniel Baum, Július Parulek, Hans-Christian Hege, and Ivan Viola. Visual analysis of biomolecular cavities: State of the art. *Comput. Graph. Forum*, 35(3):527–551, 2016. doi:10.1111/cgf.12928.

24 J. Liang, H. Edelsbrunner, P. Fu, P.V. Sudhakar, and S. Subramaniam. Analytical shape computation of macromolecules: I. molecular area and volume through alpha shape. *Proteins Structure Function and Genetics*, 33(1):1–17, 1998.

25 J. Liang, H. Edelsbrunner, P. Fu, P.V. Sudhakar, and S. Subramaniam. Analytical shape computation of macromolecules: II. inaccessible cavities in proteins. *Proteins Structure Function and Genetics*, 33(1):18–29, 1998.

26 J. Liang, H. Edelsbrunner, and C. Woodward. Anatomy of protein pockets and cavities. *Protein Science*, 7(9):1884–1897, 1998.

27 Paul Mach and Patrice Koehl. Geometric measures of large biomolecules: Surface, volume, and pockets. *Journal of Computational Chemistry*, 32(14):3023–3038, 2011. doi:10.1002/jcc.21884.

28 Talha Bin Masood and Vijay Natarajan. An integrated geometric and topological approach to connecting cavities in biomolecules. In Chuck Hansen, Ivan Viola, and Xiaoru Yuan, editors, *2016 IEEE Pacific Visualization Symposium, PacificVis 2016, Taipei, Taiwan, April 19-22, 2016*, pages 104–111. IEEE Computer Society, 2016. doi:10.1109/PACIFICVIS.2016.7465257.

29 Talha Bin Masood, Tathagata Ray, and Vijay Natarajan. Parallel computation of alpha complex for biomolecules. *CoRR*, abs/1908.05944, 2019. arXiv:1908.05944.

30 Talha Bin Masood, Sankaran Sandhya, Nagasuma R. Chandra, and Vijay Natarajan. CHEXVIS: a tool for molecular channel extraction and visualization. *BMC Bioinform.*, 16:119:1–119:19, 2015. doi:10.1186/s12859-015-0545-9.

31 Ashwin Nanjappa. *Delaunay triangulation in $\mathbb{R}^3$ on the GPU*. PhD thesis, National University of Singapore, 2012.

32 Donald R. Sheehy. An output-sensitive algorithm for computing weighted α-complexes. In *Proceedings of the 27th Canadian Conference on Computational Geometry, CCCG 2015, Kingston, Ontario, Canada, August 10-12, 2015*. Queen's University, Ontario, Canada, 2015. URL: http://research.cs.queensu.ca/cccg2015/CCCG15-papers/42.pdf.

33 Raghavendra Sridharamurthy, Talha Bin Masood, Harish Doraiswamy, Siddharth Patel, Raghavan Varadarajan, and Vijay Natarajan. Extraction of robust voids and pockets in proteins. In Lars Linsen, Bernd Hamann, and Hans-Christian Hege, editors, *Visualization in Medicine and Life Sciences III*, Mathematics and Visualization, pages 329–349. Springer, 2016. doi:10.1007/978-3-319-24523-2_15.

34 David F Watson. Computing the n-dimensional Delaunay tessellation with application to Voronoi polytopes. *The Computer Journal*, 24(2):167–172, 1981.

Relative Persistent Homology

Nello Blaser
Department of Informatics, University of Bergen, Norway
nello.blaser@uib.no

Morten Brun
Department of Mathematics, University of Bergen, Norway
morten.brun@uib.no

Abstract

The alpha complex efficiently computes persistent homology of a point cloud X in Euclidean space when the dimension d is low. Given a subset A of X, relative persistent homology can be computed as the persistent homology of the relative Čech complex $\check{C}(X, A)$. But this is not computationally feasible for larger point clouds X. The aim of this note is to present a method for efficient computation of relative persistent homology in low dimensional Euclidean space. We introduce the relative Delaunay-Čech complex $\mathrm{Del}\check{C}(X, A)$ whose homology is the relative persistent homology. It is constructed from the Delaunay complex of an embedding of X in $(d+1)$-dimensional Euclidean space.

2012 ACM Subject Classification Mathematics of computing → Algebraic topology

Keywords and phrases topological data analysis, relative homology, Delaunay-Čech complex, alpha complex

Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.18

1 Introduction

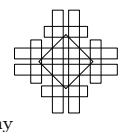
Persistent homology is receiving growing attention in the machine learning community. In that light, the scalability of persistent homology computations is of increasing importance. To date, the alpha complex is the most widely used method to compute persistent homology for large low-dimensional data sets.

Relative persistent homology has been considered several times in recent years. For example Edelsbrunner and Harrer [8] have presented an application of relative persistent homology to estimate the dimension of an embedded manifold. Relative persistent homology is also a way to introduce the concept of extended persistence [5]. De Silva and others have shown that the relative persistent homology $H_*(X, A_t)$ with an increasing family of sets A_t and a constant $X = \cup_t A_t$, and the corresponding relative persistent cohomology have the same barcode [6]. They also show that absolute persistent homology of A_t can be computed from this particular type of relative persistent homology. More recently, Pokorny and others [9] have used relative persistent homology to cluster two-dimensional trajectories. Some software, such as PHAT [2], even allows for the direct computation of relative persistent homology. For an example see the PHAT github repository.

Despite the fact that relative persistent homology has been considered in many different situations, we are not aware of a relative version of the alpha- or Delaunay-Čech complexes being used.

Our contributions are as follows.

1. We give a new elementary proof that the Delaunay-Čech complex is level homotopy equivalent to the Čech complex. This has previously been shown using discrete Morse theory [1].
2. We extend this proof to the relative versions of the Delaunay-Čech complex and the Čech complex.



3. We explain how the relative Delaunay-Čech complex can be constructed through embedding in a higher dimension.

Given finite $A \subseteq X \subseteq \mathbb{R}^d$, these contributions lead to the constuction of a filtered simplicial complex $\mathrm{Del\check{C}}(X, A)$ with persistent homology isomorphic to the relative persistent homology of Čech persistence modules $\check{\mathrm{C}}_*(X;k)/\check{\mathrm{C}}_*(A;k)$. The underlying simplicial complex of $\mathrm{Del\check{C}}(X, A)$ is the Delaunay complex of an embedding Z of X in $\mathbb{R}^{d+1}$ with the property that the projection $\mathrm{pr}\colon \mathbb{R}^{d+1} \to \mathbb{R}^d$ takes Z onto X. All simplices in the Delaunay complex of Z projecting to a subset of A are given filtration value zero. The filtration value of the remaining simplices in the Delaunay complex of Z is defined to be the Čech filtration value of their projection to $\mathbb{R}^d$. This is the content of Theorem 2.

This manuscript is structured as follows. In Section 2, we introduce relative persistent homology, and in Section 3 we construct the relative Delaunay-Čech complex. The rest of the paper serves to prove that the relative Delaunay-Čech complex is level homotopy equivalent to the relative Čech complex. Section 4 introduces Dowker Nerves, the theoretical foundation used in the proof. In Section 5, we introduce the alpha- and Delaunay-Čech complexes using the Dowker Nerve terminology and show that they are level homotopy equivalent to the Čech complex. Section 6 introduces the relative alpha- and Delaunay-Čech dissimilarities, and proves that their nerves are level homotopy equivalent to the relative Čech complex. Finally, in Section 7 we show that the nerve of the relative Delaunay-Čech dissimilarity is level homotopy equivalent to the relative Delaunay-Čech complex.

2 Relative persistent homology

Let X be a finite subset of Euclidean space $\mathbb{R}^d$. Given $t > 0$, the Čech complex $\check{\mathrm{C}}_t(X)$ of X is the abstract simplicial complex with vertex set X and with $\sigma \subseteq X$ a simplex of $\check{\mathrm{C}}_t(X)$ if and only if there exists a point $p \in \mathbb{R}^d$ with distance less than t to every point in σ. Varying t we obtain the filtered Čech complex $\check{\mathrm{C}}(X)$.

Given a subset A of X we obtain an inclusion $\check{\mathrm{C}}(A) \subseteq \check{\mathrm{C}}(X)$ of filtered simplicial complexes and an induced inclusion $\check{\mathrm{C}}_*(A;k) \subseteq \check{\mathrm{C}}_*(X;k)$ of associated chain complexes of persistence modules over the field k. The *relative persistent homology of the pair* (X, A) is defined as the homology of the factor chain complex of persistence modules $\check{\mathrm{C}}_*(X;k)/\check{\mathrm{C}}_*(A;k)$.

For X of small cardinality, the relative persistent homology can be calculated as the reduced persistent homology of the relative Čech complex $\check{\mathrm{C}}(X, A)$, where $\sigma \subseteq X$ is a simplex of $\check{\mathrm{C}}(X, A)_t$ if either $\sigma \subseteq A$ or $\sigma \in \check{\mathrm{C}}_t(X)$. However, as the cardinality of X grows, this quickly becomes computationally infeasible.

3 The relative Delaunay-Čech complex

Before delving into theory we present a filtered simplicial complex that is level homotopy equivalent to the relative Čech complex $\check{\mathrm{C}}(X, A)$ of a pair of finite subsets $A \subseteq X$ of Euclidean space $\mathbb{R}^d$. Two filtered simplicial complexes $K = (K_t)_{t\geq 0}$ and $L = (L_t)_{t\geq 0}$ are *level homotopy equivalent* if there exists a filtered simplicial map $f\colon K \to L$ so that the geometric realizaton of $f_t\colon K_t \to L_t$ is a homotopy equivalence for each t.

For convenience, we let $B = X - A$ so that X is the disjoint union of A and B. Choose $s > 0$ bigger than the maximal filtration values in the alpha complexes of A and B. The set

$$Z = A \times \{s\} \cup B \times \{-s\}$$

is an embedding of X in $\mathbb{R}^{d+1}$. Let $\mathrm{Del}(Z)$ be the Delaunay complex of Z.

▶ **Definition 1.** *The* relative Delaunay-Čech complex *of the finite subsetes $A \subseteq X$ of $\mathbb{R}^d$ is the filtered simplicial complex* $\mathrm{Del\check{C}}(X, A)$ *with* $\mathrm{Del}(Z)$ *as underlying simplicial complex and with filtration $R\colon \mathrm{Del}(Z) \to \mathbb{R}$ defined as follows: Given $\sigma \in \mathrm{Del}(Z)$, let* $\mathrm{pr}(\sigma)$ *be the projection of $\sigma \subseteq \mathbb{R}^{d+1}$ to $\mathbb{R}^d$. If* $\mathrm{pr}(\sigma)$ *is contained in A we let $R(\sigma) = 0$. Otherwise we let $R(\sigma)$ be the radius of the smallest enclosing ball of* $\mathrm{pr}(\sigma)$.

▶ **Theorem 2.** *The filtered simplicial complex* $\mathrm{Del\check{C}}(X, A)$ *is level homotopy equivalent to the relative Čech complex* $\check{\mathrm{C}}(X, A)$. *In particular, the persistent homology of* $\mathrm{Del\check{C}}(X, A)$ *is isomorphic to the relative Čech persistent homology of the pair (X, A). If X is of cardinality n, then* $\mathrm{Del\check{C}}(X, A)$ *contains $O(n^{\lceil (d+1)/2 \rceil})$ simplices.*

The statement about the size of the relative Delaunay-Čech complex is a direct consequence of the result of [10] that the Delaunay triangulation of n points in $d+1$ dimensions contains $O(n^{\lceil (d+1)/2 \rceil})$ simplices

4 Dowker nerves

A *dissimilarity* is a continuous function of the form $\Lambda\colon X \times Y \to [0, \infty]$, for topological spaces X and Y, where $[0, \infty]$ is given the order topology. A *morphism* $f\colon \Lambda \to \Lambda'$ of dissimilarities $\Lambda\colon X \times Y \to [0, \infty]$ and $\Lambda'\colon X' \times Y' \to [0, \infty]$ consists of a pair (f_1, f_2) of continuous functions $f_1\colon X \to X'$ and $f_2\colon Y \to Y'$ so that for all $(x, y) \in X \times Y$ the following inequality holds:

$$\Lambda'(f_1(x), f_2(y)) \leq \Lambda(x, y).$$

This notion of morphism is less general than for example [3, Definition 2.10], but it is simpler and suffices for our purposes.

The *Dowker Nerve* $N\Lambda$ of Λ is the filtered simplicial complex described as follows: For $t > 0$, the simplicial complex $N\Lambda_t$ consists of the finite subsets σ of X for which there exists $y \in Y$ so that $\Lambda(x, y) < t$ for every $x \in \sigma$.

Let $f\colon \Lambda \to \Lambda'$ be a morphism of dissimilarities as above and let $\sigma \in N\Lambda_t$. Given $y \in Y$ with $\Lambda(x, y) < t$ for every $x \in \sigma$ we see that

$$\Lambda'(f_1(x), f_2(y)) \leq \Lambda(x, y) < t,$$

so $f_1(\sigma) \in N\Lambda'_t$. Thus we have a simplicial map $f\colon N\Lambda \to N\Lambda'$.

Given $x \in X$ and $t > 0$, the *Λ-ball of radius t centered at x* is the subset of Y defined as

$$B_\Lambda(x, t) = \{y \in Y, \mid \Lambda(x, y) < t\}.$$

The *t-thickening of* Λ is the subset of Y defined as

$$\Lambda^t = \bigcup_{x \in X} B_\Lambda(x, t).$$

Note that by construction the set of Λ-balls of radius t is an open cover of the t-thickening of Λ.

The *geometric realization* $|K|$ of a simplicial complex K on the vertex set V is the subspace of the space $[0, 1]^V$ of functions $\alpha\colon V \to [0, 1]$ described as follows:

1. The subset $\alpha^{-1}((0, 1])$ of V consisting of elements where α is strictly positive is a simplex in K. In particular it is finite.
2. The sum of the values of α is one, that is $\sum_{v \in V} \alpha(v) = 1$.

With respect to the product topology, the subspace topology on $|K|$ is called the *strong topology* on the geometric realization. It is convenient for construction of functions into $|K|$. The *weak tooplogy* on $|K|$, which we are not going to use here, is convenient for construction of functions out of $|K|$. The homotopy type of $|K|$ is the same for these two topologies [7, p. 355, Corollary A.2.9]. Given a simplex $\sigma \in K$, the simplex $|\sigma|$ of $|K|$ is the closure of

$$\{\alpha \colon V \to [0,1] \,|\, \alpha(v) > 0 \text{ for all } v \in \sigma\}.$$

The simplices of $|K|$ are the sets of this form.

A *partition of unity subordinate to the dissimilarity* $\Lambda \colon X \times Y \to [0,\infty]$ consists of continuous maps $\varphi^t \colon \Lambda^t \to |N\Lambda_t|$ such that given $x \in X$, the closure of the set

$$\{y \in Y \mid \varphi^t(y)(x) > 0\}$$

is contained in $B_\Lambda(x,t)$. We say that Λ is *numerable* if a partition of unity subordinate to Λ exists. If Y is paracompact, then every dissimilarity of the form $\Lambda \colon X \times Y \to [0,\infty]$ is numerable [7, p. 355, paragraph after Definition A.2.10].

Let $y \in \Lambda^t$ and let $\{\varphi^t \colon \Lambda^t \to |N\Lambda_t|\}$ be a partition of unity subordinate to Λ. If $x \in X$ with $\varphi^t(y)(x) > 0$, then $\Lambda(x,y) < t$. Therefore $\varphi^t(y)$ is contained in a simplex $|\sigma|$ in $|N\Lambda_t|$ with σ contained in $\{x \in X \mid \Lambda(x,y) < t\}$. Every finite subset of this set is an element of $N\Lambda_t$. This implies that for $s \leq t$ there is a simplex of $|N\Lambda_t|$ containing both $\varphi^s(y)$ and $\varphi^t(y)$. It also implies that given another partition of unity $\{\psi^t \colon \Lambda^t \to |N\Lambda_t|\}$ subordinate to Λ there is a simplex of $|N\Lambda_t|$ containing both $\varphi^t(y)$ and $\psi^t(y)$. This is exactly the definition of contiguous maps, so φ^t and ψ^t are contiguous, and thus homotopic maps [7, Remark 2.22, p. 350]. Similarly, the diagram

$$\begin{array}{ccc} \Lambda^s & \xrightarrow{\varphi^s} & |N\Lambda_s| \\ \downarrow & & \downarrow \\ \Lambda^t & \xrightarrow{\varphi^t} & |N\Lambda_t| \end{array}$$

commutes up to homotopy [7, paragraph on the nerve starting on page 355 and ending on page 356].

Recall that a cover $\mathcal{U}$ of Y is *good* if all non-empty finite intersections of members of $\mathcal{U}$ are contractible. We now state the Nerve Lemma in the context of dissimilarities.

▶ **Theorem 3.** *If Y is paracompact and $\Lambda \colon X \times Y \to [0,\infty]$ is a dissimilarity, then there exists a partition of unity $\{\varphi^t \colon \Lambda^t \to |N\Lambda_t|\}$ subordinate to Λ. Moreover, if the cover of Λ^t by Λ-balls of radius t is a good cover, then φ^t is a homotopy equivalence.*

Proof. By the above discussion, we only need to note that the last statement about good covers is [11, Theorem 4.3]. ◀

A functorial version of the Nerve Lemma can be stated as follows:

▶ **Proposition 4.** *Let $\Lambda \colon X \times Y \to [0,\infty]$ and $\Lambda' \colon X' \times Y' \to [0,\infty]$ be dissimilarities and let $f = f_1 \times f_2 \colon X \times Y \to X' \times Y'$ be a morphism $f \colon \Lambda \to \Lambda'$ of dissimilarities. If $\{\varphi^t \colon \Lambda^t \to |N\Lambda_t|\}$ is a partition of unity subordinate to Λ and $\{\psi^t \colon (\Lambda')^t \to |N\Lambda'_t|\}$ is a partition of unity subordinate to Λ', then for every $t \geq 0$ the diagram*

$$\begin{array}{ccc} \Lambda^t & \xrightarrow{\varphi^t} & |N\Lambda_t| \\ {\scriptstyle f_2}\downarrow & & \downarrow{\scriptstyle |f_1|} \\ (\Lambda')^t & \xrightarrow{\psi^t} & |N\Lambda'_t|, \end{array}$$

commutes up to homotopy.

Proof. We show that the two compositions are contiguous. Recall that $|f_1|$ takes a point $\alpha\colon X \to [0,1]$ of $|N\Lambda_t|$ to the point $|f_1|(\alpha)$ of $|N\Lambda'_t|$ with $|f_1|(\alpha)(x') = \sum_{f_1(x)=x'} \alpha(x)$. Recall further that $\varphi^t(y)$ is contained in a simplex $|\sigma|$ in $|N\Lambda_t|$, where σ is contained in $\{x \in X \mid \Lambda(x,y) < t\}$. Then we have that for $y \in \Lambda^t$, the elements $|f_1|(\varphi^t(y))$ and $\psi^t(f_2(y))$ of $|N\Lambda'_t|$ are contained in simplices $|\sigma|$ and $|\tau|$ respectively. Both σ and τ are subsets of the set $\{x' \in X' \mid \Lambda'(x', f_2(y)) < t\}$. However every finite subset of this set is a simplex in $N\Lambda'_t$. In particular, so is the union $\sigma \cup \tau$. ◀

5 The alpha- and Delaunay-Čech complexes

Given a finite subset X of $\mathbb{R}^d$ we define the Voronoi cell of $x \in X$ as

$$\mathrm{Vor}(X,x) = \{p \in \mathbb{R}^d \mid d(x,p) \leq d(y,p) \text{ for all } y \in X\}.$$

Let $\mathbb{R}^d_d$ be Euclidean space with the discrete topology. The *discrete Delaunay dissimilarity* of X is defined as

$$\mathrm{del}^X\colon X \times \mathbb{R}^d_d \to [0,\infty], \quad \mathrm{del}^X(x,p) = \begin{cases} 0 & \text{if } p \in V(X,x) \\ \infty & \text{if } p \notin V(X,x). \end{cases}$$

The *Delaunay complex* $\mathrm{Del}(X)$ is the simplicial complex with vertex set X and with $\sigma \subseteq X$ a simplex of $\mathrm{Del}(X)$ if and only if there exists a point in $\mathbb{R}^d$ belonging to $\mathrm{Vor}(X,x)$ for every $x \in \sigma$. That is, $\mathrm{Del}(X) = N\,\mathrm{del}^X_t$ for $t > 0$.

Note that with respect to Euclidean topology, the discrete Delaunay dissimilarity is not continuous, and hence $\mathrm{del}^X\colon X \times \mathbb{R}^d \to [0,\infty]$ is not a dissimilarity. One way to deal with this is to use the Nerve Lemma for absolute neighbourhood retracts [4, Theorem 8.2.1]. In order to use Theorem 3 and Proposition 4 from above, instead we construct a continuous version of the Delaunay dissimilarity.

Given a subset σ of X and $p \in \mathbb{R}^d$, let

$$d_{\mathrm{Vor}}(p,\sigma) = \max\{d(p,\mathrm{Vor}(X,x)) \mid x \in \sigma\},$$

where for any $A \subseteq \mathbb{R}^d$, we define $d(p,A) = \inf_{a\in A}\{d(p,a)\}$.

Note that if $\sigma \notin \mathrm{Del}(X)$, the infimum ε_σ of the continuous function $d_{\mathrm{Vor}}(-,\sigma)\colon \mathbb{R}^d \to \mathbb{R}$ is strictly positive. Choose $\varepsilon > 0$ so that $2\varepsilon < \varepsilon_\sigma$ for every subset σ of X that is not in $\mathrm{Del}(X)$. Given $x \in X$ we define the ε-thickened Voronoi cell $\mathrm{Vor}(X,x)^\varepsilon$ by

$$\mathrm{Vor}(X,x)^\varepsilon = \{p \in \mathbb{R}^d \mid d(p,\mathrm{Vor}(X,x)) < \varepsilon\}.$$

By construction the nerve of the open cover $(\mathrm{Vor}(X,x)^\varepsilon)_{x\in X}$ of $\mathbb{R}^d$ is equal to $\mathrm{Del}(X)$.

Let $h\colon [0,\infty] \to [0,\infty]$ be the order preserving map

$$h(t) = \begin{cases} -\ln(1-t/\varepsilon) & \text{if } t < \varepsilon \\ \infty & \text{if } t \geq \varepsilon. \end{cases} \tag{1}$$

For $x \in X$, let $\mathrm{Del}_x\colon \mathbb{R}^d \to [0,\infty]$ be the function defined by $\mathrm{Del}_x(p) = h(d(p,\mathrm{Vor}(X,x)))$ so that $\mathrm{Del}_x(\mathrm{Vor}(X,x)) = 0$ and $\mathrm{Del}_x(\mathbb{R}^d \setminus \mathrm{Vor}(X,x)^\varepsilon) = \infty$.

The Delaunay dissimilarity of X is defined as

$$\mathrm{Del}^X\colon X \times \mathbb{R}^d \to [0,\infty], \quad \mathrm{Del}^X(x,p) = \mathrm{Del}_x(p).$$

By the above discussion we know that $N\,\mathrm{Del}^X_t = N\,\mathrm{del}^X_t = \mathrm{Del}(X)$ whenever $t > 0$.

The Čech dissimilarity of X is defined as

$$d^X \colon X \times \mathbb{R}^d \to [0, \infty],$$

where $d^X(x, p)$ is the Euclidean distance between $x \in X$ and $p \in \mathbb{R}^d$.

The alpha dissimilarity of X is defined as

$$A^X = \max(\mathrm{Del}^X, d^X) \colon X \times \mathbb{R}^d \to [0, \infty].$$

The Delaunay-Čech dissimilarity is defined as

$$\mathrm{Del\check{C}}^X \colon X \times \left(\mathbb{R}^d \times \mathbb{R}^d\right) \to [0, \infty], \quad \mathrm{Del\check{C}}^X(x, (p, q)) = \max(d^X(x, p), \mathrm{Del}^X(x, q)).$$

Note the nerve of the dissimilarity

$$\mathrm{del\check{C}}^X \colon X \times \left(\mathbb{R}^d \times \mathbb{R}^d_d\right) \to [0, \infty], \quad \mathrm{del\check{C}}^X(x, (p, q)) = \max(d({}^X x, p), \mathrm{del}^X(x, q))$$

is identical to the nerve of $\mathrm{Del\check{C}}^X$. Moreover, the Dowker nerves of the Delaunay-, Čech-, alpha- and Delaunay-Čech dissimilarities are the Delaunay-, Čech-, alpha- and Delaunay-Čech complexes respectively. For all these dissimilarities, the corresponding balls are convex, so the geometric realizations are homotopy equivalent to the corresponding thickenings. In order to see that the morphism $A^X \to d^X$ of dissimilarities induces homotopy equivalences $|NA^X_t| \xrightarrow{\simeq} |Nd^X_t|$ it suffices to note that the corresponding map $(A^X)^t \to (d^X)^t$ is the identity map. This holds because $B_{A^X}(x, t) = B_{d^X}(x, t) \cap B_{\mathrm{Del}^X}(x, t)$ and given $y \in B_{d^X}(x, t)$ we have that $y \in \mathrm{Vor}(X, x')$ for some $x' \in X$. Thus, $d^X(y, x')$ is minimal, so $d^X(y, x') \leq d^X(y, x) < t$ and $y \in B_{d^X}(x', t) \cap B_{\mathrm{Del}^X}(x', t)$.

In order to see that the morphism $\mathrm{Del\check{C}}^X \to d^X$ of dissimilarities induces homotopy equivalences $|N\,\mathrm{Del\check{C}}^X_t| \xrightarrow{\simeq} |Nd^X_t|$ we use the following lemma:

▶ **Lemma 5.** *For every $(p, q) \in (\mathrm{Del\check{C}}^X)^t$, the entire line segment between (p, p) and (p, q) is contained in $(\mathrm{Del\check{C}}^X)^t$.*

Proof. In order not to clutter notation we omit superscript X on dissimilarities. Let $\gamma \colon [0, 1] \to \mathbb{R}^d$ be the function $\gamma(s) = (1 - s)p + sq$. We claim that given $(p, q) \in \mathrm{Del\check{C}}^t$ and $s \in [0, 1]$ the point $(p, \gamma(s)) = (p, (1 - s)p + sq)$ is in $\mathrm{Del\check{C}}^t$.

If $(p, q) \in \mathrm{Del\check{C}}^t$, there exists a point $x \in X$, such that $p \in B_d(x, t)$ and $q \in B_{\mathrm{Del}}(x, t)$, that is, $d(q, \mathrm{Vor}(X, x)) < h^{\leftarrow}(t)$, where $h^{\leftarrow}$ is the generalized inverse of h. Pick $q' \in \mathrm{Vor}(X, x)$ so that $d(q, q') < h^{\leftarrow}(t)$. Let $\gamma' \colon [0, 1] \to \mathbb{R}^d$ be the function $\gamma'(s) = (1 - s)p + sq'$. Given $s \in [0, 1]$, suppose that the point $(p, \gamma'(s)) = (p, (1 - s)p + sq')$ is in $\mathrm{del\check{C}}^t$. Then there exists $x' \in X$ so that $d(x', p) < t$ and $\gamma'(s) \in V(X, x')$ and $(p, \gamma(s))$ is in $\mathrm{Del\check{C}}^t$ since the distance between $(1 - s)p + sq$ and $(1 - s)p + sq'$ is less than $h^{\leftarrow}(t)$ and $d(\gamma'(s), \mathrm{Vor}(X, x')) = 0$.

We are left to show that, given $s \in [0, 1]$, the point $(p, \gamma'(s)) = (p, (1 - s)p + sq')$ is in $\mathrm{del\check{C}}^t$. Suppose $\gamma'(s) \in \mathrm{Vor}(X, y)$ for some $s \in [0, 1)$ and some $y \in X$. We claim that then $p \in B_d(y, t)$. To see this, we may without loss of generality assume that $y \neq x$. Let H be the hyperplane in between x and y, i.e.

$$H = \{z \in X \mid d(x, z) = d(y, z)\}.$$

Let

$$H_+ = \{z \in X \mid d(x, z) \geq d(y, z)\}$$

and

$$H_- = \{z \in X \mid d(x,z) \leq d(y,z)\}.$$

Since $\gamma'(s) \in \mathrm{Vor}(X,y)$ we have $\gamma'(s) \in H_+$. Since $q \in \mathrm{Vor}(X,x)$ we have $q \in H_-$. Since the line segment between p and q either is contained in H or intersects H at most once we must have $p \in H_+$. That is, $d(y,p) \leq d(x,p) < t$, so $p \in B_d(y,t)$ as claimed. ◀

By Lemma 5, the inclusion

$$(d^X)^t = \cup_{x\in X} B_{d^X}(x,t) \to \cup_{x\in X} B_{\mathrm{Del\check{C}}^X}(x,t) = (\mathrm{Del\check{C}}^X)^t, \quad p \mapsto (p,p)$$

is a deformation retract. In particular it is a homotopy equivalence.

6 The relative Delaunay-Čech dissimilarity

In this section we consider two subsets X_1 and X_2 of d-dimensional Euclidean space $\mathbb{R}^d$.

The *Voronoi diagram* of a finite subset X of $\mathbb{R}^d$ is the set of pairs of the form $(x, \mathrm{Vor}(X,x))$ for $x \in X$, that is,

$$\mathrm{Vor}(X) = \{(x, \mathrm{Vor}(X,x)) \mid x \in X\}.$$

This may seem overly formal since the projection on the first factor gives a bijection $\mathrm{Vor}(X) \to X$. However, when we work with Voronoi cells with respect to different subsets X_1 and X_2 of $\mathbb{R}^d$ it may happen that $\mathrm{Vor}(X_1,x_1) = \mathrm{Vor}(X_2,x_2)$ even when $x_1 \neq x_2$. The *Voronoi diagram* of the pair of subsets X_1 and X_2 of $\mathbb{R}^d$ is the union

$$\mathrm{Vor}(X_1,X_2) = \mathrm{Vor}(X_1) \cup \mathrm{Vor}(X_2).$$

The *discrete Delaunay dissimilarity* of X_1 and X_2 is defined as

$$\mathrm{del}^{X_1,X_2}\colon \mathrm{Vor}(X_1,X_2) \times \mathbb{R}^d_d \to [0,\infty], \qquad \mathrm{del}^{X_1,X_2}((x,V),p) = \begin{cases} 0 & \text{if } p \in V \\ \infty & \text{if } p \notin V. \end{cases}$$

The simplicial complex $N\,\mathrm{del}_t^{X_1,X_2}$ is independent of $t > 0$. It is the *Delaunay complex* $\mathrm{Del}(X_1,X_2)$ on X_1 and X_2. In order to describe the homotopy type of this simplicial complex we thicken the Voronoi cells like we did in the previous section:

Given a subset σ of $\mathrm{Vor}(X_1,X_2)$ and $p \in \mathbb{R}^d$, let

$$d_{\mathrm{Vor}}(p,\sigma) = \max\{d(p,V) \mid (x,V) \in \sigma\}.$$

Note that if $\sigma \notin \mathrm{Del}(X_1,X_2)$, the infimum ε_σ of the continuous function $d_{\mathrm{Vor}}(-,\sigma)\colon \mathbb{R}^d \to \mathbb{R}$ is strictly positive. Choose $\varepsilon > 0$ so that $2\varepsilon < \varepsilon_\sigma$ for every subset σ of $\mathrm{Vor}(X_1,X_2)$ that is not in $\mathrm{Del}(X_1,X_2)$. Given $(x,V) \in \mathrm{Vor}(X_1,X_2)$ we define the ε-thickening V^ε of V by

$$V^\varepsilon = \{p \in \mathbb{R}^d \mid d(p,V) < \varepsilon\}.$$

By construction, the nerve of the open cover $\{(x,V^\varepsilon)\}_{(x,V)\in \mathrm{Vor}(X_1,X_2)}$ is equal to $\mathrm{Del}(X_1,X_2)$. The Delaunay dissimilarity Del^{X_1,X_2} of X_1 and X_2 is defined as

$$\mathrm{Vor}(X_1,X_2) \times \mathbb{R}^d \xrightarrow{\mathrm{Del}^{X_1,X_2}} [0,\infty], \qquad \mathrm{Del}^{X_1,X_2}((x,V),p) = h(d(p,V))$$

for $h\colon [0,\infty] \to [0,\infty]$ the order preserving map defined in Equation (1).

The inclusion $X_1 \to \mathrm{Vor}(X_1, X_2)$ taking $x \in X_1$ to $(x, \mathrm{Vor}(X_1, x))$ induces a morphism of dissimilarities $\mathrm{Del}^{X_1} \to \mathrm{Del}^{X_1,X_2}$ and an inclusion of nerves $N\,\mathrm{Del}_t^{X_1} \subseteq N\,\mathrm{Del}_t^{X_1,X_2}$ for $t > 0$.

Next, we construct the dissimilarity A^{X_1,X_2} as

$$\mathrm{Vor}(X_1, X_2) \times \mathbb{R}^d \xrightarrow{A^{X_1,X_2}} [0, \infty], \qquad ((x, V), p) \mapsto \max(d(x, p), \mathrm{Del}^{X_1,X_2}((x, V), p)).$$

Also here we have an obvious inclusion $NA_t^{X_1} \to NA_t^{X_1,X_2}$, and the A^{X_1,X_2}-balls are convex so the nerve lemma yields a homotopy equivalence

$$|NA_t^{X_1,X_2}| \simeq \bigcup_{(x,V)\in \mathrm{Vor}(X_1,X_2)} B_{A^{X_1,X_2}}((x, V), t) = \bigcup_{x\in X_1\cup X_2} B_{d^{X_1\cup X_2}}(x, t) = (X_1 \cup X_2)^t.$$

Finally, we construct the dissimilarity $\mathrm{Del\check{C}}^{X_1,X_2}$

$$\mathrm{Vor}(X_1, X_2) \times (\mathbb{R}^d \times \mathbb{R}^d) \xrightarrow{\mathrm{Del\check{C}}^{X_1,X_2}} [0, \infty],$$
$$((x, V), (p, q)) \mapsto \max(d(x, p), \mathrm{Del}^{X_1,X_2}((x, V), q))$$

Here again we have an obvious inclusion $N\,\mathrm{Del\check{C}}_t^{X_1} \to N\,\mathrm{Del\check{C}}_t^{X_1,X_2}$, and the $\mathrm{Del\check{C}}^{X_1,X_2}$-balls are convex so the nerve lemma yields a homotopy equivalence

$$|N\,\mathrm{Del\check{C}}_t^{X_1,X_2}| \simeq (\mathrm{Del\check{C}}^{X_1,X_2})^t$$

The following variant of Lemma 5 implies that $(\mathrm{Del\check{C}}^{X_1,X_2})^t$ is a deformation retract of $(X_1 \cup X_2)^t$.

▶ **Lemma 6.** *For every $(p, q) \in (\mathrm{Del\check{C}}^{X_1,X_2})^t$, the entire line segment between (p, p) and (p, q) is contained in $(\mathrm{Del\check{C}}^{X_1,X_2})^t$.*

Proof. Given $(p, q) \in (\mathrm{Del\check{C}}^{X_1,X_2})^t = (\mathrm{Del\check{C}}^{X_1})^t \cup (\mathrm{Del\check{C}}^{X_2})^t$, we have $(p, q) \in (\mathrm{Del\check{C}}^{X_i})^t$ for some $i \in \{1, 2\}$. Then also (p, p) lies in $(\mathrm{Del\check{C}}^{X_i})^t$, and Lemma 5 proves the claim. ◀

7 Nerve of the relative Delaunay-Čech dissimilarity

In this section we show that the nerve of the relative Delaunay dissimilarity is level homotopy equivalent to the relative Dealunay-Čech complex.

We fix some notation used in this section: $X_1 \subseteq \mathbb{R}^d$ and $X_2 \subseteq \mathbb{R}^d$ are finite subsets. We let s be a positive real number, we let $Z = X_1 \times \{s\} \cup X_2 \times \{-s\}$ and we let $\mathrm{pr}\colon \mathbb{R}^{d+1} \to \mathbb{R}^d$ be the projection omitting the last coordinate.

▶ **Lemma 7.** *The projection $\mathrm{pr}\colon \mathbb{R}^{d+1} \to \mathbb{R}^d$ induces a surjection*

$$\mathrm{Vor}(Z) \xrightarrow{g} \mathrm{Vor}(X_1, X_2), \qquad ((x, s), V) \mapsto (x, V(X_1, x)), \quad ((x, -s), V) \mapsto (x, V(X_2, x)),$$

with $\mathrm{pr}(V) \subseteq V(X_i, x)$ for $x \in X_i$. Given $(x, V) \in \mathrm{Vor}(X_1, X_2)$ the fiber $g^{-1}((x, V))$ consists of all elements of $\mathrm{Vor}(Z)$ of the form $((x, a), V)$ for $a \in \{\pm s\}$.

Proof. We show that $\mathrm{pr}(V) \subseteq V(X_1, x_1)$ for $((x_1, s), V) \in \mathrm{Vor}(Z)$ with $x_1 \in X_1$. Given $(p, r) \in V$ we have for all points of the form (x_1', s) for $x_1' \in X_1$ that $d((p, r), (x_1, s)) \leq d((p, r), (x_1', s))$. This implies that $d(p, x_1) \leq d(p, x_1')$, and thus $p \in V(X_1, x_1)$. We conclude that $\mathrm{pr}(V) \subseteq V(X_1, x_1)$. An analogous argument applies for elements of the form $((x_2, -s), V)$ in $\mathrm{Vor}(Z)$. ◀

Let s_1 be larger than the largest filtration value of the alpha complex of X_1. Then the function $j_1\colon \mathrm{Vor}(X_1) \to \mathrm{Vor}(Z)$ defined by $j_1(x_1, V) = ((x_1, s), V(Z, (x_1, s)))$ induces a simplicial map of nerves $\mathrm{del}(X_1) \to \mathrm{del}(Z)$ for all $s > s_1$. Similarly, there is a simplicial map $\mathrm{del}(X_2) \to \mathrm{del}(Z)$ for all $s > s_2$ when s_2 is larger than all filtration values of the alpha complex of X_2. Let $s(X_1, X_2) = \max(s_1, s_2)$.

Recall, from the previous two sections, that ε_σ is the infimum of the continuous function $d_{\mathrm{Vor}}(-, \sigma)\colon \mathbb{R}^d \to \mathbb{R}$. Choose $\varepsilon > 0$ satisfying the following two criteria:

1. $2\varepsilon < \varepsilon_\sigma$ for every subset σ of $\mathrm{Vor}(X_1, X_2)$ that is not in $\mathrm{Del}(X_1, X_2)$.
2. $2\varepsilon < \varepsilon_\sigma$ for every subset σ of $\mathrm{Vor}(Z)$ that is not in $\mathrm{Del}(Z)$.

Let $h\colon [0, \infty] \to [0, \infty]$ be the order preserving map defined in Equation (1), and let Del^Z and Del^{X_1,X_2} be constructed using h. We define a new dissimilarity

$$D\colon \mathrm{Vor}(Z) \times (\mathbb{R}^d \times \mathbb{R}^{d+1}) \to [0, \infty], \quad D((z, V), (p, q)) = \max(d(\mathrm{pr}(z), p), \mathrm{Del}^Z((z, V), q)).$$

Note that the underlying simplicial complex $\bigcup_{t>0} ND_t$ of the nerve of D is the Delaunay complex $\mathrm{del}(Z)$. The filtration value of $\sigma \in \mathrm{del}(Z)$ in the neve of D is the filtration value of $g(\sigma)$ in the nerve of $\mathrm{Del\check{C}}^{X_1,X_2}$.

▶ **Proposition 8.** *Let $X_1 \subseteq \mathbb{R}^d$ and $X_2 \subseteq \mathbb{R}^d$ be finite. Choose $s > s(X_1, X_2)$. Then $\mathrm{Vor}(Z) \xrightarrow{g} \mathrm{Vor}(X_1, X_2)$ and $\mathrm{id} \times \mathrm{pr}\colon \mathbb{R}^d \times \mathbb{R}^{d+1} \to \mathbb{R}^d \times \mathbb{R}^d$ form a morphism*

$$f = (g, \mathrm{id} \times \mathrm{pr})\colon D \to \mathrm{Del\check{C}}^{X_1,X_2}$$

of dissimilarities inducing a homotopy equivalence

$$g\colon ND_t \to N\,\mathrm{Del\check{C}}_t^{X_1,X_2}$$

for every $t > 0$.

Proof. For $i = 1, 2$ the inclusion $\mathrm{pr}(V) \subseteq V(X_i, x)$ for $((x, (-1)^{i-1}s), V) \in \mathrm{Vor}(Z)$ implies that

$$\mathrm{Del}^{X_1,X_2}(g(z, V), \mathrm{pr}(q)) \leq \mathrm{Del}^Z((z, V), q)$$

for all $((z, V), q) \in \mathrm{Vor}(Z)$. So we have a morphism $f = (g, \mathrm{id} \times \mathrm{pr})\colon D \to \mathrm{Del\check{C}}^{X_1,X_2}$.

In order to show that g induces a homotopy equivalence of geometric realizations, by the Nerve Lemma, it suffices to show that given a simplex σ of $N\,\mathrm{Del\check{C}}_t^{X_1,X_2}$, the inverse image $g^{-1}(\sigma)$ is a simplex of ND_t. Let p be a point in the intersection of the Voronoi cells in σ. Write $g^{-1}(\sigma) = \tau_1 \cup \tau_2$, where τ_1 consists of Voronoi cells with centers at height s and τ_2 consists of Voronoi cells with centers at height $-s$. Let $\sigma_1 = \{(x_1, s) \mid (x_1, V(X_1, x_1)) \in \sigma\}$ and $\sigma_2 = \{(x_2, -s) \mid (x_2, V(X_2, x_2)) \in \sigma\}$.

Suppose that τ_2 is empty. Then actually $\sigma \in \mathrm{Del\check{C}}_t^{X_1}$, and since $s > s_1$ we know that $j_1(\sigma) \in \mathrm{del}(Z)$. Since $g \circ j_1$ is the inclusion of $\mathrm{Vor}(X_1)$ in $\mathrm{Vor}(X_1, X_2) = \mathrm{Vor}(X_1) \cup \mathrm{Vor}(X_2)$ we know that $j_1(\sigma) \subseteq g^{-1}(\sigma) = \tau_1$ and that $j_1(\sigma) \in ND_t$. On the other hand, since τ_2 is empty, by Lemma 7 we know that $g^{-1}(\sigma)$ is contained in $j_1(\sigma)$, so they must be equal. We conclude that $g^{-1}(\sigma)$ is a simplex of ND_t. A similar argument applies when τ_1 is empty.

In the remaining case where both τ_1 and τ_2 are nonempty, the function

$$f\colon \mathbb{R}^{d+1} \to \mathbb{R}, \qquad f(a) = d_{\mathrm{Vor}}(a, \sigma_1) - d_{\mathrm{Vor}}(a, \sigma_2)$$

has $f((p, -s)) > 0$ and $f((p, s)) < 0$. By the intermediate value theorem there exists $t \in [-s, s]$ with $f(p, t) = 0$. Since (p, t) has the same distance to all elements of σ_1 and also has the same distance to all elements of σ_2 we conclude that (p, t) is in the intersection of the Voronoi cells in $g^{-1}(\sigma) = \tau_1 \cup \tau_2$. Thus $\mathrm{Del\check{C}}^Z((z, V), p) = 0$ and $d(\mathrm{pr}(z), p) < t$ for all $(z, V) \in g^{-1}(\sigma)$. In particular $g^{-1}(\sigma) \in ND_t$. ◀

We are now ready to compute persistent homology of $X_1 \cup X_2$ relative to X_1. The relative Delaunay-Čech complex $\mathrm{Del\check{C}}(X_1 \cup X_2, X_1)$ is the filtered simplicial complex with $\mathrm{Del\check{C}}(X_1 \cup X_2, X_1)_t = j_1(\mathrm{del}(X_1)) \cup ND_t$. Note that this is consistent with Definition 1.

▶ **Theorem 9.** *Let $X_1 \subseteq \mathbb{R}^d$ and $X_2 \subseteq \mathbb{R}^d$ be finite. Choose $s > s(X_1, X_2)$. Then the geometric realization of the filtered simplicial complex $\mathrm{Del\check{C}}(X_1 \cup X_2, X_1)$ is level homotopy equivalent to the filtered space $t \mapsto (X_1 \cup X_2)^t / X_1^t$. In particular, there is an isomorphism*

$$(H_*(\mathrm{Del\check{C}}(X_1 \cup X_2, X_1)_t))_{t>0} \cong (H_*((X_1 \cup X_2)^t, X_1^t))_{t>0}$$

of persistence modules.

Proof. Since $j_1(\mathrm{del}(X_1)$ is contractible, the geometric realization of $\mathrm{Del\check{C}}(X_1 \cup X_2, X_1)_t$ is homotopy equivalent to the quotient space $|\mathrm{Del\check{C}}(X_1 \cup X_2, X_1)_t|/|j_1(\mathrm{del}(X_1)|$. This quotient space is homeomorphic to $|ND_t|/|ND_t \cap j_1(\mathrm{Del}(X_1))|$. By Proposition 8 the map $g\colon ND_t \to N\,\mathrm{Del\check{C}}_t^{X_1,X_2}$ induces a homotopy equivalence of geometric realizations. Moreover g induces an an isomorphism $ND_t \cap j_1(\mathrm{Del}(X_1)) \to N\,\mathrm{Del\check{C}}_t^{X_1}$. Combining these two statements, g induces a homotopy equivalence $|ND_t|/|ND_t \cap j_1(\mathrm{Del}(X_1))| \to |N\,\mathrm{Del\check{C}}_t^{X_1,X_2}|/|N\,\mathrm{Del\check{C}}_t^{X_1}|$. The space $|N\,\mathrm{Del\check{C}}_t^{X_1,X_2}|$ is homotopy equivalent to the Euclidean t-thickening $(X_1 \cup X_2)^t$ of $X_1 \cup X_2$ and $|N\,\mathrm{Del\check{C}}_t^{X_1}|$ is homotopy equivalent to the Euclidean t-thickening X_1^t of X_1. ◀

This concludes the proof of Theorem 2.

References

1. Ulrich Bauer and Herbert Edelsbrunner. The Morse theory of Čech and Delaunay complexes. *Trans. Amer. Math. Soc.*, 369(5):3741–3762, 2017. doi:10.1090/tran/6991.
2. Ulrich Bauer, Michael Kerber, Jan Reininghaus, and Hubert Wagner. Phat – persistent homology algorithms toolbox. *Journal of Symbolic Computation*, 78:76–90, 2017. Algorithms and Software for Computational Topology. doi:10.1016/j.jsc.2016.03.008.
3. Nello Blaser and Morten Brun. Sparse filtered nerves, 2018. ArXiv 1810.02149. arXiv:1810.02149.
4. A. Borel and J.-P. Serre. Corners and arithmetic groups. *Comment. Math. Helv.*, 48:436–491, 1973. doi:10.1007/BF02566134.
5. David Cohen-Steiner, Herbert Edelsbrunner, and John Harer. Extending persistence using Poincaré and Lefschetz duality. *Foundations of Computational Mathematics*, 9(1):79–103, February 2009. doi:10.1007/s10208-008-9027-z.
6. Vin de Silva, Dmitriy Morozov, and Mikael Vejdemo-Johansson. Dualities in persistent (co)homology. *Inverse Problems*, 27(12):124003, November 2011. doi:10.1088/0266-5611/27/12/124003.
7. Albrecht Dold. *Lectures on algebraic topology.* Classics in Mathematics. Springer-Verlag, Berlin, 1995. Reprint of the 1972 edition. doi:10.1007/978-3-642-67821-9.
8. Herbert Edelsbrunner and John Harer. Persistent homology—a survey. In *Surveys on discrete and computational geometry*, volume 453 of *Contemp. Math.*, pages 257–282. Amer. Math. Soc., Providence, RI, 2008. doi:10.1090/conm/453/08802.
9. F. T. Pokorny, K. Goldberg, and D. Kragic. Topological trajectory clustering with relative persistent homology. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 16–23, May 2016. doi:10.1109/ICRA.2016.7487092.
10. Raimund Seidel. The upper bound theorem for polytopes: an easy proof of its asymptotic version. *Computational Geometry*, 5(2):115–116, 1995. doi:10.1016/0925-7721(95)00013-Y.
11. Žiga Virk. Rips complexes as nerves and a functorial Dowker-nerve diagram, 2019. ArXiv 1906.04028. arXiv:1906.04028.

Edge Collapse and Persistence of Flag Complexes

Jean-Daniel Boissonnat
Université Côte d'Azur, INRIA, Sophia Antipolis, France
Jean-Daniel.Boissonnat@inria.fr

Siddharth Pritam
Université Côte d'Azur, INRIA, Sophia Antipolis, France
siddharth.pritam@inria.fr

Abstract

In this article, we extend the notions of dominated vertex and strong collapse of a simplicial complex as introduced by J. Barmak and E. Miniam. We say that a simplex (of any dimension) is dominated if its link is a simplicial cone. Domination of edges appears to be a very powerful concept, especially when applied to flag complexes. We show that edge collapse (removal of dominated edges) in a flag complex can be performed using only the 1-skeleton of the complex. Furthermore, the residual complex is a flag complex as well. Next we show that, similar to the case of strong collapses, we can use edge collapses to reduce a flag filtration $\mathcal{F}$ to a smaller flag filtration $\mathcal{F}^c$ with the same persistence. Here again, we only use the 1-skeletons of the complexes. The resulting method to compute $\mathcal{F}^c$ is simple and extremely efficient and, when used as a preprocessing for persistence computation, leads to gains of several orders of magnitude w.r.t the state-of-the-art methods (including our previous approach using strong collapse). The method is exact, irrespective of dimension, and improves performance of persistence computation even in low dimensions. This is demonstrated by numerous experiments on publicly available data.

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

Keywords and phrases Computational Topology, Topological Data Analysis, Edge Collapse, Simple Collapse, Persistent homology

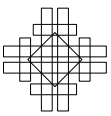
Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.19

Funding This research has received funding from the European Research Council (ERC) under the European Union's Seventh Framework Programme (FP/2007- 2013) / ERC Grant Agreement No. 339025 GUDHI (Algorithmic Foundations of Geometry Understanding in Higher Dimensions).

Acknowledgements We want to thank Marc Glisse for useful discussions and Vincent Rouvreau for his help with Gudhi.

1 Introduction

Improving the performance of computing persistent homology has been a central goal in Topological Data Analysis (TDA) since the early days of the field about 20 years ago. Very significant progress has been obtained on the two main components of the overall pipeline: the preprocessing of the sequence of complexes given as input and the computation of persistence homology (PH). The latter line of research led to improvement of the persistence algorithm and of its analysis, to efficient implementations and optimizations, and to a new generation of software [37, 8, 6, 45]. The former and complementary direction has been intensively explored with the goal of reducing the size of the complexes in the input sequence while preserving the persistent homology of the sequence, or approximating it in a controlled way [44, 30, 18, 13, 51, 41, 20, 27]. Among the most widely used complexes in TDA are the flag complexes and, in particular, the Vietoris-Rips complexes. These complexes are of great theoretical and practical interest since they are fully characterized by their graph (or 1-skeleton) and can thus be stored in a very compact way. Specific algorithms and very




efficient codes have been developed for those complexes [6, 51]. Despite all these advances, further progress has been obtained recently both for general simplicial complexes [12] and for flag complexes [11] using a special type of collapses, called strong collapses, introduced by J. Barmak and E. Miniam [5]. The basic idea is to simplify the complexes of the input sequence by using strong collapses and to compute the PH of an induced sequence of reduced simplicial complexes whose PH is the same or a close approximation of the PH of the initial sequence. In the case of flag complexes, the critical observation was that the construction of the reduced sequence can be done using only the 1-skeletons of the complexes, without constructing the full complexes, therefore saving time and space.

This paper further improves on these last results. Although the general philosophy is the same, there are some new key features that make the new method several orders of magnitude more efficient than all known methods.

1. Instead of strong collapses, we use the so-called edge collapses. In fact, we more generally define k-collapses that are identical to the *extended collapses* introduced in [4] (see also the early work of V. Welker [53]). When $k = 0$, we have strong collapses and when $k = 1$ edge collapses. Edge collapses share with strong collapses some important properties. Most notably, we can use edge collapses to reduce any flag filtration $\mathcal{F}$ to a smaller flag filtration $\mathcal{F}^c$ with the same persistence, using only the 1-skeletons of the complexes.
2. The reduction is exact and the PH of the reduced sequence is identical to the PH of the input sequence. However, the method can be easily adapted so as to produce an approximate reduction that would lead to better run time.
3. In [12] and in [11], the reduced sequence associated to a filtration was usually a tower (a sequence of simplicial complexes connected through simplicial maps), and part of the computing time was devoted to transforming this tower in another equivalent filtration using ideas from [26, 40]. There is no such need in the algorithm presented in this paper, which is another main source of improvement. Note however that the algorithm described in [11] works for flag towers while, in this paper, we restrict ourselves to flag filtrations.
4. The resulting method is simple and extremely efficient. On the theory side, we show that the edge collapse of a flag filtration can be computed in time $O(n\, n_c\, k^2)$, where n and n_c are the number of edges in the input and output 1-skeletons respectively and k is the maximal degree of a vertex in the input graph. The algorithm has been implemented. Numerous experiments on publicly available data show that preprocessing PH computation of flag complexes using edge collapse leads to unprecedented performance. The code will be soon released in the Gudhi library [37].

An outline of this paper is as follows. Section 2 recalls some basic ideas and constructions related to simplicial complexes and simple collapses. We introduce k-collapses and then edge collapses in Section 3. In Section 4, we prove that simple collapses preserve persistence. In Section 5, we provide the main algorithm that reduces a flag filtration to another flag filtration using edge collapses. Experiments are discussed in Section 6.

2 Preliminaries

In this section we provide some background material. Readers can refer to [38] for a comprehensive introduction to these topics.

Simplex, simplicial complex and simplicial map. An **abstract simplicial complex** K is a collection of subsets of a non-empty finite set X, such that for every subset A in K, all the subsets of A are in K. From now on, we will call an *abstract simplicial complex* simply

a *simplicial complex* or just a *complex*. An element of K is called a **simplex**. An element of cardinality $k+1$ is called a k-simplex and k is called its **dimension**. Given a simplicial complex K, we denote its geometric realization as $|K|$. A simplex is called **maximal** if it is not a proper subset of any other simplex in K. A sub-collection L of K is called a **subcomplex** if it is a simplicial complex itself.

A map $\psi : K \to L$ between two simplicial complexes is called a **simplicial map** if it always maps a simplex in K to a simplex in L. Simplicial maps are induced by vertex-to-vertex maps. A simplicial map $\psi : K \to L$ between two simplicial complexes K and L induces a continuous map $|\psi| : |K| \to |L|$ between the underlying geometric realizations. Any general simplicial map can be decomposed into more elementary simplicial maps, namely **elementary inclusions** (i.e., inclusions of a single simplex) and **elementary contractions** $\{\{u, v\} \mapsto u\}$ (where a vertex is mapped onto another vertex). The inverse operation of an inclusion is called a **simplicial removal**, denoted as $K \hookleftarrow L$.

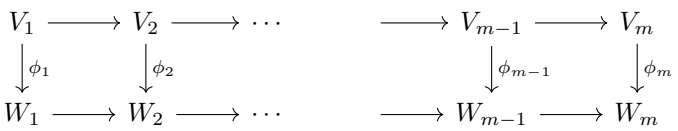
Flag complex and Neighborhood. A complex K is a **flag** or a **clique** complex if, when a subset of its vertices form a clique (i.e. any pair of vertices is joined by an edge), they span a simplex. It follows that the full structure of K is determined by its 1-skeleton (or graph) we denote by G. For a vertex v in G, the **open neighborhood** $N_G(v)$ of v in G is defined as $N_G(v) := \{u \in G \mid [uv] \in E\}$, here E is the set of edges of G. The **closed neighborhood** $N_G[v]$ is $N_G[v] := N_G(v) \cup \{v\}$. Similarly we define the closed and open neighborhood of an edge $[xy] \in G$, $N_G[xy]$ and $N_G(xy)$ as $N_G[xy] := N[x] \cap N[y]$ and $N_G(xy) := N(x) \cap N(y)$, respectively. The above definitions can be extended to any k-clique $\sigma = [v_1, v_2, ..., v_k]$ of G; $N_G[\sigma] := \bigcap_{v_i \in \sigma} N[v_i]$ and $N_G(\sigma) := \bigcap_{v_i \in \sigma} N(v_i)$.

Star, Link and Simplicial Cone. Let σ be a simplex of a simplicial complex K, the **closed star** of σ in K, $st_K(\sigma)$ is a subcomplex of K which is defined as follows, $st_K(\sigma) := \{\tau \in K |\ \tau \cup \sigma \in K\}$. The **link** of σ in K, $lk_K(\sigma)$ is defined as the set of simplices in $st_K(\sigma)$ which do not intersect with σ, $lk_K(\sigma) := \{\tau \in st_K(\sigma) | \tau \cap \sigma = \emptyset\}$. The **open star** of σ in K, $st^o_K(\sigma)$ is defined as the set $st_K(\sigma) \setminus lk_K(\sigma)$. Usually $st^o_K(\sigma)$ is not a subcomplex of K.

Let L be a simplicial complex and let a be a vertex not in L. Then the set aL defined as $aL := \{a, \tau \mid \tau \in L \text{ or } \tau = \sigma \cup a; \text{ where } \sigma \in L\}$ is called a **simplicial cone**.

Sequences of complexes. A **sequence** of simplicial complexes $\mathcal{T} : \{K_1 \xrightarrow{f_1} K_2 \xrightarrow{f_2} \cdots \xrightarrow{f_{(m-1)}} K_m\}$ connected through simplicial maps f_i is called a **simplicial tower** or simply a *tower*. When all the simplicial maps f_i are inclusions, the tower is called a **filtration**. If all the simplicial complexes K_i are flag complexes, we call it **flag towers** and **flag filtrations**.

Persistent homology. If we compute the homology classes of all the K_i, we get the sequence $\mathcal{P}(\mathcal{T}) : \{H_p(K_1) \xrightarrow{f_1^*} H_p(K_2) \xrightarrow{f_2^*} H_p(K_3) \xrightarrow{f_3^*} \cdots \xrightarrow{f^*_{(m-1)}} H_p(K_m)\}$. Here $H_p()$ denotes the homology class of dimension p with coefficients from a field $\mathbb{F}$ and f_i^* is the homomorphism induced from f_i. $\mathcal{P}(\mathcal{T})$ is a sequence of vector spaces connected through the f_i^* called a **persistence module**. More formally, a *persistence module* $\mathbb{V}$ is a sequence of vector spaces $\{V_1 \to V_2 \to V_3 \to \cdots \to V_m\}$ connected with homomorphisms $\{\to\}$ between them. A persistence module arising from a sequence of simplicial complexes captures the evolution of the topology of the sequence. Two different persistence modules $\mathbb{V} : \{V_1 \to V_2 \to \cdots \to V_m\}$ and $\mathbb{W} : \{W_1 \to W_2 \to \cdots \to W_m\}$, connected through a set of homomorphisms $\phi_i : V_i \to W_i$ are **equivalent** if the ϕ_i are isomorphisms and the following diagram commutes [14, 24].



Any persistence module can be *decomposed* into a collection of intervals of the form $[i, j)$ [14]. The multiset of all the intervals $[i, j)$ in this decomposition is called the **persistence diagram** of the persistence module. An interval of the form $[i, j)$ in the persistence diagram of $\mathcal{P}(\mathcal{T})$ corresponds to a homological feature (a "cycle") which appeared at i and disappeared at j. The persistence diagram (PD) completely characterizes the persistence module, that is, there is a bijective correspondence between the PD and the equivalence class of the persistence module [14, 58]. In other words, equivalent persistence modules have the same the same persistence diagram.

Simple collapse. Given a complex K, a simplex $\sigma \in K$ is called a **free simplex** if σ has a unique coface $\tau \in K$. The pair $\{\sigma, \tau\}$ is called a **free pair**. The action of removing a free pair: $K \to K \setminus \{\sigma, \tau\}$ is called an **elementary simple collapse.** A series of such elementary simple collapses is called a **simple collapse**. We denote it as $K \searrow L$. This operation preserves the homotopy type of the simplicial complex K, which we write $K \sim L$. In particular, there is a retraction map $|r| : |K| \to |L|$ between the underlying geometric realization of K and L which is a strong deformation retraction. A complex K' will be called **simple-collapse minimal** if there is no free pair $\{\sigma, \tau\}$ in K'. A subcomplex K^{ec} of K is called an **elementary core** of K if $K \searrow K^{ec}$ and K^{ec} is simple-collapse minimal.

Removal of a simplex. We denote by $\boldsymbol{K \setminus \sigma}$ the subcomplex of K obtained by removing σ, i.e. the complex that has all the simplices of K except the simplex σ and the cofaces of σ.

3 Edge Collapse

In this section, we first extend the definition of a dominated vertex introduced in [5] to simplices of any dimension. Given a simplex $\sigma \in K$, we denote by Σ_σ the set of maximal simplices of K that contain σ. The intersection of all the maximal simplices in Σ_σ will be denoted as $\bigcap \Sigma_\sigma := \bigcap_{\tau \in \Sigma_\sigma} \tau$.

Dominated simplex. A simplex σ in K is called a **dominated simplex** if the link $lk_K(\sigma)$ of σ in K is a simplicial cone, i.e. if there exists a vertex $v' \notin \sigma$ and a subcomplex L of K, such that $lk_K(\sigma) = v'L$. We say that the vertex v' is *dominating* σ and that σ is *dominated* by v', which we denote as $\sigma \prec v'$.

***k*-collapse.** Given a complex K, the action of removing a dominated k-simplex σ from K is called an **elementary *k*-collapse**, denoted as $K \searrow\!\searrow^k \{K \setminus \sigma\}$. A series of elementary k-collapses is called a ***k*-collapse**, denoted as $K \searrow\!\searrow^k L$. We further call a complex K ***k*-collapse minimal** if it does not have any dominated k simplices. A subcomplex K^k of K is called a ***k*-core** if $K \searrow\!\searrow^k K^k$ and K^k is k-collapse minimal.
The notion of k-collapse is the same as the notion of *extended collapse* introduced in [4]. We give it a different name to indicate the dependency on the dimension. A 0-collapse is a strong collapse as introduced in [5]. A 1-collapse will be called an **edge collapse**. It is not hard to

see that an elementary simple collapse of a k-simplex σ is a k-collapse, as it is dominated by the vertex $v = \tau \setminus \sigma$, where τ is the unique coface containing σ. Each k-collapse can be decomposed into a sequence of elementary simple collapses and therefore k-collapses preserve the simple homotopy type [53, Lemma 2.7] and [4, Lemma 8]. Therefore, like simple collapses, k-collapses induce a strong deformation retract as well on the geometric realization.

The following lemma extends a result in [5] to general k-collapse. It shows that the domination of a simplex can be characterized in terms of maximal simplices.

▶ **Lemma 1.** *A simplex $\sigma \in K$ is dominated by a vertex $v' \in K$, $v' \notin \sigma$, if and only if all the maximal simplices of K that contain σ also contain v', i.e. $v' \in \bigcap \Sigma_\sigma$.*

Proof. If $\sigma \prec v'$ then $lk_K(\sigma) = v'L$ by definition. This implies that for any maximal simplex τ in $st_K(\sigma)$, $v' \in \tau$. Therefore, $v' \in \bigcap \Sigma_\sigma$. For the reverse direction, let $v' \in \bigcap \Sigma_\sigma$. Hence, for any maximal simplex τ in $st_K(\sigma)$, we have $v' \in \tau$. Now as $v' \notin \sigma$, v' belong to all the simplices $\tau \setminus \sigma$, and thus $lk_K(\sigma) = v'L$ where $L = (\tau \setminus \sigma) \setminus v'$. Hence $\sigma \prec v'$ iff $v' \in \bigcap \Sigma_\sigma$. ◀

Lemma 1 has important algorithmic consequences. To perform a k-collapse, one simply needs to store the adjacency matrix between the k-simplices and the maximal simplices of K.

Next we study the special case of a flag complex K and characterize the domination of a simplex σ of a flag complex K in terms of its neighborhood.

▶ **Lemma 2.** *Let σ be a simplex of a flag complex K. Then σ will be dominated by a vertex v' if and only if $N_G[\sigma] \subseteq N_G[v']$.*

Proof. Assume that $N_G[\sigma] \subseteq N_G[v']$ and let τ be a maximal simplex of K that contains σ. For a vertex $x \in \tau$ and for any vertex $v \in \sigma$, the edge $[x, v] \in \tau$. Therefore $x \in N_G[\sigma] \subseteq N_G[v']$. Every vertex in τ is thus linked by an edge to v' and, since K is a flag complex and τ is maximal, v' must be in τ. This implies that all the maximal simplices that contains σ also contain v'. Hence σ is dominated by v'.

Consider the other direction. If $\sigma \prec v'$, by Lemma 1, all the maximal simplices that contain σ also contain v'. This implies $N_G[\sigma] \subseteq N_G[v']$. ◀

Lemma 2 is a generalisation of Lemma 1 in [11]. The next lemma, though elementary, is of crucial significance. Both lemmas show that edge collapses are well-suited to flag complexes.

▶ **Lemma 3.** *Let K be a flag complex and let L be any subcomplex of K obtained by edge collapse. Then L is also a flag complex.*

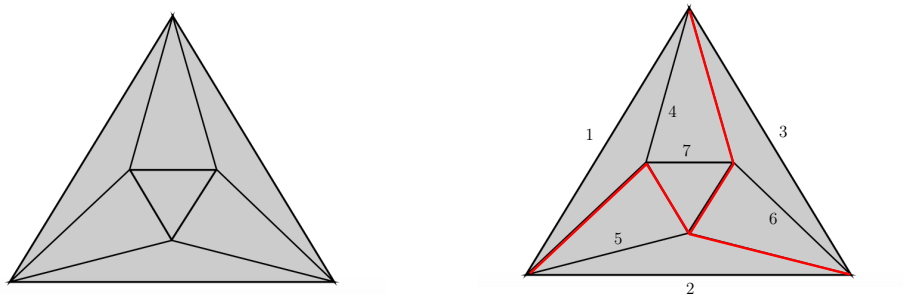


Figure 1 The above complex does not have any dominated vertex and thus cannot be 0-collapsed. However, by proceeding from the boundary edges, one can edge collapse this complex to a 1-dimensional complex. The 1-core obtained in this way is collapsible to a point using 0-collapse.

Efficiency of reduction. As will be demonstrated in Section 6, edge collapse appears to be a very efficient tool to reduce the size of a complex while preserving its homotopy type. A simple example will help giving some intuition why edge collapse can be superior to vertex collapse. See Figure 1.

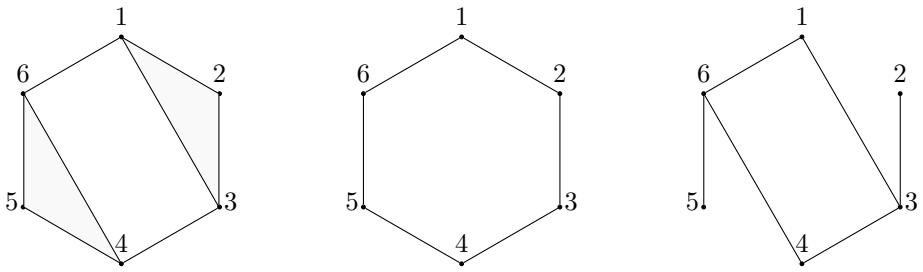


Figure 2 The complex on the left has two different 1-cores, the one in the middle is obtained after removing the inner edges $[1,3]$ and $[4,6]$, and the one in the right by removing the outer edges $[1,2]$ and $[4,5]$. Note that the one in the right can be further strong collapsed.

4 Simple Collapse and Persistence

In this section, we turn our attention to the general case of simple collapses (of which k-collapses are a special case) and provide one of the main result of this article. This can be seen as a generalization of Theorem 2 of [12].

▶ **Theorem 4.** *Let $f : K \to L$ be a simplicial map between two complexes K and L and let $K' \subset K$ and $L' \subset L$ be subcomplexes of K and L such that $K \searrow K'$ and $L \searrow L'$. Then there exists a map $f' : K' \to L'$, induced by f, such that the persistence of $f^* : H_p(K) \to H_p(L)$ and $f'^* : H_p(K') \to H_p(L')$ are the same for any integer $p \geq 0$. The induced map f' may not be simplicial. Nevertheless, it can be expressed as a combination of inclusions, contractions and removals of simplices.*

Proof. Let us consider the following diagram between the geometric realizations of the complex $|K|$, $|L|$, $|K'|$ and $|L'|$.

$$\begin{array}{ccc} |K| & \xrightarrow{|f|} & |L| \\ {\scriptstyle |i_k|}\uparrow\downarrow{\scriptstyle |r_k|} & & {\scriptstyle |i_l|}\uparrow\downarrow{\scriptstyle |r_l|} \\ |K'| & \xrightarrow{|f'|} & |L'| \end{array}$$

and the associated diagram after computing the p-th singular homology groups

$$\begin{array}{ccc} H_p^o(|K|) & \xrightarrow{|f|^*} & H_p^o(|L|) \\ {\scriptstyle |i_k|^*}\uparrow\downarrow{\scriptstyle |r_k|^*} & & {\scriptstyle |i_l|^*}\uparrow\downarrow{\scriptstyle |r_l|^*} \\ H_p^o(|K'|) & \xrightarrow{|f'|^*} & H_p^o(|L'|) \end{array}$$

Here $|r_k|$ and $|r_l|$ are the deformation retractions on the geometric realizations associated with the simple collapse and $|i_k|$ and $|i_l|$ are the inclusion maps. $H_p^o()$ denotes the singular homology and * is the induced homomorphisms by the corresponding continuous maps. The map $|f'|$ is defined as $|f'| := |r_l||f||i_k|$. Hence $|f'||r_k| = |r_l||f||i_k||r_k|$. Now observe

that, since $|r_k|$ is a deformation retraction, $|i_k||r_k|$ is homotopic to the identity over $|K|$. It follows that $|r_l||f||i_k||r_k|$ is homotopic to $|r_l||f|$. Since homotopic maps induce identical homomorphisms on the corresponding homology groups [38, Proposition 2.19], we deduce that $|f'|^*|r_k|^* = |r_l|^*|f|^*$ (commutativity). Also, since $|r_k|^*$ and $|r_l|^*$ are induced by deformation retractions, they are isomorphisms on their respective singular homology groups. We have thus proved that the above diagram commutes and that the vertical maps $|r_k|$ and $|r_l|$ are isomorphisms. This implies that the two maps $|f| : |K| \to |L|$ and $|f'| : |K'| \to |L'|$ have the same singular persistent homology.

$|f'|$ induces a map $f' := r_l \circ f \circ i_k$ between the simplicial complexes K' and L'. Note that f' can be expressed as a composition of inclusions, contractions and removals of simplices, as i_k is an inclusion, f is simplicial and r_l is a simple collapse. Also, for simplicial complexes, singular homology is isomorphic to simplicial homology [38, Theorem 2.27]. This implies that the persistent singular homology $|f'|^* : H_p^o(|K'|) \to H_p^o(|L'|)$ and the persistent simplicial homology $f'^* : H_p(K') \to H_p(L')$ are equivalent. Therefore, the persistent simplicial homologies $f^* : H_p(K) \to H_p(L)$ and $f'^* : H_p(K') \to H_p(L')$ are equivalent. ◀

The use of singular homology in the proof is due to the lack of a simplicial map associated with the retraction ($|r|$) of a simple collapse. Due to the same reason, the induced map $f' : K' \to L'$ may not be necessarily simplicial. However, as mentioned in the above proof the map f' can be expressed as a combination of inclusions, contractions and removals of simplices. When a sequence of simplicial complexes contains removals of simplices, it is called a zigzag sequence. There are algorithms [45, 42] to compute zigzag persistence but they are not as efficient as the usual algorithms for filtrations and towers.

In the next section, we consider the case of flag filtrations and show that we can restrict the way the edge collapses are performed so that the reduced filtration is also a flag filtration.

5 Edge collapse of a flag filtration

In Section 3, we have introduced edge collapse for general simplicial complexes and provided an easy criterion for edge-domination in a flag complex using only the 1-skeleton of the complex. In this section, we provide an algorithm to simplify a flag filtration by removing dominated edges (i.e. edge collapses), again using only the 1-skeleton of the complex.

We define a notion of **removable edge** to help explain how our algorithm works (Algorithm 1) and to prove its correctness. Let G be a graph and K be the associated flag complex. We say that an edge e in a graph G is removable either if it is dominated in K or if there exists a sequence of edge collapses $K \searrow\searrow^1 K^c$ such that e is dominated in the reduced complex K^c. Our algorithm is based on the fact that the flag complexes K and K^c are homotopy equivalent [53, Lemma 2.7] and [4, Lemma 8]. If $e = [u, v]$, we define the **edge-neighborhood** of an edge $e \in G$ as the set $EN_G(e) := \{[x, y], x \in \{u, v\}, y \in N_G([uv])\}$.

Algorithm. Let $\mathcal{F} : K_1 \hookrightarrow K_2 \hookrightarrow \cdots \hookrightarrow K_n$ be a flag filtration and $\mathcal{G}_\mathcal{F} : G_1 \hookrightarrow G_2 \hookrightarrow \cdots \hookrightarrow G_n$ be the associated sequence of 1-skeletons. We further assume that $G_i \hookrightarrow G_{i+1}$ is an elementary inclusion, namely the inclusion of a single edge we name e_{i+1}. The edges in $E := \{e_1, ..., e_n\}$ are thus indexed by their order in the filtration and we denote by G_i the subset $\{e_1, ..., e_i\}$. Our algorithm computes a subset of edges $E^c \subseteq E$ and attach to each edge in E^c a new index. We thus obtain a new sequence of flag complexes $\mathcal{F}^c$ corresponding to E^c, we call the *core sequence*. The construction of E^c and of the new indices is done so that $\mathcal{F}^c$ has the same persistence diagram as $\mathcal{F}$.

Let's give an intuitive presentation of the algorithm first. The central idea is to identify edges that appear to be non-removable at some point in the algorithm. We store such edges

in a set E^c. To be more specific, consider the case of the inclusion of an edge $G_{i-1} \xhookrightarrow{e_i} G_i$ such that e_i is dominated in G_i : e_i is thus removable in G_i and is not included in E^c. Suppose first that all further edges e_s are dominated in G_s, $i < s \leq n$. Then e_i remains removable and will never be put in E^c. This is consistent with the fact that e_i does not change the topology of the complexes K_s and is therefore not required when computing persistence.

Assume now that some edge e_p, $i < p \leq n$, is non-dominated in G_p. The status of e_i, that was removable in all G_s for $s < p$, may change to non-removable in G_p. Therefore, we check whether e_i is non-removable in G_p (by proceeding in the reverse filtration order) and, in the affirmative, include e_i in E^c. In turn, the fact that e_i changed from removable to non removable may change the status of the edges with smaller indices which could become non-removable after the inclusion of e_i. If such edges are found, they are also included in E^c.

Before describing the algorithm in detail, two remarks are in order. First, we do not change the status of an edge from non-removable to removable even if it has become removable: this will enforce the output sequence to be a filtration. Second, we change the filtration values of some edges: the new filtration value of an edge is the first index at which it is found to be non-removable. The second point leads to faster computation of E^c, otherwise one has to proceed backward recursively to search for new non-removable edges.

We now explain how to compute E^c. See [Algorithm 1] for the pseudo-code. The main **for** loop on line 6 (called the forward loop) iterates over the edges in the filtration $\mathcal{F}$ by increasing filtration values, i.e. in the *forward direction*, and check whether or not the current edge e_i is dominated in the graph G_i. If *not*, we insert e_i in E^c and keep its original index i.

After the insertion of an edge e_i in E^c, we proceed to the so-called backward loop ([Lines 9-26]) and look for new non-dominated edges in G_i, considering the edges by decreasing filtration values. We assign G_i to a temporary graph G, and we assign the edge-neighborhood of e_i in the graph G_i to E^{nbd} [Line 9-10]. As established in Lemma 5, the search for new non-dominated edges can be restricted to E^{nbd}. If an edge e_j is not in E^c and not in E^{nbd} [Line 13-14], e_j is still dominated : we then remove it from G [Line 22]. If $e_j \notin E^c$ and $e_j \in E^{nbd}$, then we check whether it is dominated or not. If e_j is dominated, we remove it from G [Line 19]. Otherwise, we insert e_j in E^c and assign to it the *new index* i, i.e. the index of the edge e_i that has triggered the backward search in G_i. Next we enlarge the edge-neighborhood E^{nbd} by inserting the edge-neighbors of e_j in G. We repeat this process until the last index $j = 1$. Upon termination of the forward loop [Line 6-30], we output E^c as the final set.

The computation of non-removable edges (the set E^c) is dependent on the order in which we do the backward search (the backward loop). In Algorithm 1 we chose to proceed in the reverse order of the filtration. A different choice of order might result in a different set of non-removable edges since edge collapses are order dependent as mentioned in Section 3.

We now prove the correctness of the above algorithm after some more definitions.

Critical Edges. Edges in E^c are called **critical** while edges in $E \setminus E^c$ are called **non-critical**. All edges have an original index i given by the insertion order in the input filtration $\mathcal{F}$. The critical edges received a second index j, called their **critical index**, when they are inserted in E^c. By convention, if an edge is not critical and thus has never been inserted in E^c, we will set its critical index to be ∞. Hence, at the end of Algorithm 1, each edge $e \in E$ has two indices, an original and a critical index. To make this explicit, we denote e as e_i^j. Clearly $i \leq j$. We further distinguish the cases $i = j$ and $i < j$. If $i = j$, e_i has been put in E^c during the forward loop and we call e_i a **primary critical edge**. If $i < j$, e_i has been put in E^c during the backward loop and we call it a **secondary critical edge**.

Algorithm 1 Core flag filtration algorithm.

```
1: procedure CORE-FLAG-FILTRATION(E)
2:     input : set of edges E of G_F sorted by filtration value.
3:     E^c ← ∅; i ← 1;
4:     E^nbd ← ∅
5:     G ← ∅
6:     for e_i ∈ E do                          ▷ For i = 1, ..., n in increasing order
7:         if e_i is non-dominated in G_i then
8:             Insert {e_i, i} in E^c.
9:             G ← G_i
10:            E^nbd ← EN_{G_i}(e_i)
11:            j ← i − 1
12:            for e_j in G_i do                ▷ For j = (i − 1), ..., 1 in decreasing order
13:                if e_j ∉ E^c then
14:                    if e_j ∈ E^nbd then
15:                        if e_j is non-dominated in G then
16:                            Insert {e_j, i} in E^c.
17:                            E^nbd ← E^nbd ∪ EN_G(e_j)
18:                        else
19:                            G ← G \ e_j
20:                        end if
21:                    else
22:                        G ← G \ e_j
23:                    end if
24:                end if
25:                j ← j − 1
26:            end for
27:        end if
28:        G ← ∅
29:        i ← i + 1
30:    end for
31:    return E^c                              ▷ E^c is the 1-skeleton of the core flag filtration.
32: end procedure
```

For $i = 1, ..., n$, we define the **critical graph** at index i, denoted G_i^c, as the graph whose edges are the edges in E^c with a critical index at most i. We denote the associated flag complex as K_i^c.

Correctness. We now prove some lemmas to certify the correctness of our algorithm.

The following lemma justifies the fact that the search for new critical edges during the backward loop of Algorithm 1 is restricted to the neighborhood of already found critical edges.

▶ **Lemma 5.** *Let e be an edge in a graph G and let e' be a new edge and $G' := G \cup e'$. If e is dominated in G and $e \notin EN_{G'}(e')$, then e is dominated in G'.*

Proof. Let $e \prec v'$ in G, then $N_G[e] \subseteq N_G[v']$. Plainly, $N_G[v'] \subseteq N_{G'}[v']$ and, since $e \notin EN_{G'}(e')$, $N_{G'}[e] = N_G[e]$. Therefore, $N_{G'}[e] = N_G[e] \subseteq N_{G'}[v']$ implies $e \prec v'$ in G'. ◀

The following lemma says that a non-critical edge is always removable and that a critical edge is removable until it becomes critical.

▶ **Lemma 6.** *Let e_i^j be an edge with $i < j$, then it is removable in all G_t, $i \leq t < \min(n+1, j)$.*

Proof. According to the algorithm, if $i < j$, e_i^j is dominated in G_i (j being finite or not).

1. Let us first consider the case $j = \infty$. Note that e_i^∞ is non-critical and let j_i be the smallest primary critical index greater than i. If no such index exists, set $j_i = n + 1$. We show by induction that e_i^∞ remains removable in all G_t, $i \leq t < n+1$. As shown above, it is true for $t = i$ since e_i^j is dominated in G_i. So assume that e_i^j is removable in G_{t-1} and consider the insertion of e_t in G_t, for some $t < j_i$. By definition of j_i, e_t is dominated in G_t, which implies that e_i^j is removable in G_t (in the backward sequence $e_t, e_{t-1}, \ldots, e_i$). Consider now $t = j_i$. Since e_{j_i} is a primary critical edge, it is non-dominated in G_{j_i}. According to the algorithm, a backward loop has been triggered at j_i. During this backward loop, e_i^∞ has *not* been inserted in E^c since its second critical index is ∞. This is only possible because e_i^∞ has been found to be dominated in G. Since G is initialized as G_{j_i}, it follows that e_i^∞ is removable in G_{j_i}. We can now proceed in a similar way for all $t, j_i < t < n + 1$.
2. The proof is very similar for the case $i < j \leq n$. As e_i^j has not been inserted in E^c until the backward loop triggered at index j, e_i^j remains removable in all G_t, $i \leq t < j$. ◀

Note that our statement does not imply that a critical edge e_i^j, $i < j \leq n$, can never be removable in G_t, $t \geq j$. It just means that we are sure that it will remain removable until the point it becomes critical.

▶ **Lemma 7.** *For each i, Algorithm 1 produces a sequence of elementary edge collapses such that $K_i \searrow\searrow^1 K_i^c$.*

Proof. By definition, $G_i \setminus G_i^c = \{e_t^m |\ t \leq i, m > i\}$ is the set of edges of G_i whose critical index m is greater than i, which includes the non-critical edges ($m = \infty$). Any edge $e_t^m \in G_i \setminus G_i^c$ is removable in all $K_j, j < m$ by Lemma 6. ◀

The proof of the following theorem certifying the correctness of our algorithm follows directly through the application of Lemma 7 and Theorem 4.

▶ **Theorem 8.** *Let $\mathcal{F} : K_1 \hookrightarrow K_2 \hookrightarrow \cdots \hookrightarrow K_n$ be a flag filtration and $\mathcal{G}_\mathcal{F} : G_1 \hookrightarrow G_2 \hookrightarrow \cdots \hookrightarrow G_n$ be the associated sequence of 1-skeletons, such that $G_i \hookrightarrow G_{i+1}$ is an elementary inclusion of an edge e_{i+1}. Let G_i^c be the critical graph and K_i^c be its flag complex as defined before. Then the associated flag filtration of the critical edges, $\mathcal{F}^c : K_1^c \hookrightarrow K_2^c \hookrightarrow \cdots \hookrightarrow K_n^c$ has the same persistence diagram as $\mathcal{F}$.*

Proof. Let us consider the following diagram of the geometric realizations of the flag complexes for any $i \in \{1, ..., n\}$, where K_i^c is the flag complex of the critical graph G_i^c.

$$\begin{array}{ccc} |K_i| & \hookrightarrow & |K_{i+1}| \\ \uparrow\downarrow{\scriptstyle |r_i|} & & \uparrow\downarrow{\scriptstyle |r_{i+1}|} \\ |K_i^c| & \hookrightarrow & |K_{i+1}^c| \end{array}$$

Using Lemma 7, there is an edge collapse and therefore a simple collapse from K_i to K_i^c and from K_{i+1} to K_{i+1}^c. And $|r_i|$ and $|r_{i+1}|$ are the deformation retractions induced by the corresponding edge collapses. The equivalence of the persistence modules then follows directly from the application of Theorem 4. ◀

Complexity. Write n_v for the total number of vertices, n for the total number of edges and k for the maximum degree of a vertex in G_n. We represent each graph G_i as an adjacency list, where every vertex stores a *sorted* list of at most k adjacent vertices. Additionally, we store the set of edges (E and E^c) as a separate data structure.

The cost of inserting and removing an edge from such an adjacency list is $\mathcal{O}(k)$. Since the size of $N_G[v]$ is at most k for any vertex v, the cost of computing $N_G[e]$ for an edge e is $\mathcal{O}(k)$. Checking if an edge e is dominated by a vertex $v \in N_G[e]$ reduces to checking whether $N_G[e] \subseteq N_G[v]$, see Lemma 2. Since all the lists are sorted, this operation takes $\mathcal{O}(k)$ time per vertex v, hence $\mathcal{O}(k^2)$ time in total.

Let us now analyze the worst-case time complexity of Algorithm 1. At each step i of the forward loop [Line 6], either e_i is dominated (which can be checked in $O(k^2)$ time) or a backward loop is triggered [Line 12]. The backward loop will consider all edges with (original) index at most i and check whether they are dominated or not. Writing n_c for the number of primary critical edges, the worst-case time complexity is $nk^2 + \sum_{i=1}^{n_c} n\,k^2 = \mathcal{O}(nn_ck^2)$. The space complexity is $\mathcal{O}(n)$. In practice, n_c is a small fraction of n (see Table 1).

6 Computational Experiments

Our algorithm [Algorithm 1] has been implemented for VR filtrations as a C++ module named EdgeCollapser. Our previous preprocessing method described in [11] to simplify VR filtrations using strong collapses is called VertexCollapser (previously called RipsCollapser). Both EdgeCollapser and VertexCollapser take as input a VR filtration and return the reduced flag filtration according to their respective algorithms.

We present results on five datasets **netw-sc**, **senate**, **eleg**, **HIV** and **torus**. The first four datasets are publicly available [22] and are given as the interpoint distance matrix of the points. The last dataset **torus** has 2000 points sampled in a spiraled fashion on a torus embedded in a 3-sphere of $\mathbf{R}^4$ [39]. The reported time includes the time of EdgeCollapser/VertexCollapser and the time to compute the persistence diagram (PD) using the Gudhi library [37].

The code has been compiled using the compiler "clang-900.0.38" and all computations were performed on a "2.8 GHz Intel Core i5" machine with 16 GB of available RAM. Both EdgeCollapser and VertexCollapser work irrespective of the dimension of the complexes associated to the input datasets. However, the size of the complexes in the reduced filtration, even if much smaller than in the original filtration, might exceed the capacities of the PD computation algorithm. For this reason, we introduced, as in Ripser [6], a parameter *dim* and restricts the expansion of the flag complexes to a maximal dimension *dim*.

The experimental results using EdgeCollapser are summarized in Table 1. Observe that the reduction in the number of edges done by EdgeCollapser is quite significant. The ratio between the number of initial edges and the number of critical edges is approximately 20. Therefore the reduction in the size of k-simplices can be as large as $\mathcal{O}(20^k)$. This is verified experimentally too, as the reduced complexes are small and of low dimension (column Size/Dim) compared to the input VR-complexes which are of dimensions respectively 57, 54 and 105 for the first three datasets **netw-sc**, **senate** and **eleg**.[1]

Comparison with VertexCollapser. The same set of experimental results using VertexCollapser are summarized in Table 2. VertexCollapser can be used in two modes: in the exact mode (step=0), the output filtration has the same PD as the input filtration while,

[1] The sizes of the complexes are so big that we could not compute the exact number of simplices.

in the approximate mode (step>0), a certified approximation is returned. For appropriate comparison, we use VertexCollapser in exact mode. It can be seen that EdgeCollapser is faster than VertexCollapser by approximately two orders of magnitude. The main reason for this is the efficient preprocessing algorithm behind EdgeCollapser. As it can be noticed in some cases, the reduction obtained using VertexCollapser is better than using EdgeCollapser, but even in those cases EdgeCollapser is faster than VertexCollapser.

In terms of size reduction, EdgeCollapser either outperforms VertexCollapser by a big amount or is comparable. Some intuition can be gained from the *torus* example. This is a well distributed point sets sampled from a manifold without boundary. The fact that there is no boundary implies that there are only a few number of dominated vertices, which dramatically reduces the capacity of VertexCollapser to collapse.

EdgeCollapser computes the exact PD of the input filtration while VertexCollapser has an exact and an approximate modes, Results in Table 2 are obtained using the exact mode of VertexCollapser, while results in Table 1 [11] are obtained using the approximate mode. In both cases, EdgeCollapser performs much better than VertexCollapser. An approximate version of EdgeCollapser can be easily implemented similarly to the case of VertexCollapser. Instead of triggering the backward loop of the algorithm [Line12-26] at each primary critical edge we find, we can trigger the backward loop at certain snapshot values only. See Section 5 of [11] for more details on the approximate methodology and description of snapshot.

Table 1 The columns are, from left to right: dataset (Data), number of points (Pnt), maximum value of the scale parameter (Thrsld), Initial number of edges/Critical (final) number of edges *Edge(I)/Edge(C)*, number of simplices (Size) and dimension of the final filtration (Dim), parameter (*dim*), time (in seconds) taken by Edge-Collapser and total time (in seconds) including PD computation (Tot-Time).

Data	Pnt	Thrsld	EdgeCollapser +PD				
			Edge(I)/Edge(C)	Size/Dim	*dim*	Pre-Time	Tot-Time
netw-sc	379	5.5	8.4K/417	1K/6	∞	0.62	0.73
senate	103	0.415	2.7K/234	663/4	∞	0.21	0.24
eleg	297	0.3	9.8K/562	1.8K/6	∞	1.6	1.7
HIV	1088	1050	182K/6.9K	86.9M/?	6	491	2789
torus	2000	1.5	428K/14K	44K/3	∞	288	289

Table 2 The columns are, from left to right: dataset (Data), number of points (Pnt), maximum value of the scale parameter (Thrsld), number of simplices (Size) and dimension of the final filtration (Dim), parameter (*dim*), time (in seconds) taken by VertexCollapser, total time (in seconds) including PD computation (Tot-Time), parameter *Step* (linear approximation factor) and the number of snapshots used (Snaps). For the last experiment (torus), the preprocessing was stopped after 12hrs due to the number of snapshots and the size of the complexes.

Data	Pnt	Thrsld	VertexCollapser +PD					
			Size/Dim	*dim*	Pre-Time	Tot-Time	*Step*	Snaps
netw-sc	379	5.5	175/3	∞	366.46	366.56	0	8420
senate	103	0.415	417/4	∞	15.96	15.98	0	2728
eleg	297	0.3	835K/16	∞	518.36	540.40	0	9850
HIV	1088	1050	127.3M/?	4	660	3,955	4	184
torus	2000	1.5		4	∞^*	∞	0	428K

Comparison with Ripser. Ripser is a state-of-the-art software to compute the persistent homology of VR-complexes [6]. Ripser computes the *exact* PD associated to an input filtration up to some dimension *dim*. EdgeCollapser (as well as VertexCollapser) are not really competitors of Ripser since they act more as a preprocessing of the input filtration and do not compute Persistence Homology and can be associated to any software computing PH of flag filtrations. Nevertheless, we run Ripser[2] on the same datasets as in Table 1 to demonstrate the benefit of using EdgeCollapser. Results are presented in Table 3. The main observation is that, in most of the cases, EdgeCollapser computes PD in all dimensions and outperforms Ripser, even when we restrict the dimension of the input filtration given to Ripser.

Table 3 Time is the total time (in seconds) taken by Ripser. ∞ means that the experiment ran longer than 12 hours or crashed due to memory overload.

Data	Pnt	Threshold	Ripser		Ripser		Ripser	
			dim	Time	*dim*	Time	*dim*	Time
netw-sc	379	5.5	4	25.3	5	231.2	6	∞
senate	103	0.415	3	0.52	4	5.9	5	52.3
”	”	”	6	406.8	7	∞		
eleg	297	0.3	3	8.9	4	217	5	∞
HIV	1088	1050	2	31.35	3	∞		
torus	2000	1.5	2	193	3	∞		

References

1 M. Adamaszek and J. Stacho. Complexity of simplicial homology and independence complexes of chordal graphs. *Computational Geometry: Theory and Applications*, 57:8–18, 2016.

2 D. Attali, A. Lieutier, and D. Salinas. Efficient data structure for representing and simplifying simplicial complexes in high dimensions. *International Journal of Computational Geometry and Applications (IJCGA)*, 22:279–303, 2012.

3 Dominique Attali and André 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 Dominique Attali, André Lieutier, and David Salinas. Vietoris-rips complexes also provide topologically correct reconstructions of sampled shapes. *Computational Geometry*, 46(4):448–465, 2013.

5 J. A. Barmak and E. G. Minian. Strong homotopy types, nerves and collapses. *Discrete and Computational Geometry*, 47:301–328, 2012.

6 U. Bauer. Ripser. URL: `https://github.com/Ripser/ripser`.

7 U. Bauer, M. Kerber, and J. Reininghaus. Clear and compress: Computing persistent homology in chunks. In *Topological Methods in Data Analysis and Visualization III, Mathematics and Visualization*, pages 103–117. Springer, 2014.

8 U. Bauer, M. Kerber, J. Reininghaus, and H. Wagner. PHAT – persistent homology algorithms toolbox. *Journal of Symbolic Computation*, 78, 2017.

9 J-D. Boissonnat and C. S. Karthik. An efficient representation for filtrations of simplicial complexes. In *ACM Transactions on Algorithms*, 2018.

[2] We used the command <./ripser inputData –format distances –threshold inputTh –dim inputDim >.

10 J-D. Boissonnat, C. S. Karthik, and S. Tavenas. Building efficient and compact data structures for simplicial complexes. *Algorithmica*, 79:530–567, 2017.

11 J-D. Boissonnat and S. Pritam. Computing persistent homology of flag complexes via strong collapses. *International Symposium on Computational Geometry (SoCG)*, 2019.

12 J-D. Boissonnat, S.Pritam, and D. Pareek. Strong Collapse for Persistence. In *26th Annual European Symposium on Algorithms (ESA 2018)*, volume 112, 2018.

13 M. Botnan and G. Spreemann. Approximating persistent homology in euclidean space through collapses. *Applicable Algebra in Engineering, Communication and Computing*, 26:73–101, 2015.

14 G. Carlsson and V. de Silva. Zigzag persistence. *Found Comput Math*, 10, 2010.

15 G. Carlsson, V. de Silva, and D. Morozov. Zigzag persistent homology and real-valued functions. *International Symposium on Computational Geometry (SoCG)*, pages 247–256, 2009.

16 G. Carlsson, T. Ishkhanov, V. de Silva, and A. Zomorodian. On the local behavior of spaces of natural images. *In: International Journal of Computer Vision*, 76:1–12, 2008.

17 J. M. Chan, G. Carlsson, and R. Rabadan. Topology of viral evolution. *In: Proceedings of the National Academy of Sciences*, 110:18566–18571, 2013.

18 F. Chazal and S. Oudot. Towards persistence-based reconstruction in Euclidean spaces. *International Symposium on Computational Geometry (SoCG)*, 2008.

19 C. Chen and M. Kerber. Persistent homology computation with a twist. *In European Workshop on Computational Geometry (EuroCG)*, pages 197–200, 2011.

20 Aruni Choudhary, Michael Kerber, and Sharath Raghvendra. Polynomial-Sized Topological Approximations Using the Permutahedron. In Sándor Fekete and Anna Lubiw, editors, *32nd International Symposium on Computational Geometry (SoCG 2016)*, volume 51 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 31:1–31:16, Dagstuhl, Germany, 2016. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. URL: `http://drops.dagstuhl.de/opus/volltexte/2016/5923`, `doi:10.4230/LIPIcs.SoCG.2016.31`.

21 H. Edelsbrunner D. Cohen-Steiner and J. Harer. Stability of persistence diagrams. *Discrete and Compututaional Geometry*, 37:103–120, 2007.

22 Datasets. URL: `https://github.com/n-otter/PH-roadmap/`.

23 V. de Silva and R. Ghrist. Coverage in sensor networks via persistent homology. *In: Algebraic and Geometric Topology*, 7:339 – 358, 2007.

24 H. Derksen and J. Weyman. Quiver representations. *Notices of the American Mathematical Society*, 52(2):200–206, February 2005.

25 T. K. Dey, H. Edelsbrunner, S. Guha, and D. Nekhayev. Topology preserving edge contraction. *Publications de l'Institut Mathematique (Beograd)*, 60:23–45, 1999.

26 T. K. Dey, F. Fan, and Y. Wang. Computing topological persistence for simplicial maps. In *International Symposium on Computational Geometry (SoCG)*, pages 345–354, 2014.

27 T. K. Dey, D. Shi, and Y. Wang. *SimBa: An efficient tool for approximating Rips-filtration persistence via Simplicial Batch-collapse*. In European Symp. on Algorithms (ESA), pages 35:1–35:16, 2016.

28 T. K. Dey and R. Slechta. Filtration simplification for persistent homology via edge contraction. *International Conference on Discrete Geometry for Computer Imagery*, 2019.

29 C. H. Dowker. Homology groups of relations. *The Annals of Mathematics*, 56:84–95, 1952.

30 P. Dłotko and H. Wagner. Simplification of complexes for persistent homology computations,. *Homology, Homotopy and Applications*, 16:49–63, 2014.

31 H. Edelsbrunner and J. Harer. *Computational Topology: An Introduction*. American Mathematical Society, 2010.

32 H. Edelsbrunner, D. Letscher, and A. Zomorodian. Topological persistence and simplification. *Discrete and Compututational Geometry*, 28:511–533, 2002.

33 Omer Egecioglu and Teofilo F. Gonzalez. A computationally intractable problem on simplicial complexes. *Computational Geometry*, 6:85–98, 1996.

34 B. T. Fasy, J. Kim, F. Lecci, and C. Maria:. Introduction to the R-package tda. *CoRR abs/1411.1830*, 2014. `arXiv:1411.1830`.

35 E. Fieux and J. Lacaze. Foldings in graphs and relations with simplicial complexes and posets. *Discrete Mathematics*, 312(17):2639–2651, 2012.

36 F. Le Gall. Powers of tensors and fast matrix multiplication. *ISSAC '*, 14:296–303, 2014.

37 Gudhi: Geometry understanding in higher dimensions. URL: `http://gudhi.gforge.inria.fr/`.

38 A. Hatcher. *Algebraic Topology*. Univ. Press Cambridge, 2001.

39 Benoît Hudson, Gary L. Miller, Steve Oudot, and Donald R. Sheehy. Topological inference via meshing. *International Symposium on Computational Geometry (SoCG)*, 2010.

40 M. Kerber and H. Schreiber:. Barcodes of towers and a streaming algorithm for persistent homology. *International Symposium on Computational Geometry (SoCG)*, 2017. `arXiv:1701.02208`.

41 M. Kerber and R. Sharathkumar. Approximate Čech complex in low and high dimensions. In *Algorithms and Computation*, pages 666–676. by Leizhen Cai, Siu-Wing Cheng, and Tak-Wah Lam. Vol. 8283. Lecture Notes in Computer Science, 2013.

42 C. Maria and S. Oudot. Zigzag persistence via reflections and transpositions. In *Proc. ACM-SIAM Symposium on Discrete Algorithms (SODA) pp.* 181–199, January 2015.

43 N. Milosavljevic, D. Morozov, and P. Skraba. Zigzag persistent homology in matrix multiplication time. In *International Symposium on Computational Geometry (SoCG)*, 2011.

44 K. Mischaikow and V. Nanda. Morse theory for filtrations and efficient computation of persistent homology. *Discrete and Computational Geometry*, 50:330–353, September 2013.

45 D. Mozozov. Dionysus. URL: `http://www.mrzv.org/software/dionysus/`.

46 J. Munkres. *Elements of Algebraic Topology*. Perseus Publishing, 1984.

47 N. Otter, M. Porter, U. Tillmann, P. Grindrod, and H. Harrington. A roadmap for the computation of persistent homology. *EPJ Data Science, Springer Nature*, page 6:17, 2017.

48 Steve Y. Oudot and Donald R. Sheehy. Zigzag zoology: Rips zigzags for homology inference. *Foundations of Computational Mathematics*, 15, 2015.

49 J. Perea and G. Carlsson. A Klein-bottle-based dictionary for texture representation. *In: International Journal of Computer Vision*, 107:75–97, 2014.

50 H. Schreiber. Sophia. URL: `https://bitbucket.org/schreiberh/sophia/`.

51 D. Sheehy. Linear-size approximations to the Vietoris–Rips filtration. *Discrete and Computational Geometry*, 49:778–796, 2013.

52 M. Tancer. Recognition of collapsible complexes is NP-complete. *Discrete and Computational Geometry*, 55:21–38, 2016.

53 Volkmar Welker. Constructions preserving evasiveness and collapsibility. *Discrete Mathematics*, 207(1):243–255, 1999.

54 J. H. C Whitehead. Simplicial spaces nuclei and m-groups. *Proc. London Math. Soc*, 45:243–327, 1939.

55 A. C. Wilkerson, H. Chintakunta, and H. Krim. Computing persistent features in big data: A distributed dimension reduction approach. In *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 11–15, 2014.

56 A. C. Wilkerson, T. J. Moore, A. Swami, and A. H. Krim. Simplifying the homology of networks via strong collapses. In *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 11–15, 2013.

57 A. Zomorodian. The tidy set: A minimal simplicial set for computing homology of clique complexes. In *International Symposium on Computational Geometry (SoCG)*, pages 257–266, 2010.

58 A. Zomorodian and G. Carlsson. Computing persistent homology. *Discrete and Computational Geometry*, 33:249–274, 2005.

The Topological Correctness of PL-Approximations of Isomanifolds

Jean-Daniel Boissonnat
Université Côte d'Azur, INRIA, Sophia-Antipolis, France
jean-daniel.boissonnat@inria.fr

Mathijs Wintraecken
IST Austria, Klosterneuburg, Austria
m.h.m.j.wintraecken@gmail.com

Abstract

Isomanifolds are the generalization of isosurfaces to arbitrary dimension and codimension, i.e. manifolds defined as the zero set of some multivariate vector-valued smooth function $f : \mathbb{R}^d \to \mathbb{R}^{d-n}$. A natural (and efficient) way to approximate an isomanifold is to consider its Piecewise-Linear (PL) approximation based on a triangulation $\mathcal{T}$ of the ambient space $\mathbb{R}^d$. In this paper, we give conditions under which the PL-approximation of an isomanifold is topologically equivalent to the isomanifold. The conditions are easy to satisfy in the sense that they can always be met by taking a sufficiently fine triangulation $\mathcal{T}$. This contrasts with previous results on the triangulation of manifolds where, in arbitrary dimensions, delicate perturbations are needed to guarantee topological correctness, which leads to strong limitations in practice. We further give a bound on the Fréchet distance between the original isomanifold and its PL-approximation. Finally we show analogous results for the PL-approximation of an isomanifold with boundary.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases PL-approximations, isomanifolds, solution manifolds, topological correctness

Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.20

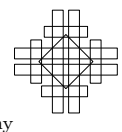
Related Version A full version of this paper is available at `https://hal.archives-ouvertes.fr/hal-02386193`.

Funding The research leading to these results has received funding from the European Research Council (ERC) under the European Union's Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreement No. 339025 GUDHI (Algorithmic Foundations of Geometry Understanding in Higher Dimensions).
Mathijs Wintraecken: Supported by the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 754411.

Acknowledgements First and foremost, we acknowledge Siargey Kachanovich for discussions. We thank Herbert Edelsbrunner and all members of his group, all former and current members of the Datashape team (formerly known as Geometrica), and André Lieutier for encouragement. We thank the reviewers for their comments which improved the exposition.

1 Introduction

Isomanifolds (also called solution manifolds) are the generalization of isosurfaces to arbitrary dimension and codimension, i.e. manifolds defined as the zero set of some multivariate vector-valued function $f : \mathbb{R}^d \to \mathbb{R}^{d-n}$. Not all submanifolds of $\mathbb{R}^d$ are isomanifolds although locally we can always write an embedded smooth manifold as the zero set of a smooth function, because it can be parametrized as a function from the tangent space to the manifold itself as a consequence of the implicit function theorem. Isosurfaces play a crucial role in medical



imaging, computer graphics and geometry processing. Higher dimensional isomanifolds are also of fundamental importance in many fields like statistics [19], dynamical systems [49], econometrics or mechanics [41].

The most widely used algorithm to approximate an isosurface is the celebrated Marching Cube algorithm (MC) [37, 42]. Extending the MC to isomanifolds of higher dimensions and codimensions leads to major difficulties and it is then preferred to use the so-called Marching Simplex algorithm (MS) where a simplicial triangulation of the ambient space is used instead of the cubical grid [1, 29, 40]. The major advantage is that the PL-approximation $\hat{\mathcal{M}}$ of $\mathcal{M}$ is well defined and easy to compute inside each d-simplex of the triangulation. Moreover, some regular triangulations, like Coxeter or Kuhn-Freudenthal triangulations do not require to be stored and can be used implicitly like the cubical grid. Hence the algorithm can be made efficient for isomanifolds of low dimensions even if they are embedded in high dimensional spaces [16].

Proving that either the MC or the MS produce good approximations has been a long lasting question. Some results were achieved within the computational geometry community in three dimensions. In [10, 44], conditions were given that ensure that $\hat{\mathcal{M}}$ is a PL-manifold close and topology equivalent (homeomorphic) to $\mathcal{M}$. Unfortunately, [10] uses a particular combination of collapses and Morse theory and [44] something akin to normal surface theory [47], both of which are specific to low dimensions. In the case of MS in higher dimensions, weaker results [2, 3] have been known for a while, e.g. bounds on the one-sided Hausdorff distance, on the approximation of tangent spaces, and manifoldness of the approximation (under strong conditions). However, it is only very recently that complete correctness results have been proved for general submanifolds of any dimension [7, 11, 12, 15]. However, the proofs in higher dimensions rely on perturbation schemes that are quite intricate and make the methods of little practical value. This is a major difference with the case of curves and surfaces where no such requirements exist [31], [46, Section 10.2].

In this paper, we restrict our attention to isomanifolds and show that, in this case, no perturbation scheme is required to obtain correct outputs. This is a major achievement with respect to effective computation and applications. Also the techniques used here are different from many of the standard tools. We, in particular, do not rely on Delaunay triangulations [48, 25, 21], nor on the closed ball property [33], Whitney's lemma [13] or collapses [4]. The current paper mainly relies on a version of the implicit function theorem from non-smooth analysis [23] with some Morse theory. Another clear difference with previous methods is that we do not provide lower bounds on the quality of the linear pieces in the Piecewise-Linear (PL) approximation. Although this is an appealing property, it is extremely difficult to satisfy in practice as mentionned above. Here we ask for less but still provide strong guarantees on $\hat{\mathcal{M}}$. Perturbation techniques can be used in an optional postprocessing step to improve the quality of the simplices of the output. They no longer play a critical role in the construction of the approximation.

The rest of this paper is subdivided in two sections. In the first section, we treat closed isomanifolds, i.e. compact manifolds without boundary. We show that, for a fine enough ambient triangulation, the output $\hat{\mathcal{M}}$ of the MS is a manifold that is isotopic to $\mathcal{M}$, close to $\mathcal{M}$ with respect to the Fréchet distance, that approximates well the tangent bundle of $\mathcal{M}$ (Theorem 20 and Corollaries 22 and Proposition 6).

In the second section, we prove similar results for isomanifolds with boundary. Extension to general isostratifolds is briefly discussed in Section 4. All proofs can be found in [17].

2 Isomanifolds (without boundary)

Let $f : \mathbb{R}^d \to \mathbb{R}^{d-n}$ be a smooth (C^2 suffices) function and suppose that 0 is a regular value of f, meaning that at every point x such that $f(x) = 0$, the Jacobian of f is non-degenerate. Then the zero set of f is an n-dimensional manifold as a direct consequence of the implicit function theorem, see for example [30, Section 3.5]. We further assume that $f^{-1}(0)$ is compact. As in [2] we consider a triangulation $\mathcal{T}$ of $\mathbb{R}^d$. The function f_{PL} is a linear interpolation of the values of f at the vertices if restricted to a single simplex $\sigma \in \mathcal{T}$. For any function $g : \mathbb{R}^d \to \mathbb{R}^{d-n}$ we write g^i, with $i = 1, \ldots, d-n$, for the components of g.

We prove that, under certain conditions, there is an isotopy from the zero set of f to the zero set of f_{PL}. The proof will be using the Piecewise-Linear (PL) map

$$F_{PL}(x, \tau) = (1 - \tau)f(x) + \tau f_{PL}(x), \tag{1}$$

which interpolates between f and f_{PL} and is based on the generalized implicit function theorem. The isotopy is in fact stronger than just the existence of a homeomorphism from the zero set of f to that of f_{PL}.

Our result in particular implies that the zero set of f_{PL} is a manifold. The fact that the zero set of f_{PL} is a manifold was proved (under strong condition) by Allgower and Georg [3, Theorem 15.4.1], without a homeomorphism with the zero set of f. The conditions here are weaker, because we do not require that the zero set avoids simplices that have dimension less than the codimension, see [3, Definition 12.2.2] and the text above [3, Theorem 15.4.1]. The idea to avoid these low dimensional simplices originates with Whitney [50], apparently unbeknownst to Allgower and George [3, 2]. Very heavy perturbation schemes for the vertices of the ambient triangulation $\mathcal{T}$ are required for the manifold to be sufficiently far from simplices in $\mathcal{T}$ that have dimension less than the codimension of the manifold [50, 15]. Various techniques have been developed to compute such perturbations with guarantees. They typically consist in perturbing the position of the sample points or in assigning weights to the points. Complexity bounds are then obtained using volume arguments. See, for example [20, 14, 11, 9]. However, these techniques suffer from several drawbacks. The constants in the complexity depend exponentially on the ambient dimension. Moreover the analysis assumes that the probability of the simplices of dimension less than the codimension to intersect the manifold is zero, which is not true when dealing with finite precision. As a result, the actual implementations we are aware of fail to work well in practice except in very simple cases.

We are, by definition, only interested in $f^{-1}(0)$ so we can ignore points that are sufficiently far from this zero set. More precisely, we observe the following: If $f^i(x)$ is positive for all x in a geometric simplex σ then so is $f^i_{PL}(x)$, because $f^i_{PL}(x)$ is a convex combination of the (positive) values at the vertices. This in turn implies that $F^i_{PL}(x, \tau)$ is positive on $\sigma \times [0, 1]$, as for each τ it is convex combination of positive numbers. The same argument holds for negative values. So we see that

▶ Remark 1. Write Σ_0 for the set of all $\sigma \in \mathcal{T}$, such that $(f^i)^{-1}(0) \cap \sigma \neq \emptyset$ for all i. Then for all τ, $\{x \mid F_{PL}(x, \tau) = 0\} \subset \Sigma_0$.

The results will be expressed using constants defined in terms of f and the ambient triangulation $\mathcal{T}$.

▶ **Definition 2.** *We define*

$$\gamma_0 = \inf_{x\in\Sigma_0} |\det(\operatorname{grad}(f^i)\cdot\operatorname{grad}(f^j))_{i,j}| \tag{2}$$

$$\gamma_1 = \sup_{x\in\Sigma_0} \max_i |\operatorname{grad}(f^i)| \tag{3}$$

$$\alpha = \sup_{x\in\Sigma_0} \max_i \|Hes(f^i)(x)\|_2 = \sup_x \max_i \|(\partial_k\partial_l f^i(x))_{k,l}\|_2, \tag{4}$$

$$D : \text{the longest edge length of a simplex in } \Sigma_0 \tag{5}$$

$$T : \text{the smallest thickness of a simplex in } \Sigma_0. \tag{6}$$

Here $\operatorname{grad}(f^i) = (\partial_j f_i)_j$ *denotes the gradient of component* f^i, $\det(\operatorname{grad}(f^i)\cdot\operatorname{grad}(f^j))_{i,j}$ *denotes the determinant of the matrix with entries* $\operatorname{grad}(f^i)\cdot\operatorname{grad}(f^j)$, $\|\cdot\|_2$ *the operator 2-norm, and* $(\partial_k\partial_l f_i)_{k,l}$ *the matrix of second order derivatives, that is the Hessian (Hes). We recall the definition of the operator norm:* $\|A\|_p = \max_{x\in\mathbb{R}^n} \frac{|Ax|_p}{|x|_p}$, *with* $|\cdot|_p$ *the p-norm on* $\mathbb{R}^n$. *The thickness is the ratio of the height over the longest edge length.*

We will assume that $\gamma_0, \gamma_1, \alpha, D, T \in (0, \infty)$. The constant γ_0 quantifies how close 0 to not being a regular value of f. The thickness is a measure of how well shaped the simplices are. A good choice for $\mathcal{T}$ is the Coxeter triangulation of type A_d, see [24, 22], or the related Freudenthal triangulations, see [34, 36, 32, 49], which can be defined for different values of D while keeping T constant. In this paper, we will thus think of all the above quantities as well as d and n as constants except D and our results will hold for D small enough.

The result

We are going to construct an ambient isotopy based on (1). The zero set of $F_{PL}(x, 0)$ (or $f(x)$) gives the smooth isosurface, while the zero set of $F_{PL}(x, 1)$ (or $f_{PL}(x)$) gives the PL approximation, that is the triangulation of the isosurface after possible barycentric subdivision. The map $\tau \mapsto \{x \mid F_{PL}(x, \tau) = 0\}$ in fact gives an isotopy. Without too much extra work we will also bound the Fréchet distance between $f(x)$ and $f_{PL}(x)$.

Proving isotopy consists of two technical steps, which consume most of the space in the proof below, as well as the use of a standard observation from Morse theory/gradient flow in the third step. The technical steps are

- Let $\sigma \in \mathcal{T}$. We first show that $\{(x, \tau) \mid F_{PL}(x, \tau) = 0\} \cap (\sigma \times [0, 1])$ is a smooth manifold, under certain conditions (Corollary 8).
- We prove that $F_{PL}^{-1}(0)$ is a manifold, under certain conditions, using techniques from nonsmooth analysis (Corollary 19).

Along the way we shall also see that $F_{PL}^{-1}(0)$ is never tangent to the $\tau = c$ planes, where c is a constant. The gradient of $(x, \tau), \mapsto \tau$ in the ambient space is $(0, 1)$. Projecting this vector onto the tangent space of $F_{PL}^{-1}(0)$ gives the gradient of $(x, \tau), \mapsto \tau$ restricted to $F_{PL}^{-1}(0)$. Because of the non-tangency property, this projection is non-zero. So the gradient field of the function $(x, \tau), \mapsto \tau$ restricted to $F_{PL}^{-1}(0)$, is piecewise smooth (because $F_{PL}^{-1}(0)$ is piecewise smooth) and never vanishes.

Now we arrive at the third step, which is similar to a standard observation in Morse theory [38, 39], with the exception that we now consider piecewise-smooth instead of smooth vector fields. We refer to Milnor [38] for an excellent introduction, see Lemma 2.4 and Theorem 3.1 in particular.

▶ **Lemma 3** (Gradient flow induced isotopies)**.** *The flow of a non-vanishing piecewise-smooth gradient vector field of a function* τ *on a compact manifold generates an isotopy from* $\tau = c_1$ *to* $\tau = c_2$, *where* c_1 *and* c_2 *are constants.*

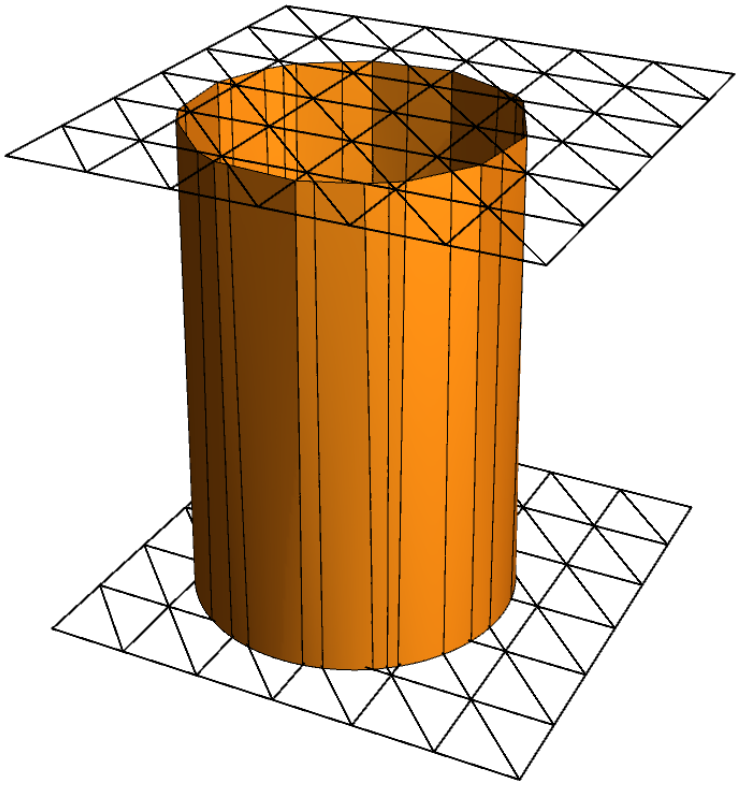


Figure 1 A pictorial overview of the proof. The τ-direction goes upwards. Similarly to Morse theory we find that $f_{PL}^{-1}(0)$ (top) and $f^{-1}(0)$ (bottom) are homeomorphic if the function τ restricted to $F_{PL}^{-1}(0)$ does not encounter a Morse critical point.

Bounds on the gradient of τ on the manifold give a bound on the Fréchet distance, which is defined as follows:

▶ **Definition 4** (Fréchet distance for embedded manifolds)**.** *Let $\mathcal{M}$ and $\mathcal{M}'$ be two homeomorphic, compact submanifolds of $\mathbb{R}^d$. Write $\mathcal{H}$ for the set of all homeomorphisms from $\mathcal{M}$ to $\mathcal{M}'$. The Fréchet distance between $\mathcal{M}$ and $\mathcal{M}'$ is*

$$d_F(\mathcal{M}, \mathcal{M}') = \inf_{h \in \mathcal{H}} \sup_{x \in \mathcal{M}} d(x, h(x)).$$

2.1 Estimates for a single simplex

We now first concentrate on a single simplex σ and write f_L for the linear function whose values on the vertices of σ coincide with f, that is f_L is the linear extension of the interpolation of f. Note that f_L coincides with f_{PL} within the geometric simplex σ (but not necessarily outside).

2.1.1 Preliminaries and variations of know results

We need a simple estimate similar to Proposition 2.1 of Allgower and George [2].

▶ **Lemma 5.** *Let $\sigma \subset \Sigma_0$ and let f_L be as described above. Then $|f_L^i(x) - f^i(x)| \leq 2D^2\alpha$ for all $x \in \sigma$.*

We will also be using an estimate similar to Proposition 2.2 of Allgower and George [2].

▶ **Proposition 6.** *Let $\sigma \subset \Sigma_0$ and let f_L be as described above. Then*

$$|\mathrm{grad} f_L^i - \mathrm{grad} f^i| = \sqrt{\sum_j (\partial_j f_L^i(x) - \partial_j f^i(x))^2} \leq \frac{4dD\alpha}{T},$$

for all x in the geometric simplex σ.

2.1.2 Estimates on the gradient inside a single simplex

We write

$$F_L(x,\tau) = (1-\tau)f(x) + \tau f_L(x). \tag{7}$$

We note that F_L extends smoothly outside σ, that is we can think of $F_L : \mathbb{R}^d \to \mathbb{R}^{d-n}$. Here and throughout we restrict ourselves to the setting where $\tau \in [0,1]$.

We now find the following

▶ **Lemma 7.** *If we write $\mathrm{grad}_{(x,\tau)}$ for the gradient that includes the τ component, we have*

$$|\det(\mathrm{grad}_{(x,\tau)}(F_L^i) \cdot \mathrm{grad}_{(x,\tau)}(F_L^j))_{i,j}| > \gamma_0 - g_1(D), \tag{8}$$

with $g_1(D) = \mathcal{O}(D)$. See Appendix A of [17] for the exact expression of g_1.

▶ **Corollary 8** ($F_L^{-1}(0)$ is a manifold in a neighbourhood of $\sigma \times [0,1]$)**.** *If $\gamma_0 > g_1(D)$ the implicit function theorem applies to $F_L(x,\tau)$ inside $\sigma \times [0,1]$. (In fact it applies to an open neighbourhood of this set). In particular, we have proven the first of our two technical steps, $\{(x,\tau) \mid F_{PL}(x,\tau) = 0\} \cap (\sigma \times [0,1])$ is a smooth manifold.*

2.1.3 Transversality with regard to the τ-direction

We will also prove the main result which we need for the third step, that is the gradient of τ restricted to $F_{PL}^{-1}(0)$, is piecewise smooth and never vanishes. We now prove that inside each $\sigma \times [0,1]$ the gradient of τ on $F_L^{-1}(0)$ is smooth and does not vanish.

We first give a simple lower bound on the lengths of vectors $v^1, \ldots, v^{d-n}$, assuming that the norms $|v^i|$ are upper bounded and the determinant of the Gram matrix is lower bounded.

▶ **Lemma 9.** *Let $v^1, \ldots, v^{d-n} \in \mathbb{R}^d$, $|v^i| \leq \gamma_1$, for all i, and assume that $\det(v^i \cdot v^j)_{i,j} > \gamma_0$. Then $|v^i| \geq \sqrt{\gamma_0}/\gamma_1^{d-n-1}$.*

We also need to bound the angle of the vectors $\mathrm{grad}_{(x,\tau)}(F_L^i)$ and the x plane, that is $\mathbb{R}^d \subset \mathbb{R}^{d+1}$. We recall the definition. If $v \in \mathbb{R}^{d+1}$ is a vector and $\Xi = \mathbb{R}^d \subset \mathbb{R}^{d+1}$, is the space spanned by the d basis vectors corresponding to the x-directions, the angle between v and Ξ is $\angle(v,\Xi) = \inf_{w \in \Xi} \angle(v,w)$.

▶ **Lemma 10.** *Let Ξ be as above. We have*

$$\tan \angle(\mathrm{grad}_{(x,\tau)}(F_L^i), \Xi) \leq \frac{2D^2\alpha}{\sqrt{\gamma_0}/\gamma_1^{d-n-1} - \frac{4dD\alpha}{T}}.$$

In particular the manifold $F_L^{-1}(0)$ inside $\sigma \times [0,1]$ is never tangent to the $\tau = c$ planes, where c is a constant.

Combining Lemma 10 and Corollary 8 gives:

▶ **Corollary 11.** *If $\gamma_0 > g_1(D)$, and $\sqrt{\gamma_0}/\gamma_1^{d-n-1} > \frac{4dD\alpha}{T}$, then inside each $\sigma \times [0,1]$ the gradient of τ on $F_L^{-1}(0)$ is smooth and does not vanish.*

2.2 Global result

We are now going to prove the global result. For this, we need to recall some definitions and results from non-smooth analysis. We refer to [23] for an extensive introduction.

▶ **Definition 12** (Generalized Jacobian, Definition 2.6.1 of [23]). *Let $F : \mathbb{R}^{d+1} \to \mathbb{R}^{d-n}$, where F is assumed to be just Lipschitz. The generalized Jacobian of F at x_0 denoted by $J_F(x_0)$, is the convex hull of all $(d-n)\times(d+1)$-matrices B obtained as the limit of a sequence of the form $J_F(x_i)$, where $x_i \to x_0$ and F is differentiable at x_i.*

Following [23, page 253] we also define:

▶ **Definition 13.** *The generalized Jacobian $J_F(x_0)$ is said to be of maximal rank provided every matrix in $J_F(x_0)$ is of maximal rank.*

Write $\mathbb{R}^{d+1} = \mathbb{R}^{n+1} \times \mathbb{R}^{d-n}$ and denote the coordinates of $\mathbb{R}^{d+1}$ by (x, y) accordingly. Fix a point (a, b), with $F(a, b) = 0 \in \mathbb{R}^{d-n}$. We now write:

▶ **Notation 14** ([23, page 256]). *$J_F(x_0, y_0)|_y$ is the set of all $(n+1)\times(n+1)$-matrices M such that, for some $(n+1)\times(d-n)$-matrix N, the $(n+1)\times(d+1)$-matrix $[N, M]$ belongs to $J_F(x_0, y_0)$.*

With these definitions and notations we now have:

▶ **Theorem 15** (The generalized implicit function theorem [23, page 256]). *Suppose that $J_F(a, b)|_y$ is of maximal rank. Then there exists an open set $U \subset \mathbb{R}^{n+1}$ containing a such that there exists a Lipschitz function $g : U \to \mathbb{R}^{d-n}$, such that $g(a) = b$ and $F(x, g(x)) = 0$ for all $x \in U$.*

We recall the definition of F_{PL},

$$F_{PL}(x, \tau) = (1-\tau)f(x) + \tau f_{PL}(x). \tag{1}$$

Further recall that the closed star of a vertex v in a simplicial complex is the closure of all simplices in the complex that contain v.

Because of the definition of α, see (4), and Proposition 6, we have that $\operatorname{grad}_{(x,\tau)} F_{PL}(x, \tau)$ and $\operatorname{grad}_{(x,\tau)} F_{PL}(\tilde{x}, \tau)$ are close if x and $\tilde{x}$ are. In particular,

▶ **Lemma 16.** *Let v be a vertex in $\mathcal{T}$, $x_1, x_2 \in \mathrm{star}(v)$, and $\tau_1, \tau_2 \in [0, 1]$, such that $\operatorname{grad}_{(x,\tau)} F^i_{PL}(x_1, \tau_1)$ and $\operatorname{grad}_{(x,\tau)} F^i_{PL}(x_2, \tau_2)$ are well defined, then*

$$|\operatorname{grad}_{(x,\tau)} F^i_{PL}(x_1, \tau_1) - \operatorname{grad}_{(x,\tau)} F^i_{PL}(x_2, \tau_2)| \leq \frac{10d^2 D\alpha}{T} + 4\gamma_1 D + 4D^2\alpha.$$

We now immediately have the same bound on points in the convex hull of a number of such vectors:

▶ **Corollary 17.** *Suppose we are in the setting of Lemma 16 and $x_0, x_1, \ldots, x_m \in \mathrm{star}(v)$, $\tau_0, \ldots, \tau_m \in [0, 1]$, and suppose that $\mu_1, \ldots, \mu_m$ are positive weights such that $\mu_1 + \cdots + \mu_m = 1$ then,*

$$\left| \operatorname{grad}_{(x,\tau)} F^i_{PL}(x_0, \tau_0) - \sum_{k=1}^{m} \mu_k \operatorname{grad}_{(x,\tau)} F^i_{PL}(x_k, \tau_k) \right| \leq \frac{10d^2 D\alpha}{T} + 4\gamma_1 D + 4D^2\alpha.$$

Using Lemma 7, we see

▶ **Lemma 18.** *Let v be a vertex in $\mathcal{T}$, $x_1, \ldots, x_m \in star(v)$, and $\tau_1, \ldots, \tau_m \in [0,1]$, such that $\operatorname{grad}_{(x,\tau)} F^i_{PL}(x_k, \tau_k)$, $k = 0, \ldots, m$ are well defined. If we moreover assume $D \leq 1$, and $\frac{6dD\alpha}{T} \leq \gamma_1$ we have that*

$$\left| \det \left(\left(\sum_{k=1}^{m} \mu_k \operatorname{grad}_{(x,\tau)} F^i_{PL}(x_k, \tau_k) \right) \cdot \left(\sum_{k=1}^{m} \mu_k \operatorname{grad}_{(x,\tau)} F_{PL}(x_k, \tau_k) \right) \right)_{i,j} \right| \geq \gamma_0 - g_2(D),$$

with $g_2(D) = \mathcal{O}(D)$. See Appendix A of [17] for the exact expression of g_2.

From the previous lemma, we immediately have that

▶ **Corollary 19** ($\{(x,\tau) \mid F_{PL}(x,\tau) = 0\}$ is a manifold)**.** *If $D \leq 1$, $\frac{6dD\alpha}{T} \leq \gamma_1$, and $\gamma_0 > g_2(D)$ the generalized implicit function theorem, Theorem 15, applies to $F_{PL}(x,\tau) = 0$. In particular, $\{(x,\tau) \mid F_{PL}(x,\tau) = 0\}$ is a manifold.*

We notice that this bound is stronger than the bound in Corollary 8, that is $g_1(D) \leq g_2(D)$. This means that $F^{-1}_{PL}(0)$ is a Piecewise-Smooth manifold if the conditions of Corollary 19 hold. The second technical step of the proof is now also completed.

The fact that $F_L(x,\tau) = 0$ is a Piecewise-Smooth manifold and Corollary 11 give that the gradient of τ is a Piecewise-Smooth vector field whose flow we can integrate to give a isotopy ι from the zero set of f to that of f_{PL}.

We summarize in a theorem:

▶ **Theorem 20.** *If, $D \leq 1$, $\frac{6dD\alpha}{T} \leq \gamma_1$, $\sqrt{\gamma_0}/\gamma_1^{d-n-1} > \frac{4dD\alpha}{T}$, and $\gamma_0 > g_2(D)$ then the zero set of f_{PL} is a manifold isotopic to the zero set of f. We stress that one can satisfy all conditions by choosing D sufficiently small.*

2.2.1 Fréchet distance

To bound the Fréchet distance (d_F) between the zero sets of $f(x)$ and f_{PL}, it suffices to bound the angle that the gradient of τ, as restricted to $F^{-1}_{PL}(0)$), makes with the (ambient) τ-direction.

For this we will use the angle bound of Lemma 10, together with some estimates that are similar in spirit to those in [8, Lemma C.13].

▶ **Lemma 21.** *Let $v^1, \ldots, v^{d-n} \in \mathbb{R}^{d+1}$, $|v^i| \leq \tilde{\gamma}_1$, for all i, and assume that $\det(v^i \cdot v^j)_{i,j} > \tilde{\gamma}_0 > 0$. Let e_τ be a unit vector. If for all i, $\cos(\angle v^i, e_\tau) \leq \phi_0$, then for any $w \in \operatorname{span}(v^1, \ldots, v^{d-n})$*

$$\cos \angle(w, e_\tau) \leq \frac{(d-n) d^{d-n-1} \phi_0 \tilde{\gamma}_1^{d-n}}{\sqrt{\tilde{\gamma}_0}}.$$

Let e_τ be the τ direction and let g_τ be the gradient of τ restricted to $F^{-1}_{PL}(0)$, whenever it exists. We want to bound the angle of g_τ and the τ-direction. Because the isotopy ι is given by integrating the gradient flow and we have a bound on the norm of the gradient, the Fréchet distance is bounded (by the norm of the gradient because the time of the flow is 1).

There is one subtlety, because the manifold is only Piecewise-Smooth, we need to take into account the points where g_τ is not uniquely defined. Because for each simplex σ, F_L extends to a neighbourhood of $\sigma \times [0,1]$, there exists a limit of $g_\tau(x_i, \tau_i)$ for any sequence (x_i, τ_i) that lies in $\operatorname{int}(\sigma) \times [0,1]$, where int denotes the interior. This means that if we bound g_τ for each simplex we also bound its limits, where the limits are as just described.

We are now ready to combine Lemmas 10, 21, and Theorem 20.

▶ **Corollary 22** (Bound on the Fréchet distance). *Suppose that the conditions of Theorem 20 are satisfied. Then,* $d_F(f^{-1}(0), f_{PL}^{-1}(0)) \leq \tan \arcsin g_3(D)$, *with* $g_3(D) = \mathcal{O}(D^2)$, *where we think of* γ_0, γ_1, d, n, T *and* α *as constants. See Appendix A of [17] for the exact expression of* g_2.

The most important thing to observe is that $\tan(\arcsin(x)) = \frac{x}{\sqrt{1-x^2}}$, so that we find that $d_F(f^{-1}(0), f_{PL}^{-1}(0)) = \mathcal{O}(D^2)$, where we think of γ_0, γ_1, d, n, T and α as constants.

3 Isomanifolds with boundary

We will now consider isomanifolds with boundary. By this we mean that on top of the function $f : \mathbb{R}^d \to \mathbb{R}^{d-n}$, we'll have another function $f_\partial : \mathbb{R}^d \to \mathbb{R}$ and the set we consider is $M = f^{-1}(0) \cap f_\partial^{-1}([0, \infty))$. This is a manifold with boundary if the gradients of f^i span a $(d-n)$-dimensional space at each point of $f^{-1}(0)$ and the gradients of f^i and f_∂ span a $(d-n+1)$-dimensional space at each point of $\partial M = f^{-1}(0) \cap f_\partial^{-1}(0)$, as a consequence of the submersion theorem. We will again write f_{PL} for the PL interpolation of f. Similarly we write $f_{\partial,PL}$ for the PL interpolation of f_∂. We prove that, under certain conditions, there is an isotopy from $f^{-1}(0) \cap f_\partial^{-1}([0, \infty))$ to $f_{PL}^{-1}(0) \cap f_{\partial,PL}^{-1}([0, \infty))$. The conditions are very similar to the conditions we have before, but of course we need to include bounds on the gradient of $f_{\partial,PL}$.

Overview of the proof

We will again construct an isotopy, but in this case it will consist of two steps.

- In the **first step**, we isotope the part of $f^{-1}(0)$ that is far from $f_\partial^{-1}(0)$ to its piecewise linear approximation, while leaving the part of $f^{-1}(0)$ that is close to $f_\partial^{-1}(0)$ smooth. We will denote the result by $M_1 = (F_{PL,1}(\cdot, 1))^{-1}(0)$, see (9).
- In the **second step**, we consider a (small) tubular neighbourhood around $f_\partial^{-1}(0)$ as restricted to M_1 by looking at all $f_\partial^{-1}(\epsilon)$ for $|\epsilon|$ sufficiently small.[1] We then isotope $M_1 \cap f_\partial^{-1}(\epsilon)$ to its piecewise linear approximation. Again the isotopy is chosen in such a way that for ϵ relatively large (for the points such that M_1 is already Piecewise-Linear) it leaves $M_1 \cap f_\partial^{-1}(\epsilon)$ invariant. This gives an isotopy of a tubular neighbourhood of $\partial M_1 = M_1 \cap f_\partial^{-1}(0)$ to its Piecewise-Linear approximation.

We will first partition the manifold in two parts using a smooth bump function $\phi : \mathbb{R} \to [0, 1]$ that is zero in a neighbourhood of zero and $\phi(y) = 1$ if $|y| > y_0$, for some $y_0 > 0$. Such bump functions can be easily constructed, see for example [35, Section 2.2]. We will be using the function $\phi\left(\sum_i (f^i)^2 + f_\partial^2\right)$.

The first step will be using the zero set of the following function:

$$F_{PL,1}(x, \tau) = \left(1 - \tau\phi\left(\sum_i (f^i)^2 + f_\partial^2\right)\right) f(x) + \tau\phi\left(\sum_i (f^i)^2 + f_\partial^2\right) f_{PL}(x), \tag{9}$$

on which we'll apply the same gradient flow argument as before.

The resulting set M_1 is the same zero set of f_{PL} as before if we stay sufficiently far away from ∂M and the isotopy leaves the manifold invariant close to ∂M. In particular, $\partial M_1 = \partial M$.

[1] We stress that ϵ may be negative.

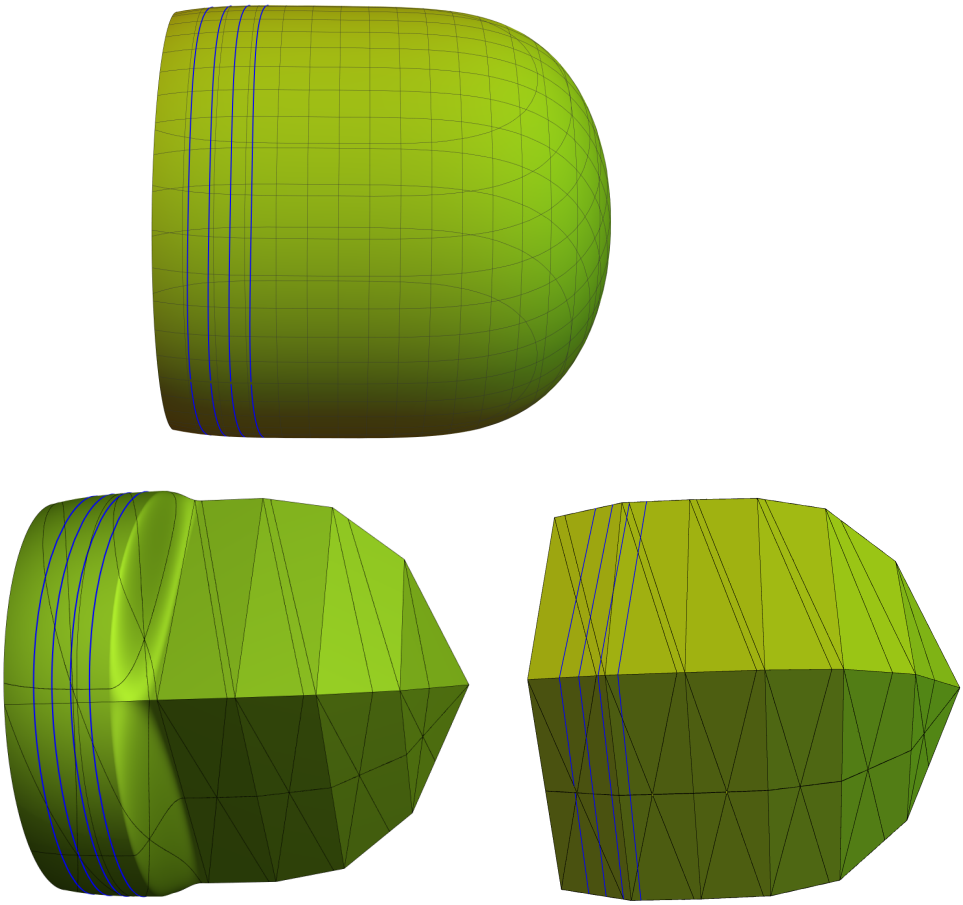


Figure 2 Top: we see the original isosurface with $f_\partial^{-1}(-1/10)$, $f_\partial^{-1}(0)$, $f_\partial^{-1}(1/10)$, and $f_\partial^{-1}(2/10)$ indicated in blue. Bottom left: we see that at the end of Step 1 the neighbourhood of the boundary is intact, while the rest has been isotoped to a Piecewise-Linear approximation. Bottom right: we have also isotoped the neighbourhood of the boundary to a Piecewise-Linear approximation by isotoping $f_\partial^{-1}(\epsilon)$, to its Piecewise-Linear approximation for all sufficiently small ϵ.

In the second step, we define an isotopy that will act only on a small neighbourhood of ∂M. Consider the sets $B_1(\epsilon) = M_1 \cap f_\partial^{-1}(\epsilon)$ and, for each ϵ, define the function

$$\begin{aligned} F_{PL,2,\epsilon}(x,\tau) = &\left(1 - \tau\psi\left(\sum_i (f^i)^2 + f_\partial^2\right)\right)(F_{PL,1}(x,1), f_\partial(x) - \epsilon) \\ &+ \tau\psi\left(\sum_i (f^i)^2 + f_\partial^2\right)(f_{PL}(x), f_{\partial,PL}(x) - \epsilon), \end{aligned} \tag{10}$$

where $\psi : \mathbb{R} \to [0,1]$ is now a smooth bump function that is 1 in a sufficiently large neighbourhood of zero (somewhat larger than y_0) and zero outside some compact set. We stress that $F_{PL,2,\epsilon}$ is a mapping from $\mathbb{R}^d \times [0,1]$ to $\mathbb{R}^{d-n+1}$. Using the result for isomanifolds (with some modifications), we can prove that each individual set $B_1(\epsilon)$ is isotopic to $f_{PL}^{-1}(0) \cap f_{\partial,PL}^{-1}(\epsilon)$ for small ϵ while, for sufficiently large ϵ, it leaves the set invariant.

3.1 Step 1

The proof closely follows the proof for the case without boundary in Section 2. The main technical difficulty will be to provide bounds that serve as the counterparts of Lemmas 7 and 18. To be able to do so, we first need to discuss bounds on the bump functions ϕ and ψ.

3.1.1 Bump functions

Following [35, Section 2.2], we write

$$\zeta_1(x) = \begin{cases} 0 & \text{if } x \leq 0 \\ e^{-1/x} & \text{if } x > 0 \end{cases}$$

For $0 < y_1 < y_2$ we write $\zeta_2(x) = \zeta_1(x - y_1)\zeta_1(y_2 - x)$. Then we define $\phi_l : \mathbb{R} \to [0, 1]$ by $\phi_l(x) = \int_x^{y_2} \zeta_2(x')\mathrm{d}x' \Big/ \int_{y_1}^{y_2} \zeta_2(x')\mathrm{d}x'$. Finally define $\phi_b : \mathbb{R} \to [0, 1]$ by $\phi_b(x) = \phi_l(|x|)$, and let $\phi(x) = 1 - \phi_b(x)$.

▶ **Lemma 23.** *We have $\phi_b(x) \in [0, 1]$ and, writing $2y_1 = y_2 = y_0$,*

$$\partial_x(\phi_l(x)) \leq 2\frac{e^{\frac{4}{3(y_2-y_1)}}}{y_2 - y_1} = 4\frac{e^{\frac{2}{3y_0}}}{y_0} = \gamma_\phi. \tag{11}$$

3.1.2 Inside a single simplex

Similarly to Lemma 7, we now give a condition that ensures that the zero set of $F^i_{PL,1}(x, \tau)$ is smooth inside $\sigma \times [0, 1]$. In fact, similarly to (7), we define

$$\begin{aligned} F^i_{L,1}(x, \tau) &= \left(1 - \tau\phi\left(\sum_l (f^l)^2 + f_\partial^2\right)\right) f^i(x) + \tau\phi\left(\sum_l (f^l)^2 + f_\partial^2\right) f^i_L(x) \\ &= f(x) + \tau\phi\left(\sum_i (f^i)^2 + f_\partial^2\right)(f^i_L(x) - f^i(x)), \end{aligned}$$

where ϕ is as defined above. Observe that $F^i_{L,1}(x, \tau)$ can be extended to a neighbourhood of $\sigma \times [0, 1]$.

▶ Remark 24. For the constants, it is better if y_0 can be chosen as large as possible, but we need y_1 to be quite a bit larger than y_0. In turn, we cannot choose y_1 arbitrarily large because this would mean that the gradient field $\operatorname{grad} f_\partial|_{f^{-1}(0)}$ (seen as restricted on $f^{-1}(0)$) would never vanish. The latter is in general impossible thanks to the hairy ball theorem [18].

We introduce the following definition that complements Definition 2:

▶ **Definition 25.**

$$\gamma_2 = \sup_{x \in \Sigma_0} \left|\operatorname{grad}\left(\sum_l (f^l)^2 + f_\partial^2\right)\right| = 2 \sup_{x \in \Sigma_0} \left|\sum_l f^l \operatorname{grad} f^l + f_\partial \operatorname{grad} f_\partial\right| \tag{12}$$

We have then the analog of Lemma 7:

▶ **Lemma 26.** *We have :*

$$|\det(\operatorname{grad}_{(x,\tau)} F^i_{L,1}(x, \tau) \cdot \operatorname{grad}_{(x,\tau)} F^j_{L,1}(x, \tau))_{i,j}| > \gamma_0 - g_4(D),$$

with $g_4(D) = \mathcal{O}(D)$. The exact expression of g_4 is given in [17, Appendix A].

The following corollary is then the analog of Corollary 8:

▶ **Corollary 27** ($F^{-1}_{L,1}(0)$ is a manifold)**.** *If $\gamma_0 > g_4(D)$, where $g_4(D) = O(D)$ is as in Lemma 26, then $F^{-1}_{L,1}(0)$ is a smooth manifold inside an ϵ neighbourhood of $\sigma \times [0, 1]$.*

3.1.3 Transversality with regard to the τ-direction

We note that, similarly to Lemma 10, we have

▶ **Lemma 28.** *Let Ξ be as in Lemma 10 and γ_ϕ as in* (11).

$$\tan \angle(\mathrm{grad}_{(x,\tau)}(F_{L,1}), \Xi) \leq \frac{2D^2\alpha}{\sqrt{\gamma_0}/\gamma_1^{d-n-1} - \gamma_2\gamma_\phi 2D^2\alpha - \frac{4dD\alpha}{T}}.$$

In particular, if $\sqrt{\gamma_0}/\gamma_1^{d-n-1} > \gamma_2\gamma_\phi 2D^2\alpha + \frac{4dD\alpha}{T}$, $F_{L,1}^{-1}(0)$ (if it is a manifold) is never tangent to the $\tau = c$ planes, where c is a constant.

Now, similarly to Corollary 11, we find that

▶ **Corollary 29** (Transversality with respect to τ for Step 1)**.** *Suppose that $\gamma_0 > g_4(D)$ and that $\sqrt{\gamma_0}/\gamma_1^{d-n-1} > \gamma_2\gamma_\phi 2D^2\alpha + \frac{4dD\alpha}{T}$. Then, inside each $\sigma \times [0,1]$, the gradient of τ on $F_{L,1}^{-1}(0)$ is smooth and does not vanish.*

3.1.4 Global result

We now have to prove that $F_{PL,1}^{-1}(0)$ is a manifold. For this, we shall use a bound similar to the one given in Lemma 18, so that we are able to apply the generalized implicit function theorem if this bound is satisfied. But first of all, we need the following bound, which is similar to Lemma 16.

▶ **Lemma 30.** *Assuming that the gradients are well defined, we have $|\mathrm{grad}_{(x,\tau)}F_{PL,1}^i(x_1,\tau_1) - \mathrm{grad}_{(x,\tau)}F_{PL,1}^i(x_2,\tau_2)| \leq g_5(D)$, with $g_5(D) = \mathcal{O}(D)$. The expression for g_5 is given in [17, Appendix A].*

Just as in Corollary 17, we immediately have the same bound on points in the convex hull of a number of such vectors:

▶ **Corollary 31.** *Suppose we are in the setting of Lemma 30 and $x_0, x_1, \ldots, x_m \in star(v)$, $\tau_0, \ldots, \tau_m \in [0,1]$, such that $\mathrm{grad}_{(x,\tau)}F_{PL,1}^i(x_i,\tau_i)$ is well defined for all i. Further assume that $\mu_1, \ldots, \mu_m$ are positive weights such that $\mu_1 + \cdots + \mu_m = 1$. Then,*

$$\left|\mathrm{grad}_{(x,\tau)}F_{PL,1}^i(x_0,\tau_0) - \sum_{k=1}^m \mu_k \mathrm{grad}_{(x,\tau)}F_{PL,1}^i(x_k,\tau_k)\right| \leq g_5(D).$$

▶ **Lemma 32.** *Under the same conditions as in Lemma 18,*

$$\det\left(\left(\sum_{k=1}^m \mu_k \,\mathrm{grad}_{(x,\tau)}F_{PL,1}^i(x_k,\tau_k)\right)\cdot\left(\sum_{k=1}^m \mu_k \,\mathrm{grad}_{(x,\tau)}F_{PL,1}^j(x_k,\tau_k)\right)\right)_{i,j}$$
$$\geq \gamma_0 - g_4(D) - g_6(D),$$

with $g_6(D) = \mathcal{O}(D)$. The exact expression of g_6 is given in [17, Appendix A].

Lemma 32 immediately yields that

▶ **Corollary 33** ($F_{PL,1}^{-1}(0)$ is a manifold)**.** *If, $\gamma_0 > g_4(D) + g_6(D)$ the generalized implicit function theorem, Theorem 15, applies to $F_{PL,1}(x,\tau) = 0$. In particular $F_{PL,1}^{-1}(0)$ is a manifold.*

We stress again that inside the set $\{x | \phi\left(\sum_i (f^i)^2(x) + f_\partial^2(x)\right) = 1\}$ the zero set of $F_{PL,1}(x,1)$ coincides with the zero set of $f_{PL}(x)$.

3.2 Step 2

Before we can proceed we have to specify the bump function ψ. We suppose that

$$\psi(x) = \begin{cases} 1 & \text{if } |x| \leq \frac{101}{100} y_0 \\ 0 & \text{if } |x| \geq 2y_0. \end{cases}$$

In particular we pick $\psi(x) = \phi_b(x)$, with the choice $y_1 = \frac{101}{100} y_0$ and $y_2 = 2y_0$.

First we stress that the zero set of $F_{PL,2,\epsilon}(x, 1)$ coincides with the zero set of $(f_{PL}(x), f_{\partial,PL}(x) - \epsilon)$, provided that $\psi(\sum_i f_i(x)^2 + f_\partial(x)^2) = 1$.

Secondly, we now claim the following:

▶ **Lemma 34.** *The zero set of $F_{PL,2,\epsilon}(x, 1)$ is a subset of the zero set of $f_{PL}(x)$, for each ϵ.*

The technical result that remains to be proven is the counterpart of Theorem 20 for $F_{PL,2,\epsilon}(x, \tau)$ and for each sufficiently small ϵ. To be precise it suffices for $\epsilon \leq 2y_0$. We remark that it is likely that this bound on ϵ can be improved.

We again follow the same path to prove this result. That is we first concentrate on a single simplex and prove that inside that simplex the zero set of $F_{PL,2,\epsilon}$ is a smooth manifold on which the gradient of τ as restricted to the manifold does not vanish. We then prove that is the zero set of $F_{PL,2,\epsilon}$ is globally a manifold.

3.2.1 Assumptions and notations

Because we are now faced with both $f(x)$ and $f_\partial(x)$ we need to introduce a bound on how far the gradients of all there are from being colinear. We write

$$f_B(x) = (f(x), f_\partial(x)). \tag{13}$$

Before we were only interested in the set Σ_0, similarly here we sometimes concentrate on a neighbourhood of the zero set of both f_∂ and f. Therefore we write B_ν for all $\sigma \in \mathcal{T}$ such that $(\sum_l (f^l)^2 + (f_\partial)^2)^{-1}([-2y_0, 2y_0]) \cap \sigma \neq \emptyset$.

We define γ_0^B in terms of the determinant of the Gram matrix of the gradients, that is

$$\gamma_0^B = \inf_{x \in B_\nu \cap \Sigma_0} |\det(\operatorname{grad}(f_B^i) \cdot \operatorname{grad}(f_B^j))_{i,j}|. \tag{14}$$

We note that because we take the gradients we can just ignore the ϵ constant. For the lengths of the gradients of f_B we define,

$$\gamma_1^B = \sup_{x \in \Sigma_0} \max_i |\operatorname{grad}(f_B^i)|, \tag{15}$$

for all $1 \leq i \leq d - n + 1$. Similarly to α, we define β as the bound on the operator 2-norm of all Hessians of f_B, that is

$$\beta = \sup_{x \in \Sigma_0} \max_i \|\operatorname{Hes}(f_B^i)\|_2 = \sup_{x \in \Sigma_0} \max_i \|(\partial_k \partial_l f_B^i)_{k,l}\|_2. \tag{16}$$

We stress that we have chosen our definitions such that $\alpha \leq \beta$.

We use the same notation for the ambient triangulation $\mathcal{T}$, the lower bound on the thickness of the simplices T and upper bound on the longest edge length D. We also need to introduce a bound on the differential of the bump function ψ. Similarly to (11) we define,

$$\gamma_\psi = 2\frac{e^{\frac{4}{3(y_2 - y_1)}}}{y_2 - y_1} = 2\frac{e^{\frac{4}{3(2y_0 - \frac{101}{100} y_0)}}}{2y_0 - \frac{101}{100} y_0} = \frac{200}{99} \frac{e^{\frac{400}{297 y_0}}}{y_0}, \tag{17}$$

because we picked $y_1 = \frac{101}{100} y_0$ and $y_2 = 2y_0$, for ψ.

3.2.2 Inside a single simplex

Similarly to Lemma 26, we now give a condition that ensure that the zero set of $F_{PL,2,\epsilon}(x,\tau)$ is smooth inside $\sigma \times [0,1]$. In fact similarly to (7), we define

$$F_{L,2,\epsilon}(x,\tau) = \left(1 - \tau\psi\left(\sum_i (f^i)^2 + f_\partial^2\right)\right)(F_{L,1}(x,1), f_\partial(x) - \epsilon) + \tau\psi\left(\sum_i (f^i)^2 + f_\partial^2\right)(f_L(x), f_{\partial,L}(x) - \epsilon),$$

which can be extended to a neighbourhood of $\sigma \times [0,1]$.

▶ **Lemma 35.** *For all ϵ, $\det(\mathrm{grad}_{(x,\tau)} F^i_{L,2,\epsilon}(x,\tau) \cdot \mathrm{grad}_{(x,\tau)} F^j_{L,2,\epsilon}(x,\tau))_{i,j} \geq \gamma_0^B - g_7(D)$ with $g_7(D) = \mathcal{O}(D)$. The exact expression of g_7 is given in [17, Appendix A].*

▶ **Corollary 36** ($F^{-1}_{L,2,\epsilon}(0)$ is a manifold)**.** *We have that $F^{-1}_{L,2,\epsilon}(0)$ is a smooth manifold inside an small neighbourhood of $\sigma \times [0,1]$ provided $\gamma_0^B > g_7(D)$, with $g_7(D)$ as in Lemma 35. As usual this can always be satisfied by choosing the triangulation fine enough, that is D sufficiently small.*

3.2.3 Transversality with regard to the τ-direction

Once more similarly to Lemma 10, we have

▶ **Lemma 37.** *Let Ξ be as in Lemma 10. We have*

$$\tan\angle(\mathrm{grad}_{(x,\tau)}(F_{L,2,\epsilon}), \Xi) \leq \frac{2D^2\beta}{\sqrt{\gamma_0^B}/(\gamma_1^B)^{d-n-2} - (\gamma_2(2\gamma_\phi + \gamma_\psi) + 1)2D^2\beta - \frac{12dD\beta}{T}}.$$

In particular the manifold $F^{-1}_{L,2,\epsilon}(0)$ inside $\sigma \times [0,1]$, if well defined, is never tangent to the $\tau = c$ planes, where c is a constant, if

$$\sqrt{\gamma_0^B}/(\gamma_1^B)^{d-n-2} > (\gamma_2(2\gamma_\phi + \gamma_\psi) + 1)2D^2\beta + \frac{12dD\beta}{T}.$$

Now similary to Corollary 11, we find that

▶ **Corollary 38** (Transversality with respect to τ for Step 2)**.** *Suppose that the conditions of Corollary 36 are satisfied. If moreover*

$$\sqrt{\gamma_0^B}/(\gamma_1^B)^{d-n-2} > (\gamma_2(2\gamma_\phi + \gamma_\psi) + 1)2D^2\beta + \frac{12dD\beta}{T},$$

then inside each $\sigma \times [0,1]$ the gradient of τ on $F^{-1}_{L,2,\epsilon}(0)$ is smooth and does not vanish.

3.2.4 Global result

We now have to prove that $F^{-1}_{PL,2,\epsilon}(0)$ is a manifold, for all sufficiently small ϵ. For this we shall use a bound similar to the one given in Lemma 18, so that we are able to apply the generalized implicit function theorem if this bound is satisfied. For this, we first need the following bound, which is similar to Lemma 30.

▶ **Lemma 39.** *Let v be a vertex in $\mathcal{T}$, $x_1, x_2 \in star(v)$, and $\tau_1, \tau_2 \in [0,1]$, such that $\mathrm{grad}_{(x,\tau)} F^i_{PL,2,\epsilon}(x_1,\tau_1)$ and $\mathrm{grad}_{(x,\tau)} F^i_{PL,2,\epsilon}(x_2,\tau_2)$ are well defined, then*

$$|\mathrm{grad}_{(x,\tau)} F^i_{PL,2,\epsilon}(x_1,\tau_1) - \mathrm{grad}_{(x,\tau)} F^i_{PL,2,\epsilon}(x_2,\tau_2)| \leq g_8(D),$$

with $g_8(D) = \mathcal{O}(D)$. The exact expression of g_8 is given in [17, Appendix A].

Just as in Corollary 17, we immediately have the same bound on points in the convex hull of a number of such vectors:

▶ **Corollary 40.** *Suppose $x_0, x_1, \ldots, x_m \in star(v)$, $\tau_0, \ldots, \tau_m \in [0,1]$, such that $\operatorname{grad}_{(x,\tau)} F^i_{PL,2,\epsilon}(x_i, \tau_i)$ is well defined for all i. Further assume that $\mu_1, \ldots, \mu_m$ are positive weights such that $\mu_1 + \cdots + \mu_m = 1$. Then,*

$$\left| \operatorname{grad}_{(x,\tau)} F^i_{PL,2,\epsilon}(x_0, \tau_0) - \sum_{k=1}^{m} \mu_k \operatorname{grad}_{(x,\tau)} F^i_{PL,2,\epsilon}(x_k, \tau_k) \right| \leq g_8(D).$$

▶ **Lemma 41.** *Under the same conditions as in Corollary 40,*

$$\det \left(\left(\sum_{k=1}^{m} \mu_k \operatorname{grad}_{(x,\tau)} F^i_{PL,2,\epsilon}(x_k, \tau_k) \right) \cdot \left(\sum_{k=1}^{m} \mu_k \operatorname{grad}_{(x,\tau)} F^j_{PL,2,\epsilon}(x_k, \tau_k) \right) \right)_{i,j} \geq \gamma_0^B - g_7(D) - g_9(D),$$

where $g_9(D) = \mathcal{O}(D)$. See Appendix A of [17] for the g_9.

Lemma 41 immediately yields that

▶ **Corollary 42** (The generalized implicit function theorem in Step 2)**.** *If, $\gamma_0^B > g_7(D) + g_9(D)$ the generalized implicit function theorem, Theorem 15, applies to $F_{PL,1}(x, \tau) = 0$. In particular $F^{-1}_{PL,1}(0)$ is a manifold.*

We stress that this condition only needs to be satisfied in a when $\sum_l (f^l)^2 + (f_\partial)^2$ is small, outside this neighbourhood the isotopy leaves the zero set invariant.

▶ **Theorem 43.** *If,*

$$\begin{aligned}
\sqrt{\gamma_0}/\gamma_1^{d-n-1} &> \gamma_2\gamma_\phi 2D^2\alpha + \frac{4dD\alpha}{T} && \text{(Corollary 29)}\\
\gamma_0 &> g_4(D) + g_6(D) && \text{(Corollaries 27 and 33)}\\
\sqrt{\gamma_0^B}/(\gamma_1^B)^{d-n-2} &> (\gamma_2(2\gamma_\phi + \gamma_\psi) + 1)2D^2\beta + \frac{12dD\beta}{T} && \text{(Corollary 38)}\\
\gamma_0^B &> g_7(D) + g_9(D), && \text{(Corollaries 36 and 42)}
\end{aligned}$$

then $f^{-1}(0) \cap f_\partial^{-1}([0,\infty))$ is isotopic to $f_{PL}^{-1}(0) \cap f_{\partial,PL}^{-1}([0,\infty))$. We stress that one can satisfy all conditions by choosing D sufficiently small. See Appendix A of [17] for the $g_i(D)$.

4 Isostratifolds

There is no obstruction in principle that prevents us from extending the approach above to isostratifolds. By isostratifolds we mean stratifolds that are given by the zero sets of functions and inequalities. See [17], for a short discussion of the approach. However, finding the precise constants involved would become prohibitively lengthy. Apart from some Delaunay based work on triangulations of stratifolds in three dimensions [43, 45, 28, 27, 26], we are not aware of similar results. Significant effort did go in the detection of strata, in this case in arbitrary dimension, see for example [6, 5].

References

1 Eugene L. Allgower and Kurt Georg. Simplicial and continuation methods for approximating fixed points and solutions to systems of equations. *Siam review*, 22(1):28–85, 1980.

2 Eugene L. Allgower and Kurt Georg. Estimates for piecewise linear approximations of implicitly defined manifolds. *Applied Mathematics Letters*, 2(2):111–115, 1989.

3 Eugene L. Allgower and Kurt Georg. *Numerical continuation methods: an introduction*, volume 13. Springer Science & Business Media, 1990.

4 Dominique Attali and André Lieutier. Geometry-driven collapses for converting a Čech complex into a triangulation of a nicely triangulable shape. *Discrete & Computational Geometry*, 54(4):798–825, December 2015. doi:10.1007/s00454-015-9733-7.

5 P. Bendich, S. Mukherjee, and B. Wang. Stratification learning through homology inference. In *2010 AAAI Fall Symposium Series*, 2010.

6 Paul Bendich, David Cohen-Steiner, Herbert Edelsbrunner, John Harer, and Dmitriy Morozov. Inferring local homology from sampled stratified spaces. In *Proceedings of the IEEE Symposium on Foundations of Computer Science*, pages 536–546, 2007.

7 J.-D. Boissonnat, R. Dyer, and A. Ghosh. The Stability of Delaunay Triangulations. *International Journal of Computional Geometry & Applications*, 23(4-5):303–334, 2013. doi:10.1142/S0218195913600078.

8 J-D. Boissonnat, M. Rouxel-Labbé, and M. Wintraecken. Anisotropic triangulations via discrete Riemannian Voronoi diagrams. *SIAM Journal on Computing*, 48(3):1046–1097, 2019. doi:10.1137/17M1152292.

9 Jean-Daniel Boissonnat, Frédéric Chazal, and Mariette Yvinec. *Geometric and Topological Inference*. Cambridge Texts in Applied Mathematics. Cambridge University Press, 2018. doi:10.1017/9781108297806.

10 Jean-Daniel Boissonnat, David Cohen-Steiner, and Gert Vegter. Isotopic implicit surface meshing. *Discrete & Computational Geometry*, 39(1):138–157, March 2008. doi:10.1007/s00454-007-9011-4.

11 Jean-Daniel Boissonnat, Ramsay Dyer, and Arijit Ghosh. Delaunay stability via perturbations. *International Journal of Computational Geometry & Applications*, 24(02):125–152, 2014.

12 Jean-Daniel Boissonnat, Ramsay Dyer, and Arijit Ghosh. Delaunay Triangulation of Manifolds. *Foundations of Computational Mathematics*, 45:38, 2017. doi:10.1007/s10208-017-9344-1.

13 Jean-Daniel Boissonnat, Ramsay Dyer, Arijit Ghosh, André Lieutier, and Mathijs Wintraecken. Local conditions for triangulating submanifolds of Euclidean space. Preprint, July 2019. URL: https://hal.inria.fr/hal-02267620.

14 Jean-Daniel Boissonnat and Arijit Ghosh. Manifold reconstruction using tangential Delaunay complexes. *Discrete & Computational Geometry*, 51(1):221–267, 2014.

15 Jean-Daniel Boissonnat, Siargey Kachanovich, and Mathijs Wintraecken. Triangulating submanifolds: An elementary and quantified version of Whitney's method. Preprint, December 2018. URL: https://hal.inria.fr/hal-01950149.

16 Jean-Daniel Boissonnat, Siargey Kachanovich, and Mathijs Wintraecken. Sampling and Meshing Submanifolds in High Dimension. Preprint, November 2019. URL: https://hal.inria.fr/hal-02386169.

17 Jean-Daniel Boissonnat and Mathijs Wintraecken. The topological correctness of PL-approximations of isomanifolds, 2019. URL: https://hal.archives-ouvertes.fr/hal-02386193.

18 L. E. J. Brouwer. Über Abbildung von Mannigfaltigkeiten. *Mathematische Annalen*, 71(4):598–598, December 1912. doi:10.1007/BF01456812.

19 Yen-Chi Chen. Solution manifold and its statistical applications, 2020. arXiv:2002.05297.

20 S-W. Cheng, T. K. Dey, and E. A. Ramos. Manifold reconstruction from point samples. In *Proc. 16th ACM-SIAM Symp. Discrete Algorithms*, pages 1018–1027, 2005.

21 S.-W. Cheng, T. K. Dey, and J. R. Shewchuk. *Delaunay Mesh Generation*. Computer and information science series. CRC Press, 2013.

22 Aruni Choudhary, Siargey Kachanovich, and Mathijs Wintraecken. Coxeter triangulations have good quality. Preprint, December 2017. URL: https://hal.inria.fr/hal-01667404.

23 Frank H. Clarke. *Optimization and Nonsmooth Analysis*, volume 5 of *Classics in applied mathematics*. SIAM, 1990.

24 Harold S. M. Coxeter. Discrete groups generated by reflections. *Annals of Mathematics*, pages 588–621, 1934.

25 T. K. Dey. *Curve and Surface Reconstruction; Algorithms with Mathematical Analysis*. Cambridge University Press, 2007.

26 Tamal K. Dey and Joshua A. Levine. Delaunay meshing of piecewise smooth complexes without expensive predicates. *Algorithms*, 2(4):1327–1349, 2009. doi:10.3390/a2041327.

27 Tamal K. Dey and Andrew G. Slatton. Localized Delaunay refinement for volumes. *Computer Graphics Forum*, 30(5):1417–1426, 2011. doi:10.1111/j.1467-8659.2011.02016.x.

28 Tamal Krishna Dey and Andrew G. Slatton. Localized Delaunay refinement for piecewise-smooth complexes. In Guilherme Dias da Fonseca, Thomas Lewiner, Luis Mariano Peñaranda, Timothy M. Chan, and Rolf Klein, editors, *Symposium on Computational Geometry 2013, SoCG '13, Rio de Janeiro, Brazil, June 17-20, 2013*, pages 47–56. ACM, 2013. doi:10.1145/2462356.2462376.

29 Akio Doi and Akio Koide. An efficient method of triangulating equi-valued surfaces by using tetrahedral cells. *IEICE TRANSACTIONS on Information and Systems*, E74-D, 1991.

30 J. J. Duistermaat and J. A. C. Kolk. *Multidimensional Real Analysis I: Differentiation*. Number 86 in Cambridge Studies in Advanced Mathematics. Cambridge University Press, 2004.

31 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.

32 B. Curtis Eaves. *A course in triangulations for solving equations with deformations*, volume 234. Lecture Notes in Economics and Mathematical Systems, 1984.

33 Herbert Edelsbrunner and Nimish R. Shah. Triangulating topological spaces. *International Journal of Computational Geometry & Applications*, 7(04):365–378, 1997.

34 Hans Freudenthal. Simplizialzerlegungen von beschrankter flachheit. *Annals of Mathematics*, pages 580–582, 1942.

35 M. W. Hirsch. *Differential Topology*. Springer-Verlag, 1976.

36 Harold W. Kuhn. Some combinatorial lemmas in topology. *IBM Journal of research and development*, 4(5):518–524, 1960.

37 William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *ACM SIGGRAPH Computer Graphics*, volume 21, pages 163–169. ACM, 1987.

38 J. Milnor. *Morse Theory*. Princeton University Press, 1969.

39 John Milnor. *Lectures on the H-Cobordism Theorem*. Princeton University Press, 1965. URL: http://www.jstor.org/stable/j.ctt183psc9.

40 Chohong Min. Simplicial isosurfacing in arbitrary dimension and codimension. *Journal of Computational Physics*, 190(1):295–310, 2003.

41 Timothy S. Newman and Hong Yi. A survey of the marching cubes algorithm. *Computers & Graphics*, 30(5):854–879, 2006. doi:10.1016/j.cag.2006.07.021.

42 Timothy S Newman and Hong Yi. A survey of the marching cubes algorithm. *Computers & Graphics*, 30(5):854–879, 2006.

43 Steve Oudot, Laurent Rineau, and Mariette Yvinec. Meshing Volumes Bounded by Smooth Surfaces. *Computational Geometry, Theory and Applications*, 38:100–110, 2007. URL: https://hal.inria.fr/inria-00070382.

44 Simon Plantinga and Gert Vegter. Isotopic approximation of implicit curves and surfaces. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 245–254. ACM, 2004.

45 Laurent Rineau. *Meshing Volumes bounded by Piecewise Smooth Surfaces*. Theses, Université Paris-Diderot - Paris VII, November 2007. URL: `https://tel.archives-ouvertes.fr/tel-00410864`.

46 Mael Rouxel-Labbé, Mathijs Wintraecken, and Jean-Daniel Boissonnat. Discretized Riemannian Delaunay Triangulations. Research Report RR-9103, INRIA Sophia Antipolis - Méditerranée, October 2017. URL: `https://hal.inria.fr/hal-01612924`.

47 Jennifer Schultens. *Introduction to 3-manifolds*, volume 151. American Mathematical Society, 2014.

48 Jonathan Richard Shewchuk. Lecture notes on Delaunay mesh generation, 1999.

49 Michael J. Todd. *The computation of fixed points and applications*, volume 124. Lecture Notes in Economics and Mathematical Systems, 1976.

50 H. Whitney. *Geometric Integration Theory*. Princeton University Press, 1957.

Minimum Bounded Chains and Minimum Homologous Chains in Embedded Simplicial Complexes

Glencora Borradaile
Oregon State University, Corvallis, OR, USA
glencora@eecs.oregonstate.edu

William Maxwell
Oregon State University, Corvallis, OR, USA
maxwellw@oregonstate.edu

Amir Nayyeri
Oregon State University, Corvallis, OR, USA
nayyeria@oregonstate.edu

Abstract

We study two optimization problems on simplicial complexes with homology over $\mathbb{Z}_2$, the minimum bounded chain problem: given a d-dimensional complex $\mathcal{K}$ embedded in $\mathbb{R}^{d+1}$ and a null-homologous $(d-1)$-cycle C in $\mathcal{K}$, find the minimum d-chain with boundary C, and the minimum homologous chain problem: given a $(d+1)$-manifold $\mathcal{M}$ and a d-chain D in $\mathcal{M}$, find the minimum d-chain homologous to D. We show strong hardness results for both problems even for small values of d; $d = 2$ for the former problem, and $d = 1$ for the latter problem. We show that both problems are APX-hard, and hard to approximate within any constant factor assuming the unique games conjecture. On the positive side, we show that both problems are fixed-parameter tractable with respect to the size of the optimal solution. Moreover, we provide an $O(\sqrt{\log \beta_d})$-approximation algorithm for the minimum bounded chain problem where β_d is the dth Betti number of $\mathcal{K}$. Finally, we provide an $O(\sqrt{\log n_{d+1}})$-approximation algorithm for the minimum homologous chain problem where n_{d+1} is the number of $(d+1)$-simplices in $\mathcal{M}$.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases computational topology, algorithmic complexity, simplicial complexes

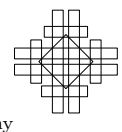
Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.21

Related Version A full version of this paper is available at `https://arxiv.org/abs/2003.02801`.

Funding This material is based upon work supported by the National Science Foundation under Grant Nos. CCF-1617951 and CCF-1816442.

1 Introduction

Simplicial complexes are best known as a generalization of graphs, but have more structure than other generalizations such as hypergraphs. Despite the structure, simplicial complexes are sufficiently expressive to make many algorithmic questions computationally intractable. For example, the generalization of shortest path that we examine in this work is NP-hard in 2-dimensional simplicial complexes [16]. Since planar graphs (1-dimensional simplicial complexes embeddable in $\mathbb{R}^2$) exhibit structure that is algorithmically useful, resulting in more efficient or more accurate algorithms than for general graphs, we ask whether 2-dimensional simplicial complexes that are embeddable in $\mathbb{R}^3$ (and more generally, d-complexes emdeddable in $\mathbb{R}^{d+1}$) also have sufficient structure that can be exploited algorithmically. To this end, we examine the algebraic generalization of the shortest path problem in graphs to simplicial




complexes of higher dimension. This restriction via embedding in Euclidean space would still result in a useful algorithmic tool, given the connection of embedded simplicial complexes to meshes arising from physical systems.

Formally we study the *minimum bounded chain problem* which is the algebraic generalization of the shortest path problem in graphs [23]. The goal of the minimum bounded chain problem is to find a subcomplex whose boundary is a given input cycle C. More precisely: Given a d-dimensional simplicial complex $\mathcal{K}$ and a null-homologous $(d-1)$-dimensional cycle $C \subset \mathcal{K}$, find a minimum-cost d-chain $D \subset \mathcal{K}$ whose boundary $\partial D = C$. The requirement that the cycle be null-homologous is necessary and sufficient for the existence of a solution and we study the problem in the context of $\mathbb{Z}_2$-homology.[1] In $\mathbb{Z}_2$-homology, a d-chain is a subset of d-simplices of the simplicial complex. We see this as a generalization of the shortest path problem in graphs as follows: Let $\mathcal{K}$ be a one dimensional simplicial complex (i.e. a graph). A pair of vertices in the same connected component, s and t, is a null-homologous 0-chain and the minimum 1-chain whose boundary is $\{s, t\}$ is the shortest (s, t)-path. Grady has written on why this generalization is useful in the context of 3D graphics [18].

The minimum bounded chain problem is closely related to the minimum homologous chain problem which asks: given a d-chain D, find a minimum-cost d-chain X such that the symmetric difference of D and X form the boundary of a $(d+1)$-chain. Alternatively, X is the minimum-cost d-chain that is homologous to D. Dunfield and Hirani [16] show the minimum bounded and homologous chain problems are equivalent under additional assumptions. We study the minimum homologous chain problem for d-chains in $(d+1)$-manifolds.

1.1 Our results

We present approximation and fixed-parameter tractable algorithms for the minimum bounded chain and the minimum homologous chain problem. In this paper we consider both problems in the context of simplicial homology over $\mathbb{Z}_2$. We denote by n_d the number of d-simplices of the d-dimensional simplicial complex $\mathcal{K}$. Two of our results assume the unique games conjecture. For an overview of the unique games conjecture and its impact on computational topology we refer the reader to the work of Growchow and Tucker-Foltz [19].

▶ **Theorem 1.** *There exists an $O(\sqrt{\log \beta_d})$-approximation algorithm for the minimum bounded chain problem for a simplicial complex $\mathcal{K}$ embedded in $\mathbb{R}^{d+1}$, with dth Betti number β_d.*

▶ **Theorem 2.** *There exists an $O(15^k \cdot k \cdot n_d^3)$ time exact algorithm for the minimum bounded chain problem for simplicial complexes embedded in $\mathbb{R}^{d+1}$, where k is the number of d-simplices in the optimal solution.*

▶ **Theorem 3.** *There exists an $O(\sqrt{\log n_{d+1}})$-approximation algorithm for the minimum homologous chain problem for d-chains in $(d+1)$-manifolds.*

▶ **Theorem 4.** *There exists an $O(15^{k \cdot k \cdot n_d^3})$ time exact algorithm for the minimum homologous chain problem for d-chains in $(d+1)$-manifolds, where k is the size of the optimal solution.*

The running times for the first two theorems is computed assuming that the dual graph of the complex in $\mathbb{R}^{d+1}$ is available. The last two theorems hold, more generally, for weak pseudomanifolds studied by Dey et al. in [14].

On the hardness side, we show that constant factor approximation algorithms for these problems (minimum bounded chain and minimum homologous chain) are unlikely.

[1] Formal definitions are presented in Section 2.

▶ **Theorem 5.** *The minimum bounded chain problem is*
(i) *hard to approximate within a* $(1+\varepsilon)$ *factor for some* $\varepsilon > 0$ *assuming* $P \neq NP$*, and*
(ii) *hard to approximate within any constant factor assuming the unique games conjecture, even if* $\mathcal{K}$ *is a* 2*-dimensional simplicial complex embedded in* $\mathbb{R}^3$ *with input cycle* C *embedded on the boundary of the unbounded volume in* $\mathbb{R}^3 \setminus \mathcal{K}$*.*

▶ **Theorem 6.** *The minimum homologous chain problem is*
(i) *hard to approximate within a* $(1+\varepsilon)$ *factor for some* $\varepsilon > 0$ *assuming* $P \neq NP$*, and*
(ii) *hard to approximate within any constant factor assuming the unique games conjecture, even when the input chain is a* 1*-cycle on an orientable* 2*-manifold.*

1.2 Related Work

1.2.1 Chain problems over $\mathbb{Z}$ and $\mathbb{R}$

Research on the minimum bounded chain problem is limited to the case of $\mathbb{Z}$-homology, where linear programming techniques can be employed algorithmically. Sullivan described the problem as the discretization of the minimal spanning surface problem [28] with Kirsanov reducing the problem to an instance of minimum cut in the dual graph [23]. Sullivan's work is on the closely related cellular complexes, but under the same restrictions we study (embedded in $\mathbb{R}^d$) and Kirsanov studies the problem in embedded simplicial complexes.

Likewise, research on minimum homologous chain has largely worked in $\mathbb{Z}$-homology. Dey, Hirani and Krishnamoorthy formulate the minimum homologous chain problem over $\mathbb{Z}$ as an integer linear program and describe topological conditions for the linear program to be totally unimodular (and so, poly-time solvable) [13]. Of course, integer linear programming approaches do not extend to $\mathbb{Z}_2$-homology.

This linear programming approach was then applied to the minimum bounded chain problem (over $\mathbb{Z}$) by Dunfield and Hirani [16]. Moreover, they show the minimum bounded chain problem is NP-complete via a reduction from 1-in-3 SAT. The gadget they use was originally used by Agol, Hass and Thurston to show that the minimal spanning area problem is NP-complete [2].

Linear programming techniques have also been used by Chambers and Vejdemo-Johansson to solve the minimum bounded chain problem in the context of $\mathbb{R}$-homology [9]. In $\mathbb{R}$-homology Carvalho et al provide an algorithm finding a (not necessarily minimum) bounded chain in a manifold by searching the dual graph of the manifold [7].

1.2.2 Chain problems over $\mathbb{Z}_2$

Special cases of the minimum homologous chain problem have been studied in $\mathbb{Z}_2$ homology. The homology localization problem is the case when the input chain is a cycle. The homology localization problem over $\mathbb{Z}_2$ in surface-embedded graphs is known to be NP-hard via a reduction from maximum cut by Chambers et al. [8]; our reduction is from the complement problem minimum uncut. On the algorithmic side, Erickson and Nayyeri provide a $2^{O(g)}n \log n$ time algorithm where g is the genus of the surface [17]. Using the idea of annotated simplices, Busaryev et al. generalize this algorithm for homology localization of 1-cycles in simplicial complexes; the algorithm runs in $O(n^\omega) + 2^{O(g)}n^2 \log n$ time where ω is the exponent of matrix multiplication, and g is the first homology rank of the complex [6].

Using a reduction from the nearest codeword problem Chen and Freedman showed that homology localization with coefficients over $\mathbb{Z}_2$ is not only NP-hard, but it cannot be approximated within any constant factor in polynomial time [11]. These hardness results

hold for a 2-dimensional simplicial complex, but not necessarily for 2-dimensional complexes embedded in $\mathbb{R}^3$. They also give a polynomial-time algorithm for the special case of d-dimensional simplicial complex that is embedded in $\mathbb{R}^d$. (This is different from our setting of a d-dimensional simplicial complex that is embedded in $\mathbb{R}^{d+1}$; however the algorithm also reduces to a minimum cut problem in a dual graph, much like that of Kirsanov and Gortler.)

1.2.3 Algebraic formulations

The minimum bounded chain problem over $\mathbb{Z}_2$ can be stated as a linear algebra problem, but this has little algorithmic use since the resulting problems are intractable. The algebraic formulation is to find a vector x of minimum Hamming weight that solves an appropriately defined linear system $Ax = b$. (It is possible to reduce in the reverse direction, but the resulting complex is not embeddable in general, and so provides no new results.)

In coding theory this algebraic problem is a well studied decoding problem known as maximum likelihood decoding, and it was shown to be NP-hard by Berlekamp, McEliece and van Tilborg [4, 29]. Downey, Fellows, Vardy and Whittle show that maximum likelihood decoding is W[1]-hard [15]. Further, Austrin and Khot show that maximum likelihood decoding is hard to approximate within a factor of $2^{(\log n)^{1-\epsilon}}$ under the assumption that $\mathsf{NP} \not\subseteq \mathsf{DTIME}(2^{(\log n)^{O(1)}})$ [3]. This work was continued by Bhattacharyya, Gadekar, Ghosal and Saket who showed that maximum likelihood decoding is still W[1]-hard when the problem is restricted to $O(k \log n) \times O(k \log n)$ sized matrices for some constant k [5].

1.2.4 Paper organization

In Section 2, we give formal definitions for the paper. In Section 3, we present our approximation algorithms and fixed-parameter tractable algorithms. In Section 4, we present our hardness results.

2 Preliminaries

2.1 Simplicial complexes

Given a set of vertices V we define an *abstract simplicial complex* $\mathcal{K}$ to be a subset of the power set of V such that the following property holds: if $\sigma \in \mathcal{K}$ and $\tau \subset \sigma$ then $\tau \in \mathcal{K}$. We call any $\sigma \in \mathcal{K}$ a *simplex* and define the dimension of σ to be $|\sigma| - 1$ if $|\sigma| - 1 = d$ we call σ a d-simplex. Further, we call 0-simplices, 1-simplices, and 2-simplices *vertices*, *edges*, and *triangles*. We define the dimension of $\mathcal{K}$ to be equal to the largest dimension of any simplex in $\mathcal{K}$. If $\mathcal{K}$ has dimension d we refer to $\mathcal{K}$ as a d-simplicial complex or d-complex. We refer to any subset of a d-simplex σ as a *face* of σ.

2.2 Homology

In this paper we work in simplicial homology with coefficients over the finite field $\mathbb{Z}_2$. Here we briefly define the concepts from homology that will be used throughout this paper. We assume familiarity with the basics of algebraic topology, and refer the reader to standard references [20, 25] for the details.

Given a simplicial complex $\mathcal{K}$ we define the dth *chain group* of $\mathcal{K}$ to be the free abelian group, with coefficients over $\mathbb{Z}_2$, generated by the d-simplices in $\mathcal{K}$. We denote the chain group as $C_d(\mathcal{K})$ and note that its elements are expressed as formal sums $\bigoplus \alpha_i \sigma_i$ where

$\alpha_i \in \mathbb{Z}_2$ and $\sigma_i \in \mathcal{K}$ is a d-simplex. We call the elements of the chain group *chains* or more specifically d-chains. When working over $\mathbb{Z}_2$ there is a one-to-one correspondence between d-chains and sets of d-simplices in $\mathcal{K}$. It follows that adding two d-chains over $\mathbb{Z}_2$ is the same thing as taking the symmetric difference of their corresponding sets. Hence, we use the notation $\sigma \oplus \tau$ to denote the sum of two d-chains. By abuse of notation we will also use $\oplus$ to denote the symmetric difference of sets, but the context should always be clear.

For a d-simplex σ we define its *boundary* $\partial\sigma$ to be the sum of the $(d-1)$-simplices contained in σ. We extend this operation linearly to obtain the *boundary operator* on chain groups, $\partial_d \colon C_d(\mathcal{K}) \to C_{d-1}(\mathcal{K})$. We will often drop the subscript when the context is clear. Note that the composition $\partial_{d-1}\partial_d$ is always equal to the zero map. If $\partial\sigma = \tau$ we say that σ is bounded by τ. We call a chain σ a *cycle* if $\partial\sigma = 0$.

By $Z_d(\mathcal{K})$ we denote the dth *cycle group* of $\mathcal{K}$. This is subgroup of $C_p(\mathcal{K})$ generated by the d-simplices in $\ker \partial_d$. Similarly, by $B_d(\mathcal{K})$ we denote the dth *boundary group* of $\mathcal{K}$, which is the subgroup of $C_p(\mathcal{K})$ generated by the d-simplices in $\operatorname{im} \partial_{d+1}$. Since $\partial_{d+1}\partial_d = 0$ we have that $B_d(\mathcal{K})$ is a subgroup of $Z_d(\mathcal{K})$. We define the dth *homology group* of $\mathcal{K}$, denoted $H_d(\mathcal{K})$, to be the quotient group $Z_d(\mathcal{K})/B_d(\mathcal{K})$. The dth Betti number of $\mathcal{K}$, denoted β_d, is defined to be the dimension of $H_d(\mathcal{K})$. We call a d-chain σ null-homologous if it is a boundary, that is $\sigma \in B_d(\mathcal{K})$. Further, we call two d-chains σ and τ *homologous* if their difference is a boundary, that is $\sigma \oplus \tau \in B_d(\mathcal{K})$.

2.3 Embeddings and duality

Given a d-complex $\mathcal{K}$ an embedding of $\mathcal{K}$ is a function $f \colon \mathcal{K} \to \mathbb{R}^{d+1}$ such that f restricted to any simplex in $\mathcal{K}$ is an injection. Further, for any two simplices $\sigma, \tau \in \mathcal{K}$ we require that $f(\sigma) \cap f(\tau) = f(\sigma \cap \tau)$. That is, the images of two simplices only intersect at their common faces. The function f is an *embedding* of the abstract simplicial complex $\mathcal{K}$. In this paper we make no distinction between $\mathcal{K}$ and an embedding of $\mathcal{K}$. Hence, we use the notation $\mathcal{K}$ to refer to both and refer to $\mathcal{K}$ as an *embedded simplicial complex*.

The Alexander duality theorem, a higher dimensional analog of the Jordan curve theorem, states that $\mathbb{R}^{d+1} \setminus \mathcal{K}$ is partitioned into $\beta_d + 1$ connected components. Exactly one of these connected components is unbounded, and we refer to the unbounded component as V_∞. Using this partition we define the dual graph $\mathcal{K}^*$ of $\mathcal{K}$. $\mathcal{K}^*$ has one vertex for each connected component of $\mathbb{R}^{d+1} \setminus \mathcal{K}$ with the vertex corresponding to V_∞ denoted by v_∞. Further, $\mathcal{K}^*$ has one edge for each d-simplex in $\mathcal{K}$. There is an edge between two vertices representing connected components V_1 and V_2 in $\mathcal{K}$ if there is a d-simplex contained in the intersection of the topological closures of V_1 and V_2. Note that $\mathcal{K}^*$ can have parallel edges and self-loops. Since each d-simplex can be in the closure of at most two connected components we have a one-to-one correspondence between d-simplices in $\mathcal{K}$ and edges in $\mathcal{K}^*$. If S is a set of d-simplices in $\mathcal{K}$ we denote their corresponding edges in $\mathcal{K}^*$ by S^*. Similar to planar graphs, there is a duality between d-cycles in $\mathcal{K}$ and edge cuts in $\mathcal{K}^*$. There exists a one-to-one correspondence between d-cycles in $\mathcal{K}$ and minimal edge cuts in $\mathcal{K}^*$. We refer to this correspondence as cycle/cut duality, and it will play a central role in many of our proofs.

By $\mathsf{shell}(\mathcal{K})$ we denote the *outer shell* of $\mathcal{K}$. This is defined to be the subcomplex of $\mathcal{K}$ consisting of all d-simplices whose corresponding edges in $\mathcal{K}^*$ are incident to v_∞. Equivalently, it is also the subcomplex of $\mathcal{K}$ consisting of all d-simplices contained in the boundary of V_∞.

We endow the embedding of a simplicial complex $\mathcal{K}$ with the subspace topology inherited from $\mathbb{R}^{d+1}$. We call $\mathcal{K}$ a d-dimensional *manifold* if every point in its embedding is contained in a neighborhood homeomorphic to $\mathbb{R}^d$. If every point in the embedding of $\mathcal{K}$ is contained in a neighborhood homeomorphic to either $\mathbb{R}^d$ or the d-dimensional half-space we call $\mathcal{K}$ a *manifold with boundary*.

2.4 Graph cuts

Let $G = (V, E)$ be a graph. For any two subsets $V_1, V_2 \subset V$ a (V_1, V_2)-cut is a set of edges E' such that the graph $G' = (V, E \setminus E')$ contains no path from V_1 to V_2. Often we will consider $(S, \overline{S})$-cuts for some $S \subset V$ where $\overline{S}$ denotes the complement of S in V. By E_S we refer to the edge set corresponding to all edges that have one endpoint in S and the other in $\overline{S}$, which is the minimum $(S, \overline{S})$-cut. We extend this notation to vertices. For any two vertices $s, t \in V$ an (s, t)-cut refers to a set of edges whose removal disconnects s from t.

2.5 The minimum bounded/homologous chain problems

Now we give the formal statement of the minimum bounded chain problem. Given a d-dimensional simplicial complex $\mathcal{K}$ and a $(d-1)$-cycle C contained in $\mathcal{K}$ the *minimum bounded chain* problem $(\mathcal{K}, C)$ asks to find a d-chain X with $\partial X = C$ such that the cost of X is minimized. The cost of X is given by its ℓ_1 norm $\|X\|_1$. Here we are treating X as an n-dimensional indicator vector where n is the number of d-simplices in $\mathcal{K}$. The simplicial complex $\mathcal{K}$ may be weighted by assigning a real number to each d-simplex in $\mathcal{K}$. In this case the cost of X is given by $\langle W, X \rangle$, where W is a vector assigning weights to the d-simplices of $\mathcal{K}$.

Now let D be a d-chain, which may or may not be a cycle. The minimum homologous chain problem asks to find a minimum d-chain X such that $X = D \oplus \partial V$ for some $(d+1)$-chain V, equivalently, the minimum d-chain X such that $D \oplus X$ is null-homologous. The cost of X as well as the weighted problem are defined the same as in the previous paragraph.

In this paper, we study the minimum bounded chain problem for complexes embedded in $\mathbb{R}^{d+1}$, and the minimum homologous chain problem for d-chains in $(d+1)$-manifolds.

3 Approximation algorithm and fixed-parameter tractability

In this section, we describe approximation algorithms and parameterized algorithms for both minimum bounded chain and minimum homologous chain problems. Our algorithms work with the dual graph of the input space. In order to simplify our presentation we assume that the dual graph of the input complex contains no loops. The following lemma shows that we can make this assumption without any loss of generality. The proof can be found in the full version of the paper.

▶ **Lemma 7.** *In polynomial time we can preprocess an instance of the minimum bounded chain problem $(\mathcal{K}, C)$ into a new instance $(\mathcal{K}', C')$ such that (i) $(\mathcal{K}')^*$ contains no loops and (ii) an α-approximation algorithm for $(\mathcal{K}', C')$ implies an α-approximation algorithm for $(\mathcal{K}, C)$.*

3.1 Reductions to the minimum cut completion problem

Given $G = (V, E)$ and $E' \subseteq E$, the *minimum cut completion* problem asks for a cut $(S, \overline{S})$ with edge set E_S that minimizes $|E_S \oplus E'|$. First, we show that the minimum cut completion problem generalizes the minimum bounded chain problem.

▶ **Lemma 8.** *For any d-dimensional instance of the minimum bounded chain problem, $(\mathcal{K}, C)$, there exists an instance of the minimum cut completion problem $(G = (V, E), E')$ that can be computed in polynomial time, and a one-to-one correspondence between cuts in G and d-chains with boundary C in $\mathcal{K}$. Moreover, if the cut $(S, \overline{S})$ with edge set E_S in G corresponds to the d-chain Q in $\mathcal{K}$ then $|E_S \oplus E'| = |Q|$.*

Proof. Let F be any d-chain such that $\partial F = C$, such an F can be computed in polynomial time, by solving the linear system. In turn, let $G = \mathcal{K}^*$, and $E' = F^*$.

Now, let Q be any d-chain such that $\partial Q = C$. So, $\partial(Q \oplus F) = 0$. Thus, by cycle/cut duality $Q \oplus F$ partitions $\mathbb{R}^{d+1}$, let $(S, \overline{S})$ be the corresponding dual cut in $\mathcal{K}^*$, and let E_S be the edge set of this cut. We have $|E_S \oplus E'| = |E_S \oplus F^*| = |Q^*| = |Q|$.

On the other hand, let $(S, \overline{S})$ be a cut in $\mathcal{K}^*$, with edge set E_S. By cycle/cut duality $\partial E_S^* = 0$. Now, let $Q = E_S^* \oplus F$. It follows that $\partial Q = C$. Moreover, we have $|Q| = |E_S^* \oplus F| = |E_S \oplus F^*| = |E_S^* \oplus E'|$. ◀

We show via a similar argument that the cut completion problem also generalizes the minimum homologous chain problem when the input complex is a *weak pseudomanifold* (see the full version of the paper for the proof). A weak pseudomanifold is a pure d-complex such that every $(d-1)$-simplex is a face of at most two d-simplices. Weak pseudomanifolds generalize manifolds and the definition was first introduced by Dey et al. in [14]. Although recognizing d-manifolds is undecidable [12], weak pseudomanifolds can be recognized in polynomial time.

▶ **Lemma 9.** *For any d-dimensional instance of the minimum homologous chain problem $(\mathcal{M}, D)$, where $\mathcal{M}$ is a weak pseudomanifold, there exists an instance of the minimum cut completion problem $(G = (V, E), E')$ that can be computed in polynomial time, and a one-to-one correspondence between cuts in G and d-chains in $\mathcal{M}$ that are homologous to D. Moreover, if the cut $(S, \overline{S})$ with edge set E_S in G corresponds to the d-chain Q in $\mathcal{K}$ then $|E_S \oplus E'| = |Q|$.*

3.2 Algorithms for the minimum cut completion problem

We show an $O(\sqrt{\log |V|})$-approximation algorithm and a fixed-parameter tractable algorithm for the cut completion problem. We obtain both of these results via reduction to *2CNF Deletion*: given an instance of 2SAT, find the minimum number of clauses to delete to make the instance satisfiable. Agarwal et al. [1] show an $O(\sqrt{\log n})$-approximation algorithm for 2CNF Deletion, where n is the number of clauses, and Razgon and O'Sullivan show that the problem is fixed-parameter tractable.

▶ **Lemma 10** (Agarwal et al.[1], Theorem 3.1)**.** *There is a randomized polynomial-time algorithm for finding an $O(\sqrt{\log n})$-approximation for the minimum disagreement 2CNF Deletion problem.*

▶ **Lemma 11** (Razgon and O'Sullivan [27], Theorem 7)**.** *Let B be an instance of 2CNF Deletion problem with m clauses that admits a solution of size k. There is an $O(15^k \cdot k \cdot m^3)$ time exact algorithm for solving B.*

The next lemma shows similar results for the cut completion problem.

▶ **Lemma 12.** *For the cut completion problem $(G = (V, E), E')$,*

- **(i)** *there is a randomized polynomial-time $O(\sqrt{\log |V|})$-approximation algorithm, and*
- **(ii)** *there is an $O(15^k \cdot k \cdot |E|^3)$ time exact algorithm, where k is the size of the optimal solution.*

Proof. Let $G = (V, E)$, and $E' \subseteq E$. We show a 2CNF Deletion instance B_G such that for any cut $(S, \overline{S})$ with edge set E_S, the number of unsatisfied clauses in B_G is exactly $|E_S \oplus E'|$. The statement of the lemma will follow from Lemma 10 and 11.

Let B_G be the instance of the 2CNF Deletion problem defined on G as follows:

- For each vertex $v \in V$, we have variable $b(v)$.
- For each edge $(u, v) \in E$:
  - if $(u, v) \in E'$, we add $b(u) \vee b(v)$ and $\neg b(u) \vee \neg b(v)$ to B, and
  - if $(u, v) \notin E'$, we add $b(u) \vee \neg b(v)$ and $\neg b(u) \vee b(v)$ to B.

  (Note that in both cases, any assignment of $b(u)$ and $b(v)$ satisfies at least one of the clauses. Again in both cases, assignments exist that satisfy both clauses.)

Let $(S, \overline{S})$ be a cut with edge set E_S. Let b_S be the natural boolean vector that corresponds to the cut: $b(v) = [v \in S]$ for all $v \in V$. We show that $|E_S \oplus E'|$ is equal to the number of clauses that are not satisfied in B_G. Specifically, we show (I) for each edge $(u, v) \in E_S \oplus E'$, exactly one of its corresponding clauses is satisfied, and (II) for each edge $(u, v) \notin E_S \oplus E'$ both of its corresponding clauses are satisfied.

If $(u, v) \in E_S \oplus F$ there are two cases to consider: (I.1) $(u, v) \in E_S$ and $(u, v) \notin E'$, that is $b(u) \neq b(v)$ and the corresponding clauses are $b(u) \vee \neg b(v)$ and $\neg b(u) \vee b(v)$. Exactly one of the clauses is satisfied. (I.2) $(u, v) \notin E_S$ and $(u, v) \in E'$, that is $b(u) = b(v)$, and the corresponding clauses are $b(u) \vee b(v)$ and $\neg b(u) \vee \neg b(v)$; exactly one of the clauses is satisfied.

If $(u, v) \notin E_S \oplus E'$ there are two cases to consider: (II.1) $(u, v) \in E_S$ and $(u, v) \in E'$, that is $b(u) \neq b(v)$ and the corresponding clauses are $b(u) \vee b(v)$ and $\neg b(u) \vee \neg b(v)$. Both of the clauses are satisfied. (II.2) $(u, v) \notin E_S$ and $(u, v) \notin E'$, that is $b(u) = b(v)$, and the corresponding clauses are $b(u) \vee \neg b(v)$ and $\neg b(u) \vee b(v)$. Both of the clauses are satisfied. ◀

3.3 Wrap up (Proofs of Theorems 1, 2, 3, and 4)

Lemma 8 and Lemma 9 show that the bounded chain problem and the minimum homologous chain problem are special cases of the cut completion problem, and Lemma 12 shows that we obtain $O(\sqrt{\log |V|})$-approximation algorithm and $O(15^k \cdot k \cdot |E|^3)$ time exact algorithm for the cut completion problem. The number of vertices $|V|$ translates to β_d for simplicial complexes embedded in $\mathbb{R}^{d+1}$ (Theorem 1), and n_{d+1}, the number of $(d+1)$-dimensional simplices for $(d+1)$-manifolds (Theorem 3). The number of edges $|E|$ translates to n_d in both simplicial complexes embedded in $\mathbb{R}^{d+1}$ (Theorem 2) $(d+1)$-manifolds (Theorem 4).

4 Hardness of approximation

In this section, we show it is unlikely that either of the minimum bounded chain or minimum homologous chain problems admit constant factor approximation algorithms, even for their low dimensional instances. Our hardness results follow from reductions from the minimum cut completion problem, defined in the previous section.

4.1 Minimum bounded chain to minimum cut completion

We show that the minimum cut completion problem reduces to a 2-dimensional instance of the minimum bounded chain problem $(\mathcal{K}, C)$, where $\mathsf{shell}(\mathcal{K})$ is in fact a manifold and C is a (possibly not connected) cycle on $\mathsf{shell}(\mathcal{K})$. Our hardness of approximation result for the minimum bounded chain problem is based on this reduction.

▶ **Lemma 13.** *Let $(G = (V, E), E')$ be any instance of the minimum cut completion problem. There exists an instance of the 2-dimensional minimum bounded chain problem $(\mathcal{K}, C)$ with C on the outer shell of $\mathcal{K}$ that can be computed in polynomial time, and a one-to-one correspondence between cuts in G and 2-chains with boundary C in $\mathcal{K}$. Moreover, if the cut $(S, \overline{S})$ with edge set E_S in G corresponds to the 2-chain Q in $\mathcal{K}$ then*

$$\frac{|Q|}{\tau} - 1 \leq |E_S \oplus E'| \leq \frac{|Q|}{\tau},$$

where $\tau = 58m + 2$ *and* m *is the number of edges in* G.

Proof. Our construction is simple in high-level. We start from any embedding of G in $\mathbb{R}^3$, and we thicken it to obtain a space, in which each edge corresponds to a tube. We insert a disk in the middle of each tube; we call these disks *edge disks*. Then we triangulate all of the 2-dimensional pieces. The dual of the complex that we build is almost G, except for one extra vertex corresponding to its outer volume, and a set of extra edges, all incident to the extra vertex. We give our detailed construction below.

We consider the following piecewise linear embedding of G in $\mathbb{R}^3$; let n and m be the number of vertices and edges of G, respectively. First, map the vertices of G into $\{1, 2, \ldots, n\}$ on the x-axis. Now, consider $m + 2$ planes $h_0, h_1, \ldots, h_{m+1}$ all containing the x-axis with normals being evenly spaced vectors ranging from $(0, 1, 1)$ to $(0, 1, -1)$. We use $h_1, \ldots, h_m$ for drawing the edges G. We arbitrarily assign edges of G to these plane, so each plane will contain exactly one edge. Each edge is drawn on its plane as a three-segment curve; the first and the last segment are orthogonal to x-axis and the middle one is parallel. All edges are drawn in the upper half-space of $\mathbb{R}^3$. See Figure 1, left.

Next, we place an axis parallel cube around each vertex. The size of the cubes must be so that they do not intersect, fix the width of each cube to be $1/10$. We refer to these cubes as *vertex cubes*. Then, we replace the part of each edge outside the cubes with a cubical tube, called *edge tube*. We choose the thickness of these tubes sufficiently small so that they are disjoint. We also puncture the cubes so that the union of all vertex cubes and edges tubes form a surface; see Figure 1, left. (This surface will have genus $m - n + 1$ by Euler's formula, which is the dimension of the cycle space of G)

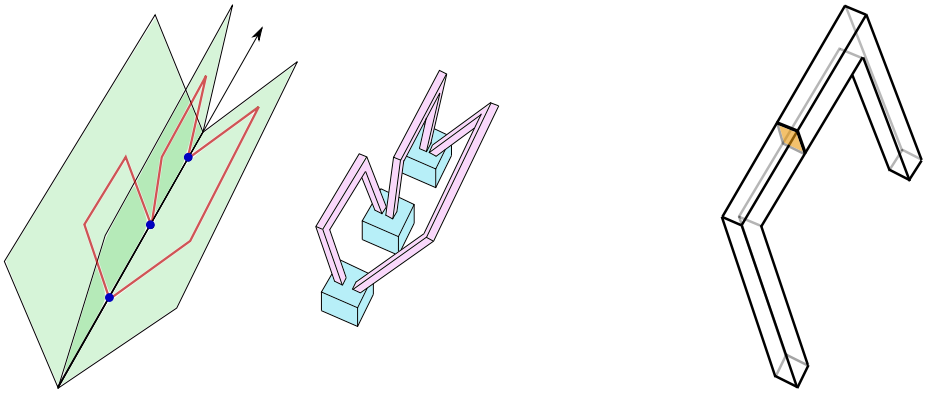


Figure 1 Left: an embedding of K_3 in $\mathbb{R}^3$, and the thickened surface composed of blue vertex cubes and pink edge tubes, right: an edge tube subdivided by an edge square.

Next, we subdivide each tube by placing a square in its middle; see Figure 1, right. We refer to these squares as *edge squares*. Edge squares partition the inside of the surface into n volumes. We observe that each of these volumes contains exactly one vertex of the drawing of G, thus, we call them *vertex volumes*.

For our reduction to work, we need that the weight of each 2-cycle to be dominated by the weight of its edge squares. To achieve that we finely triangulate each edge square. For an edge tube, we first subdivide its surface to 16 quadrangles as shown in Figure 2, left. Then, we obtain a triangulation with 32 triangles by splitting each quadrangle into two triangles.

For a vertex cube, note that all the punctures are on the top face by our construction. We split all the other faces by dividing each of them into two triangles. For the top face, we can obtain a triangulation in polynomial time; this triangulation will have $4\deg(v)+8$ triangles by Euler's formula, where $\deg(v)$ is the degree of the vertex corresponding to the cube. Therefore, the triangulation of each vertex cube will have $4\deg(v)+18$ triangles, see Figure 2, right. Therefore, there are $\left(\sum_{v\in V} 4\deg(v)+18\right)+32m \leq 58m$ triangles that are not part of edge squares. Finally, we triangulate each edge square into $58m+2$ triangles so that the cost of one edge square is greater than the sum of all triangles not contained in edge squares. This triangulation can be done by starting with a square made up of two triangles and repeatedly subdividing triangles by inserting a new vertex in the interior and connecting it to the corners with edges. The subdivision is performed by inserting a vertex into the interior of the triangle and connecting it with an edge to each vertex on the boundary of the triangle. The result is a new complex, homeomorphic to the original, with two additional triangles. Overall, our complex $\mathcal{K}$ has $O(m^2)$ triangles.

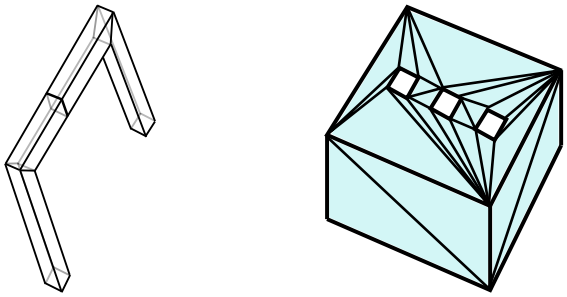


Figure 2 Left: subdividing the surface of an edge-tube to quadrangles, right: triangulating the surface of a vertex cube.

We are now done with the construction of $\mathcal{K}$. Let B be the set of all triangles in edge squares that correspond to edges in E'. Then, let $C=\partial B$. We show an almost cost preserving one-to-one correspondence between cuts in the cut completion problem in G and chains with boundary C in $\mathcal{K}$.

Let $(S,\overline{S})$ be a cut with edge set E_S, note that the cost of this cut is $|E_S \oplus E'|$ in the cut completion problem (G,E'). In $\mathcal{K}$, let $\mathcal{V}_S$ be the symmetric difference of the vertex volumes that correspond to vertices of S. The total weight of $\mathcal{V}_S$ is between $|E_S|(58m+2)$ and $|E_S|(58m+2)+58m$. Similarly, the total weight of $\mathcal{V}_S \oplus B$ is between $|E_S \oplus E'|(58m+2)$ and $|E_S \oplus E'|(58m+2)+58m$. Since we cannot get an exact count on the number of edges in the subgraph induced by S we have a range of values for the weight of $\mathcal{V}_S$ instead of an exact weight. However, if E_S and $E_{S'}$ are two cuts with $|E_S|<|E_{S'}|$ then the weight of $\mathcal{V}_S$ is strictly less than the weight of $\mathcal{V}_{S'}$ by the construction of the edge squares.

On the other hand, let Q be a 2-chain with boundary C in $\mathcal{K}$. As C does not intersect the interior of any edge square, for each edge square either Q contains all of its triangles or none of them. Also, $Q \oplus B$ has no boundary, thus its complement $\mathbb{R}^3 \setminus (Q \oplus B)$ is disconnected. The interior of each vertex volume is completely inside one of the connected components of $\mathbb{R}^3 \setminus (Q \oplus B)$, as by the construction $Q \oplus B$ must either contain the entire vertex volume or none of it. Now, let S be the set of all vertices whose corresponding vertex volumes are in the unbounded connected component of $\mathbb{R}^3 \setminus (Q \oplus B)$. The edges of the cut $(S,\overline{S})$ correspond to edge squares in $Q_s \oplus B$, where Q_s is the set of edge square triangles of Q. As B is in one-to-one correspondence to E', it follows that the cut completion cost of $(S,\overline{S})$ is $\frac{|Q_s|}{58m+2}$.

We have $|Q| = |Q_s| + |Q_r|$ where Q_r is the set of triangles in Q not contained in edge squares. The size of $|Q_s|$ is $58m + 2$ per edge square, and $|Q_r| \leq 58m$ by construction. It follows that we have our desired inequality,

$$\frac{Q}{58m+2} - 1 \leq |E_S \oplus E'| \leq \frac{Q}{58m+2}. \quad \blacktriangleleft$$

The next lemma shows that an approximation algorithm for the minimum bounded chain problem implies an approximation algorithm with almost the same quality for the minimum cut completion problem.

▶ **Lemma 14.** *Let $(G = (V, E), E')$ be any instance of the minimum cut completion problem. For any $\alpha \geq 1$ and any $\varepsilon > 0$, there exists an instance of the 2-dimensional minimum bounded chain problem $(\mathcal{K}, C)$ that can be computed in polynomial time, such that an α-approximation algorithm for $(\mathcal{K}, C)$ implies a $((1+\varepsilon)\alpha)$-approximation algorithm for (G, E'), and C is on the outer shell of $\mathcal{K}$.*

Proof. Let $\varepsilon > 0$. Given an α-approximation algorithm for the minimum bounded chain problem, we describe an $((1+\varepsilon)\alpha)$-approximation algorithm for the cut completion problem. Let $G = (V, E)$, and $E' \subseteq E$ be any instance of the cut completion problem, and let $(S_{opt}, \overline{S_{opt}})$ with edge set be an optimal solution for this instance. Our algorithm considers two cases, based on whether $|E_{S_{opt}} \oplus E'| < 1/\varepsilon$ or not. It solves the problem under each assumption and outputs the best solution it obtains in the end.

If $|E_{S_{opt}} \oplus E'| < 1/\varepsilon$, then our algorithm finds the optimal solution in $O(n^{1/\varepsilon + O(1)})$ time by considering all subsets of edges E'' of size at most $1/\varepsilon$ as candidates for $E_{S_{opt}} \oplus E'$. From all candidates, we return the minimum E'' such that $E'' \oplus E'$ is a cut. Note this is an exact algorithm, so in this case we find the optimal solution.

Otherwise, if $|E_{S_{opt}} \oplus E'| \geq 1/\varepsilon$, we use the given α-approximation algorithm for the minimum bounded chain problem for a simplicial complex $\mathcal{K}$, and chain C that corresponds to (G, E') by Lemma 13. Note that $\mathcal{K}$ is an unweighted simplicial complex piecewise linearly embedded in $\mathbb{R}^3$ and C is a cycle in its outer shell.

Let Q_{opt} be the corresponding 2-chain to $(S_{opt}, \overline{S_{opt}})$ in $\mathcal{K}$. Thus, $\frac{Q_{opt}}{\tau} - 1 \leq |E_{S_{opt}} \oplus E'| \leq \frac{|Q_{opt}|}{\tau}$. In addition, let Q be the surface with boundary C that the α-approximation algorithm finds, so $|Q| \leq \alpha \cdot |Q_{opt}|$. Finally, let $(S, \overline{S})$ be the cut corresponding to Q in G via the one-to-one correspondence of Lemma 13. Therefore, $\frac{Q}{\tau} - 1 \leq |E_S \oplus E'| \leq \frac{|Q|}{\tau}$. Putting everything together,

$$|E_S \oplus E'| \leq \frac{|Q|}{\tau} \leq \alpha \cdot \frac{|Q_{opt}|}{\tau} \leq \alpha \cdot \left(|E_{S_{opt}} \oplus E'| + 1\right). \tag{1}$$

Since $|E_{S_{opt}} \oplus E'| \geq 1/\varepsilon$, we have: $|E_{S_{opt}} \oplus E'| + 1 \leq (1+\varepsilon) \cdot |E_{S_{opt}} \oplus E'|$. Therefore, together with (1), we have a $((1+\varepsilon)\alpha)$-approximation algorithm, as desired. $\blacktriangleleft$

4.2 Minimum homologous cycle to minimum cut completion

We show a similar reduction from the cut completion problem to the minimum homologous cycle problem for 1-dimensional cycles on orientable 2-manifolds. The minimum homologous cycle problem is the special case of the minimum homologous chain problem when the input chain is required to be a cycle, so showing hardness of approximation for it implies hardness of approximation for the more general minimum homologous chain problem.

▶ **Lemma 15.** *Let $(G = (V, E), E')$ be any instance of the minimum cut completion problem. For any $\alpha \geq 1$, there exists an instance of the 1-dimensional minimum homologous cycle problem $(\mathcal{M}, D)$ that can be computed in polynomial time such that an α-approximation for $(\mathcal{M}, D)$ implies an α-approximation for (G, E').*

Proof. We construct a 2-manifold $\mathcal{M}$ as in the proof of Lemma 13, but we omit the edge squares. Each edge of G corresponds to a cycle with 4 edges in $\mathcal{M}$; these cycles are the boundaries of the omitted edge squares. We call these cycles *edge rings*. The connected components of $\mathcal{M}$ after removing the edge rings correspond to the vertices of G, we call these connected components *vertex regions*. We set D to be equal to the set of edge rings corresponding to E'. Intuitively, if X is the minimum cycle homologous to D we do not want $X \oplus D$ to intersect the interior of any vertex region. That is, $X \oplus D$ is a collection of edge rings and corresponds to a cut in G. To achieve this, we subdivide each edge not contained in an edge ring into a long path. The result is an embedded graph with non-triangular faces, which is not a simplicial complex. To fix this, we triangulate the inside of each non-triangular face such that the shortest path between any two vertices on the face remains the shortest path after the triangulation. Given any α-approximation of the new complex we can obtain a smaller solution using only the edge rings, which corresponds to a cut in G. Our formal construction follows.

Let $\tau = 4\lceil\alpha\rceil|E| + 1$; we subdivide each edge not contained in an edge ring τ times. For each face of length $\ell > 3$ we triangulate by adding $\ell+1$ concentric cycles, each with ℓ vertices, labeled $\gamma_0, \ldots, \gamma_\ell$, where γ_0 is the original face from the subdivided version of $\mathcal{M}$. By $v_{i,j}$ we denote the jth vertex in γ_i. We add the edges $(v_{i,j}, v_{i+1,j}$ and $(v_{i,j}, v_{i,j+1 \mod \ell})$. To complete the triangulation we add one additional vertex $\overline{v}$ at the center of γ_ℓ and add an edge between it and each vertex on γ_ℓ. We call the new simplicial complex $\mathcal{M}'$. See Figure 3 for an example.

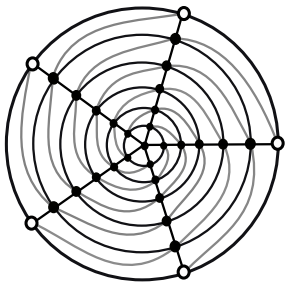


Figure 3 Subdividing a face of length five; the outer face with white vertices is the original face.

Let $(S_{opt}, \overline{S_{opt}})$ be an optimal solution to the minimum cut completion instance (G, E'). Suppose we can compute an α-approximation C of the minimum homologous cycle instance $(\mathcal{M}', D)$, hence $|C| \leq \alpha|C_{opt}|$. By our construction an optimal solution to $(\mathcal{M}', D)$ has the same size as an optimal solution to $(\mathcal{M}, D)$. As C is a cycle, if C crosses a cycle γ_0 it must cross it an even number of times. For any two consecutive vertices $u, v \in \gamma_0$ in C we replace the path between them with the shortest path contained in γ_0. We call the new cycle C', since $C' \leq C$ we have that C' is also an α-approximation for $(\mathcal{M}', D)$. Note that C' is a union of edge rings, otherwise $|C'| > \alpha|C_{opt}|$. It follows that C' corresponds to a cut $E_{S'}$ with $|C'| = 4|E_{S'} \oplus E'|$. Hence, we have $|E_{S'} \oplus E'| \leq \alpha|E_{S_{opt}} \oplus E'|$. Thus, $E_{S'}$ is an α-approximation for (G, E'). ◀

4.3 Wrap up

It remains to show that the cut completion problem is hard to approximate. We show this via a straightforward reduction from the *minimum uncut problem*: given a graph $G = (V, E)$, find a cut with minimum number of uncut edges. Note that the optimal cuts for the minimum uncut problem and the maximum cut problem coincide, yet, approximation algorithms for one problem do not necessarily imply approximation algorithm for the other one.

▶ **Lemma 16.** *The minimum uncut problem is a special case of the minimum cut completion problem.*

Proof. Consider the cut completion problem for $G = (V, E)$, and let $E' = E$. Let $(S, \overline{S})$ be any cut with edge set E_S. The cut completion cost of this cut is

$$|E_S \oplus E'| = |E_S \oplus E| = |E \setminus E_S|,$$

which is the number of uncut edges by $(S, \overline{S})$. ◀

Now, we are ready to prove our hardness results.

Proof of Theorem 5 and 6. The minimum uncut problem is hard to approximate within $(1 + \varepsilon)$ for some $\varepsilon > 0$ [26]. In addition, it is hard to approximate within any constant factor assuming the unique games conjecture [24, 21, 10, 22]. By Lemma 16, the cut completion problem generalizes the minimum uncut problem. Finally, by Lemma 15 and 14, for any $\alpha > 1$ and $\varepsilon > 0$, an α-approximation algorithm for the minimum bounded chain problem or the minimum homologous cycle problem implies a $((1 + \varepsilon)\alpha)$-approximation algorithm, or an α-approximation for the cut completion problem, respectively. ◀

References

1. Amit Agarwal, Moses Charikar, Konstantin Makarychev, and Yury Makarychev. $O(\sqrt{\log n})$ approximation algorithms for min uncut, min 2CNF deletion, and directed cut problems. In *Proceedings of the Thirty-seventh Annual ACM Symposium on Theory of Computing*, STOC '05, pages 573–581, New York, NY, USA, 2005. ACM. doi:10.1145/1060590.1060675.
2. Ian Agol, Joel Hass, and William Thurston. The computational complexity of knot genus and spanning area. *Transactions of the American Mathematical Society*, 358(09):3821–3851, September 2006. doi:10.1090/s0002-9947-05-03919-x.
3. Per Austrin and Subhash Khot. A simple deterministic reduction for the gap minimum distance of code problem. In *Proceedings of the 38th International Colloquim Conference on Automata, Languages and Programming - Volume Part I*, ICALP'11, pages 474–485, Berlin, Heidelberg, 2011. Springer-Verlag.
4. E. Berlekamp, R. McEliece, and H. van Tilborg. On the inherent intractability of certain coding problems (corresp.). *IEEE Transactions on Information Theory*, 24(3):384–386, May 1978. doi:10.1109/TIT.1978.1055873.
5. Arnab Bhattacharyya, Ameet Gadekar, Suprovat Ghoshal, and Rishi Saket. On the hardness of learning sparse parities. In *24th Annual European Symposium on Algorithms, ESA 2016, August 22-24, 2016, Aarhus, Denmark*, pages 11:1–11:17, 2016.
6. Oleksiy Busaryev, Sergio Cabello, Chao Chen, Tamal K. Dey, and Yusu Wang. Annotating simplices with a homology basis and its applications. In Fedor V. Fomin and Petteri Kaski, editors, *Algorithm Theory – SWAT 2012*, pages 189–200, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.
7. J. Carvalho, Mikael Vejdemo-Johansson, Danica Kragic, and Florian Pokorny. An algorithm for calculating top-dimensional bounding chains. *PeerJ Computer Science*, 4:e153, May 2018. doi:10.7717/peerj-cs.153.
8. Erin W. Chambers, Jeff Erickson, and Amir Nayyeri. Minimum cuts and shortest homologous cycles. In *Proceedings of the Twenty-fifth Annual Symposium on Computational Geometry*, SCG '09, pages 377–385, New York, NY, USA, 2009. ACM. doi:10.1145/1542362.1542426.
9. Erin Wolf Chambers and Mikael Vejdemo-Johansson. Computing minimum area homologies. *Comput. Graph. Forum*, 34(6):13–21, September 2015. doi:10.1111/cgf.12514.

10 Shuchi Chawla, Robert Krauthgamer, Ravi Kumar, Yuval Rabani, and D. Sivakumar. On the hardness of approximating multicut and sparsest-cut. *Comput. Complex.*, 15(2):94–114, June 2006. doi:10.1007/s00037-006-0210-9.

11 Chao Chen and Daniel Freedman. Hardness results for homology localization. *Discrete & Computational Geometry*, 45(3):425–448, April 2011. doi:10.1007/s00454-010-9322-8.

12 A.V. Chernavsky and V.P. Leksine. Unrecognizability of manifolds. *Annals of Pure and Applied Logic*, 141(3):325–335, 2006. Papers presented at the Second St. Petersburg Days of Logic and Computability Conference on the occasion of the centennial of Andrey Andreevich Markov, Jr. doi:10.1016/j.apal.2005.12.011.

13 Tamal K. Dey, Anil N. Hirani, and Bala Krishnamoorthy. Optimal homologous cycles, total unimodularity, and linear programming. *SIAM J. Comput.*, 40(4):1026–1044, July 2011. doi:10.1137/100800245.

14 Tamal K. Dey, Tao Hou, and Sayan Mandal. Computing minimal persistent cycles: Polynomial and hard cases. *ArXiv*, abs/1907.04889, 2019. arXiv:1907.04889.

15 Rod G. Downey, Michael R. Fellows, Alexander Vardy, and Geoff Whittle. The parametrized complexity of some fundamental problems in coding theory. *SIAM J. Comput.*, 29(2):545–570, October 1999. doi:10.1137/S0097539797323571.

16 Nathan M. Dunfield and Anil N. Hirani. The least spanning area of a knot and the optimal bounding chain problem. In *Proceedings of the Twenty-seventh Annual Symposium on Computational Geometry*, SoCG '11, pages 135–144, New York, NY, USA, 2011. ACM. doi:10.1145/1998196.1998218.

17 Jeff Erickson and Amir Nayyeri. Minimum cuts and shortest non-separating cycles via homology covers. In *Proceedings of the Twenty-second Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '11, pages 1166–1176, Philadelphia, PA, USA, 2011. Society for Industrial and Applied Mathematics. URL: http://dl.acm.org/citation.cfm?id=2133036.2133124.

18 Leo Grady. Minimal surfaces extend shortest path segmentation methods to 3D. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32:321–334, 2010.

19 Joshua A. Grochow and Jamie Tucker-Foltz. Computational Topology and the Unique Games Conjecture. In Bettina Speckmann and Csaba D. Tóth, editors, *34th International Symposium on Computational Geometry (SoCG 2018)*, volume 99 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 43:1–43:16, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.SoCG.2018.43.

20 Allen Hatcher. *Algebraic topology*. Cambridge University Press, 2002.

21 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, April 2007. doi:10.1137/S0097539705447372.

22 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, March 2015. doi:10.1145/2629614.

23 Danil Kirsanov and Steven Gortler. A discrete global minimization algorithm for continuous variational problems, August 2004.

24 Pasin Manurangsi and Luca Trevisan. Mildly exponential time approximation algorithms for vertex cover, balanced separator and uniform sparsest cut. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2018, August 20-22, 2018 - Princeton, NJ, USA*, pages 20:1–20:17, 2018. doi:10.4230/LIPIcs.APPROX-RANDOM.2018.20.

25 James R. Munkres. *Elements of Algebraic Topology*. Addison Wesley Publishing Company, 1984. URL: http://www.worldcat.org/isbn/0201045869.

26 Christos Papadimitriou and Mihalis Yannakakis. Optimization, approximation, and complexity classes. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, STOC '88, pages 229–234, New York, NY, USA, 1988. ACM. doi:10.1145/62212.62233.

27 Igor Razgon and Barry O'Sullivan. Almost 2-sat is fixed-parameter tractable. *J. Comput. Syst. Sci.*, 75(8):435–450, December 2009. doi:10.1016/j.jcss.2009.04.002.

28 John M. Sullivan. *A Crystalline Approximation Theorem for Hypersurfaces*. PhD thesis, Princeton University, 1990. URL: http://torus.math.uiuc.edu/jms/Papers/thesis.

29 A. Vardy. The intractability of computing the minimum distance of a code. *IEEE Trans. Inf. Theor.*, 43(6):1757–1766, November 1997.

On Rectangle-Decomposable 2-Parameter Persistence Modules

Magnus Bakke Botnan
Vrije Universiteit Amsterdam, The Netherlands
m.b.botnan@vu.nl

Vadim Lebovici
École Normale Supérieure, Paris, France
vadim.lebovici@ens.fr

Steve Oudot
Inria Saclay, Palaiseau, France
steve.oudot@inria.fr

Abstract

This paper addresses two questions: (1) can we identify a sensible class of 2-parameter persistence modules on which the rank invariant is complete? (2) can we determine efficiently whether a given 2-parameter persistence module belongs to this class? We provide positive answers to both questions, and our class of interest is that of rectangle-decomposable modules. Our contributions include: (a) a proof that the rank invariant is complete on rectangle-decomposable modules, together with an inclusion-exclusion formula for counting the multiplicities of the summands; (b) algorithms to check whether a module induced in homology by a bifiltration is rectangle-decomposable, and to decompose it in the affirmative, with a better complexity than state-of-the-art decomposition methods for general 2-parameter persistence modules. Our algorithms are backed up by a new structure theorem, whereby a 2-parameter persistence module is rectangle-decomposable if, and only if, its restrictions to squares are. This local condition is key to the efficiency of our algorithms, and it generalizes previous conditions from the class of block-decomposable modules to the larger one of rectangle-decomposable modules. It also admits an algebraic formulation that turns out to be a weaker version of the one for block-decomposability. Our analysis focuses on the case of modules indexed over finite grids, the more general cases are left as future work.

2012 ACM Subject Classification Mathematics of computing → Algebraic topology

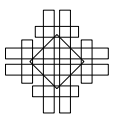
Keywords and phrases topological data analysis, multiparameter persistence, rank invariant

Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.22

Related Version A full version of the paper is available at `https://arxiv.org/abs/2002.08894`.

1 Introduction

A *persistence module* M over a subset $U \subseteq \mathbb{R}^d$ is a collection of vector spaces $\{M_t\}_{t\in U}$ and linear maps $\rho_s^t := M(s \leq t)\colon M_s \to M_t$ with the property that ρ_s^s is the identity map and $\rho_t^u \circ \rho_s^t = \rho_s^u$ for all $s \leq t \leq u \in U$. Here $s \leq t$ if and only if $s_i \leq t_i$ for all $i \in \{1, 2, \ldots, d\}$. In the language of category theory, a persistence module M is a functor $M\colon U \to \mathsf{vec}$ where vec is the category of vector spaces and the partially ordered set U is considered as a category in the obvious way. In this setting, *morphisms* between persistence modules are natural transformations $M \Rightarrow N$ between functors, defined by collections of linear maps $\{\varphi_t : M_t \to N_t\}_{t\in U}$ such that $\varphi_t \circ M(s \leq t) = N(s \leq t) \circ \varphi_s$ for all $s \leq t \in U$. Their kernels, images and cokernels, as well as products, direct sums and quotients of persistence modules, are defined pointwise at each index $t \in U$. Similarly, an *isomorphism* between two persistence modules is a natural isomorphism between them. We will refer to the case $d = 1$ as *single-parameter persistence*, and for $d \geq 2$ we will use the term *multi-parameter persistence*.



▶ Remark 1. Throughout this paper we will work exclusively with finite-dimensional vector spaces over a fixed field $\mathbf{k}$. When finite-dimensionality is emphasized we will refer to the persistence module as being pointwise finite-dimensional (pfd).

Single-parameter persistence modules are typically obtained through the application of homology to a filtered topological space. This process is known as *persistent homology* and has found a wide range of applications to the sciences, as well as to other parts of mathematics such as symplectic geometry. See [14, 20] for an introduction to persistent homology. What makes such persistence modules particularly amenable to data analysis is that they can be completely described by multisets of intervals in $\mathbb{R}$ called *barcodes* [12]. Such a collection of intervals can then in turn be used to extract topological information from the data at hand, and further utilized in statistics and machine learning. We now give an example of this structure theorem in the simple case of $U = \{1, 2, 3\} \subseteq \mathbb{R}$.

▶ **Example 2.** Consider the following sequence of vector spaces and linear maps

$$\mathbf{k}^2 \xrightarrow{\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}} \mathbf{k}^2 \xrightarrow{[1\ -1]} \mathbf{k}.$$

By replacing the basis $\{e_1, e_2\}$ of the middle vector space $\mathbf{k}^2$ with the basis $\{e_1, e_1 + e_2\}$ we get the following matrix representations of the linear maps

$$\mathbf{k}^2 \xrightarrow{\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}} \mathbf{k}^2 \xrightarrow{[1\ 0]} \mathbf{k} = \left(\mathbf{k} \xrightarrow{1} \mathbf{k} \xrightarrow{1} \mathbf{k}\right) \oplus \left(\mathbf{k} \xrightarrow{1} \mathbf{k} \to 0\right).$$

The two persistence modules on the right-hand side are uniquely specified by their supports $\{1, 2, 3\}$ and $\{1, 2\}$, respectively. Their supports give rise to the barcode which in this case is given by $\{\{1, 2, 3\}, \{1, 2\}\}$.

As illustrated by Example 2, a persistence module can be recovered from its barcode thanks to the notion of *indicator modules*: for $X \times Y \subseteq \mathbb{R}^2$ and a subset $Q \subseteq X \times Y$, the indicator module of Q, denoted $\mathbf{k}_Q$, is defined by

$$\mathbf{k}_{Q,t} = \begin{cases} \mathbf{k} & (t \in Q) \\ 0 & (t \notin Q) \end{cases} \qquad \mathbf{k}_Q(s \leq t) = \begin{cases} \mathrm{Id}_{\mathbf{k}} & \text{if } s \text{ and } t \in Q, \\ 0 & \text{else.} \end{cases}$$

By convention, we set $\mathbf{k}_\emptyset = 0$. A persistence module is an *interval module* if it is the indicator module of an interval[1]. Note that, just as choosing a basis for a vector space is not canonical, there may be many ways of decomposing a single-parameter persistence module into a direct sum of such interval modules. However, just as for the dimension of a finite-dimensional vector space, the associated barcode given by the multiset of interval supports of the summands is independent of the chosen decomposition [1].

Another desirable property of single-parameter persistence modules M is that they are completely described up to isomorphism by the *rank invariant*, i.e. the collection of ranks $r(s, t) = \mathrm{rank}(M(s \leq t))$ for all $s \leq t$. This can easily be verified in the previous example, and more generally, for any pfd persistence module M indexed over a finite set $[\![1, n]\!]$, the following inclusion-exclusion formula (also known as the *persistence measure* [9, 11]) gives the multiplicity $m(s, t)$ of any interval $[\![s, t]\!]$ in the barcode of M:

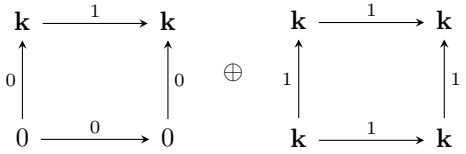
$$m(s, t) = r(s, t) - r(s - 1, t) - r(s, t + 1) + r(s - 1, t + 1). \tag{1}$$

[1] In the poset $X \times Y$, we say that Q is an interval if it is convex and zigzag path-connected, i.e if between any two points $p, q \in Q$, there exists a zigzag path $p \leq p_1 \geq p_2 \leq \cdots \geq p_n \leq q$ with p_i's in Q.

Many applications do however naturally come equipped with multiple parameters, and for such applications it is natural to consider multi-parameter persistence modules, see e.g. the introduction of [2] for an example of how multi-parameter persistence connects to hierarchical clustering. Let us first consider the simplest instantiation of 2-parameter persistence modules, namely modules indexed by the square $S = \{a = (0,0), b = (1,0), c = (0,1), d = (1,1)\} \subseteq \mathbb{R}^2$.

▶ **Example 3.** The persistence module on the left-hand side below can be transformed into the one on the right-hand side via a change of basis at the vertices:

In turn, the persistence module on the right-hand side is the direct sum



Just as in Example 2, these persistence modules are completely defined by their support. We define the barcode of the aforementioned persistence module to be the (multi-)set of supports of its summands, namely $\{\{c,d\},\{a,b,c,d\}\}$.

Although commutative diagrams like the one in the previous example may appear unwieldy at first glance, such persistence modules can – just as in the single-parameter case – be completely described (up to isomorphism) by a multiset of elements from

$$\mathcal{I} := \{\{a\},\{b\},\{c\},\{d\},\{a,b\},\{a,c\},\{b,d\},\{c,d\},\{a,b,c\},\{b,c,d\},\{a,b,c,d\}\}, \tag{2}$$

called *intervals* in the 2×2 grid. See e.g. Figure 13 in [15]. However, in contrast to the single-parameter case, the rank invariant on persistence modules indexed by S is no longer a *complete* invariant, i.e. it does not fully determine the isomorphism type of such modules. For instance, two persistence modules with barcodes $\{\{a,b,c\},\{a\}\}$ and $\{\{a,b\},\{\{a,c\}\}$ are non-isomorphic yet exhibit the same rank invariant.

▶ **Example 4.** Consider the following two persistence modules:

The diagram to the left can easily be seen to be composed of two interval summands in the 3×2 grid. By contrast, the diagram to the right is indecomposable: there exists no change of basis for which this persistence module can be written as a direct sum of persistence modules in a non-trivial way. Again, the two modules have the same rank invariant.

In the setting of no more than four columns and two rows, results from the field of representation theory of quivers show that there exists a finite set of building blocks (indecomposable modules) from which every persistence module can be built (via direct sums, and up to isomorphism). Based on this, one can associate a well-defined barcode-like structure to such a module by counting the multiplicity of every summand in the decomposition. The inclusion of such grids into topological data analysis was inspired by a problem in materials science [15]. For five or more columns the theory becomes increasingly complex. In particular, for six or more columns there is no way to parametrize a set of building blocks in any reasonable way[2]. This is a major obstacle to the development of the theory of multi-parameter persistence.

A natural question to consider then is whether one can endow multi-parameter persistence modules with additional structure in order to enforce nice decomposition theorems akin to that of single-parameter persistence. One such setting coming from computational topology was identified in [3, 8], and further generalized in [10], where it is shown that the so-called *strongly exact* 2-parameter persistence modules indexed over $\mathbb{R}^2$ are determined (up to isomorphism) by a multiset of particularly simple planar rectangular regions called *blocks*. Basically, a block is either an upper-right or lower-left quadrant, or a horizontal or vertical infinite band. The great advantage of this condition is that it can be checked locally: a 2-parameter persistence module (called a *bimodule* for short) is block-decomposable if, and only if, its restriction to any square as in Example 3 is block-decomposable.

Contributions. In this paper we address two important follow-up questions:

- Can we work out conditions such as above for larger classes of bimodules?
- Can we identify classes of bimodules for which the rank invariant is complete?

Our answers to both questions are positive, and the two classes of bimodules turn out to be the same, namely that of *rectangle-decomposable* bimodules, which by definition are determined (up to isomorphism) by a multiset of *rectangles*, i.e subsets R of the form $R = I \times J \subseteq \mathbb{R}^2$ where I and J are intervals in $\mathbb{R}$. Specifically, a bimodule is rectangle-decomposable if it decomposes into a direct sum of *rectangle modules*, i.e. indicator modules of rectangles.

Our local condition for rectangle decomposability, called *weak exactness*, is a weaker version of the condition for block decomposability, in that it allows all types of rectangular shapes in the local squares' decompositions, as opposed to just blocks. More precisely, calling $\mathcal{R}$ the following subset of $\mathcal{I}$:

$$\mathcal{R} = \{\{a\}, \{b\}, \{c\}, \{d\}, \{a, b\}, \{a, c\}, \{b, d\}, \{c, d\}, \{a, b, c, d\}\}, \tag{3}$$

▶ **Definition 5** (Weak exactness). *Given subsets X, Y of $\mathbb{R}$, a persistence module $M\colon X \times Y \subseteq \mathbb{R} \times \mathbb{R} \to$ vec is* weakly exact *if the barcode of the following square*

$$\begin{array}{ccc} M_{(s_x,t_y)} & \xrightarrow{\rho^t_{(s_x,t_y)}} & M_t \\ {\scriptstyle \rho_s^{(s_x,t_y)}}\uparrow & & \uparrow{\scriptstyle \rho^t_{(t_x,s_y)}} \\ M_s & \xrightarrow[\rho_s^{(t_x,s_y)}]{} & M_{(t_x,s_y)} \end{array} \tag{4}$$

consists of elements from $\mathcal{R}$ for all indices $s \leq t$ in $X \times Y$.

By comparison, the strong exactness condition replaces $\mathcal{R}$ by $\mathcal{B} = \mathcal{R} \setminus \{\{b\}, \{c\}\}$.

[2] The underlying graph, called a quiver, is known to be of *wild* representation type.

▶ **Example 6.** The persistence module to the left below is strongly exact, while the one to the right is only weakly exact, and the persistence modules in Example 4 are not even weakly exact (each time the weak or strong exactness condition fails on the outermost rectangle):

Our analysis focuses on the case of modules indexed over finite grids[3], the more general cases are left as future work. Our contributions summarize as follows:

- In Section 2 we prove that the rank invariant is complete on the class of rectangle-decomposable bimodules (Theorem 8). To this end, we generalize the inclusion-exclusion formula (1) to our setting. Note that our result also follows indirectly from an inclusion-exclusion formula for a generalization of the rank invariant for interval-decomposable modules [17, Prop. 7.13], but that we provide an explicit statement together with a simple and direct proof.
- In Section 3 we show that the rank invariant of a (1-critical) simplicical bifiltration with a total of n simplices can be computed in $O(n^4)$ time (Theorem 9). In conjunction with our inclusion-exclusion formula, this yields an $O(n^4)$ time algorithm for computing the barcode of a persistence bimodule that is known to be rectangle-decomposable (Corollary 10). This is an improvement over merely applying some state-of-the-art algorithm for computing decompositions of general 2-parameter persistence modules, which would take $O(n^{2\omega+1})$ time where $2 \leq \omega < 2.373$ is the exponent for matrix multiplication [13].
- In Section 4 we propose an algebraic formulation of our weak exactness condition (Definition 11). This formulation turns out to be equivalent to Definition 5 and to global rectangle decomposability (the central mathematical result in the paper), specifically:
  ▶ **Theorem 7.** *Let M be a pfd persistence module indexed over $X \times Y$, where X, Y are finite subsets of $\mathbb{R}$. Then, M is rectangle-decomposable if and only if M is weakly exact.*
- In Section 5 we leverage this result to derive an $O(n^{2+\omega})$-time algorithm for checking the rectangle-decomposability of persistence bimodules induced in homology from (1-critical) simplicical bifiltrations with at most n simplices (Theorem 19). Once again, this is an improvement over applying some state-of-the-art algorithm for computing decompositions of general 2-parameter persistence modules and then checking the summands one by one.
- Finally, in Section 6 we show how rectangle-decomposable modules arise from (sufficiently tame) real-valued functions on a topological space. This is then used to give a new proof of the *pyramid basis theorem* of [3].

2 Completeness of the rank invariant

Suppose in this section that X, Y are subsets of $\mathbb{Z}$.

▶ **Theorem 8.** *The isomorphism type of any pfd rectangle-decomposable persistence module M over $X \times Y$ is fully determined by the rank invariant of M.*

[3] A *finite grid* is the product of two finite subsets of $\mathbb{R}$. Note that any finite grid can be identified with a grid of the form $[\![1, n]\!] \times [\![1, m]\!]$ for appropriate choices of $n, m \in \mathbb{N}$.

The proof consists in showing that the multiplicity $m(s,t)$ of each individual rectangle module $\mathbf{k}_{[\![s_x,t_x]\!]\times[\![s_y,t_y]\!]}$ in the decomposition of M is given by the inclusion-exclusion formula (7) below, which involves only the rank invariant $r : X \times Y \to \mathbb{N}$ of M. This formula is the analogue, in the category of rectangle-decomposable pfd bimodules, of the inclusion-exclusion formula (1) for counting the multiplicities of interval summands in one-parameter persistence.

Fix arbitrary indices $s \leq t \in X \times Y$. Recall that the rank of $(A \oplus B)(s \leq t)$ is equal to the sum of the ranks of $A(s \leq t)$ and $B(s \leq t)$. Meanwhile, for any summand $\mathbf{k}_R$ of M, the rank of $\mathbf{k}_R(s \leq t)$ is 1 if $s,t \in R$ and 0 otherwise. Therefore, $r(s,t)$ counts (with multiplicity) the number of summands of M whose rectangle support contains both s and t. Then, denoting by $m(s,t^+)$ the number of (rectangle) summands whose support contains t and has s as lower-left corner, we have the folllowing inclusion-exclusion formula:

$$m(s,t^+) = r(s,t) - r((s_x-1,s_y),t) - r((s_x,s_y-1),t) + r((s_x-1,s_y-1),t). \tag{5}$$

This formula can be interpreted as follows: a rectangle containing t has s as lower-left corner if and only if it contains s but neither (s_x-1,s_y) nor (s_x,s_y-1); and it contains both (s_x-1,s_y) and (s_x,s_y-1) if and only if it contains (s_x-1,s_y-1).

Using the same approach at t, we can now compute the number $m(s,t)$ of summands of M whose support has s as lower-left corner and t as upper-right corner (i.e. is the rectangle $[\![s_x,t_x]\!] \times [\![s_y,t_y]\!]$). The corresponding inclusion-exclusion formula is:

$$m(s,t) = m(s,t^+) - m(s,(t_x+1,t_y)^+) - m(s,(t_x,t_y+1)^+) + m(s,(t_x+1,t_y+1)^+). \tag{6}$$

Combining (5) and (6) together gives the desired inclusion-exclusion formula for the multiplicity $m(s,t)$ of the summand $\mathbf{k}_{[\![s_x,t_x]\!]\times[\![s_y,t_y]\!]}$ in the decomposition of M from the rank invariant, hence completing the proof of Theorem 8, namely:

$$\begin{aligned} m(s,t) \quad = \quad & r(s,t) - r((s_x-1,s_y),t) \\ & -r((s_x,s_y-1),t) + r((s_x-1,s_y-1),t) \\ & -r(s,(t_x+1,t_y)) + r((s_x-1,s_y),(t_x+1,t_y)) \\ & +r((s_x,s_y-1),(t_x+1,t_y)) - r((s_x-1,s_y-1),(t_x+1,t_y)) \\ & -r(s,(t_x,t_y+1)) + r((s_x-1,s_y),(t_x,t_y+1)) \\ & +r((s_x,s_y-1),(t_x,t_y+1)) - r((s_x-1,s_y-1),(t_x,t_y+1)) \\ & +r(s,(t_x+1,t_y+1)) - r((s_x-1,s_y),(t_x+1,t_y+1)) \\ & -r((s_x,s_y-1),(t_x+1,t_y+1)) + r((s_x-1,s_y-1),(t_x+1,t_y+1)). \end{aligned} \tag{7}$$

3 Computing the rank invariant and rectangle decompositions

Let F be a simplicial bifiltration with n simplices in total. Assume without loss of generality that F is indexed over the grid $G = [\![1,n]\!] \times [\![1,n]\!]$, for any larger indexing grid must contain arrows with identity maps that can be pre- or post-composed, and any smaller grid can be enlarged by inserting arrows with identity maps. Assume further that F is 1-*critical*, meaning that the set $\{t \in G \mid \sigma \in F_t\}$ has a unique minimal element for any simplex σ entering the filtration. We also fix a homology degree p.

▶ **Theorem 9.** *Given the above input, the rank invariant of the persistence bimodule M induced by F in p-th homology can be computed in $O(n^4)$ time.*

As we have not seen any proof of this result in the literature, below we provide an algorithm with the desired complexity. But before this, let us point out that this theorem, combined with the inclusion-exclusion formula (7), gives an $O(n^4)$-time algorithm to compute the barcode of F assuming that M is rectangle-decomposable: once the rank invariant of M has been computed, iterate over all the pairs (s,t) with $s \leq t \in G$ and, for each one of them, apply the formula in constant time to get the multiplicity of the rectangle module $\mathbf{k}_{[\![s_x,t_x]\!]\times[\![s_y,t_y]\!]}$ in the decomposition of M. Thus,

▶ **Corollary 10.** *Computing the decomposition of a rectangle-decomposable module over $X \times Y$ induced in homology by a 1-critical bifiltration with n simplices in total can be done in $O(n^4)$ time.*

This complexity compares favorably to that of the currently best known algorithm for computing direct-sum decompositions of general persistence bimodules[4], which is $O(n^{2\omega+1})$ where $2 \leq \omega < 2.373$ is the exponent for matrix multiplication [13].

Let us now provide the algorithm for Theorem 9. First, we compute a free resolution of M of size $O(n^2)$ from F in $O(n^3)$ time using the algorithm of [18]. This free resolution takes the form of an exact sequence as follows[5]:

$$0 \longrightarrow M_\zeta \xrightarrow{\psi} M_\eta \xrightarrow{\varphi} M_\gamma \longrightarrow M \longrightarrow 0,$$

where M_γ, M_η and M_ζ are free bigraded modules, equipped with bases $(\gamma_1, \cdots, \gamma_k)$, $(\eta_1, \cdots, \eta_l)$, $(\zeta_1, \cdots, \zeta_m)$ respectively, of sizes $k, l, m \leq n$. The elements γ_i are called *generators*, while the η_j are called *relations* and the ζ_r are called *relations on relations*. Each one of them is homogeneous and thus assigned a unique grade, denoted by $\mathrm{gr}(\cdot)$. The morphisms of free bigraded modules φ, ψ are given as $k \times l$ and $l \times m$ matrices respectively, with coefficients in $\mathbf{k}$. Exactness implies that ψ is injective and $M \cong \operatorname{Coker} \varphi$.

Ignoring first the relations and the relations on relations (i.e. assuming that $l = m = 0$ hence $M_\eta = M_\zeta = 0$), the rank invariant $r : [\![1,n]\!]^2 \times [\![1,n]\!]^2 \to \mathbb{N}$ is given by:

$$\forall s \leq t \in G, \quad r(s,t) = \#\{i \mid \mathrm{gr}(\gamma_i) \leq s\}.$$

Computing the numbers on the right-hand side for all $s \leq t \in G$ is easily done in $O(n^4)$ time by dynamic programming: first we iterate over the $k \leq n$ generators to fill in an $n \times n$ table storing at each index s the number of generators having s as their grade; then we build a $n^2 \times n^2$ lookup table by iterating over the indices (s,t) in lexicographic order and using the following recurrence formula[6]:

$$\begin{aligned}
\#\{i \mid \mathrm{gr}(\gamma_i) \leq s\} &= \#\{i \mid \mathrm{gr}(\gamma_i) < s\} + \#\{i \mid \mathrm{gr}(\gamma_i) = s\} \\
&= \#\{i \mid \mathrm{gr}(\gamma_i) \leq (s_x - 1, s_y)\} + \#\{i \mid \mathrm{gr}(\gamma_i) \leq (s_x, s_y - 1)\} \\
&\quad - \#\{i \mid \mathrm{gr}(\gamma_i) \leq (s_x - 1, s_y - 1)\} + \#\{i \mid \mathrm{gr}(\gamma_i) = s\}.
\end{aligned}$$

Once this is done, we can take the relations into account (still ignoring relations on relations, i.e. assuming that $m = 0$ hence $M_\zeta = 0$). Each relation η_j gives a $\mathbf{k}$-linear constraint on a subset Υ of the generators, encoded in the j-th column of the matrix of φ: $\sum_{u \in \Upsilon} \alpha_u \gamma_u = 0$,

[4] Let us also point out that our approach does not suffer from the limitation of the algorithm of [13], which is that no two generators or relations in a minimal presentation of M can have the same grade.

[5] Such resolutions exist by Hilbert's Syzygy theorem. As mentioned in [18], although their algorithm only computes a free presentation, it adapts readily to compute a free resolution with the same time complexity – simply reapply their kernel basis computation procedure to the algorithm's output matrix.

[6] The formula adapts to the cases where $s_x = 1$ or $s_y = 1$ by merely removing the invalid terms.

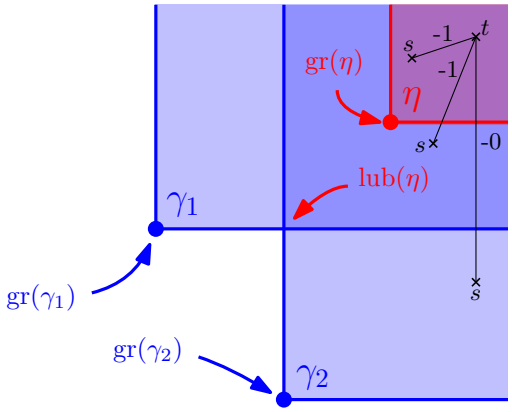


Figure 1 A relation η between two generators γ_1, γ_2, and its effect on $r(s,t)$ for various $s \leq t$.

where each α_u belongs to $\mathbf{k}\backslash\{0\}$. Call $\mathrm{lub}(\eta_j)$ the least upper bound of the set $\{\mathrm{gr}(\gamma_u) \mid u \in \Upsilon\}$ in the product order $\leq$ on G. The effect of η_j on the rank invariant is to decrement it at indices $s \leq t$ such that $s \geq \mathrm{lub}(\eta_j)$ and $t \geq \mathrm{gr}(\eta_j)$, as illustrated in Figure 1. Hence the following formula for the update of $r(s,t)$:

$$r(s,t) \quad \longleftarrow \quad r(s,t) - \#\{j \mid \mathrm{lub}(\eta_j) \leq s \text{ and } \mathrm{gr}(\eta_j) \leq t\}.$$

The numbers on the right-hand side can be computed in $O(n^4)$ time again using dynamic programming: first we build a $n^2 \times n^2$ table storing at each index (s,t) the number of relations η_j such that $\mathrm{gr}(\eta_j) = t$ and $\mathrm{lub}(\eta_j) = s$; then for each index $t \in G$ we build an intermediate $n \times n$ lookup table by iterating over the indices s in lexicographic order and using the following recurrence formula[6]:

$$\begin{aligned}\#\{j \mid \mathrm{lub}(\eta_j) \leq s \text{ and } \mathrm{gr}(\eta_j) = t\} \quad =& \#\{j \mid \mathrm{lub}(\eta_j) \leq (s_x - 1, s_y) \text{ and } \mathrm{gr}(\eta_j) = t\} \\ &+ \#\{j \mid \mathrm{lub}(\eta_j) \leq (s_x, s_y - 1) \text{ and } \mathrm{gr}(\eta_j) = t\} \\ &- \#\{j \mid \mathrm{lub}(\eta_j) \leq (s_x - 1, s_y - 1) \text{ and } \mathrm{gr}(\eta_j) = t\} \\ &+ \#\{j \mid \mathrm{lub}(\eta_j) = s \text{ and } \mathrm{gr}(\eta_j) = t\};\end{aligned}$$

finally, we build the lookup table for the rank invariant r by iterating over the indices (s,t) in lexicographic order and using the following recurrence formula[6]:

$$\begin{aligned}\#\{j \mid \mathrm{lub}(\eta_j) \leq s \text{ and } \mathrm{gr}(\eta_j) \leq t\} \quad =& \quad \#\{j \mid \mathrm{lub}(\eta_j) \leq s \text{ and } \mathrm{gr}(\eta_j) \leq (t_x - 1, t_y)\} \\ &+ \#\{j \mid \mathrm{lub}(\eta_j) \leq s \text{ and } \mathrm{gr}(\eta_j) \leq (t_x, t_y - 1)\} \\ &- \#\{j \mid \mathrm{lub}(\eta_j) \leq s \text{ and } \mathrm{gr}(\eta_j) \leq (t_x - 1, t_y - 1)\} \\ &+ \#\{j \mid \mathrm{lub}(\eta_j) \leq s \text{ and } \mathrm{gr}(\eta_j) = t\}.\end{aligned}$$

Once this is done, we can take the relations on relations into account. Each one of them, say ζ_r, gives a $\mathbf{k}$-linear constraint on a subset Ξ of the relations, encoded in the r-th column of the matrix of ψ: $\sum_{v \in \Xi} \beta_v \eta_v = 0$, where each β_v belongs to $\mathbf{k} \setminus \{0\}$. Calling $\mathrm{lub}(\zeta_r)$ the least upper bound of the set $\{\mathrm{lub}(\eta_v) \mid v \in \Xi\}$, the effect of ζ_r on the rank invariant is to compensate for one of the relations by incrementing $r(s,t)$ at indices $s \leq t$ such that $s \geq \mathrm{lub}(\zeta_r)$ and $t \geq \mathrm{gr}(\zeta_r)$. Hence the following formula for the update of $r(s,t)$:

$$r(s,t) \quad \longleftarrow \quad r(s,t) + \#\{r \mid \mathrm{lub}(\zeta_r) \leq s \text{ and } \mathrm{gr}(\zeta_r) \leq t\}.$$

The numbers on the right-hand side can be computed in $O(n^4)$ time using the same two-stage dynamic programming scheme as introduced for relations.

All in all, the algorithm takes $O(n^4)$ time. The injectivity of ψ means that the relations on relations are linearly independent, so the correctness of the output table representing the rank invariant follows by design. This completes the proof of Theorem 9.

4 Algebraic formulation of weak exactness

As shown in [5, 10], a persistence module $M\colon X \times Y \subseteq \mathbb{R}\times\mathbb{R} \to \mathsf{vec}$ is strongly exact if, and only if, the following sequence induced by (4) is exact for all indices $s \leq t \in X \times Y$:

$$M_s \xrightarrow{(\rho_s^{(t_x,s_y)},\rho_s^{(s_x,t_y)})} M_{(t_x,s_y)} \oplus M_{(s_x,t_y)} \xrightarrow{\rho^t_{(t_x,s_y)}-\rho^t_{(s_x,t_y)}} M_t. \tag{8}$$

Similarly, we can characterize weak exactness (Definition 5) algebraically:

▶ **Definition 11** (Algebraic weak exactness). *A persistence module $M\colon X \times Y \subseteq \mathbb{R}\times\mathbb{R} \to \mathsf{vec}$ is called* algebraically weakly exact *if the following equalities hold for all $s \leq t \in X \times Y$:*

$$\operatorname{Im}\rho_s^t = \operatorname{Im}\rho^t_{(t_x,s_y)} \cap \operatorname{Im}\rho^t_{(s_x,t_y)},$$
$$\operatorname{Ker}\rho_s^t = \operatorname{Ker}\rho_s^{(t_x,s_y)} + \operatorname{Ker}\rho_s^{(s_x,t_y)}.$$

This condition holds in particular when the sequence (8) is exact, but not only. Indeed, as can be checked easily, any rectangle (not just block) module is algebraically weakly exact. So is any rectangle-decomposable pfd persistence bimodule, since the property is obviously preserved under taking direct sums of pfd persistence bimodules. The converse holds as well:

▶ **Theorem 12** (Decomposition of algebraically weakly exact pfd bimodules). *For any algebraically weakly exact pfd persistence module M over a finite grid $(X \times Y, \leq)$, there is a unique multiset $\mathcal{R}M$ of rectangles of $X \times Y$ such that:*

$$M \cong \bigoplus_{R\in\mathcal{R}M} \mathbf{k}_R.$$

Since this result holds in particular for persistence bimodules indexed over squares, it ensures that a pfd persistence module over a square is algebraically weakly exact if, and only if, it is rectangle-decomposable. Hence the equivalence between weak exactness (Definition 5) and algebraic weak exactness (Definition 11), and the correctness of Theorem 7.

The rest of this section is devoted to the proof of Theorem 12. From this point on, and until the end of the section, whenever we talk about *weak exactness* we refer consistently to the algebraic formulation from Definition 11.

4.1 A preliminary remark concerning submodules and summands

A morphism $f : M \to N$ between two persistence modules over $(X\times Y, \leq)$ is a *monomorphism* (resp. *epimorphism*) if for every $t \in X \times Y$, $f_t : M_t \to N_t$ is injective (resp. surjective). We say that a monomorphism $f : M \to N$ between two persistence modules M and N *splits* if there is a morphism $g : N \to M$ such that $g \circ f = \mathrm{Id}_M$. If every monomorphism with domain M splits, we say that M is an *injective persistence module*.

It is not true that any submodule of a persistence module is a summand. However, if $f : M \to N$ is a monomorphism between two persistence modules M and N which splits, it is well known that there is a direct sum decomposition $N \cong M \oplus \mathrm{Coker}(f)$. Therefore, an injective submodule of a persistence module is a summand thereof. In our analysis we will often use the following result:

▶ **Lemma 13.** *For any indices $k \in [\![1,n]\!]$ and $l \in [\![1,m]\!]$, the indicator module $\mathbf{k}_{[\![1,k]\!]\times[\![1,l]\!]}$ is an injective persistence module over $[\![1,n]\!] \times [\![1,m]\!]$.*

Proof. This lemma is a consequence of [5, Lem. 2.1] since the subset $[\![1,k]\!] \times [\![1,l]\!]$ is clearly a directed ideal of the poset $[\![1,n]\!] \times [\![1,m]\!]$, following the definition of [5, Sec. 2.1]. ◀

4.2 Proof of Theorem 12

Uniqueness of the decomposition follows directly from Krull-Schmidt-Remak-Azumaya's theorem [1], since the endomorphism ring of any rectangle module is clearly isomorphic to $\mathbf{k}$ and thus local. We therefore focus on the existence of a decomposition into rectangle summands. Our proof proceeds by induction on the poset of grid dimensions (n,m), also viewed as a subposet of $\mathbb{R}^2$ equipped with the product order:

- Our base cases are when $n = 1$ or $m = 1$. The result is then a direct consequence of Gabriel's theorem [16], which asserts that M decomposes as a direct sum of interval modules, each interval being a rectangle of width 1.

- Fix $n > 1$ and $m > 1$, and assume that the result is true for all grids of sizes $n' \times m'$ such that $(n',m') < (n,m)$. Fix a persistence module M over $[\![1,n]\!] \times [\![1,m]\!]$ that is pfd and weakly exact. Observe that M has finite *total dimension* $\sum_{t\in[\![1,n]\!]\times[\![1,m]\!]} \dim M_t$, so we know from a simple induction that M decomposes as a direct sum of indecomposables – see [5, Theorem 1.1] for a more general statement. As any summand of a weakly exact module is again weakly exact, we may restrict our attention to pfd indecomposable modules. For the sake of contradiction, assume that M is pfd, weakly exact, indecomposable, and not isomorphic to a rectangle module. Then:

▶ **Lemma 14.** *The map $\rho_{(1,1)}^{(n,m)}$ is zero.*

Proof. Suppose the contrary. Then we have $\operatorname{Ker}\rho_{(1,1)}^{(n,m)} \subsetneq M_{(1,1)}$. Let $\alpha \in M_{(1,1)} \setminus \operatorname{Ker}\rho_{(1,1)}^{(n,m)}$. The submodule N of M spanned by the collection of images $(\rho_{(1,1)}^{(i,j)}(\alpha))_{(i,j)\in[\![1,n]\!]\times[\![1,m]\!]}$ is isomorphic to $\mathbf{k}_{[\![1,n]\!]\times[\![1,m]\!]}$, an injective persistence module by Lemma 13. Hence, N is a summand of M, contradicting that M is not isomorphic to a rectangle module. ◀

▶ **Lemma 15.** *The space $M_{(1,1)}$ maps injectively to the nodes of the grid $[\![1,n-1]\!]\times[\![1,m-1]\!]$.*

Proof. Let us restrict M to the grid $[\![1,n-1]\!] \times [\![1,m]\!]$. The restriction – denoted by N – may no longer be indecomposable, however it is still pfd and weakly exact, therefore our induction hypothesis asserts that N decomposes as a finite (internal) direct sum where each summand is isomorphic to some rectangle module. Consider any one of these summands, say $N' \cong \mathbf{k}_{R'}$, such that $(1,1) \in R'$. Then, we claim that $(n-1,1) \in R'$ as well. Indeed, otherwise, one can extend N' to a persistence module over $[\![1,n]\!] \times [\![1,m]\!]$ by putting zero spaces on the last column n. This yields an injective rectangle submodule of M (Lemma 13), and therefore a rectangle summand of M – a contradiction.

By our claim, $M_{(1,1)}$ maps injectively to the nodes $(i,1)$ for $i \in [\![1,n-1]\!]$. Similary, by restricting M to the grid $[\![1,n]\!] \times [\![1,m-1]\!]$, we deduce that $M_{(1,1)}$ maps injectively to the nodes $(1,j)$ for $j \in [\![1,m-1]\!]$. Then, by weak exactness, we have

$$\forall (i,j) \in [\![1,n-1]\!] \times [\![1,m-1]\!],\ \operatorname{Ker}\rho_{(1,1)}^{(i,j)} = \operatorname{Ker}\rho_{(1,1)}^{(i,1)} + \operatorname{Ker}\rho_{(1,1)}^{(1,j)} = 0,$$

so $M_{(1,1)}$ maps injectively to all the nodes of the grid $[\![1,n-1]\!] \times [\![1,m-1]\!]$. ◀

▶ **Lemma 16.** *The spaces $M_{(1,1)}$ and $M_{(n,m)}$ are zero.*

Proof. By weak exactness and Lemma 14, we have

$$M_{(1,1)} = \operatorname{Ker} \rho_{(1,1)}^{(n,m)} = \operatorname{Ker} \rho_{(1,1)}^{(n,1)} + \operatorname{Ker} \rho_{(1,1)}^{(1,m)}.$$

Assuming for a contradiction that $M_{(1,1)} \neq 0$, we have that at least one of the two terms on the right-hand side of the above equation must be non-zero – say $\operatorname{Ker} \rho_{(1,1)}^{(n,1)} \neq 0$. Let $\alpha \neq 0$ be an element in that kernel. By Lemma 15, its images at the nodes of $[\![1, n-1]\!] \times [\![1, m-1]\!]$ are non-zero. Meanwhile, its images at the nodes of $\{n\} \times [\![1, m]\!]$ are zero, by composition. There are two cases:

- Either $\rho_{(1,1)}^{(1,m)}(\alpha) = 0$, in which case the images of α at the nodes of $[\![1, n]\!] \times \{m\}$ are also zero, which implies that the persistence submodule of M spanned by the images of α is isomorphic to $\mathbf{k}_{[\![1,n-1]\!]\times[\![1,m-1]\!]}$.
- Or $\rho_{(1,1)}^{(1,m)}(\alpha) \neq 0$, in which case, for all $i \in [\![1, n-1]\!]$, we have

  $$\alpha \notin \operatorname{Ker} \rho_{(1,1)}^{(1,m)} \overset{\text{(Lemma 15)}}{=} \operatorname{Ker} \rho_{(1,1)}^{(1,m)} + \operatorname{Ker} \rho_{(1,1)}^{(i,1)} = \operatorname{Ker} \rho_{(1,1)}^{(i,m)},$$

  which implies that the images of α at the nodes of $[\![1, n-1]\!] \times \{m\}$ are non-zero as well. Hence, the persistence submodule of M spanned by the images of α is isomorphic to $\mathbf{k}_{[\![1,n-1]\!]\times[\![1,m]\!]}$.

In both cases, the persistence submodule of M spanned by the images of α is an injective rectangle module (Lemma 13), hence a rectangle summand of M – a contradiction.

By applying vector-space duality pointwise to M, we obtain an indecomposable module M^* of the grid $[\![1, n]\!]^{\mathrm{op}} \times [\![1, m]\!]^{\mathrm{op}}$ – which is isomorphic to $[\![1, n]\!] \times [\![1, m]\!]$ as a poset. This persistence module is still pfd, and still weakly exact as well since the equations of weak exactness are stable under vector-space duality (kernels become images, sums become intersections, and vice-versa). Hence, by the first part of the proof, $M^*_{(1,1)} = 0$, i.e the space at node (n, m) of M is zero, hence the result. ◀

▶ **Lemma 17.** *The space $M_{(1,m)}$ is zero.*

Proof. Assume for a contradiction that $M_{(1,m)} \neq 0$. Call N the restriction of M to the grid $[\![1, n-1]\!] \times [\![1, m]\!]$. By our induction hypothesis, N decomposes as a finite (internal) direct sum where each summand is isomorphic to some rectangle module. Since $M_{(1,m)} \neq 0$, at least one of these rectangles contains the node $(1, m)$. Among such rectangles, take one – say $R' = [\![1, i]\!] \times [\![j, m]\!]$ – that has lowest lower-left corner, and call N' the corresponding summand of N. Denote by N'' the rest of the internal decomposition of N, i.e. $N = N' \oplus N''$.

First, we claim that $i = n - 1$. Indeed, otherwise we can extend N' to a rectangle persistence submodule $\bar{N}'$ of M by putting zero spaces on the last column n, and N'' to another persistence submodule $\bar{N}''$ by putting the internal spaces of M on the last column, so that $M = \bar{N}' \oplus \bar{N}''$ – a contradiction.

Second, we claim that $j \in [\![2, m-1]\!]$. Indeed, $j \geq 2$ since by Lemma 16 we know that $M_{(1,1)} = 0$. Meanwhile, if j were equal to m, then N' would go to zero on the last column of $[\![1, n]\!] \times [\![1, m]\!]$ since $M_{(n,m)} = 0$ by Lemma 16, and so we could extend N to a rectangle persistence submodule $\bar{N}'$ of M by putting zero spaces on the last column, and N'' to another persistence submodule $\bar{N}''$ by putting the internal spaces of M on the last column, so that $M = \bar{N}' \oplus \bar{N}''$ – a contradiction.

Consider now the space $N_{(1,j)} = M_{(1,j)}$, and take a generator α of the subspace $N'_{(1,j)} \cong \mathbf{k}$. By Lemma 16 we know that the map $\rho^{(n,m)}_{(1,j)}$ is zero, so by weak exactness we have $\alpha = \alpha_h + \alpha_v$ for some $\alpha_h \in \operatorname{Ker} \rho^{(n,j)}_{(1,j)}$ and $\alpha_v \in \operatorname{Ker} \rho^{(1,m)}_{(1,j)}$. We claim that $\alpha_h \notin N''_{(1,j)}$. Indeed, otherwise we would have

$$\rho^{(1,m)}_{(1,j)}(\alpha) = \rho^{(1,m)}_{(1,j)}(\alpha_h) + \rho^{(1,m)}_{(1,j)}(\alpha_v) = \rho^{(1,m)}_{(1,j)}(\alpha_h) \in \rho^{(1,m)}_{(1,j)}(N''_{(1,j)}) \subseteq N''_{(1,m)},$$

thus contradicting our assumption that $N = N' \oplus N''$ with the support of N' containing $(1, m)$. Likewise, for any node $t \in R'$ we have $\rho^{t}_{(1,j)}(\alpha_h) \notin N''_t$, for otherwise we would get a contradiction from

$$\rho^{(t_x,m)}_{(1,j)}(\alpha) = \rho^{(t_x,m)}_{(1,j)}(\alpha_h) = \rho^{(t_x,m)}_{t}(\rho^{t}_{(1,j)}(\alpha_h)) \in \rho^{(t_x,m)}_{t}(N''_t) \subseteq N''_{(t_x,m)}.$$

Thus, the persistence submodule N^h of N generated by α_h is isomorphic[7] to N' and in direct sum with N''. We can therefore exchange N' for N^h in the internal decomposition of N. Since N^h is mapped to zero on the last column of $[\![1, n]\!] \times [\![1, m]\!]$, we can extend it to a rectangle persistence submodule $\bar{N}^h$ of M by putting zero spaces on the last column, meanwhile we can extend N'' to another persistence submodule $\bar{N}''$ by putting the internal spaces of M on the last column, so that $M = \bar{N}^h \oplus \bar{N}''$ – a contradiction. ◀

▶ **Lemma 18.** *$M_{(1,j)} = 0$ for all $j \in [\![1, m]\!]$.*

Proof. The result is already proven[8] for $j = m$ by Lemma 17. Let then $j \in [\![1, m-1]\!]$. Call N the restriction of M to the grid $[\![1, n]\!] \times [\![1, m-1]\!]$. By our induction hypothesis, N decomposes as a finite (internal) direct sum where each summand is isomorphic to some rectangle module. Assuming for a contradiction that some summand N' has a support R' that intersects the first column, we know from Lemma 17 that N' maps to zero at node $(1, m)$. By composition, N' maps to zero as well at the nodes on the last row m. Therefore, as in the proof of Lemma 17, we can extend N' to a rectangle summand of M by putting zero spaces on row m, thus reaching a contradiction. ◀

It follows from Lemma 18 that M itself is not supported outside the rectangle $R = [\![2, n]\!] \times [\![1, m]\!]$. The induction hypothesis (applied to the restriction of M to R) implies then that M decomposes as a direct sum of rectangle modules, which raises a contradiction. This concludes the induction step and the proof of Theorem 12.

5 Algorithm for checking rectangle decomposability

As in Section 3, let F be a simplicial bifiltration with n simplices in total, and let us assume without loss of generality that F is indexed over the grid $G = [\![1, n]\!] \times [\![1, n]\!]$. We further assume that F is 1-*critical*, and we fix a homology degree p.

Given this input, how fast can we check whether the persistence bimodule M induced in p-th homology decomposes into rectangle summands? An obvious solution is to first decompose M from the data of F, then to check the summands one by one. As explained in Section 3, the currently best known algorithm for decomposition runs in time $O(n^{2\omega+1})$, where $2 \leq \omega < 2.373$ is the exponent for matrix multiplication [13]. The advantage of the algebraic weak exactness condition from Section 4 is that it can be checked locally, which reduces the total running time to $O(n^{2+\omega})$. Below we sketch the algorithm:

[7] Note that we do not need to check that α_h goes to zero when leaving R', since by assumption R' reaches row m and, as we saw earlier, $i = n - 1$ so R' reaches column $n - 1$ as well.

[8] It is also proven for $j = 1$ by Lemma 16, although we do not use this fact in the proof.

1. Compute the rank invariant $r : [\![1,n]\!]^2 \times [\![1,n]\!]^2 \to \mathbb{N}$ of M.
2. Compute invariants for kernels and images, denoted by $\kappa : [\![1,n]\!]^2 \times [\![1,n]\!]^2 \to \mathbb{N}$ and $\iota : [\![1,n]\!]^2 \times [\![1,n]\!]^2 \to \mathbb{N}$ respectively, which return the dimensions of $\operatorname{Ker} \rho_s^{(s_x,t_y)} + \operatorname{Ker} \rho_s^{(t_x,s_y)}$ and of $\operatorname{Im} \rho_{(s_x,t_y)}^t \cap \operatorname{Im} \rho_{(t_x,s_y)}^t$ respectively at indices $s \leq t$, and zero elsewhere.
3. For each pair of indices $s \leq t$, check whether $r(s,t) = \iota(s,t)$ and $r(s,s) - r(s,t) = \kappa(s,t)$. If any such equality fails, then answer that M is not rectangle-decomposable. Otherwise, answer that M is rectangle-decomposable.

We now provide further implementation details and analyze the algorithm on the fly:

Step 1 has already been detailed in Section 3 and runs in $O(n^4)$ time.

Step 3 obviously runs in $O(n^4)$ time, and its correctness comes from the commutativity of the square in (4): indeed, commutativity implies that $\operatorname{Im} \rho_s^t \subseteq \operatorname{Im} \rho_{(s_x,t_y)}^t \cap \operatorname{Im} \rho_{(t_x,s_y)}^t$ and $\operatorname{Ker} \rho_s^{(s_x,t_y)} + \operatorname{Ker} \rho_s^{(t_x,s_y)} \subseteq \operatorname{Ker} \rho_s^t$, so checking weak exactness for this square amounts to checking equality between the dimensions of the various spaces involved, hence the equations.

For Step 2, we first compute, for each $t = (j,l) \in G$, the barcode of the zigzag module[9] induced in homology by the following zigzag of simplicial complexes:

$$F_{(1,l)} \longrightarrow \cdots \longrightarrow F_{(j-1,l)} \longrightarrow F_t \longleftarrow F_{(j,l-1)} \longleftarrow \cdots \longleftarrow F_{(j,1)} \ . \tag{9}$$

We then do the same with the following zigzag, for each $s = (i,k) \in G$:

$$F_{(i,n)} \longleftarrow \cdots \longleftarrow F_{(i,k+1)} \longleftarrow F_s \longrightarrow F_{(i+1,k)} \longrightarrow \cdots \longrightarrow F_{(n,k)} \ . \tag{10}$$

Then, for each indices $(i,k) = s \leq t = (j,l)$, by restriction, the dimension of $\operatorname{Im} \rho_{(i,l)}^t \cap \operatorname{Im} \rho_{(j,k)}^t$ is given by the number of intervals in the barcode of (9) that span the subzigzag $F_{(i,l)} \longrightarrow F_t \longleftarrow F_{(j,k)}$, while (dually) the dimension of $\operatorname{Ker} \rho_s^{(i,l)} + \operatorname{Ker} \rho_s^{(j,k)}$ is given by $r(s,s)$ minus the number of intervals in the barcode of (10) that span the subzigzag $F_{(i,l)} \longleftarrow F_s \longrightarrow F_{(j,k)}$ (the proof of these simple facts is given in [6, Appendix A]). Regarding the running time: since the zigzags (9)-(10) involve $O(n)$ simplex insertions and deletions each, their barcode computation takes $O(n^\omega)$ using the algorithm based on fast matrix multiplication [19]. Then, each barcode having $O(n)$ intervals, the computation of the dimensions and their storage in tables of integers representing the invariants κ and ι takes $O(n)$. This is true for each choice of indices $s \leq t$, hence a total running time in $O(n^{2+\omega} + n^3) = O(n^{2+\omega})$. As a consequence,

▶ **Theorem 19.** *Checking the rectangle-decomposability of the bimodule induced in homology by a 1-critical bifiltration with n simplices in total can be done in $O(n^{2+\omega})$ time.*

6 An example of rectangle-decomposable module

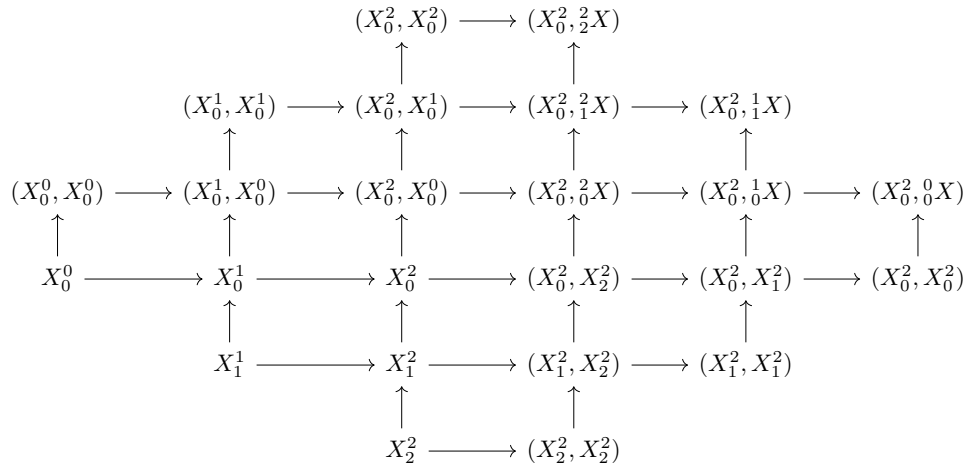
In [8] the authors show that a large pyramidal diagram can be associated to a sufficiently tame real valued function $f : X \to \mathbb{R}$. We briefly recall their construction. Under the assumption that the function is of *Morse type*, there exists a finite set of *critical values* $a_1 < a_2 < \ldots < a_n$, and we may choose real values s_i satisfying

$$-\infty < s_0 < a_1 < s_1 < \cdots < a_n < s_n < +\infty. \tag{11}$$

[9] A zigzag module is a persistence module indexed over a poset of the form $\bullet \longleftrightarrow \bullet \longleftrightarrow \cdots \longleftrightarrow \bullet$, where double-headed arrows mean that the actual arrows can be oriented either forward or backward. Such modules always decompose into direct sums of interval modules [4, 7].

Here the idea is that the preimage of $[s_i, s_{i+1}]$ deformation retracts onto the fiber over a_{i+1}, and that the fiber is constant (up to homotopy) between critical values. This gives a way of studying how the topology of the fibers connect across scales.

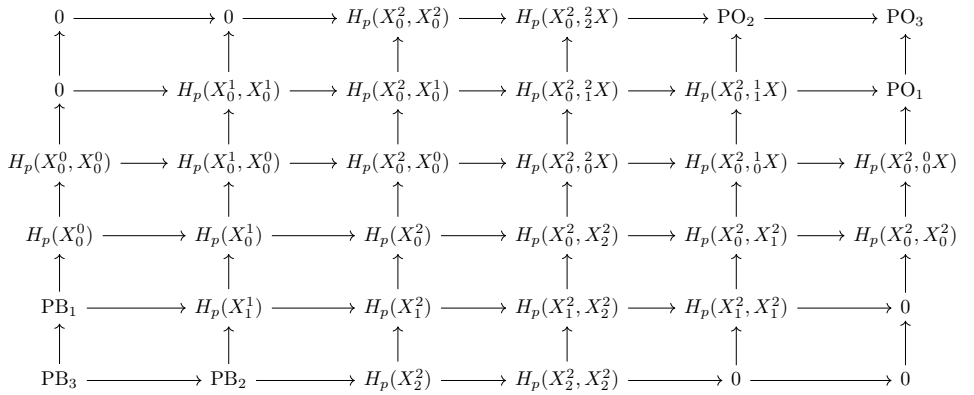
Denoting $X_i^j = f^{-1}[s_i, s_j]$ and ${}_i^jX = X_0^i \cup X_j^n$, obvious inclusions yield a commutative diagram, such as the following one for $n = 2$:



Building on the work of [8], it is shown in [3] that the above diagram, upon application of homology, decomposes into a direct sum of interval modules, where each interval is the intersection of a rectangle in $\mathbb{Z}^2$ with the pyramid above. This result is referred to as the *pyramid basis theorem*. We now give a new proof of this fact using Theorem 12. More precisely, we show the following:

▶ **Theorem 20** (Pyramid basis theorem)**.** *The homology pyramid as constructed in [8] is interval-decomposable, where the intervals are restrictions of rectangles in $\mathbb{Z}^2$ to the pyramid.*

To simplify the notation we prove the case for $n = 2$ and it will be evident that the argument generalizes. First, extend the homology diagram to a bimodule on a finite grid as follows:



Here PB_i denotes pullback and PO_i denotes pushout. Inductively these are defined (up to canonical isomorphism) by

$$\mathrm{PB}_1 = \ker\left(H_p(X_0^0) \oplus H_p(X_1^1) \to H_p(X_0^1)\right)$$
$$\mathrm{PB}_2 = \ker\left(H_p(X_1^1) \oplus H_p(X_2^2) \to H_p(X_1^2)\right)$$
$$\mathrm{PB}_3 = \ker\left(\mathrm{PB}_1 \oplus \mathrm{PB}_2 \to H_p(X_1^1)\right).$$

and dually for the pushouts, with kernels replaced by cokernels. By Theorem 12 it suffices to show that the extended diagram is weakly exact. The fact that any square with vertices on the original "pyramid" is strongly exact (i.e. the sequence (8) induced by such a square is exact) follows from the exactness of the relative Mayer–Vietoris sequence. Morever, as remarked in [5, Section 5.1], the extension of the pyramid to pullbacks and pushouts preserves strong exactness (and thus weak exactness). It remains to consider squares with a 0 vector space as either its top-left or bottom-right corner. The fact that such squares are weakly exact is an easy consequence of commutativity. We conclude that the bimodule shown above is weakly exact and therefore rectangle-decomposable. The restrictions of the rectangle summands to the original homology pyramid give the intervals in the *pyramid basis theorem*.

References

1 Gorô Azumaya. Corrections and supplementaries to my paper concerning Krull-Remak-Schmidt's theorem. *Nagoya Mathematical Journal*, 1:117–124, 1950.

2 Ulrich Bauer, Magnus B Botnan, Steffen Oppermann, and Johan Steen. Cotorsion torsion triples and the representation theory of filtered hierarchical clustering. *arXiv preprint*, 2019. arXiv:1904.07322.

3 Paul Bendich, Herbert Edelsbrunner, Dmitriy Morozov, Amit Patel, et al. Homology and robustness of level and interlevel sets. *Homology, Homotopy and Applications*, 15(1):51–72, 2013.

4 Magnus Bakke Botnan. Interval decomposition of infinite zigzag persistence modules. *Proceedings of the American Mathematical Society*, 145(8):3571–3577, 2017.

5 Magnus Bakke Botnan and William Crawley-Boevey. Decomposition of persistence modules. *To appear in the Proceedings of the AMS*, 2018. arXiv:1811.08946.

6 Magnus Bakke Botnan, Vadim Lebovici, and Steve Oudot. On rectangle-decomposable 2-parameter persistence modules, 2020. arXiv:2002.08894.

7 Gunnar Carlsson and Vin De Silva. Zigzag persistence. *Foundations of computational mathematics*, 10(4):367–405, 2010.

8 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.

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

10 Jérémy Cochoy and Steve Oudot. Decomposition of exact pfd persistence bimodules. *Discrete and Computational Geometry*, 2019. To appear, currently available as arXiv preprint arXiv:1605.09726.

11 David Cohen-Steiner, Herbert Edelsbrunner, and John Harer. Stability of persistence diagrams. *Discrete & Computational Geometry*, 37(1):103–120, 2007.

12 William Crawley-Boevey. Decomposition of pointwise finite-dimensional persistence modules. *Journal of Algebra and its Applications*, 14(05):1550066, 2015.

13 Tamal K Dey and Cheng Xin. Generalized persistence algorithm for decomposing multi-parameter persistence modules. *arXiv preprint arXiv:1904.03766*, 2019.

14 Herbert Edelsbrunner and John Harer. Persistent homology-a survey. *Contemporary mathematics*, 453:257–282, 2008.

15 Emerson G Escolar and Yasuaki Hiraoka. Persistence modules on commutative ladders of finite type. *Discrete & Computational Geometry*, 55(1):100–157, 2016.

16 Peter Gabriel. Unzerlegbare Darstellungen I. *manuscripta mathematica*, 6(1):71–103, March 1972. doi:10.1007/BF01298413.

17 Woojin Kim and Facundo Memoli. Generalized persistence diagrams for persistence modules over posets. *arXiv preprint arXiv:1810.11517*, 2018.

18 Michael Lesnick and Matthew Wright. Computing minimal presentations and betti numbers of 2-parameter persistent homology. *arXiv preprint arXiv:1902.05708*, 2019.

19 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.

20 Steve Y Oudot. *Persistence theory: from quiver representations to data analysis*, volume 209. American Mathematical Society Providence, RI, 2015.

Robust Anisotropic Power-Functions-Based Filtrations for Clustering

Claire Brécheteau
Laboratoire de Mathématiques Jean Leray & École Centrale de Nantes, France
claire.brecheteau@ec-nantes.fr

Abstract

We consider robust power-distance functions that approximate the distance function to a compact set, from a noisy sample. We pay particular interest to robust power-distance functions that are anisotropic, in the sense that their sublevel sets are unions of ellipsoids, and not necessarily unions of balls. Using persistence homology on such power-distance functions provides robust clustering schemes. We investigate such clustering schemes and compare the different procedures on synthetic and real datasets. In particular, we enhance the good performance of the anisotropic method for some cases for which classical methods fail.

2012 ACM Subject Classification Theory of computation → Unsupervised learning and clustering

Keywords and phrases Power functions, Filtrations, Hierarchical Clustering, Ellipsoids

Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.23

Related Version A full version of the paper is available at `https://hal.archives-ouvertes.fr/hal-02397100`.

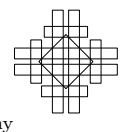
Supplementary Material At `https://hal.archives-ouvertes.fr/hal-02397100`, the source code is available, as an annex file.

Acknowledgements I am extremely grateful to Samuel Tapie, for his suggestion to use tangency of ellipsoids at their first intersection point, to derive the expression of their intersection radius.

1 Introduction

Often data can be represented as a point cloud $\mathbb{X}$ in a Euclidean space $\mathbb{R}^d$. Grouping data into clusters as homogeneous and well-separated as possible is the purpose of clustering. When no label is know in advance, we talk about unsupervised clustering. Topological data analysis (TDA) tools are designed to understand the shape of the data. Thereby, such tools may help to understand the shape of clusters in which to group the data. In this paper, we develop and study a TDA-based unsupervised clustering scheme. In addition, our method detects and removes points that do not really belong to any cluster; the outliers.

Clustering datasets is of extreme importance in multiple domains including medicine and social networks among others. The classical k-means method clusters data into isotropic clusters. In particular, the trimmed version of k-means of [14] that removes outliers, supplies balls-shaped clusters. These two algorithms have been extended by [2, 5] for Bregman-balls-shaped clusters, see also tclust [17] for ellipsoidal clusters. Such methods are well-suited for data generated according to mixtures of distributions which sublevel-set are Bregman balls themselves. For more general datasets, for instance, a sample of point from a disconnected manifold, these methods are no longer appropriate. Spectral clustering methods [27] perform such tasks, but are not robust to outliers. DBSCAN [19] is an algorithm based on a fixed upper-level set of an approximation of the density, and consequently, does not provide a multiscale information. Via a dendrogram, the classical single-linkage hierarchical clustering algorithm provides such a multiscale information. The dendrogram encodes information about the connectivity of unions of balls centered at points in $\mathbb{X}$, or equivalently, of the sublevel




sets of the distance function to $\mathbb{X}$. For a fixed radius r, the Čech complex is a simplicial complex defined as the collection of simplices (vertex, edge, triangle, tetrahedron) for which the r-balls centered at the vertices have a non-empty common intersection. We call 1-skeleton its subcomplex (a graph) that contains only vertices and edges. The non-decreasing family of such graphs indexed by $r \in \mathbb{R}$ is called a filtration. Single-linkage is a persistence-based method since is based on the persistence, prominence or equivalently lifetime of the connected components into this graph filtration, however, it is not robust to outliers. The algorithm ToMATo in [12] is robust and persistence-based. Indeed, it is based on a graph filtration built from a neighborhood graph and a (robust) distance-like function whose values guide the appearance of vertices and edges in the graph filtration. An example of robust distance function that Chazal et al. consider in [12] is given by the distance-to-measure (DTM) [10]. Note that the graph is a priori not intrinsic to the distance function, which may cause bad clustering. For instance, an edge that links two vertices with small distance-function value but intersects an area with large distance function value, may link two clusters that should not be. This problem was the cause of failure of the single-linkage method for data corrupted by outliers. Alternative filtrations that do not suffer from this problem are the DTM-filtration [1], or the power filtrations [7], based on the 1-skeleton of the Čech filtration associated to the sublevel sets of a power distance function: a function of type $x \mapsto \min_{i\in I} \|x - m_i\|^2 + \omega_i$ for some $(m_i)_{i\in I}$ in $\mathbb{R}^d$ and $(\omega_i)_{i\in I}$ in $\mathbb{R}$. Some approximations of the DTM that are power functions have been introduced and studied in the literature: the k-witnessed distance [18], the power distance [7], the c-PDTM [6] whose sublevel sets are unions of c balls, and the c-PLM [4] whose sublevel sets are unions of c ellipsoids, with c possibly much smaller than the sample size. The last two functions are robust to outliers since their construction is based on the principle of trimmed least squares [26].

Contributions

By replacing balls with ellipsoids, we enlarge the notion of weighted Čech filtration into the anisotropic weighted Čech filtration. We derive an expression for the radius of intersection of two ellipsoids. We introduce a clustering algorithm based on persistence. Such a clustering algorithm can be run from any graph filtration, in particular, from the 1-skeleton of the anisotropic weighted Čech filtration, which corresponds to the filtration of sublevel sets of an anisotropic power function. We experiment this algorithm on the filtration of the c-PLM [4].

Practical interests

A clustering algorithm based on the persistence filtration of the sublevel sets of a power function is pertinent since unlike ToMATo, the graph is intrinsic to the distance function. So, no additional parameters are required for the algorithm. The main advantage of using an anisotropic power function is that its sublevel sets are ellipsoids. Much less ellipsoids are required than balls to Hausdorff-approximate a compact manifold with intrinsic dimension smaller than the ambient dimension. The clustering scheme can also be applied to decompose a set of points generated on a polygonal line into segments. Once the ellipsoids computed, the persistence algorithm runs fast. Its complexity in terms of number of comparisons is at worst $O(c^4)$, with c, the number of ellipsoids, which is much smaller than the sample size. Most importantly, the robustness of the persistence algorithm relies on the robustness of the distance function. The c-PLM [4] is robust to outliers. The guaranty for the clustering method follows from the $\|\cdot\|_\infty$-distance closeness between the power distance function and the distance function to the underlying manifold $\mathcal{X}$, relatively to the minimal distance between the connected components of $\mathcal{X}$. Note that such a proximity condition is sufficient but not necessary, as illustrated by the different numerical examples, with the c-PLM.

Organisation of the paper

In Section 2, we recall the notions of power function and weighted Čech filtration, the filtration of the nerves of its sublevel sets, that we extend to anisotropic power functions. We prove some stability and approximation properties for such filtrations. Examples of robust power filtrations are also displayed. The main clustering algorithm, Algorithm 1 is given in Section 3. This algorithm applies to any filtration of graphs, including the graph filtrations obtained as the 1-skeleton of a weighted Čech filtration. We enumerate other types of filtrations that fit into this framework. Finally, we implement Algorithm 1 with the robust anisotropic aforementioned power function in Section 4. We compare this method to other clustering methods on synthetic and real datasets.

2 Power-functions-based filtrations for robust clustering

In the sequel, we will recall the notion of filtration for subsets of $\mathbb{R}^d$ (equipped with the Euclidean norm $\|\cdot\|$) and for simplicial complexes. We will consider a class of functions for which filtrations associated to sublevel sets are easily represented by filtrations of simplicial complexes, making the evolution of their connected components tractable: the power functions. In addition, we will give an example of robust power-functions [6] that can be built from a probability distribution or a pointset $\mathbb{X}$. Their sublevel sets are unions of c balls, with c possibly much smaller than the size of $\mathbb{X}$. Most importantly, we will also give an example of a robust anisotropic power-function, whose sublevel sets are unions of c ellipsoids [4]. Both of these power functions will be considered in the next sections for clustering purposes.

2.1 Generalities on filtrations

A filtration indexed by a time set $T \subset \mathbb{R}$ is a family $(V^t)_{t\in T}$ of subsets of $\mathbb{R}^d$, non-decreasing for the inclusion (i.e. $\forall t \leq t', V^t \subset V^{t'}$). A typical example is the filtration of the sub-level sets of a function $f : \mathbb{R}^d \mapsto \mathbb{R}$, $\left(f^{-1}((-\infty, t])\right)_{t\in T}$. For any simplex S with finite vertex set $\mathbb{X}$, a filtration of simplicial complexes of S is a non-decreasing family $(S^t)_{t\in T}$ of subcomplexes of S, meaning that for every $t \leq t'$, any simplex of S^t is also a simplex of $S^{t'}$.

The interleaving pseudo-distance between two filtrations $(V^t)_{t\in T}$ and $(W^t)_{t\in T}$ is defined as the smallest $\epsilon > 0$ such that $(V^t)_{t\in T}$ and $(W^t)_{t\in T}$ are ϵ-interleaved, i.e. such that: $\forall t \in T$, $V^t \subset W^{t+\epsilon}$ and $W^t \subset V^{t+\epsilon}$. This definition extends to simplicial complexes. Note that the sub-level-sets filtrations of two functions f and g satisfying $\|f - g\|_\infty \leq \epsilon$ are ϵ-interleaved. We will see in Section 3 that the notion of interleaving is primordial, since it measures the difference of topology between two filtrations. In particular, the stability of our sub-level-sets-based clustering scheme will be guarantied from the closeness of the functions.

2.2 Power-functions-based filtrations

In this paper, we consider classes of functions whose sub-level sets filtration has a sparse representation, the power functions. The sublevel sets of these functions can be represented by simplicial complexes in so-called weighted Čech filtrations. We will consider two types of power functions, the isotropic and the anisotropic ones.

2.2.1 The isotropic case

An isotropic power function is a function $f_{\mathbf{m},\boldsymbol{\omega}} : \mathbb{R}^d \to \mathbb{R}$ defined from an index set $I = [\![1, c]\!]$, a family of centers $\mathbf{m} = (m_i)_{i\in I}$ in $\mathbb{R}^d$ and a family of weights $\boldsymbol{\omega} = (\omega_i)_{i\in I}$ in $\mathbb{R}$ by $f_{\mathbf{m},\boldsymbol{\omega}} : x \mapsto \min_{i\in I} \|x - m_i\|^2 + \omega_i$. A simple example of power function is the squared

Euclidean distance function to a set of points $\mathbb{X}$, $\mathrm{d}^2_{\mathbb{X}} : x \in \mathbb{R}^d \mapsto \min_{m\in\mathbb{X}} \|x-m\|^2$. The sublevel sets of $f_{\mathbf{m},\boldsymbol{\omega}}$, $V^t_{\mathbf{m},\boldsymbol{\omega}} = f^{-1}_{\mathbf{m},\boldsymbol{\omega}}((-\infty,t])$, are unions of at most c balls $\mathcal{B}^t_i = \overline{\mathrm{B}}(m_i, \sqrt{t-\omega_i})$ with $\overline{\mathrm{B}}(m,r) = \{x \in \mathbb{R}^d \mid \|x-m\| \leq r\}$. Note that $\mathcal{B}^t_i$ is empty for $t < \omega_i$ and two balls $\mathcal{B}^t_i$ and $\mathcal{B}^t_j$ intersect if and only if $t \geq t_{i,j}$ with $t_{i,j} = \frac{(\omega_j-\omega_i)^2+2(\omega_j+\omega_i)\|m_j-m_i\|^2+\|m_j-m_i\|^4}{4\|m_j-m_i\|^2}$. The connectivity of $V^t_{\mathbf{m},\boldsymbol{\omega}}$ can be encoded in a graph $\mathcal{G}^t_{\mathbf{m},\boldsymbol{\omega}}$, whose vertices are indices $i \in I$ such that $\omega_i \leq t$ and whose edges are pairs of vertices $[i,j]$ such that $t_{i,j} \leq t$. Indeed, $\mathcal{G}^t_{\mathbf{m},\boldsymbol{\omega}}$ and $V^t_{\mathbf{m},\boldsymbol{\omega}}$ have the same number of connected components, and m_i and m_j are in the same connected component in $V^t_{\mathbf{m},\boldsymbol{\omega}}$ if and only if i and j are also in the same component in $\mathcal{G}^t_{\mathbf{m},\boldsymbol{\omega}}$.

More generally, the topological information of $V^t_{\mathbf{m},\boldsymbol{\omega}}$ (number of connected components, loops, voids etc.) can be encoded in the weighted Čech complex $\mathrm{Cech}_{\mathbf{m},\boldsymbol{\omega}}(t)$, defined as the nerve of the union of balls $(\mathcal{B}^t_i)_{i\in I}$: $\mathrm{Cech}_{\mathbf{m},\boldsymbol{\omega}}(t) = \{\sigma \subset I \mid \bigcap_{i\in\sigma} \mathcal{B}^t_i \neq \emptyset\}$, [1, 7, 3]. According to the Nerve Lemma [20, Corollary 4G.3], any sublevel set $V^t_{\mathbf{m},\boldsymbol{\omega}}$ is homotopic to $\mathrm{Cech}_{\mathbf{m},\boldsymbol{\omega}}(t)$ and thus contains the same topological information. For computational reasons, the weighted Vietoris-Rips filtration is frequently considered as a provably good surrogate for the weighted Čech filtration $(\mathrm{Cech}_{\mathbf{m},\boldsymbol{\omega}}(t))_{t\in T}$. The weighted Vietoris-Rips complex $\mathrm{VR}_{\mathbf{m},\boldsymbol{\omega}}(t)$ is the flag complex of $\mathcal{G}^t_{\mathbf{m},\boldsymbol{\omega}}$ ($\mathcal{G}^t_{\mathbf{m},\boldsymbol{\omega}}$ is the 1-skeleton of the weighted Čech complex): $\mathrm{VR}_{\mathbf{m},\boldsymbol{\omega}}(t) = \{\sigma \subset I \mid \forall i,j \in \sigma, \mathcal{B}^t_i \cap \mathcal{B}^t_j \neq \emptyset\}$. Indeed, as a direct consequence of [3, Theorem 3.2] which is a generalization of the non-weighted case in [15, Theorem 2.5.], if the weights in $\boldsymbol{\omega}$ are non-negative, then these two filtrations are interleaved:

$$\forall 0 < t' \leq \frac{d+1}{2d}t, \mathrm{VR}_{\mathbf{m},\boldsymbol{\omega}}(t') \subset \mathrm{Cech}_{\mathbf{m},\boldsymbol{\omega}}(t) \subset \mathrm{VR}_{\mathbf{m},\boldsymbol{\omega}}(t). \tag{1}$$

These notions can all be extended to anisotropic power functions.

2.2.2 The anisotropic case

Consider $I = [\![1,c]\!]$, centers $\mathbf{m} = (m_i)_{i\in I}$ in $\mathbb{R}^d$, weights $\boldsymbol{\omega} = (\omega_i)_{i\in I}$ in $\mathbb{R}$ and matrices $\boldsymbol{\Sigma} = (\Sigma_i)_{i\in I}$ in $\mathcal{M}_d$, the set of definite positive symmetric matrices. An anisotropic power function is a function $f_{\mathbf{m},\boldsymbol{\omega},\boldsymbol{\Sigma}} : \mathbb{R}^d \to \mathbb{R}$ defined from I, $\mathbf{m}$, $\boldsymbol{\omega}$ and $\boldsymbol{\Sigma}$ by $f_{\mathbf{m},\boldsymbol{\omega},\boldsymbol{\Sigma}} : x \mapsto \min_{i\in I} \|x-m_i\|^2_{\Sigma_i^{-1}} + \omega_i$. For any matrix $\Sigma \in \mathcal{M}_d$ and $x \in \mathbb{R}^d$, $\|x\|_{\Sigma^{-1}} = \sqrt{x^T\Sigma^{-1}x}$ denotes the Σ-Mahalanobis norm of x. The sublevel sets of $f_{\mathbf{m},\boldsymbol{\omega},\boldsymbol{\Sigma}}$, $V^t_{\mathbf{m},\boldsymbol{\omega},\boldsymbol{\Sigma}} = f^{-1}_{\mathbf{m},\boldsymbol{\omega},\boldsymbol{\Sigma}}((-\infty,t])$, are unions of at most c ellipsoids $\mathcal{E}^t_i = \overline{\mathrm{B}}_{\Sigma_i}(m_i, \sqrt{t-\omega_i}) = \{x \in \mathbb{R}^d \mid \|x-m_i\|^2_{\Sigma_i^{-1}} \leq t-\omega_i\}$. Again, $\mathcal{E}^t_i$ is empty for $t < \omega_i$ and the intersection time $t_{i,j}$ of $\mathcal{E}^t_i$ and $\mathcal{E}^t_j$ is given below. The relative question of the emptiness of the intersection of two ellipsoids is tackled in [28, 25].

▶ **Proposition 1.** *Consider two ellipsoids $\mathcal{E}^t_i = \overline{\mathrm{B}}_{\Sigma_i}(m_i, \sqrt{t-\omega_i})$ and $\mathcal{E}^t_j = \overline{\mathrm{B}}_{\Sigma_j}(m_j, \sqrt{t-\omega_j})$ with $\omega_i \leq \omega_j$ in $\mathbb{R}$, m_i and m_j in $\mathbb{R}^d$, $\Sigma_i = P_iD_iP_i^T$ and $\Sigma_j = P_jD_jP_j^T$ in $\mathcal{M}_d$, with two positive diagonal matrices D_i and D_j and two orthogonal matrices P_i and P_j from the spectral theorem. Set $\tilde{\Sigma} = \sqrt{D_i}P_i^T\Sigma_j^{-1}P_i\sqrt{D_i} = \tilde{P}\tilde{D}\tilde{P}^T$, for orthogonal and diagonal matrices $\tilde{P}$ and $\tilde{D} = diag(\lambda_1, \lambda_2, \ldots, \lambda_d)$, and $\tilde{m} = \tilde{P}^T\sqrt{D_i^{-1}}P_i^T(m_j - m_i)$. Ellipsoids $\mathcal{E}^t_i$ and $\mathcal{E}^t_j$ intersect if and only if $t \geq t_{i,j}$ for $t_{i,j} = \omega_j$ when $\|\tilde{m}\| \leq \sqrt{\omega_j - \omega_i}$, and $t_{i,j} = \omega_j + \sum_{k=1}^d \left(\frac{\lambda\tilde{m}_k}{\lambda+\lambda_k}\right)^2 \lambda_k$ when $\|\tilde{m}\| > \sqrt{\omega_j - \omega_i}$. The positive number λ is the unique solution of the following equation:*

$$\sum_{k=1}^d \frac{\lambda_k - \lambda^2}{(\lambda+\lambda_k)^2}\lambda_k\tilde{m}_k^2 = \omega_j - \omega_i. \tag{2}$$

The proof is based on the fact that the ellipsoids $\mathcal{E}_i^t$ and $\mathcal{E}_j^t$ are tangent at their first intersection point, and the corresponding gradients are collinear. In the context of isotropy (i.e. for $\Sigma_i = \Sigma_j = I_d$, the identity matrix of $\mathbb{R}^d$) $\tilde{m} = m_j - m_i$, and when $\|m_j - m_i\| > \sqrt{\omega_j - \omega_i}$, (2) has a unique positive solution given by $\lambda = \frac{\omega_i - \omega_j + \|m_j - m_i\|^2}{\omega_j - \omega_i + \|m_j - m_i\|^2}$. We recover the merging time $t_{i,j}$ given in Section 2.2.1. Now, define $\mathcal{G}^t_{\mathbf{m},\boldsymbol{\omega},\boldsymbol{\Sigma}}$, $\mathrm{Cech}_{\mathbf{m},\boldsymbol{\omega},\boldsymbol{\Sigma}}(t)$ and $\mathrm{VR}_{\mathbf{m},\boldsymbol{\omega},\boldsymbol{\Sigma}}(t)$, the anisotropic counterparts of $\mathcal{G}^t_{\mathbf{m},\boldsymbol{\omega}}$, $\mathrm{Cech}_{\mathbf{m},\boldsymbol{\omega}}(t)$ and $\mathrm{VR}_{\mathbf{m},\boldsymbol{\omega}}(t)$. The nerve lemma still applies, since unions of ellipsoids are contractible. Although this paper is mostly based on the study of connected components for clustering, anisotropic weighted Čech and Vietoris-Rips filtrations are primordial to have a tractable estimation of the topology of compact sets from suitable approximations as finite unions of ellipsoids. In fact, as their isotropic counterparts (1), these filtrations are interleaved, provided that the eigenvalues of the matrices in $\boldsymbol{\Sigma}$ are positive.

▶ **Proposition 2.** *If $\boldsymbol{\omega}$ is a set on non-negative weights in $\mathbb{R}$ and $\boldsymbol{\Sigma}$ a family of matrices with eigenvalues in $[\lambda_{\min}, \lambda_{\max}]$ for some $\lambda_{\min} > 0$, then for every $t > 0$ and $0 < t' \leq \frac{\lambda_{\min}}{\lambda_{\max}} \frac{d+1}{2d} t$,*

$$\mathrm{VR}_{\mathbf{m},\boldsymbol{\omega},\boldsymbol{\Sigma}}(t') \subset \mathrm{Cech}_{\mathbf{m},\boldsymbol{\omega},\boldsymbol{\Sigma}}(t) \subset \mathrm{VR}_{\mathbf{m},\boldsymbol{\omega},\boldsymbol{\Sigma}}(t). \tag{3}$$

The condition of non-negative weights is not too restrictive since for general weights, it suffices to replace $\boldsymbol{\omega}$, t and t' by $\boldsymbol{\omega} - \min_{i\in I} \omega_i$, $t - \min_{i\in I} \omega_i$ and $t' - \min_{i\in I} \omega_i$ in the proposition. Then, the condition on t' becomes $\min_{i\in I} \omega_i < t' \leq \frac{\lambda_{\min}}{\lambda_{\max}} \frac{d+1}{2d} t + \left(1 - \frac{\lambda_{\min}}{\lambda_{\max}} \frac{d+1}{2d}\right) \min_{i\in I} \omega_i$. As noted in [15], when $\lambda_{\min}$ equals $\lambda_{\max}$ and the weights in $\boldsymbol{\omega}$ are null, the term $\frac{\lambda_{\min}}{\lambda_{\max}} \frac{d+1}{2d}$ is optimal. When $\mathbf{m}$ is the set of vertices of a regular d-simplex, the left inclusion is an equality.

Often, less ellipsoids than balls are required to describe a compact set $\mathcal{X}$, for a fixed level of precision (e.g. for the Hausdorff distance). For instance, a segment in $\mathbb{R}^2$, and more generally, any d'-dimensional submanifold in $\mathbb{R}^d$, with $d' < d$. For this reason, anisotropic Čech and Vietoris-Rips filtrations are pertinent tools to compute and store the topological information about $\mathcal{X}$ efficiently. The requisite condition is that we dispose of an anisotropic power function that is a good approximation of $\mathrm{d}^2_{\mathcal{X}}$. Such examples of functions follow.

2.3 Examples of filtrations based on robust power functions

2.3.1 Isotropic robust power functions

Set $\mathbb{X}$, a set of n points generated on the neighborhood of a compact subset $\mathcal{X}$ of $\mathbb{R}^d$. In order to face the non robustness of the distance function to $\mathbb{X}$, $\mathrm{d}_{\mathbb{X}}$, Chazal et al. have introduced the notion of distance-to-measure (DTM), in [10]. The DTM is a counterpart of $\mathrm{d}_{\mathbb{X}}$ robust to noise and outliers. Its robustness follows from some parameter $k \in [\![1, n]\!]$, the number of nearest-neighbors $X^1, X^2, \ldots, X^k$ of x in $\mathbb{X}$, used to estimate $\mathrm{d}_{\mathbb{X}}(x)$. The DTM $\mathrm{d}_{\mathbb{X},k}$ is defined by $\mathrm{d}^2_{\mathbb{X},k} : x \mapsto \frac{1}{k}\sum_{i=1}^k \|x - X^i\|^2 = \|x - m_{x,k}\|^2 + v_{x,k}$ with $m_{x,k} = \sum_{i=1}^k X^i$, the mean of the k nearest neighbours of x in $\mathbb{X}$ and $v_{x,k} = \frac{1}{k}\sum_{i=1}^k \|X^i - m_{x,k}\|^2$ their variance. Note that $\mathrm{d}_{\mathbb{X},1}$ coincides with $\mathrm{d}_{\mathbb{X}}$ and is not robust, whereas $\mathrm{d}_{\mathbb{X},n}(x)$ is the distance of x to the barycenter of the point cloud $\mathbb{X}$, up to some factor, which is robust, but very poor in terms of topological information. The DTM is actually a weighted power function [18]:

$$\mathrm{d}^2_{\mathbb{X},k}(x) = \inf_{y\in\mathbb{R}^d} \|x - m_{y,k}\|^2 + v_{y,k}. \tag{4}$$

This follows from the fact that the mean distance between x and its k nearest neighbors is not larger than the mean distance between x and the k nearest neighbors of any other point $y \in \mathbb{R}^d$. This infimum is actually a minimum over a set of c points $\mathbf{y} = (y_i)_{i\in[\![1,c]\!]}$ in $\mathbb{R}^d$, with

c of order $\binom{n}{k}$. A power approximation of the DTM, the k-witnessed distance, was defined in [18] by replacing $\mathbb{R}^d$ by $\mathbb{X}$ in (4). Its sublevel sets are unions of n balls. An approximation of the DTM with c (possibly much smaller than n) balls, the c-PDTM, was defined in [6], by replacing $\mathbb{R}^d$ by a set $\mathbf{y}_{c,k}$ of c points in $\mathbb{R}^d$. This set $\mathbf{y}_{c,k}$ is a minimum of a "k-means"-type criterion [24], $\mathbf{y} \mapsto \sum_{i=1}^{n} \min_{y\in\mathbf{y}} \|X_i - m_{y,k}\|^2 + v_{y,k}$, for $\mathbf{y}$ with cardinality c. Morally, $\mathbf{y}_{c,k}$ is chosen such that on average on $\mathbb{X}$, $x \mapsto \min_{y\in\mathbf{y}} \|x - m_{y,k}\|^2 + v_{y,k}$ is small. Note that the graph of the c-PDTM is necessarily above the graph of the DTM. According to [6], for a sample on a regular d'-dimensional manifold, c can be chosen of order $n^{\frac{d'}{d'+4}}$, which is much smaller than n. Moreover, the c-PDTM is a good approximation of $\mathrm{d}^2_{\mathcal{X}}$, despite noise.

2.3.2 An anisotropic robust power function

An anisotropic version of the c-PDTM has been introduced in [4], the c-power likelihood to measure (c-PLM). It consists in replacing Euclidean norms with Mahalanobis norms. For every $x \in \mathbb{R}^d$ and $\Sigma \in \mathcal{M}_d$, set X^1, X^2,... X^k the k-nearest neighbors of x in $\mathbb{X}$, for the Σ^{-1}-Mahalanobis norm: $\|X^i - x\|_{\Sigma^{-1}} \leq \|X^j - x\|_{\Sigma^{-1}}$ for every $i \leq j$. Denote by $m_{x,\Sigma,k}$ their mean, and by $v_{x,\Sigma,k} = \frac{1}{k}\sum_{i=1}^{k} \|X^i - m_{x,\Sigma,k}\|^2_{\Sigma^{-1}}$ their variance, relative to the Σ-Mahalanobis norm. Set $\boldsymbol{\theta}_{c,k}$, a family of c pairs $(y, \Sigma) \in \mathbb{R}^d \times \mathcal{M}_d$ that minimizes (or which criterion is as close as possible to the optimal criterion, in case of non existence of a minimum) the following "k-means"-type criterion $R_{c,k}$ among all $\boldsymbol{\theta}$s of cardinality c: $R_{c,k}(\boldsymbol{\theta}) = \sum_{i=1}^{n} \min_{(y,\Sigma)\in\boldsymbol{\theta}} \|X_i - m_{y,\Sigma,k}\|^2_{\Sigma^{-1}} + v_{y,\Sigma,k} + \log(\det(\Sigma))$. The term $\log(\det(\Sigma))$ prevents optimal covariance matrices to be degenerated, with Σ^{-1} going to 0. In some sense, minimizing such a criterion boils down to fit Gaussian distributions to the data set $\mathbb{X}$, at best. The c-PLM is the power function defined from $\boldsymbol{\theta}_{c,k}$ by: $x \mapsto \min_{(y,\Sigma)\in\boldsymbol{\theta}_{c,k}} \|x - m_{y,\Sigma,k}\|^2_{\Sigma^{-1}} + v_{y,\Sigma,k} + \log(\det(\Sigma))$. A modification of the criterion $R_{c,k}$ has been introduced in [4], to remove some datapoints ($|\mathbb{X}| - sig$ for some parameter sig), when $\mathbb{X}$ is corrupted with outliers. The criterion is given by $R_{c,k,sig}(\boldsymbol{\theta}) = \min_{(i_1,i_2,\ldots,i_{sig})\in[\![1,|\mathbb{X}|]\!]} \sum_{j=1}^{sig} \min_{(y,\Sigma)\in\boldsymbol{\theta}} \|X_{i_j} - m_{y,\Sigma,k}\|^2_{\Sigma^{-1}} + v_{y,\Sigma,k} + \log(\det(\Sigma))$.

Iterative Lloyd-type algorithms [22] provide local minima $\tilde{\boldsymbol{\theta}}_{c,k}$ and $\tilde{\boldsymbol{\theta}}_{c,k,sig}$ for the criteria $R_{c,k}$ and $R_{c,k,sig}$ [4]. These algorithms run in $O(ncd^2 + nkd^2 + n\log(n)c)it$ operations, with it the number of iterations of the algorithm. They consist, given $\boldsymbol{\theta} = (\mathbf{y}, \boldsymbol{\Sigma})$, in splitting the space $\mathbb{R}^d$ into weighted $\boldsymbol{\Sigma}$-curved Voronoi cells, replacing centers $\mathbf{y}$ by the centroid of the cells, and updating the matrices in $\boldsymbol{\Sigma}$ by a close formula from the points in the cells and ellipsoids. To compute $\tilde{\boldsymbol{\theta}}_{c,k,sig}$, a trimming step is added at each iteration. For clustering, disposing of a local minimum is enough, as enhanced in the numerical illustration section, since we can remove bad centers in $\tilde{\boldsymbol{\theta}}_{c,k}$ or in $\tilde{\boldsymbol{\theta}}_{c,k,sig}$ with the parameter *Threshold* in Algorithm 1.

3 Persistence-based clustering from power-functions-based filtrations

3.1 Persistence for power-functions-based filtrations

Set $f_{\mathbf{m},\boldsymbol{\omega},\boldsymbol{\Sigma}} : x \in \mathbb{R}^d \mapsto \min_{i\in I} \|x - m_i\|^2_{\Sigma_i^{-1}} + \omega_i$, an anisotropic power-function indexed by a set $I = [\![1, c]\!]$ and with the ω_is sorted in non-decreasing order. As above-mentioned, the sublevel sets $V^t = f^{-1}_{\mathbf{m},\boldsymbol{\omega},\boldsymbol{\Sigma}}((-\infty, t])$ are unions of at most c ellipsoids $\mathcal{E}_i^t = B_{\Sigma_i}(m_i, \sqrt{t - \omega_i})$, non empty as soon as $t \geq \omega_i$. In particular, each sublevel set of $f_{\mathbf{m},\boldsymbol{\omega},\boldsymbol{\Sigma}}$ contains at most c connected components. Each connected component of V^t, V_i^t is indexed by the smallest index $i \in I$ such that m_i belongs to the component. With a language abuse, we call connected component V_i, the family of connected components $(V_i^t)_{t\in T}$ that gets born at time $t = b_i = \omega_i$ and dies at a time $t = d_i$ when V_i^t merges with another connected component V_j^t for some

$j \leq i$. Note that $d_1 = \infty$. The lifetime of the component V_i^t, $d_i - b_i$, is called persistence or prominence of the component i. This merging information is encoded in a barcode or a dendrogram. In these two representations, each line is associated to a component V_i, has length $d_i - b_i$, and begins at the height b_i. The dendrogram is obtained from the barcode by linking the bars associated to merging components, at a height given by the merging time.

When $\mathbf{m}$ is a point set $\mathbb{X}$, $\Sigma_i = I_d$ and $\omega_i = 0$ for every i, clustering points accordingly to the connected components of V^t boils down to the classical single-linkage clustering procedure, with $t > 0$, calibrated in accordance with the dendrogram. This procedure is not robust to outliers. In this paper, we consider an adjacent procedure, similar to the ToMATo algorithm [12], based on the prominence of components. To be precise, in the clustering scheme, a component V_i cannot merge with another component V_j at a time t larger than $\omega_i + Stop$, for some parameter $Stop$. In other words, components with large prominence will never die in this clustering procedure. This is the purpose of Algorithm 1 in the next section.

In order to better visualize the prominence of the components, we represent their lifetimes in a persistence diagram. A persistence diagram is a multiset of points $(b_i, d_i) \in \mathbb{R}^2$ that lie above the diagonal $b = d$. Each point (b_i, d_i) is associated to a connected component V_i. The notion of persistence diagram was introduced by Edelsbrunner et al. in [16], in the broader framework of homology, and allows to compute lifetimes of additionnal features such as loops, voids etc. It is defined for filtrations that are regular enough, on triangulable spaces such as $\mathbb{R}^d$. The proper notion of regularity is the notion of q-tameness [11]. In [7, Proposition 3.5], Buchet et al. proved that the DTM is q-tame. The proof of [7] can be straightforwardly adjusted for distance functions to compact sets and most importantly, for anisotropic power functions, provided that the eigenvalues of the matrices Σ_i are all positive.

Since distance to compact sets, distance-to-measure and anisotropic power functions are q-tame, the persistence diagrams associated to their filtrations are well defined. They can be compared through the bottleneck distance, a distance between two diagrams D and D' defined as the minimal value of $\max_{x \in D, y \in D'} |y - \phi(x)|_\infty$ among functions ϕ that pair points in D with points in D', with some points potentially paired to diagonal points. Diagrams associated to interleaved filtrations are close, according to the following theorem.

▶ **Theorem 3** (Stability of persistence diagrams [11, 9, 13]). *If two filtrations V and W are q-tame and ϵ-interleaved, then the persistence diagrams of these filtrations are ϵ-close in bottleneck distance.*

According to Theorem 3, the persistence diagram of any anisotropic power function $f_{\mathbf{m},\boldsymbol{\omega},\boldsymbol{\Sigma}}$ that is $\epsilon - \|\cdot\|_\infty$ close to $\mathrm{d}_{\mathcal{X}}$ is ϵ-bottleneck close to the persistence diagram of the sublevel sets of $\mathrm{d}_{\mathcal{X}}$. Consequently, prominence of the connected components of $\mathcal{X}$ can be deduced from the diagram associated to $f_{\mathbf{m},\boldsymbol{\omega},\boldsymbol{\Sigma}}$, for ϵ small enough. This bottleneck closeness occurs with large probability for a regular manifold $\mathcal{X}$ for the c-PDTM built from a noisy sample from $\mathcal{X}$, according to [6]. No such result has been proved yet for the c-PLM. Anyway, intuitively, its sublevel sets are good approximations of the manifold $\mathcal{X}$, with the advantage that they are made of less ellipsoids, and that these ellipsoids are oriented accordingly to the manifold, i.e. with large eigenvalues on the tangent space and small eigenvalues on its orthogonal. This will be confirmed in the numerical illustrations section.

By construction, the persistence diagram (for connected components) associated to the filtration of the sublevel sets of $f_{\mathbf{m},\boldsymbol{\omega},\boldsymbol{\Sigma}}$ coincides with the persistence diagram associated to the anisotropic weighted Čech complex $\mathrm{Cech}(f_{\mathbf{m},\boldsymbol{\omega},\boldsymbol{\Sigma}})$. Consequently, we can forget about the ellipsoids and focus on the simplicial complex filtration, which can be computed and stored efficiently, in a $c \times c$ matrix $\mathrm{Mat} = (t_{i,j})_{i,j \in I}$. Such a matrix contains the times of appearance of vertices and of merging of connected components in $\mathrm{Cech}(f_{\mathbf{m},\boldsymbol{\omega},\boldsymbol{\Sigma}})$. The clustering scheme of this paper exposed just below is based on such a merging matrix Mat.

3.2 An algorithm for persistence-based clustering

Consider $(\mathcal{G}^t)_{t\in\mathbb{R}}$ a filtration of sub-graphs of $\mathcal{G}$, a graph with c nodes. Based on this filtration, we define an algorithm, strongly inspired from the ToMATo algorithm [12]. The clustering scheme is guided by the persistence of the connected components in $(\mathcal{G}^t)_{t\in\mathbb{R}}$, and preserves components with large prominence. We assume that the nodes of $\mathcal{G}$ are labeled such that the node labeled i gets born before the node labeled j, when $i \leq j$. The procedure is as follows. A connected component gets born when a node gets born, with the same label. A component changes of label at each time t for which it merges with a component with smaller label in $\mathcal{G}^t$, unless its prominence is larger than some parameter *Stop*. The prominence of a node or a component is defined as the lifetime of the component in the filtration (i.e. the elapsed time between the birth of the node and the time t such that a node with smaller index is present in its connected component in $\mathcal{G}^t$). The resulting clustering is given by the label of the nodes at time $t = +\infty$. It contains exactly labels of edges with a prominence larger than *Stop*. In this clustering scheme, we decide that nodes born after some time parameter *Threshold* are not relevant; they are removed. This procedure is implemented in Algorithm 1.

Algorithm 1 Persistence-based Clustering Algorithm.

Data: Mat, Threshold, Stop
Result: Color, Birth, Death
`Initialization` ;
c ← max{i | Mat[i,i] ≤ Threshold} ; Mat ← Mat[1:c,1:c] ;
Birth ← [Mat[i,i] for i in 1:c] ; Death ← [∞ for i in 1:c] ;
indice ← 1 ; I ← 1 ; time ← Mat[I,I] ; Color ← [] ;
while *time* < ∞ **do**
  if *time = Mat[I,I]* **then**
    `Component I appears` ;
    indice ← indice + 1 ; Mat[I,I] ← ∞ ; Color[I] ← I;
  else
    (col_max, col_min) ← (max(Color[I],Color[J]) , min(Color[I],Color[J]));
    if *time - Birth[col_max]* ≤ *Stop* **then**
      `Components col_max and col_min merge` ;
      Replace all entries col_max by col_min in Color ;
      Death[col_max] ← time ;
    else
      `Component col_max will never die` ;
    end
    Mat[i,j] ← ∞ for every i, j ≤ min(indice,c) such that
    (Color[i],Color[j]) ∈ {(col_min, col_max), (col_max, col_min)};
  end
  I,J ← arg $\min_{i,j\leq \min(\text{indice},c)}$ Mat[i,j] ; time ← Mat[I,J]
end

This algorithm requires a merging matrix $\text{Mat} = (t_{i,j})_{i,j\in I}$, with $I = [\![1, c]\!]$. We define its coefficients by $t_{i,i}$, the birth time of the node i in the filtration $(\mathcal{G}^t)_{t\in T}$; for $i > j$, $t_{i,j}$ the birth time of the edge $[i, j]$ and for $i < j$, $t_{i,j} = \infty$. The vector *Color* contains the resulting clustering, the vector *Birth*, the birth time of the components and *Death* their death time. Note that *Death*[1] is always $+\infty$. When $(\mathcal{G}_t)_{t\in T}$ is the filtration of the sublevel sets of some power function $f_{\mathbf{m},\boldsymbol{\omega},\boldsymbol{\Sigma}}$, the matrix Mat has coefficients given by $t_{i,i} = \omega_i$ and for $i > j \geq 1$, $t_{i,j}$ the intersecting time of the ellipsoids $\mathcal{E}_i^t$ and $\mathcal{E}_j^t$, given by Proposition 1.

In practice, to label points in $\mathbb{X}$ (generated around $\mathcal{X}$), we consider an approximation of $\mathrm{d}^2_{\mathcal{X}}$ based on a family $\mathbf{m}$ of c centers. Set $\mathbf{m}'$, the centers not removed and labeled by Algorithm 1, and $\boldsymbol{\omega}'$ and $\boldsymbol{\Sigma}'$ the corresponding parameters. Clustering points in $\mathbb{X}$ is made accordingly to these labels and to the Voronoi decomposition of $\mathbb{R}^d$, based on $\mathbf{m}'$, $\boldsymbol{\omega}'$ and $\boldsymbol{\Sigma}'$: $x \in \mathbb{X}$ has the same label as m'_i if $\|x - m'_i\|^2_{\Sigma'^{-1}_i} + \omega'_i \leq \|x - m'_j\|^2_{\Sigma'^{-1}_j} + \omega'_j$ for every j. Since $f_{\mathbf{m}^*,\boldsymbol{\omega}^*,\boldsymbol{\Sigma}^*}$ approximates $\mathrm{d}^2_{\mathcal{X}}$, in order to deal with outliers, we remove (i.e. assign the label 0) the points $x \in \mathbb{X}$ for which $f_{\mathbf{m}',\boldsymbol{\omega}',\boldsymbol{\Sigma}'}(x)$ is the largest. Note that a power function is homogeneous to the square of a distance function. Therefore, for positive weights $\boldsymbol{\omega}$, it could be more appropriate to consider the filtration of sublevel sets of $\sqrt{f_{\mathbf{m},\boldsymbol{\omega},\boldsymbol{\Sigma}}}$ instead of $f_{\mathbf{m},\boldsymbol{\omega},\boldsymbol{\Sigma}}$.

The best complexity of Algorithm 1 ($O(c^3)$ comparisons) is obtained when $Stop = \infty$, with $2c$ iterations of the algorithm. The worst complexity ($O(c^4)$) is obtained when $Stop = 0$, with $O(c^2)$ iterations. This is fast when c is much smaller than the sample size (e.g. for c-PLM and c-PDTM), and does not depend on the dimension. In the experiments of Section 4, Algorithm 1 runs much faster than the computation of the c-PLM and the c-PDTM.

In practice, just as Chazal et al. [12], we recommend to run Algorithm 1 several times. A first time with $Threshold = Stop = \infty$ to calibrate the parameter $Threshold$, in order to remove bad nodes (i.e. nodes with late birth and short lifetime). A second time with this parameter $Threshold$ and $Stop = \infty$, to measure the prominence of the components and select the number of clusters (via the parameter $Stop$), as the number of components with prominence much larger than others. More details on the calibration of these two parameters, from the persistence diagrams $(Birth[i], Death[i])_{i \in I}$, are given in Section 4.1. The final clustering is obtained from $Color$, after running Algorithm 1 with these two parameters.

Giving a sense to an optimal minimal prominence $Stop$ is possible for distance functions. For instance, for the sublevel-sets filtration of $\mathrm{d}_{\mathcal{X}}$, $Stop$ can be chosen as half of the minimal distance between two distinct components of $\mathcal{X}$. Consequently, for any $\epsilon - \|\cdot\|_\infty$-close approximation of $\mathrm{d}_{\mathcal{X}}$, taking $Stop - \epsilon$ leads to a perfect clustering, provided that $2\epsilon < Stop$.

The parameter $Threshold$ is primordial, especially for the c-PLM function. Indeed, the algorithm for the c-PLM is based on $\tilde{\boldsymbol{\theta}}_{c,k}$, a local minimizer of the criterion $R_{c,k}$. Consequently, some ellipsoids $\mathcal{E}_i$ are far from the support, or in a wrong direction. Thus, their weight ω_i (and thus $Birth[i]$) is large with respect to other well-placed ellipsoids, due to a large variance term $v_{y_i,\Sigma_i,k}$. Such bad ellipsoids are removed for a suitable parameter $Threshold$.

3.3 Connection to other persistence-based clustering methods

In the sequel, we display different graph filtrations, to be used for persistence-based clustering, with Algorithm 1. For each of these filtrations, we give a summarize of the corresponding matrices Mat, in Table 1, with the convention that $t_{i,i} \leq t_{j,j}$ when $i \leq j$.

ToMATo Algorithm [12] rests on a graph filtration based on a graph $\mathcal{G}$ and a function f defined on the nodes of $\mathcal{G}$. Morally, $\mathcal{G}^t$ is the sub-graph of $\mathcal{G}$ that contains the nodes i such that $f(i) \leq t$, and the edges $[i,j]$ if and only if i and j are in $\mathcal{G}^t$. Chazal et al. mostly studied this method for $\mathcal{G}$, a Rips graph of a set $\mathbb{X} \subset \mathbb{R}^d$, and for $f(i)$, the DTM to $\mathbb{X}$ at X_i.

The DTM-filtration [1] corresponds to the 1-skeleton of the nerve of the union of balls $\left(\bigcup_{x \in \mathbb{X}} \overline{\mathrm{B}}(x, r_t(x))\right)_{t>0}$ with $r_t(x) = -\infty$ for $t < \mathrm{d}_{\mathbb{X},k}(x)$ and $r_t(x) = (t^p - \mathrm{d}^p_{\mathbb{X},k}(x))^{\frac{1}{p}}$ for $t \geq \mathrm{d}_{\mathbb{X},k}(x)$, for some $p \geq 1$ and with the convention that $\overline{\mathrm{B}}(x, -\infty)$ is empty. In Table 1, we give the coefficients for $p = 1$. The DTM-filtration with $p = 2$ was actually introduced in [7], leading to what we call Power filtration, which coincides with the sublevel-sets filtration of the square of a power distance. We also consider additional power-functions-based filtrations, from the k-witnessed distance [18], the c-PDTM [6] and the c-PLM [4].

Table 1 Coefficients of Mat for the different methods, with the notation $f = \mathrm{d}_{\mathbb{X},k}$ for the DTM to $\mathbb{X}$ with number of nearest neighbors parameter k.

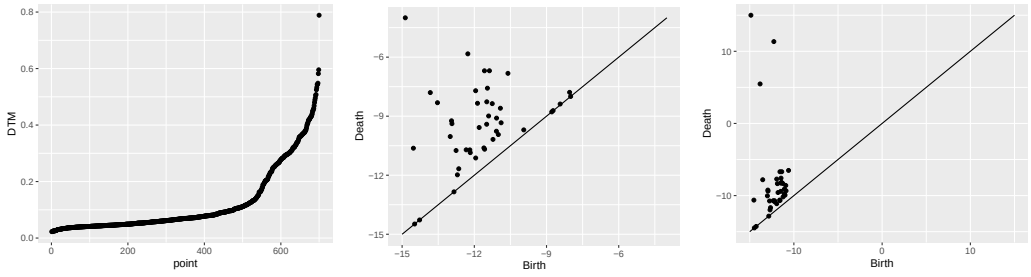
Method	$t_{i,i}$	$t_{i,j}$ for $i < j$
ToMATo	$f(i)$	$\max(f(i), f(j))(\mathbb{1}_{[i,j]\in\mathcal{G}})^{-1}$
DTM-filtration	$f(i)$	$\left(\frac{\|X_i - X_j\| + f(i) + f(j)}{2}\right)\mathbb{1}_{\|X_i - X_j\| > \lvert f(i) - f(j)\rvert} + f(i)\mathbb{1}_{f(i)-f(j)\geq\|X_i - X_j\|}$
$f_{\mathbf{m},\boldsymbol{\omega}}$	ω_i	$\frac{(\omega_j-\omega_i)^2+2(\omega_j+\omega_i)\|m_j-m_i\|^2+\|m_j-m_i\|^4}{4\|m_j-m_i\|^2}$
$\sqrt{f_{\mathbf{m},\boldsymbol{\omega}}}$	$\sqrt{\omega_i}$	$\sqrt{\frac{(\omega_j-\omega_i)^2+2(\omega_j+\omega_i)\|m_j-m_i\|^2+\|m_j-m_i\|^4}{4\|m_j-m_i\|^2}}$
$f_{\mathbf{m},\boldsymbol{\omega},\boldsymbol{\Sigma}}$	ω_i	Given by Proposition 1

Power filtration	$\sqrt{f_{\mathbf{m},\boldsymbol{\omega}}}$ with $\mathbf{m} = \mathbb{X}$ and $\boldsymbol{\omega} = (f^2(x))_{x\in\mathbb{X}}$
Witnessed	$\sqrt{f_{\mathbf{m},\boldsymbol{\omega}}}$ with $(\mathbf{m},\boldsymbol{\omega}) = (m_{x,k}, v_{x,k})_{x\in\mathbb{X}}$
c-PDTM	$f_{\mathbf{m},\boldsymbol{\omega}}$ with $(\mathbf{m},\boldsymbol{\omega}) = (m_{y,k}, v_{y,k})_{y\in\mathbf{y}_{c,k}}$
c-PLM	$f_{\mathbf{m},\boldsymbol{\omega},\boldsymbol{\Sigma}}$ with $(\mathbf{m},\boldsymbol{\omega},\boldsymbol{\Sigma}) = (m_{y,\Sigma,k}, v_{y,\Sigma,k} + \log(\det(\Sigma)), \Sigma)_{(y,\Sigma)\in\theta_{c,k}}$

4 Numerical illustrations

4.1 A complete illustration of the method

Consider the target $\mathcal{X}$, a set of three curves in $\mathbb{R}^2$. We generate $\mathbb{X} = (X_i)_{i\in[\![1,N_s+N_o]\!]}$, a set of $N_s = 500$ signal points $(X_i = Y_i + Z_i)_{i\in[\![1,N_s]\!]}$, with Y_i uniform on $\mathcal{X}$ and Z_i Gaussian with standard deviation $\sigma = 0.02$; corrupted by $N_o = 200$ outliers, uniform on $[-1.5, 2.5]^2$. We compare the clustering scheme based on Algorithm 1 with the sublevel sets of the c-PLM, to the target labels in Figure 2 (left). Parameters are set to $c = 50$ centers, $k = 10$ nearest neighbors, $sig = 520$ points to consider as signal, and $it = 100$ iterations and $n\_ini = 10$ initializations to compute a suitable local optimum $\tilde{\boldsymbol{\theta}}_{c,k,sig}$ of the c-PLM-criterion $R_{c,k,sig}$. Since the DTM $\mathrm{d}_{\mathbb{X},k}$ is large for outliers, we select sig from the curve $([\mathrm{d}_{\mathbb{X},k}(X_i), i \in [\![1, N_s + N_o]\!]]$ in non-decreasing order), as the point of slope break ; see Figure 1 (left). The DTM can be replaced by any not-trimmed approximation of the c-PLM.



Sorted vector of DTM on points in $\mathbb{X}$, to select sig

Persistence diagram, to select $Threshold$

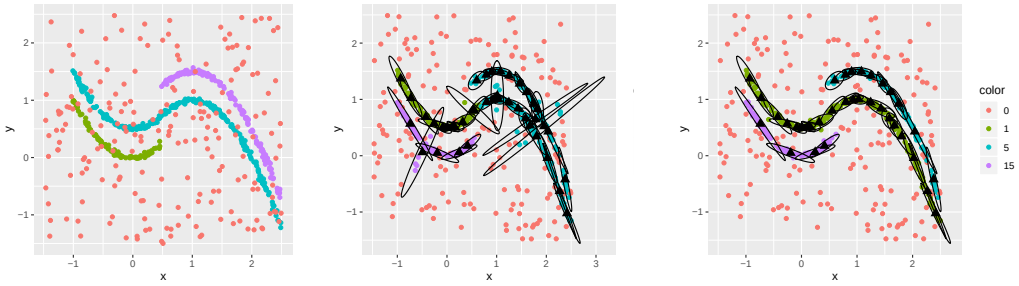
Persistence diagram, to select $Stop$

Figure 1 Parameters selection heuristics.

We run Algorithm 1 a first time with the parameters $Threshold = \infty$ and $Stop = \infty$, and display the persistence diagram $(Birth[i], Death[i])_{i\in[\![1,c]\!]}$, in Figure 1 (middle). In order to have 3 clusters, we select $Stop = 5.62$, the height of a line parallel to the diagonal, separating 3 points from the others. We run Algorithm 1 a second time with this new parameter, which results in the clustering $\mathcal{C}_1$ of Figure 2 (middle). A sublevel set of the function $f_{\tilde{\boldsymbol{\theta}}_{c,k}}$ is represented by the union of ellipses. Note that some ellipses have a bad position. This results

in a bad clustering. We use the parameter $Threshold$ to remove them. In Figure 1 (middle), 6 points are on the right side, separated from the other points with a vertical line (of abscissa -10.27). Then, we run Algorithm 1 with $Threshold = -10.27$ and $Stop = \infty$. According to the persistence diagram in Figure 1 (right), since 3 points are well-separated from the other ones with a large band parallel to the diagonal (containing a line parallel to the diagonal, with height 12), we recover the number of clusters, 3, and set $Stop = 12$. The clustering $\mathcal{C}_2$ obtained with $Threshold = -10.27$ and $Stop = 12$ is represented in Figure 2 (right). The bad ellipses have been removed. Denote by $\tilde{\boldsymbol{\theta}}'_{c,k,sig}$, the subfamily of $\tilde{\boldsymbol{\theta}}_{c,k,sig}$ made of centers not removed by the procedure. The color of any point x in Figure 2 (right) is given by the label in $Color$ (label returned by the Algorithm 1) of its associated center (y, Σ) in $\tilde{\boldsymbol{\theta}}'_{c,k,sig}$. This is the center (y, Σ) such that $f_{\tilde{\boldsymbol{\theta}}'_{c,k,sig}}(x) = \|x - m_{y,\Sigma,k}\|^2_{\Sigma^{-1}} + v_{y,\Sigma,k} + \log(\det(\Sigma)))$. The labels of the $|\mathbb{X}| - sig$ points with largest $f_{\tilde{\boldsymbol{\theta}}'_{c,k,sig}}$-value are set to 0.

Note that for large datasets, computing $\tilde{\boldsymbol{\theta}}'_{c,k,sig}$ may take some time. We can compute it from a sub-sample of $\mathbb{X}$, run Algorithm 1, and label points in $\mathbb{X}$ accordingly.



Original labels $\mathcal{C}_1$, Threshold $= \infty$ $\mathcal{C}_2$, Threshold = - 10.27

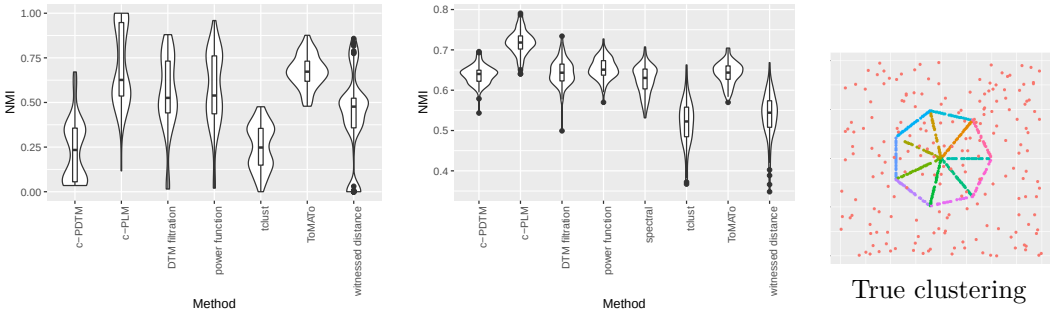
Figure 2 Two resulting clusterings, with ellipses.

We compare the performance of the two clusterings $\mathcal{C}_1$ and $\mathcal{C}_2$. In terms of outliers detection, this can be assessed via the proportion of signal points labeled as outliers (0.034 for $\mathcal{C}_1$, 0.016 for $\mathcal{C}_2$) and as the proportion of outliers labeled as signal points (0.185 for $\mathcal{C}_1$, 0.14 for $\mathcal{C}_2$). As expected from Figure 2, removing bad ellipses reduces these proportions and thus improves the outliers detection performance. In terms of clusters recovering, the normalized mutual information (NMI) is classically used. It equals 1 for a perfect clustering and 0 for a terrible clustering. When considering outliers as a cluster with label 0, we got $NMI = 0.586$ for $\mathcal{C}_1$ and $NMI = 0.841$ for $\mathcal{C}_2$. The NMI computed on the signal points labeled as signal points is $NMI = 0.634$ for $\mathcal{C}_1$ and $NMI = 1$ for $\mathcal{C}_2$, a perfect clustering.

4.2 Comparison of the different methods on synthetic datasets

We compare different clustering methods on two synthetic datasets : the previous dataset with 3 curves, and datapoints from a polygonal curve of 14 segments, as in [8]. We set parameters to $N_s = 500$, $N_o = 200$, $\sigma = 0.02$, $c = 50$, $k = 10$, $it = 100$, $n\_ini = 10$ and $Threshold$ chosen such that 10 means are removed from the c-PLM-centers $\tilde{\boldsymbol{\theta}}_{c,k,sig}$. For the ToMATo algorithm we set $r = 0.12$, the radius of the Rips graph. We used the function dbscan from the R packages dbscan [19], with parameters $eps = 0.15$ and $minPts = 10$; tclust and specc from the tclust [17] and kernlab [21] R packages.

For the three curves, the parameter r for ToMATo is chosen such that the graph is not connected, the clusterings are acceptable but have more than 3 clusters. The c-PLM often performs perfectly, and sometimes performs poorly, since the number of bad ellipses removed



Example of the 3 curves

Example of the 14 segments

True clustering

Figure 3 Violin plots representing the NMI computed on signal points, detected as signal points.

is fixed to 10 and not calibrated according to the heuristics, and their is some instability. We observe the same clustering problem as in Figure 2 (middle) for the other methods since the lines are close, compared to the distance between sample points from the same line. For the polygonal line of 14 segments, all methods except the c-PLM and tclust put centers of clusters on massive parts of $\mathbb{X}$ (the center and the intersections of 3 segments). For the c-PLM and tclust, most clusters coincide with segments. Nonetheless, their is some instability (much less pronounced for the c-PLM), since the algorithms are based on local minimizers.

4.3 Applications to real datasets

4.3.1 Recovering fleas species, based on 6 measurements

We picked the dataset flea from the R-package tourr [29], initially from [23]. This dataset contains records of 6 measurements for 74 males insects from the Palaeartic, from three different species : Heptapotamica, Concinna, Heikertingeri. The variables correspond to measurements on the tarsus, the aedeagus and the head. We normalized data so that the mean and variance of each of the 6 variables are respectively 0 and 1. In Table 2, we computed the NMI between the true species and the clustering returned by different methods. We ran each algorithm 10 times with at most 100 iterations. For every k-nearest-neighbours-based algorithm, we set $k = 10$. For ToMATo, we set $r = 1.9$ so that the graph is connected ; for the c-PLM and the c-PDTM, $c = 50$ and for dbscan, $eps = 1.5$ and $minPts = 10$. The 3-PDTM and 3-PLM methods consists in clustering data according to the weighted Voronoi cells given by the optimal centers and covariance matrices.

Table 2 NMI between clustering of fleas and their true specie.

Without Algorithm 1	k-means 0.825	tclust 0.848	DBSCAN 0.647	Spectral 1	3-PLM 1	3-PDTM 1
With Algorithm 1	ToMATo 0.628	Witnessed 0.906	power 1	DTM-filt. 1	c-PLM hier. 1	c-PDTM hier. 1

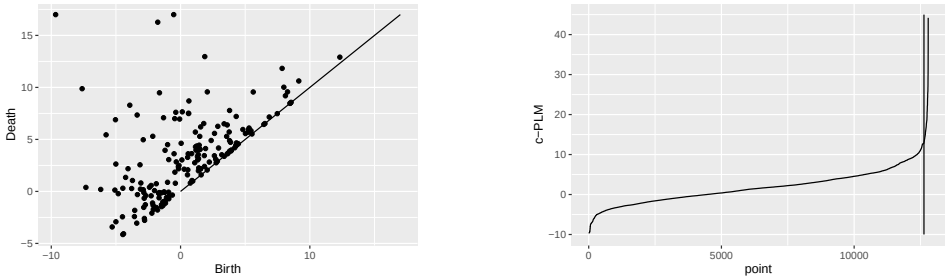
The methods based on the decomposition of $\mathbb{R}^6$ into 3 (weighted and/or curved) Voronoi cells are efficient: at most 3 bad labels for k-means and tclust and all labels correct for their "robust" versions, the 3-PDTM and the 3-PLM. The perfect performance of these two last functions is due to the weights that force the centers of cells to lie in massive areas for $\mathbb{X}$. The bad performance of ToMATo is due to the difficulty to select the parameter r for the Rips

graph, the small number of points, and the fact that the inverse of the DTM should be used instead of the DTM, as recommended by the authors. Nonetheless, we made the choice to use the DTM since the other methods (witnessed distance, power function, DTM-filtration, c-PLM and c-PDTM) are based on filtrations from approximations of the DTM, and almost all of these methods perform perfectly. The method dbscan performs poorly since it labels 14 points as outliers. Nonetheless, the points considered as signal are well clustered.

4.3.2 Clustering a earthquake dataset

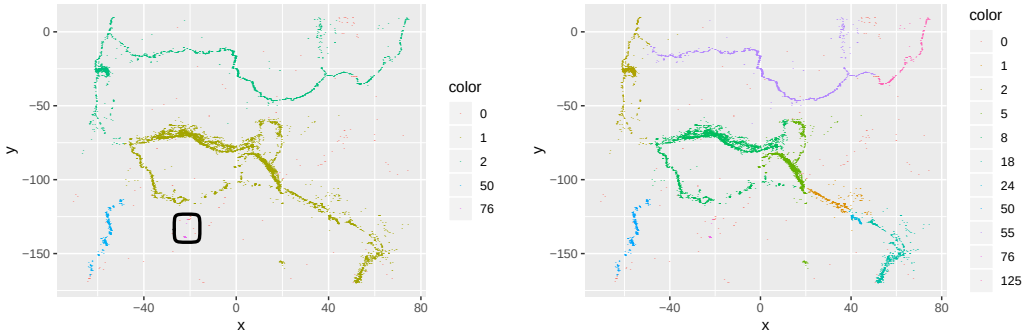
We consider a set of 12790 points representing the longitude and latitude of earthquakes of magnitude non smaller than 5.0, between the 01/01/1970 and the 01/01/2010. This dataset was picked from the website `http://earthquake.usgs.gov/earthquakes/eqarchives/epic/`.

We used Algorithm 1 with an approximation of the c-PLM based on a sub-sample of 2000 points from the dataset, with parameters $c = 200$, $k = 10$ and for $it = 50$ iterations. We restricted matrices Σ to have eigenvalues smaller than 50 by thresholding them. The persistence diagram in Figure 4 suggests that the dataset has 4 or 10 clusters. Moreover, the curve of the sorted values of the c-PLM approximation on the pointset in Figure 4 suggests to keep $sig = 12250$ points as signal points. See Figure 5 for the corresponding clustering.



Persistence diagram | Number of signal points selection

Figure 4 Parameters selection heuristics.



Clustering, 4 groups | Clustering, 10 groups

Figure 5 Earthquake clustering with Algorithm 1, for the c-PLM function.

References

1 Hirokazu Anai, Frédéric Chazal, Marc Glisse, Yuichi Ike, Hiroya Inakoshi, Raphaël Tinarrage, and Yuhei Umeda. DTM-based filtrations. In *35th International Symposium on Computational Geometry*, volume 129 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages Art. No. 58, 15. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2019.

2 Arindam Banerjee, Srujana Merugu, Inderjit S. Dhillon, and Joydeep Ghosh. Clustering with bregman divergences. *J. Mach. Learn. Res.*, 6:1705–1749, December 2005. URL: `http://dl.acm.org/citation.cfm?id=1046920.1194902`.

3 Gregory Bell, Austin Lawson, Joshua Martin, James Rudzinski, and Clifford Smyth. Weighted persistent homology. *Involve*, 12(5):823–837, 2019. `doi:10.2140/involve.2019.12.823`.

4 Claire Brécheteau. Robust shape inference from a sparse approximation of the gaussian trimmed loglikelihood. Unpublished, 2018.

5 Claire Brécheteau, Aurélie Fischer, and Clément Levrard. Robust bregman clustering. In revision, 2018.

6 Claire Brécheteau and Clément Levrard. A k-points-based distance for robust geometric inference. To appear in Bernoulli, 2017.

7 Mickaël Buchet, Frédéric Chazal, Steve Y. Oudot, and Donald R. Sheehy. Efficient and robust persistent homology for measures. *Comput. Geom.*, 58:70–96, 2016. `doi:10.1016/j.comgeo.2016.07.001`.

8 Mickaël Buchet, Tamal K. Dey, Jiayuan Wang, and Yusu Wang. Declutter and resample: towards parameter free denoising. *J. Comput. Geom.*, 9(2):21–46, 2018.

9 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*, SCG '09, pages 237–246, New York, NY, USA, 2009. ACM. `doi:10.1145/1542362.1542407`.

10 Frédéric Chazal, David Cohen-Steiner, and Quentin Mérigot. Geometric Inference for Measures based on Distance Functions. *Foundations of Computational Mathematics*, 11(6):733–751, 2011. `doi:10.1007/s10208-011-9098-0`.

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

12 Frédéric Chazal, Leonidas J. Guibas, Steve Y. Oudot, and Primoz Skraba. Persistence-based clustering in Riemannian manifolds. *J. ACM*, 60(6):Art. 41, 38, 2013. `doi:10.1145/2535927`.

13 David Cohen-Steiner, Herbert Edelsbrunner, and John Harer. Stability of persistence diagrams. *Discrete Comput. Geom.*, 37(1):103–120, 2007. `doi:10.1007/s00454-006-1276-5`.

14 J. A. Cuesta-Albertos, A. Gordaliza, and C. Matrán. Trimmed k-means: an attempt to robustify quantizers. *Ann. Statist.*, 25(2):553–576, 1997. `doi:10.1214/aos/1031833664`.

15 Vin de Silva and Robert Ghrist. Coverage in sensor networks via persistent homology. *Algebr. Geom. Topol.*, 7:339–358, 2007. `doi:10.2140/agt.2007.7.339`.

16 Herbert Edelsbrunner, David Letscher, and Afra Zomorodian. Topological persistence and simplification. *Discrete Comput. Geom.*, 28(4):511–533, 2002. Discrete and computational geometry and graph drawing (Columbia, SC, 2001). `doi:10.1007/s00454-002-2885-2`.

17 Heinrich Fritz, Luis A. Garcia-Escudero, and Agustin Mayo-Iscar. tclust: An R package for a trimming approach to cluster analysis. *Journal of Statistical Software*, 47(12):1–26, 2012. URL: `http://www.jstatsoft.org/v47/i12/`.

18 Leonidas Guibas, Dmitriy Morozov, and Quentin Mérigot. Witnessed k-distance. *Discrete Comput. Geom.*, 49(1):22–45, 2013. `doi:10.1007/s00454-012-9465-x`.

19 Michael Hahsler, Matthew Piekenbrock, and Derek Doran. dbscan: Fast density-based clustering with R. *Journal of Statistical Software*, 91(1):1–30, 2019. `doi:10.18637/jss.v091.i01`.

20 Allen Hatcher. *Algebraic topology*. Cambridge University Press, Cambridge, 2002.

21 Alexandros Karatzoglou, Alex Smola, Kurt Hornik, and Achim Zeileis. kernlab – an S4 package for kernel methods in R. *Journal of Statistical Software*, 11(9):1–20, 2004. URL: http://www.jstatsoft.org/v11/i09/.

22 Stuart P. Lloyd. Least squares quantization in PCM. *IEEE Trans. Inform. Theory*, 28(2):129–137, 1982. doi:10.1109/TIT.1982.1056489.

23 Alexander A. Lubischew. On the use of discriminant functions in taxonomy. *Biometrics*, pages 455–477, 1962.

24 J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, pages 281–297, Berkeley, Calif., 1967. University of California Press. URL: https://projecteuclid.org/euclid.bsmsp/1200512992.

25 Stephen B Pope. Algorithms for ellipsoids. Technical Report FDA-08-01, Sibley School of Mechanical & Aerospace Engineering, Cornell University Ithaca, New York 14853, 2008.

26 P. J. Rousseeuw and A. M. Leroy. *Robust Regression and Outlier Detection*. John Wiley & Sons, New York, 1987.

27 Ulrike von Luxburg. A tutorial on spectral clustering. *Stat. Comput.*, 17(4):395–416, 2007. doi:10.1007/s11222-007-9033-z.

28 Wenping Wang, Jiaye Wang, and Myung-Soo Kim. An algebraic condition for the separation of two ellipsoids. *Comput. Aided Geom. Design*, 18(6):531–539, 2001. doi:10.1016/S0167-8396(01)00049-8.

29 Hadley Wickham, Dianne Cook, Heike Hofmann, and Andreas Buja. tourr: An R package for exploring multivariate data with projections. *Journal of Statistical Software*, 40(2):1–18, 2011. URL: http://www.jstatsoft.org/v40/i02/.

Geometric Secluded Paths and Planar Satisfiability

Kevin Buchin
Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands
k.a.buchin@tue.nl

Valentin Polishchuk
Communications and Transport Systems, ITN, Linköping University, Sweden
firstname.lastname@liu.se

Leonid Sedov
Communications and Transport Systems, ITN, Linköping University, Sweden
firstname.lastname@liu.se

Roman Voronov
Institute of Mathematics and Information Technologies, Petrozavodsk State University, Russia
rvoronov@petrsu.ru

Abstract

We consider paths with low *exposure* to a 2D polygonal domain, i.e., paths which are seen as little as possible; we differentiate between *integral* exposure (when we care about how long the path sees every point of the domain) and *0/1* exposure (just counting whether a point is seen by the path or not). For the integral exposure, we give a PTAS for finding the minimum-exposure path between two given points in the domain; for the 0/1 version, we prove that in a simple polygon the shortest path has the minimum exposure, while in domains with holes the problem becomes NP-hard. We also highlight connections of the problem to minimum satisfiability and settle hardness of variants of planar min- and max-SAT.

2012 ACM Subject Classification Theory of computation; Theory of computation → Computational geometry

Keywords and phrases Visibility, Route planning, Security/privacy, Planar satisfiability

Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.24

Related Version A full version [6], including the omitted details, is available at `https://arxiv.org/abs/1902.06471`.

Acknowledgements We thank Mike Paterson for raising the question of finding minimum-exposure paths, and the anonymous reviewers for the comments improving the presentation of the paper; we also acknowledge discussions with Irina Kostitsyna, Joe Mitchell and Topi Talvitie. Part of the work was done at the workshop on Distributed Geometric Algorithms held in the University of Bologna Centre at Bertinoro Aug 25-31, 2019. VP and LS are supported by the Swedish Transport Administration and the Swedish Research Council.

1 Introduction and Related work

Both visibility and motion planning are textbook subjects in computational geometry – see, e.g., the respective chapters in the handbook [21] and the books [20, 34]. Visibility meets routing in a variety of geometric computing tasks. Historically, the first approach to finding shortest paths was based on searching the visibility graph of the domain; visibility is vital also in computing *minimum-link* paths, i.e., paths with fewest edges [25, 31, 32, 39]. "Visibility-driven" path planning has attracted also some recent interest [3, 37, 44]. In addition to the theoretical considerations, visibility and motion planning are closely coupled in practice: computer vision and robot navigation go hand-in-hand in many courses and real-world applications.

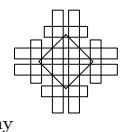


Table 1 Hardness of minimum-exposure paths in polygonal environments.

	0/1 exposure	Integral exposure
With holes	Hard (Thm. 1)	PTAS (Thm. 4)
Simple	P (Thm. 2)	

Table 2 Hardness of versions of planar opt2SAT (see Section 4 for definitions).

	min2SAT	max2SAT
V-cycle	Hard (Thm. 7)	Hard (Thm. 7)
VC-cycle	Hard (Thm. 7)	
Separable	P (Thm. 5)	
Monotone	P (Cor. 6)	

The question of *hiding* a path in a polygonal domain was first raised in a SoCG'88 paper [19]: it considered the *robber route problem* in which the goal is to minimize the length traveled within sight of at least one of a number of *threats* (each threat being a point); the problem reduces to finding the shortest path in the $0/1/\infty$ metric that assigns a cost of 1 to the union of the visibility polygons of the threats, and 0 to the rest of the domain (and infinite weight to the complement of the domain, where travel is forbidden). Our settings are different from [19] in two aspects: (1) we have a *continuum* of the threats (every point in the domain is a threat) and (2) in the integral version, we care for how long threats are seen from points along the path (formally: we integrate the visible area along the path); in other words, we account for the "intensity" of the visibility from the threats.

Lately, motivated by the rise of the Internet of things (IoT) and mobile computing, there has been a surge of research on anonymity, security, confidentiality and other forms of privacy preservation (in particular, in geometric environments [4]), studying paths with minimum exposure to sensors in a network [16, 17, 38, 43]. The standard model, again, assumes a finite number of point sensors, so the visibility is changing discretely, as the path goes in/out of a sensor coverage. To our knowledge, Lebeck, Mølhave and Agarwal [26, 27] were the first to introduce integration of the visibility *continuously* changing along the path (which is also one of our models). Our paper is different from Lebeck et al. in that we give algorithms with provable theoretical performance in continuous domains under the usual notion of distance-independent visibility. Lebeck et al. presented strategies with outstanding practical performance on discretized terrains, in the more realistic model of visibility deteriorating with distance.

Minimizing the integral exposure can be viewed as an extension of the *weighted region problem (WRP)* [2, 9–11, 14, 24, 33, 35] to the case of continuously changing weight, where the weight of a point is the area of its visibility polygon; in the WRP the input is a weighted polygonal subdivision of the domain (with a constant weight assigned to each cell of the subdivision) and the goal is to find the path minimizing the integral of the weight along the path. The computational complexity of the WRP is open; PTASs for the problem have running times that depend not only on the complexity of the subdivision, but also on various parameters of the input like ratio of max/min weight, largest coordinate and angles of the regions, etc. (the parameters differ between the algorithms, see [41, Ch. 31] for details). Integration of other measures of "local quality" (different from visibility) for points along a path was the subject also in the study of *high-quality* paths [1, 45] and related research [46, 47].

Recent papers [8, 18, 29, 42] explored paths adjacent to few vertices in graphs; such paths were dubbed *secluded* in [8]. Our paper may thus be viewed as studying geometric versions of the secluded path problem.

Contributions and Roadmap. In Section 2 we prove that in a polygonal domain with holes it is NP-hard to find a path, between two given points, minimizing the area seen from the path; the reduction is from minSAT (find the truth assignment to Boolean variables so as to

satisfy the minimum number of given disjunctive clauses). In Section 2.1 we complement the hardness by showing that in a simple polygon, shortest paths are the ones that see minimum area; even more generally, we prove that in a polygon with holes, a locally shortest path sees less area than any path of the same homotopy type (because for a small number of holes the homotopy types can be efficiently enumerated, this implies that the problem is FPT parameterized by the number of holes). Section 3 gives a PTAS for minimizing the integral of the seen area along the path; we first give a generic scheme for building a piecewise-constant approximation of the visibility area for points in the domain, and then in Section 3.2 present details of an implementation which allows applying a PTAS for WRP on our "pixels" with approximately constant seen area. Finally, in Section 4 we further explore the connection between path hiding and minSAT, and determine hardness of versions of planar minSAT (and maxSAT).

Tables 1 and 2 summarize the results. We leave open designing an approximation algorithm for minimizing the seen area, as well as the complexity of the integral version of the problem.

Notation and Problems formulation. We use $|\cdot|$ to denote the measure of a set, i.e., length of a segment and area of a 2D set. Let P be a polygonal domain with n vertices and $s, t \in P$ be two given points in it. For a point $p \in P$ let $V(p) \subseteq P$ be the visibility polygon of p, i.e., the set of points seen by p. We study the following problems:

- Geometric Secluded Path: Find the s-t path that sees as little area of P as possible (the area seen by a path is defined as the area seen by at least one point of the path, i.e., the so called *weak* visibility region of the path).
- Integral Geometric Secluded Path: Find the s-t path π that minimizes the integral of the area of the visibility polygon over the points along the path, $\int_\pi |V(p)| \, \mathrm{d}p$.

2 Minimizing seen area

We prove that exposure minimization is NP-hard in general, but in simple polygons the minimum-exposure path is the shortest path.

▶ **Theorem 1.** *Geometric Secluded Path is NP-hard.*

Proof. We reduce from min2SAT: find truth assignment for a set of n variables, satisfying the minimum number of given two-literal disjunctive clauses. (Inside this proof n will denote the number of variables and c the number of clauses.) Figure 1, left illustrates the construction. A variable gadget is an isosceles triangle. The triangles for the variables are stacked into a *Christmas tree*, with s and t placed at the top and the root respectively. Going through the left (resp. right) vertex of a triangle represents setting the variable to True (resp. False). The clauses are all put on a horizontal line above the Christmas tree so that the segment between any literal and any clause does not intersect the tree. Each clause is connected to its literals, and all connections (including the ones forming the Christmas tree edges) are thin corridors forming the domain; a clause gadget is simply the intersection of the two corridors. The idea of the reduction is to have an s-t path go through all variable gadgets, choosing whether to go through the variable or its negation in every gadget: the fewer clause gadgets are seen, the fewer clauses are satisfied.

A few technicalities have to be taken care of:

- Two variable–clause corridors, leading to different clauses, may intersect *midway*, meaning that the intersection area may be seen twice. We have to make sure that the area of such a midway intersection is much smaller than the clause gadget area. Being smaller by a

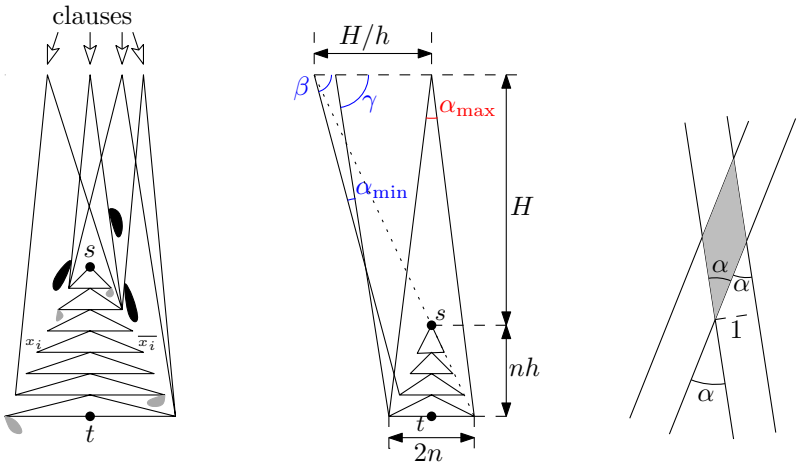


Figure 1 Left: The reduction from min2SAT. All segments are thin corridors of P. Some leakage-blocking high-area chambers are shown black and some area equalizers are shown gray (both black and gray belong to the domain). Middle: the largest angle at a clause and the smallest angle at a midway intersection. The c clauses are spread evenly on the segment of width $2H/h$; thus, the distance between clauses (the base of the triangle with angle $\alpha_{\min}$ at the apex) is $\frac{2H}{h(c-1)}$. Right: Midway intersection of unit-width corridors is area-$\frac{1}{\sin\alpha}$ rhombus with side $\frac{1}{\sin\alpha}$ and angle α.

factor $4c^3$ will suffice: even if parts of a corridor are seen due to the midway intersections with all (at most $2c-1$) other corridors, the total seen corridor's midway area will still be smaller (by a factor $\approx 2c$) than the area of a single clause gadget. Moreover, with such small midway intersections, they may be neglected altogether when counting the areas of clause gadgets seen from literals: the total areas of all (at most $4c^2$ midway intersections) will be smaller by at least a factor of c than the area of a single clause gadget. To reduce areas of the midway intersections in comparison to the clause gadgets areas, we put the clause gadgets high above the Christmas tree – at height H, to be determined later (Fig. 1, middle). The area of intersection of two corridors (Fig. 1, right) is inversely proportional to the sine of the angle between the corridors (the corridors are all of the same width), so the smallest-area clause gadget would be the one for the clause $x_n \vee \overline{x_n}$ placed directly above the apex of the Christmas tree (since we do not control which clause goes where on the clauses line, we have to consider the worst case); let $\alpha_{\max}$ be the angle between the corridors defining the gadget. Assuming the height of every variable gadget triangle is h and their bases have lengths $2, 4, \ldots, 2n$ (refer to Fig. 1, middle),

$$\alpha_{\max} = 2\arctan\frac{n}{H+nh}.$$

On the other hand, the *smallest angle* between two interesting corridors that do not lead to the same clause (i.e., the smallest angle that may define the area of a midway intersection) can be formed by corridors leading to last and last-but-one clause from the last-but-one and last variables x_n, x_{n-1} resp. (changing the endpoints of the corridors would only increase the angle of intersection); the angle is

$$\alpha_{\min} = \gamma - \beta = \arctan\frac{H+nh}{H/h - \frac{2H}{h(c-1)} - n} - \arctan\frac{H+(n-1)h}{H/h-(n-1)}.$$

By trigonometric formulas, the ratio $\sin\alpha_{\min}/\sin\alpha_{\max}$, after being squared a constant number of times, is a ratio of polynomials. This ratio tends to infinity as H grows; hence, at a polynomially large H, the ratio becomes larger than $4c^2$, as we need.

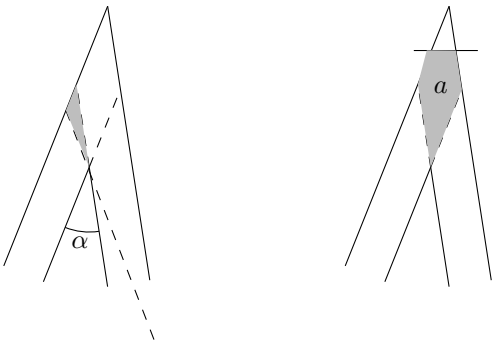


Figure 2 Left: Area seen by one literal only (gray) is negligible for small α. Right: Decreasing clause gadget area.

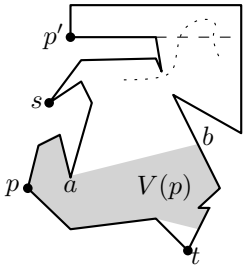


Figure 3 $V(p)$ is shaded; ab is the essential cut of p. A dotted path, crossing the cut of p' (dashed), can be shortcut along the cut.

- We make sure that the area, around a clause gadget, seen from one literal but not from the other (Fig. 2, left), is negligible in comparison with the clause gadget area (seen from both literals of the clause). This is already taken care of by the above, as the whole construction is made tall (large H).
- Leakage of paths from the Christmas tree into variable–clause corridors is prevented by attaching a large-area chamber to each corridor (between the literal and the first intersection of two corridors), so that a path going through the corridor would see the whole area of the chamber. To ensure that the area of a single chamber is larger than the area seen by any path through the Christmas tree, the whole construction is scaled up while keeping the width of the corridors fixed: since the areas available for the chambers grow quadratically with the scaling factor and the areas seen along the corridors grow linearly, a polynomial scaling will suffice to ensure that the chambers areas are large enough to prevent the path going anywhere except through the variable gadgets.
- We attach area equalizers to the literals so that no matter whether the path passes through the variable or its negation, it sees the same non-clause area (the areas may be different between the different variables; we only make sure that for any single variable the seen non-clause area does not depend on whether the variable is set to true or false by the path).
- In the construction so far, different clause gadgets may have different areas; let a denote the smallest area of a clause gadget. We make sure that all clause gadgets have area a, which can be done e.g., by appropriately cutting off the clause gadgets from the top (Fig 2, right).

Now, all s-t paths, going through the Christmas tree only, will see the same non-clause area A. The total area seen by a path is then $\approx A + ka$ where k is the number of clauses seen by the path, which is the same as the number of clauses satisfied by the truth assignment set by the path (we say that the seen area is *approximately* equal to $A + ka$ because of the non-counted areas that may be seen – midway intersections and parts seen by one literal only – which we made sure to be negligible in comparison with a). ◀

In Section 4 we discuss why we could not use *planar* min2SAT to prove hardness of Geometric Secluded Path, avoiding dealing with the crossings.

2.1 Simple polygons

We show that in a simple polygon shortest paths see least area:

▶ **Theorem 2.** *If P is a simple polygon, the shortest s-t path is the solution to* Geometric Secluded Path.

Proof. The visibility polygon $V(p)$ of a point $p \in P$ is bounded by edges and chords of P, with each chord connecting a vertex of the polygon to a point on its boundary. If P is a simple polygon and p does not see s ($s \notin V(p)$), then there is a unique chord separating p from s; the chord is called the *essential cut* of p [7] (Fig. 3).

If an essential cut does not separate s from t, then the shortest s-t path does not cross the cut, for otherwise, the path could be shortcut along the cut. That is, the shortest path crosses those and only those cuts that separate s from t. But any other path also has to cross all such cuts, i.e., has to see all the points seen by the shortest path. ◀

For polygons with a small number of holes one may go through all homotopy types of simple (without self-intersections) s-t paths: a simple argument shows that a shortcut of a path sees less than the original path, and hence the locally shortest path is the secluded path within its homotopy class.

3 A PTAS for minimizing integral exposure

In Section 3.1 we give a generic way to partition the domain in such a way that the visible area is approximately constant within a cell of the partition; then in Section 3.2 we present details of a slightly different partitioning, having straight-line edges, on which a PTAS for the WRP can be applied to find the path with approximately minimum integral exposure.

3.1 Reduction to WRP with curved regions

We first compute the *visibility graph* of P, i.e., the graph connecting pairs of mutually visible vertices of the domain, and extend every edge of the graph in both directions maximally within P. The extensions of the visibility edges split P into $O(n^4)$ cells such that the visibility polygon $V(p)$ is combinatorially the same for any point p within one cell of the subdivision; the subdivision is called the *visibility decomposition* of P [5]. In particular, the area $|V(p)|$ is given by the same formula for any point p in one cell σ of the decomposition. Specifically, the rays from p through the seen vertices of P split $V(p)$ into $O(n)$ triangles (Fig. 4, left). The side of any triangle, opposite to p, is a subset of an edge of P; we call this side the *base* of the triangle. Each of the other, non-base sides is formed by a ray passing through a vertex r' of P and ending at a point r on the base. (In Section 3.2 we will differentiate between *fixed-endpoint* sides for which $r = r'$ is an endpoint of the base and *rotating* rays which rotate around r' if p moves; here we treat both types of sides with a single formula, since fixed-endpoint sides may be viewed as a special case of rotating sides with $r = r'$.)

To write the formula for the area of the triangle pqr, we follow [12, Appendix A.1] and assume that the base is the x-axis and that both $p = (x, y)$ and $r' = (a, b)$ lie above the base ($y, b \geq 0$); then the abscissa of r is $x - y(x - a)/(y - b)$ (Fig. 4, right). Let q' be the vertex that defines the other side, pq, of pqr; to simplify the formulas, assume w.l.o.g. that q' lies on the y-axis: $q' = (0, d)$. The abscissa of q is then $x - yx/(y - d)$, and the area $y|rq|/2$ of the triangle pqr is

$$|pqr| = \frac{y^2}{2}\left(\frac{x-a}{y-b} - \frac{x}{y-d}\right) \tag{1}$$

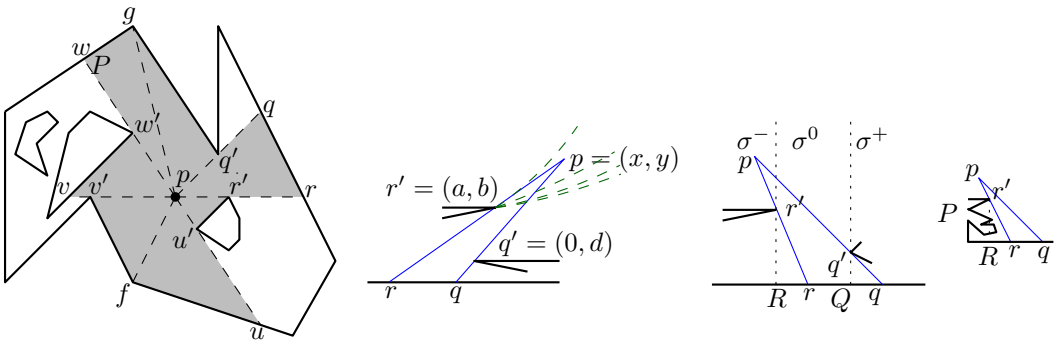


Figure 4 Left: Domain P with 3 holes and a point $p \in P$; $V(p)$ is shaded. Triangle pqr has two rotating sides, triangles puf, pvw', pwg have one fixed-endpoint and one rotating side; the other triangles have two fixed-endpoint sides. Right: $|pqr| = y|rq|/2$. Green dashed curves are level sets of $|pqr|$.

Figure 5 Left: For $p \in \sigma^-$, $|r'Rr|$ is subtracted from C while $|q'Qq|$ is added; for $p \in \sigma^0$, both areas are added; for $p \in \sigma^+$, $|r'Rr|$ is added while $|q'Qq|$ is subtracted. Right: $r'R$ is not fully inside P.

Next, to obtain a piecewise-constant $(1+\varepsilon)$-approximation of the area $|V((x,y))|$ visible from point $(x,y) \in P$, we use level sets of the area function (1). For a given area A, the equality $|pqr| = A$ is attained along the curve γ_A

$$x = \frac{2A/y^2 + a/(y-b)}{1/(y-b) - 1/(y-d)}. \tag{2}$$

Consider a cell σ of the visibility decomposition. We split σ with the curves γ_{A_i} for a set $\mathcal{A} = (A_1, \dots, A_i, \dots)$ of areas forming geometric progression with common ratio $1+\varepsilon$: $A_i = (1+\varepsilon)A_{i-1}$. Let S_i denote the set of points p for which the area of the triangle pqr is between A_{i-1} and A_i (that is, $S_i = \{p \in \sigma : A_{i-1} < |pqr| \le A_i\}$ are the points between $\gamma_{A_{i-1}}$ and γ_{A_i}). We call S_i a *curved sector* because in equation (2), we have $\lim_{y\to b} x(y) = a$ for any A, i.e., all curves γ_A have $r' = (a,b)$ as a common point. (We put a GeoGebra graphics to play with the level sets to see how they look at `https://www.geogebra.org/m/cvxvhfcf`.) We assign the same *weight* A_i to all points in the curved sector; this way, for $i > 1$ the weight of any point $p \in S_i$ is within factor $1+\varepsilon$ of the area of the triangle pqr:

$$|pqr| \le A_i \le (1+\varepsilon)|pqr| \qquad \forall p \in S_i, \forall i > 1 \tag{3}$$

For every cell σ of the visibility decomposition, we overlay the level sets from each of the $O(n)$ triangles of $V(p)$ for $p \in \sigma$. We confine the level sets to the cell, i.e., for each curve γ_A use only the intersection $\gamma_A \cap \sigma$. We call each cell of the overlay a *region* and set the weight of the region to the sum of the weights of the curved sectors whose intersection forms the region.

To bound the number of level sets used (i.e., to determine the first area A_1 in the geometric sequence $\mathcal{A}$ and the needed length of the sequence), assume that vertices of P have integer coordinates and let L denote the largest coordinate. (This model and its variants are common for WRP; in particular, the running times of known solutions for WRP [2,9–11,24,33] depend on L.) Now, consider a triangulation T of P – any point $p \in P$ lies inside a triangle τ of T and sees all of the triangle; thus the area $|V(p)|$ is at least the area of τ. Since τ has integer coordinates, by Pick's Theorem [22] the area of the triangle is at least $1/2$:

$$|V(p)| \ge 1/2 \tag{4}$$

We are now ready to prove that it suffices to have

$$A_1 = \frac{\varepsilon}{2n} \tag{5}$$

Indeed, suppose $V(p)$ consists of K triangles of areas $\Delta_1, \ldots, \Delta_K$ and let $A_1, \ldots, A_K$ be the weights of the curved sectors that form the region to which p belongs; the weight of the region is thus $w(p) = A_1 + \cdots + A_K$. Classify the triangles as "small" and "large", with the former having area at most A_1 (and thus having p lie in the sector S_1) and the latter having area larger than A_1 (with p in a sector S_i for $i > 1$); let $l = \{k : \Delta_k > A_1\}$ be the indices of the large triangles. By (3), for every large triangle $k \in l$, $A_k \le (1+\varepsilon)\Delta_k$. Since $K \le n$, we have

$$w(p) = \sum_{k\in l} A_k + \sum_{k\notin l} A_k \le (1+\varepsilon)\sum_{k\in l}\Delta_k + nA_1 \le (1+\varepsilon)|V(p)| + \varepsilon\frac{1}{2} \le (1+2\varepsilon)|V(p)| \tag{6}$$

where the last inequality is due to (4).

▶ **Proposition 3.** *If WRP on N regions with curved boundaries of constant algebraic complexity can be $(1+\varepsilon)$-approximated in time $T(N, \frac{1}{\varepsilon})$, then a $(1+\varepsilon)^2$-approximation to the minimum integral exposure path can be found in time $T(\frac{n^{10}}{\varepsilon^2}\log^2(nL), \frac{1}{\varepsilon})$.*

Proof. For an upper bound on the sector weight, note that obviously $\forall p \in P$, $|V(p)| \le L^2$. Hence, the number of needed level sets is at most $\log_{1+\varepsilon}(2nL^2) = O(\frac{1}{\varepsilon}\log(nL))$. The level sets are defined for each of the $O(n^3)$ triples $r', q', \bar{qr}$ where r', q' are vertices and $\bar{qr}$ is the side of P containing qr; thus overall there are $O(\frac{n^3}{\varepsilon}\log(nL))$ level set curves. Since each curve γ_A has constant algebraic degree (cf. (2)), any two curves intersect $O(1)$ times, so the complexity of the overlay of the level sets inside the cell σ of the visibility decomposition is $O((\frac{n^3}{\varepsilon})^2\log^2(nL))$. Since there are $O(n^4)$ cells, our construction splits P into $O(\frac{n^{10}}{\varepsilon^2}\log^2(nL))$ regions of constant weight.

By (6), region weights approximate the visibility area to within $1+\varepsilon$ (use $\varepsilon{:=}\varepsilon/2$ to get rid of the factor 2 in front of ε); hence finding a $(1+\varepsilon)$-approximate solution to the WRP on our regions provides a $(1+\varepsilon)^2$-approximation to the minimum integral exposure path. Formally, let π^* be the minimum integral exposure path (the optimal solution to Integral Geometric Secluded Path), let $\bar{\pi}$ be the minimum-weight path through our regions (the optimal solution to WRP) and let π be the $(1+\varepsilon)$-approximate solution to WRP; then

$$\int_\pi |V(p)|\,\mathrm{d}p \;\le\; \int_\pi w(p)\,\mathrm{d}p \;\le\; (1+\varepsilon)\int_{\bar{\pi}} w(p)\,\mathrm{d}p \;\le\; (1+\varepsilon)\int_{\pi^*} w(p)\,\mathrm{d}p \;\le\; (1+\varepsilon)^2\int_{\pi^*}|V(p)|\,\mathrm{d}p \tag{7}$$

where the first inequality is due to the left inequality of (3), the second is because π approximates $\bar{\pi}$, the third is because $\bar{\pi}$ is optimal w.r.t. w, and the last one is due to the right inequality in (3). ◀

3.2 A detailed implementation

Applicability of Proposition 3 remains questionable due to absence of an algorithm for WRP with curved regions boundaries. In this section we present another, direct approach to reduce our problem to WRP on a *polygonal* subdivision. We refine the visibility decomposition (without affecting the asymptotic complexity) and recalculate the area functions so that they have *linear* levels. This way, the regions in the overlay of the level sets are convex, so existing WRP solutions can be applied directly.

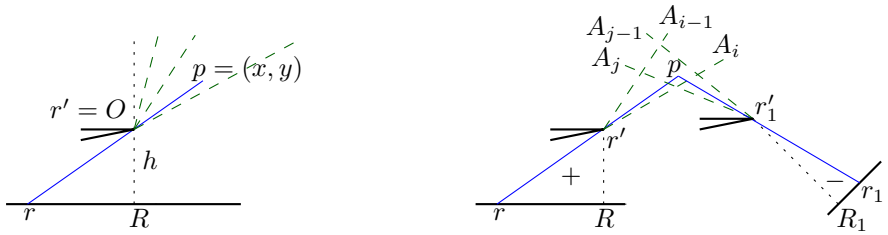


Figure 6 Left: Green dashed are level sets of the area function Δ_r (9). Right: A_i contributes positively to $w(p)$ while A_j comes with minus into $w(p)$, because for p in this region, $r' \in \oplus$ ($r'Rr \in V(p)$) while $r'_1 \in \ominus$ ($r'_1R'r'_1 \notin V(p)$).

Specifically, we differentiate between fixed-endpoint and rotating sides of the triangles into which $V(p)$ is split: the former end at a vertex of P while the latter rotate around a vertex if p moves (see Fig. 4, left). Triangles whose both sides are fixed-endpoint are easy to handle: (while the area of each individual triangle changes as p moves,) the *total* area of all such triangles remains the same (moving p just redistributes the area between the triangles, "stealing" from some and "giving" to others). We therefore call such triangles *fixed*.

Consider now a triangle pqr whose both sides pq, pr are rotating around vertices q', r' resp. (this is the most general case: if one of the sides, say, pq' is fixed, we can just assume $q = q'$); assume that rq is horizontal (Fig. 5, left). We refine the visibility decomposition by extending the vertical segments through each of r', q' maximally up and down; let R, Q be the feet of the perpendiculars dropped from r' and q' resp. onto the supporting line of pq (any of $r'R, q'Q$ may lie only partially inside P, as in Fig. 5, right – this is not an issue). Note that $|Rr'pq'Q|$ may be added to the fixed-triangles areas – the total area of all fixed triangles plus the areas of the pentagons $Rr'pq'Q$ for all the triangles with p as the apex does not depend on p (while p remains in the same cell). Denote this total area by C. The area $|V(p)|$ is obtained from C by adding/subtracting the areas of the triangles $r'Rr$ for all vertices r' on which a side of a triangle of $V(p)$ rotates – whether $|r'Rr|$ is added or subtracted depends on whether the triangle is in $V(p)$ or not:

$$|V(p)| = C + \sum_{r' \in \oplus} \Delta_r - \sum_{r' \in \ominus} \Delta_r \tag{8}$$

where $\Delta_r = |r'Rr|$ and $\oplus$ (resp. $\ominus$) is the set of vertices whose triangles $r'Rr$ are visible (resp. invisible) from p.

Assume that r' is the origin O and that the supporting line of rR is the horizontal line $y = -h$, and let $p = (x, y)$ with $x \geq 0$ (Fig. 6, left). Then

$$\Delta_r = \frac{h^2}{2} \frac{x}{y} \tag{9}$$

and a level set $\gamma_A = \{p = (x, y) : \Delta_r = A\}$ of the function (9) is a ray (emanating from the origin) of constant x/y: since the height $r'R$ of the triangle is fixed, Δ_r is constant whenever r is fixed. As in Section 3.1, we draw the rays for a set $\mathcal{A} = (A_1, \ldots, A_i, \ldots)$ of areas forming geometric progression with common ratio $1+\varepsilon$ and assign the weight A_i to all points in the sector $S_i = \{p \in \sigma : A_{i-1} < \Delta_r \leq A_i\}$ between $\gamma_{A_{i-1}}$ and γ_{A_i} (we again use the weight $A_1 = \varepsilon/(2n)$ for points between γ_0 and γ_{A_1}). Also as in Section 3.1, we define a *region* as a cell in the overlay of the rays emanating from the vertices r' of P. Finally, the weight $w(p)$ of any point p in a region is determined by C and the weights of the sectors forming the region: for a vertex $r' \in \oplus$ the weights of the sectors of r' are added to regions weights; for a vertex $r' \in \ominus$, the weights are subtracted (Fig. 6, right).

The fact that our subdivision into regions provides a $(1+\varepsilon)$-approximation to $|V(p)|$ can be argued similarly to Section 3.1:

▶ **Theorem 4.** *If WRP on N regions can be $(1+\varepsilon)$-approximated in time $T(N, \frac{1}{\varepsilon})$, then a $(1+\varepsilon)^3$-approximation to the minimum integral exposure path can be found in time $T(\frac{n^4}{\varepsilon}\log(nL), \frac{1}{\varepsilon})$.*

4 On planar optimal satisfiability

In this section we return to the (non-integral) Geometric Secluded Path problem (Section 2) and elaborate on its connections to planar satisfiability, identifying, in particular, polynomially solvable and hard versions of planar minSAT and maxSAT.

For a SAT instance with variables V and clauses C, the graph $G = (V \cup C, E)$ of the instance is the bipartite graph whose vertices are the variables and the clauses, and whose edges connect each variable to a clause whenever the variable or its negation appears in the clause. In a *planar* SAT, G is planar. Planar SAT has been the staple starting point for hardness reduction in computational geometry. In many cases, hardness of geometric problems was proved using *restricted* hard versions of planar SAT, such as:

V-cycle SAT: G remains planar after adding a cycle through V (G is no longer bipartite)

VC-cycle SAT: G remains planar after adding a cycle through $V \cup C$ (this version, as well as V-cycle SAT were defined already in the original paper on planar SAT [28])

Separable SAT: A further restriction of V-cycle SAT: for any variable x, the V-cycle separates clauses containing x from the clauses containing $\overline{x}$; in other words, no variable x has an x-containing clause and a $\overline{x}$-containing clause on the same side of the V-cycle (this version is from [28, Lemma 1], but has no name there; we take the name from [40])

Monotone SAT: In any clause, all variables are either non-negated or all variables are negated (this version is defined for general, not only for planar SAT).

See [15, 36, 40] for in-depth treatment of restricted planar SAT versions and their uses.

When proving hardness of Geometric Secluded Path in Section 2 (Theorem 1) we spent considerable effort on dealing with crossings between variable–clause connectors. A natural question is why we did not reduce from planar minSAT. The answer is that to avoid crossings, our reduction should better start from separable minSAT (Fig. 7, left), so that for any variable x, the connections from literal x reside on one side of the Christmas tree and the connections from $\overline{x}$ – on its other side (otherwise, a connection from, say, $\overline{x}$ would cross the Christmas tree itself; Fig. 7, middle). However:

▶ **Theorem 5.** *Separable minSAT can be solved in polynomial time.*

Proof. Let A be the clauses on one side of the variable chain and $B = C \setminus A$ – the clauses on the other side. Construct the "clause conflict" graph H [30] whose vertices are the clauses and whose edges connect two clauses whenever one contains the negation of a literal in the other (Fig. 7, right). For any edge, at least one of the conflicting clauses will be satisfied in any truth assignment; thus, every edge in the graph will be incident to a satisfied clause. In particular, solving the minSAT is equivalent to finding minimum vertex cover (VC) in H. By the separability, for any variable x, all clauses with x are in A and all clauses with $\overline{x}$ are in B (or vice versa); thus, any edge of H connects a clause in A with a clause in B, i.e., H is bipartite, and the VC in it can be found in polynomial time. ◀

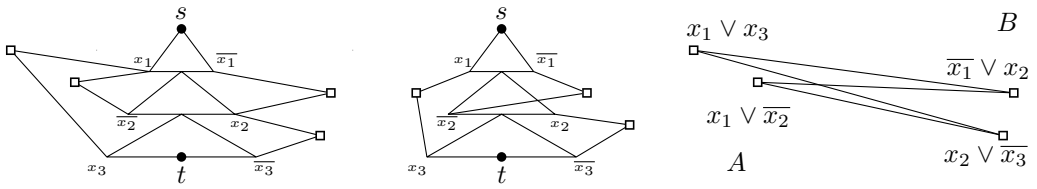


Figure 7 Left: Reduction from separable minSAT to Geometric Secluded Path would have no crossings (note that some variables have their negations on one side of the Christmas tree, while others – on the other; this is fine, since the definition of separable SAT requires separability *locally* for each variable; the separability does not have to be consistent across all variables). Middle: In non-separable minSAT, clause $\overline{x_1} \vee \overline{x_2}$ could be seen not only from s-t path via $\overline{x_2}$ but also from s-t path via x_2 due to the crossing with the Christmas tree. Right: the graph H (which happens to be $K_{2,2}$) for the instance on the left.

Note that the above proof does not use the planarity. In particular, monotone minSAT can be solved similarly: the clauses with all positive variables can form the set A and the clauses with all negative variables – set B in the graph H from the proof. We thus have:

▶ **Corollary 6.** *Monotone minSAT (planar or not) can be solved in polynomial time.*

In the full version [6], we prove NP-hardness of V- and VC-cycle min2SAT, as well as hardness of all four versions of planar max2SAT (these do not have relation to secluded paths; we give the proofs just for completeness of our treatment of planar optSAT):

▶ **Theorem 7.** *The following planar versions of max2SAT are NP-hard: V-cycle, VC-cycle, monotone, separable. V- and VC-cycle min2SAT are NP-hard.*

5 Conclusion

We studied minimum-exposure paths in polygonal domains. We showed that minimizing seen area is hard in polygons with (large number of) holes, while in polygons with a small number of holes the s-t path that sees least area can be found in polynomial time. We also gave a PTAS for finding an s-t path minimizing the integral of the seen area along the path. Finally, we discussed the connection between the geometric secluded paths and optimizing planar satisfiability, and identified hard and easy cases of planar optSAT (while the planar optSAT variants, which we proved hard, were not used in reductions in this paper, we hope that they may be useful in other settings). We conclude with some remarks on each of the problems studied.

Minimizing seen area and Secluded paths in graphs

Recall that in Secluded Path (the original, graph problem) the goal is to find an s-t path adjacent to fewest vertices of the graph (vertices of the path itself are also counted as adjacent to the path). The problem was proved hard in [8]. Our proof of hardness of Geometric Secluded Path (Theorem 1) gives an alternative proof of hardness of Secluded Path in graphs: simply remove equalizers and leakage-blocking chambers from Fig. 1 (no need to care about midway intersections and all the other geometric technicalities) and add a large number of extra vertices adjacent to each clause vertex (Fig. 8, left). While our proof is simpler than the ones in [8], it is less powerful because Chechik et al. [8] showed also hardness of approximation. In fact, the reduction in [8], shown here on Fig. 8, right, may also be seen as reduction from minSAT (in view of the connection between minSAT and VC

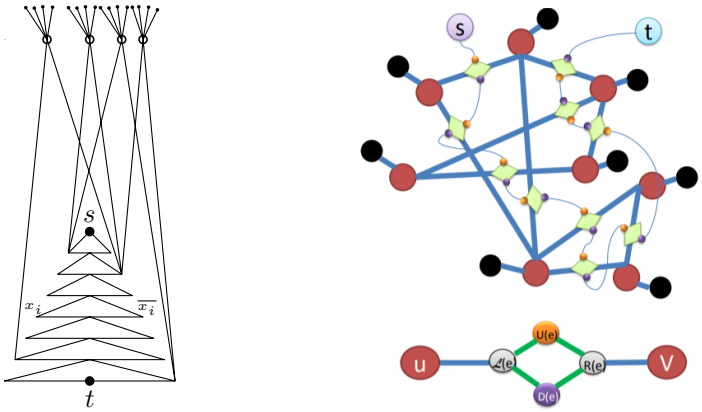


Figure 8 Left: Our reduction from min2SAT to Secluded Path. To avoid high-degree vertices at the clauses (hollow), the s-t path will go via the Christmas tree, setting the variables; the number of seen (i.e., adjacent) clause vertices is the number of satisfied clauses. Right: The reduction from VC in a graph G [8, Fig. 3]: the new graph G' has new vertices s and t, and an s-t path (thin blue) crossing all edges (thick blue) is added to G, with every crossing (lightgreen rhombi) turned into a gadget (bottom) where the s-t path chooses which vertex of G (red) the path will see; leaking into the original vertices of G (red) is prevented in G' by attaching high-weight vertices (black).

in the clause conflict graph – see proof of Theorem 5): the choices that the s-t path makes in the edges of the original graph G may be seen as setting the truth values to the variables (similarly to how the path through our Christmas tree does it).

A natural question, arising in view of the effort we spent dealing with the crossings in Section 2 when proving hardness of Geometric Secluded Path (Theorem 1), is why we did not reduce from Secluded Path in *planar* graphs. The answer is that we are not aware of a hardness result for the problem in planar graphs. Indeed, even though Chechik et al.'s hardness proof for *general* graphs (refer to Fig. 8, right) could reduce from VC in a *planar* graph G, in order to keep the planarity also in the resulting graph G' (in which the secluded s-t path is sought), the added path (crossing all edges of G) must cross each edge exactly once, meaning that it is an Euler path in the planar dual of G, meaning that the dual has vertices of even degree only, meaning that G has faces with even number of edges, meaning it has only even cycles, meaning it is bipartite, meaning VC is polynomial in it. (Strictly speaking, since we need only an Euler path through the edges, not Euler cycle, G may have 2 odd faces – we believe VC is still polynomial in such graphs).

The PTAS for integral seen area minimization

Several remarks on the complexity of our solution:

- A faster algorithm for our problem could potentially be obtained by using a "1D" discretization of edges of the visibility decomposition (instead of creating a 2D "grid" of regions, as we do), as done in many algorithms for WRP (and related problems on minimizing path integral [1,45]). Such a solution, however, would require knowing the optimal path connecting points on the boundary of the same cell of the decomposition. This, may be quite complicated, as it amounts to minimizing the integral of a function with $\Omega(n)$ terms, for which an analytical solution might not exist (though an approximation may be possible).
- An algorithm for WRP with regions whose boundaries are curves of constant algebraic degree could be interesting and would lead to a solution of our problem just using the generic scheme from Section 3.1. The biggest stumbling block for the design of such an

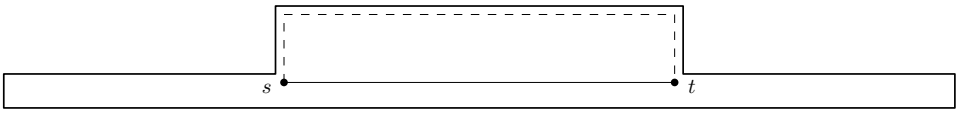


Figure 9 Shortest s-t path (solid) sees the niches behind s and t for its whole length; stepping to the side (dashed path) decreases the integral exposure.

algorithm may be the non-convexity of the regions, implying that a segment between two points on the boundary of a region is not guaranteed to stay inside the region. It may be possible that WRP techniques could be adapted to handle our regions from Section 3.1 by approximating their boundaries with piecewise-linear functions (since we are looking only for a $(1+\varepsilon)$-optimal path, the fineness of such piecewise-linear approximation would also be controlled by ε).

- Since our problem is an extension of WRP to the case of continuously changing weight, it may be tricky to establish hardness of the problem, as the complexity of WRP has remained open for many years (see [14] for a recent proof of algebraic complexity of WRP). Differently from 0/1 exposure (Theorem 3), even in simple polygons the shortest path does not necessarily minimize the integral exposure (Fig. 9).

Optimal 2-satisfiability

Few observations on min2SAT and max2SAT:

- Monotone minSAT is an example of the tractable class of submodular function minimization [23].
- Planar max2SAT has a PTAS [13, Thm. 8.8].
- If in a separable max2SAT with VC cycle, the cycle also separates the variables at the clauses (i.e., if at each clause the connections from the two variables come from the different sides of the cycle), then the problem can be solved in polynomial time by reduction to separable min2SAT.

References

1 Pankaj K Agarwal, Kyle Fox, and Oren Salzman. An efficient algorithm for computing high-quality paths amid polygonal obstacles. *ACM Transactions on Algorithms (TALG)*, 14(4):46, 2018.

2 Lyudmil Aleksandrov, Anil Maheshwari, and J-R Sack. Determining approximate shortest paths on weighted polyhedral surfaces. *Journal of the ACM (JACM)*, 52(1):25–53, 2005.

3 Esther M Arkin, Alon Efrat, Christian Knauer, Joseph Mitchell, Valentin Polishchuk, Günter Rote, Lena Schlipf, and Topi Talvitie. Shortest path to a segment and quickest visibility queries. *Journal of Computational Geometry*, 7(2):77–100, 2016. Special issue on SoCG'15.

4 Boris Aronov, Alon Efrat, Ming Li, Jie Gao, Joseph SB Mitchell, Valentin Polishchuk, Boyang Wang, Hanyu Quan, and Jiaxin Ding. Are friends of my friends too social? limitations of location privacy in a socially-connected world. In *Proceedings of the Eighteenth ACM International Symposium on Mobile Ad Hoc Networking and Computing*, pages 280–289. ACM, 2018.

5 Boris Aronov, Leonidas J Guibas, Marek Teichmann, and Li Zhang. Visibility queries and maintenance in simple polygons. *Discrete & Computational Geometry*, 27(4):461–483, 2002.

6 Kevin A. Buchin, Valentin Polishchuk, Leonid Sedov, and Roman Voronov. Geometric secluded paths and planar satisfiability. *CoRR*, abs/1902.06471, 2019. `arXiv:1902.06471`.

7 Svante Carlsson, Håkan Jonsson, and Bengt Nilsson. Finding the shortest watchman route in a simple polygon. *Discrete & Computational Geometry*, 22(3):377–402, 1999. doi:10.1007/PL00009467.

8 Shiri Chechik, Matthew P Johnson, Merav Parter, and David Peleg. Secluded connectivity problems. *Algorithmica*, 79(3):708–741, 2017.

9 Siu-Wing Cheng, Jiongxin Jin, and Antoine Vigneron. Triangulation refinement and approximate shortest paths in weighted regions. In *Proceedings of the twenty-sixth annual ACM-SIAM symposium on Discrete algorithms*, pages 1626–1640. SIAM, 2014.

10 Siu-Wing Cheng, Hyeon-Suk Na, Antoine Vigneron, and Yajun Wang. Approximate shortest paths in anisotropic regions. *SIAM Journal on Computing*, 38(3):802–824, 2008.

11 Siu-Wing Cheng, Hyeon-Suk Na, Antoine Vigneron, and Yajun Wang. Querying approximate shortest paths in anisotropic regions. *SIAM Journal on Computing*, 39(5):1888–1918, 2010.

12 Otfried Cheong, Alon Efrat, and Sariel Har-Peled. Finding a guard that sees most and a shop that sells most. *Discrete & Computational Geometry*, 37(4):545–563, 2007.

13 Nadia Creignou, Sanjeev Khanna, and Madhu Sudan. *Complexity classifications of boolean constraint satisfaction problems*, volume 7. SIAM, 2001.

14 Jean-Lou De Carufel, Carsten Grimm, Anil Maheshwari, Megan Owen, and Michiel Smid. A note on the unsolvability of the weighted region shortest path problem. *Computational Geometry*, 47(7):724–727, 2014.

15 Erik Demaine. Algorithmic lower bounds: Fun with hardness proofs. MIT OCW.

16 Hristo N Djidjev. Efficient computation of minimum exposure paths in a sensor network field. In *International Conference on Distributed Computing in Sensor Systems*, pages 295–308. Springer, 2007.

17 Hao Feng, Lei Luo, Yong Wang, Miao Ye, and Rongsheng Dong. A novel minimal exposure path problem in wireless sensor networks and its solution algorithm. *International Journal of Distributed Sensor Networks*, 12(8):1550147716664245, 2016.

18 Fedor V Fomin, Petr A Golovach, Nikolay Karpov, and Alexander S Kulikov. Parameterized complexity of secluded connectivity problems. *Theory of Computing Systems*, 61(3):795–819, 2017.

19 Laxmi Gewali, Alex C. Meng, Joseph S. B. Mitchell, and Simeon C. Ntafos. Path planning in 0/1/infinity weighted regions with applications. In Herbert Edelsbrunner, editor, *Proceedings of the Fourth Annual Symposium on Computational Geometry, Urbana-Champaign, IL, USA, June 6-8, 1988*, pages 266–278. ACM, 1988.

20 Subir Ghosh. *Visibility Algorithms in the Plane*. Cambridge University Press, New York, NY, USA, 2007.

21 J.E. Goodman and J. O'Rourke, editors. *Handbook of Discrete and Computational Geometry*. Discrete Mathematics and Its Applications. Taylor & Francis, 2nd edition, 2004.

22 Branko Grünbaum and Geoffrey C Shephard. Pick's theorem. *The American Mathematical Monthly*, 100(2):150–161, 1993.

23 Dorit S Hochbaum. Complexity and approximations for submodular minimization problems on two variables per inequality constraints. *Discrete Applied Mathematics*, 250:252–261, 2018.

24 Rajasekhar Inkulu and Sanjiv Kapoor. A polynomial time algorithm for finding an approximate shortest path amid weighted regions. *Preprint*, 2015.

25 Irina Kostitsyna, Maarten Löffler, Valentin Polishchuk, and Frank Staals. On the complexity of minimum-link path problems. *JoCG*, 8(2):80–108, 2017. Special Issue on SoCG'16. URL: http://jocg.org/index.php/jocg/article/view/328, doi:10.20382/jocg.v8i2a5.

26 Niel Lebeck, Thomas Mølhave, and Pankaj K Agarwal. Computing highly occluded paths on a terrain. In *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 14–23. ACM, 2013.

27 Niel Lebeck, Thomas Mølhave, and Pankaj K Agarwal. Computing highly occluded paths using a sparse network. In *Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 3–12. ACM, 2014.

28 David Lichtenstein. Planar formulae and their uses. *SIAM journal on computing*, 11(2):329–343, 1982.

29 Max-Jonathan Luckow and Till Fluschnik. On the computational complexity of length-and neighborhood-constrained path problems. *arXiv preprint*, 2018. arXiv:1808.02359.

30 Madhav V Marathe and SS Ravi. On approximation algorithms for the minimum satisfiability problem. *Information Processing Letters*, 58(1):23–29, 1996.

31 J. Mitchell, G. Rote, and G. Woeginger. Minimum-link paths among obstacles. *Alg-ca'92*, 8(1):431–459, 1992.

32 Joseph Mitchell, Valentin Polishchuk, and Mikko Sysikaski. Minimum-link paths revisited. *CGTA*, 47(6):651–667, 2014. doi:10.1016/j.comgeo.2013.12.005.

33 Joseph SB Mitchell and Christos H Papadimitriou. The weighted region problem: finding shortest paths through a weighted planar subdivision. *Journal of the ACM (JACM)*, 38(1):18–73, 1991.

34 Joseph O'Rourke. *Art Gallery Theorems and Algorithms*. The International Series of Monographs on Computer Science. Oxford University Press, New York, NY, 1987.

35 Christos H Papadimitriou. An algorithm for shortest-path motion in three dimensions. *Information Processing Letters*, 20(5):259–263, 1985.

36 Alexander Pilz. Planar 3-sat with a clause/variable cycle. *arXiv preprint*, 2017. arXiv:1710.07476.

37 Valentin Polishchuk and Leonid Sedov. Gender-aware facility location in multi-gender world. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 100. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

38 Yuning Song, Liang Liu, Huadong Ma, Athanasios V Vasilakos, et al. A biology-based algorithm to minimal exposure problem of wireless sensor networks. *IEEE Trans. Network and Service Management*, 11(3):417–430, 2014.

39 Subhash Suri. A linear-time algorithm for minimum link paths inside a simple polygon. *Computer Vision, Graphics and Image Processing*, 35(1):99–110, 1986.

40 Simon Tippenhauer and Wolfgang Muzler. On planar 3-sat and its variants. *Fachbereich Mathematik und Informatik der Freien Universitat Berlin*, 2016.

41 Csaba D Toth, Joseph O'Rourke, and Jacob E Goodman. *Handbook of discrete and computational geometry*. Chapman and Hall/CRC, 2017.

42 René van Bevern, Till Fluschnik, and Oxana Yu. Tsidulko. Parameterized algorithms and data reduction for safe convoy routing. In *18th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems, ATMOS 2018, August 23-24, 2018, Helsinki, Finland*, pages 10:1–10:19, 2018.

43 Giacomino Veltri, Qingfeng Huang, Gang Qu, and Miodrag Potkonjak. Minimal and maximal exposure path algorithms for wireless embedded sensor networks. In *Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 40–50. ACM, 2003.

44 Haitao Wang. Quickest visibility queries in polygonal domains. In Boris Aronov and Matya Katz, editors, *33rd International Symposium on Computational Geometry, SoCG 2017, July 4-7, 2017, Brisbane, Australia*, volume 77 of *LIPIcs*, pages 61:1–61:16. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017.

45 Ron Wein, Jur Van Den Berg, and Dan Halperin. Planning high-quality paths and corridors amidst obstacles. *The International Journal of Robotics Research*, 27(11-12):1213–1231, 2008.

46 Ron Wein, Jur Van Den Berg, and Dan Halperin. Planning near-optimal corridors amidst obstacles. In *Algorithmic Foundation of Robotics VII*, pages 491–506. Springer, 2008.

47 Ron Wein, Jur P Van den Berg, and Dan Halperin. The visibility–voronoi complex and its applications. *Computational Geometry*, 36(1):66–87, 2007.

The Next 350 Million Knots

Benjamin A. Burton
The University of Queensland, Brisbane, Australia

Abstract

The tabulation of all prime knots up to a given number of crossings was one of the founding problems of knot theory in the 1800s, and continues to be of interest today. Here we extend the tables from 16 to 19 crossings, with a total of 352 152 252 distinct non-trivial prime knots.

The tabulation has two major stages: (1) a combinatorial enumeration stage, which involves generating a provably sufficient set of candidate knot diagrams; and (2) a computational topology stage, which involves identifying and removing duplicate knots, and certifying that all knots that remain are topologically distinct. In this paper we describe the many different algorithmic components in this process, which draw on graph theory, hyperbolic geometry, knot polynomials, normal surface theory, and computational algebra. We also discuss the algorithm engineering challenges in solving difficult topological problems systematically and reliably on hundreds of millions of inputs, despite the fact that no reliably fast algorithms for these problems are known.

2012 ACM Subject Classification Mathematics of computing → Topology

Keywords and phrases Computational topology, knots, 3-manifolds, implementation

Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.25

Supplementary Material `https://regina-normal.github.io/data.html`

Funding Supported by the Australian Research Council, Discovery Project DP150104108.

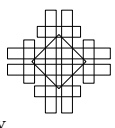
1 Introduction

Knot tabulation is one of the oldest problems in knot theory, dating back to the 1800s when it was thought to be fundamental to the structure of atoms [1]. The last major tabulation work in the literature dates back to 1998, where Hoste, Thistlethwaite and Weeks enumerate all 1 701 936 prime knots with ≤ 16 crossings [23]. In this paper we make the next major leap, tabulating all 352 152 252 topologically distinct non-trivial prime knots with ≤ 19 crossings.

The history of knot theory dates back to Gauss [1], but the first serious tabulation was done by Tait, Kirkman and Little from 1876–1899, culminating in Little's tables for ≤ 10 crossings [25]. Conway extended the tables to 11 crossings in the 1960s [10], and in 1974 Perko unearthed a duplicate 10-crossing pair that had until then gone unnoticed. Dowker and Thistlethwaite extended them to 13 crossings in the 1980s [14], followed by Hoste, Thistlethwaite and Weeks' most recent 16-crossing tables in 1998.

Our goal then is to build a *census* of all non-trivial prime knots that can be drawn with ≤ 19 crossings. Each knot should appear in the census exactly once, up to topological equivalence and/or reflection, using a diagram that uses the fewest crossings possible.

We build our census in two stages. In the first stage, described in section 3, we enumerate a set of *candidate diagrams* that are guaranteed to include every knot in the census, but which may also include unwanted (non-prime) knots and/or duplicates (where the same topological knot appears with different diagrams). In the second stage, described in sections 4–6, we remove duplicate and unwanted knots from our set of candidates, and certify that all remaining knots are prime and topologically distinct.

36th International Symposium on Computational Geometry (SoCG 2020).
Editors: Sergio Cabello and Danny Z. Chen; Article No. 25; pp. 25:1–25:17


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

Figure 1 Examples of knot projections.

Computationally, the first stage is enormously easier than the second, both in human effort and computational resources. The first stage uses only combinatorial tools, and took just a few days on a single machine. The second stage involves combinatorics, hyperbolic geometry, knot polynomials, normal surface theory, and computational algebra, and took several months (walltime) with the assistance of a cluster of a few hundred machines.

This project has motivation beyond the "popular science" aspect of knot tabulation (which of course is important for different reasons). An immediate use is to give topologists access to richer data sets for experimentation and exploration. More broadly, computational low-dimensional topology has made enormous progress in recent decades – despite a landscape of algorithmic problems that often range from exponential to tower-of-exponentially slow, are often enormously complex, sometimes unimplementable and occasionally undecidable, practitioners are nevertheless equipping themselves with diverse and sophisticated software tools that can solve these problems effectively in practice. This project showcases how far we have come: we are able to systematically solve complex and difficult problems with at some stages *billions* of inputs, with not one case left unresolved. In a sense, this project offers a blueprint for what large-scale topological computation can look like going into the future.

The bulk of the code is implemented as part of *Regina* [4], and can be downloaded from the master branch of the git repository [7]. We also make significant use of other software, notably *SnapPy* for hyperbolic geometry [13], *GAP* for computational algebra [18], and *plantri* for graph enumeration [3], and we note in sections 3–6 where these tools are used.

The final census includes 352 152 252 knots, including 352 151 858 hyperbolic knots, 380 satellite knots, and 14 torus knots. The full tables, including a more detailed statistical breakdown, can be downloaded from `https://regina-normal.github.io/data.html`.

The computations described here drawn on many diverse branches of mathematics, which we cannot hope to describe properly in this paper. Instead we try to give the flavour of each technique as it is used, and offer the reader references for further reading.

2 Preliminaries

A *knot* is a piecewise-linear simple closed curve in $\mathbb{R}^3$, formed from finitely many line segments. Two knots are *topologically equivalent* if one can be continuously deformed into the other in $\mathbb{R}^3$ without introducing self-intersections. A *non-trivial* knot is one that cannot be deformed into a simple planar polygon.

We typically represent a knot using a *diagram*: a projection of the knot onto the plane with only finitely many double points called *crossings* at which two "strands" of the projection cross transversely, and with no other multiple points at all. See Figure 1 for examples.

The *composition* of knots K and L is formed by cutting them open and connecting them with a bridge, as shown in Figure 2; a *prime* knot is one that cannot be expressed as the composition of two non-trivial knots.

Any two diagrams that represent equivalent knots can be connected through a sequence of *Reidemeister moves*: R1 (twist/untwist); R2 (overlap or pull apart two strands); and R3 (move a strand over a crossing). See Figure 3 for illustrations.

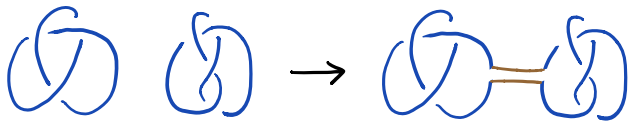


Figure 2 The composition of two knots.

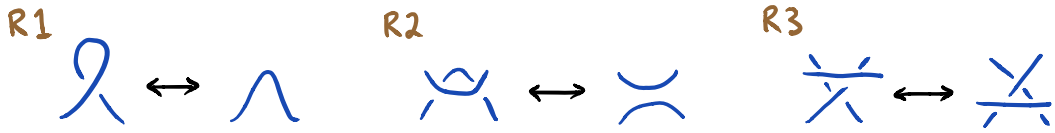


Figure 3 The three Reidemeister moves.

Knots have a close relationship with 3-manifolds. If we extend $\mathbb{R}^3$ with a point at infinity to form the 3-sphere S^3, then the *complement* of a knot K is the 3-manifold $\overline{K}$ formed by drilling out a small regular neighbourhood of K from S^3. Two knots are equivalent if and only if their complements are topologically equivalent; that is, homeomorphic [19].

Thurston's work [31] shows that every knot is exactly one of a *hyperbolic knot*, a *torus knot*, or a *satellite*. Hyperbolic knots have complements that admit a complete hyperbolic metric; torus knots are drawn on an unknotted torus (Figure 4, left); and satellites essentially embed one non-trivial knot inside the neighbourhood of another (Figure 4, right).

3 Stage one: Enumeration

The first stage of building our census is to enumerate a set of candidate diagrams, guaranteed to include each knot from our census at least once.

We do this by first building a candidate set of *model graphs* where each crossing becomes a vertex of degree 4. We reduce this set of graphs using a notion of *flype equivalence*, and then for each graph on n vertices we (essentially) resolve the vertices into crossings in each of the 2^n possible ways. The details follow in sections 3.1–3.3 below.

3.1 Enumeration of model graphs

Here we enumerate our initial set of candidate model graphs. These are all planar 4-regular graphs, and we enumerate not just the graphs but also their possible planar embeddings.

Our first observation is that embedding a graph in the plane is equivalent to embedding the graph in the *sphere* and then selecting one of the resulting cells to be the "outer cell" (i.e.,

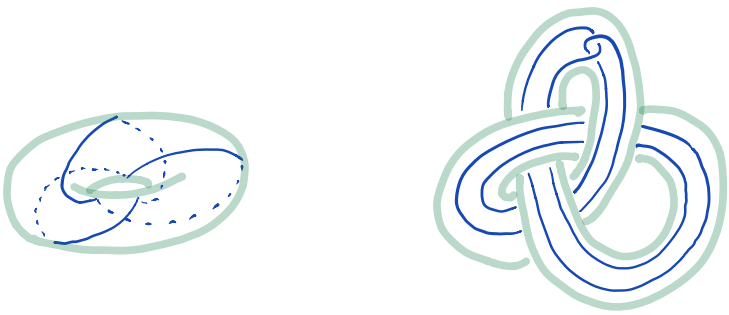


Figure 4 A torus knot and a satellite knot.

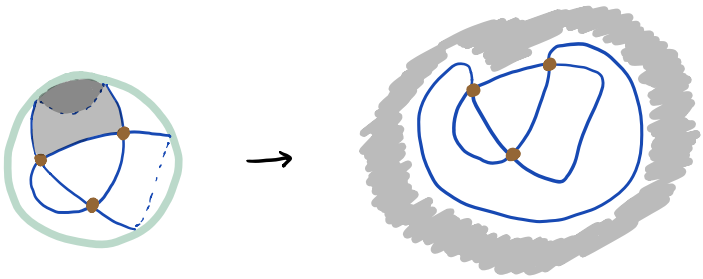


Figure 5 Converting a spherical embedding to a planar embedding.

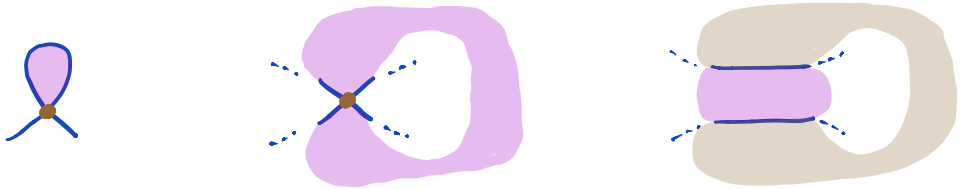


Figure 6 Cells that are not allowed for embeddings of model graphs.

the exterior of the planar embedding); see Figure 5. For our census, it does not matter which cell on the sphere becomes the outer cell: choosing a different outer cell simply corresponds to rotating the knot through 3-dimensional space. We can therefore enumerate 4-regular graphs and their embeddings in the *sphere*, which gives substantially less output.

We put further conditions on this enumeration:

1. We do not allow a cell with just one edge, since any resulting knot diagram can be simplified with move R1 (Figure 6, left).
2. We do not allow a cell that touches itself along a vertex, since any resulting knot diagram can be "untwisted" to use fewer crossings (Figure 6, centre).
3. We do not allow two cells that touch along multiple edges, since any resulting knot diagram would be a composition of two knots, and so would either be non-prime or could be simplified to fewer crossings (Figure 6, right).
4. We insist that, if we follow a path through the graph by always exiting a vertex through the opposite edge from which we enter, then this path must traverse the entire graph. This ensures that any resulting diagram represents a single connected knot, and not a link with multiple disconnected components.

We perform this graph enumeration using the software *plantri* [3]. Conditions 1–3 are supported natively through *plantri* options `-c2m2`, and for condition 4 we extend *plantri* with a simple filter that is tested before each graph is output.

The resulting set: 823 708 396 graphs with spherical embeddings.

3.2 Flype equivalence

A *flype* is a move that involves twisting a section of a knot diagram. Specifically, we begin with a connected region A of a diagram that has exactly four outgoing strands, and where two of these strands immediately meet to form a crossing. The move involves twisting region A upside-down, so that the original crossing disappears, and instead the other two outgoing strands form a new crossing in its place. See Figure 7 (left) for an illustration. We can define a similar move on a model graph, as shown in Figure 7 (right).

Figure 7 Performing a flype on a knot and a model graph.

Figure 8 Forced resolutions for subgraphs.

Tait first observed that, if two model graphs G, H are related by a flype, then any knot diagram modelled by G can be flyped into a knot diagram modelled by H with exactly the same number of crossings [23]. We can therefore partition our model graphs into equivalence classes where the graphs in each class are related by sequences of flypes, and it is enough to use just one representative from each class to build our census of knots.

Because we are dealing with hundreds of millions of graphs, we implement this using union-find [11]. The trade-off is that we need to keep all model graphs in memory simultaneously: for 19 crossings and heavily optimised data structures this required 41 GB of RAM.

The resulting set: 51 280 976 graphs with spherical embeddings.

3.3 Resolving into knot diagrams

For each vertex v of each model graph G, there are two ways of resolving v into a crossing (according to which strand runs above or below the other). This means that, if the graph G has n vertices, it resolves into 2^n knot diagrams.

This expands the size of our candidate set by many orders of magnitude, and so even at this early stage we implement some simple heuristics to reduce the output size:

- Whenever we have a subgraph σ that is (i) two parallel edges or (ii) two parallel edges that form a triangle with a third crossing, we resolve the subgraph in one of the two ways that does not allow a simplification using move R2 or a twist; see Figure 8. As a result, we use only two of the total (four or eight) possible resolutions of S.
- For each knot diagram K, we perform a short random sequence of R3 moves. If we ever obtain a diagram that can be simplified using R1 or R2, we delete K from our output.

The resulting set: 21 004 314 525 candidate knot diagrams.

4 Stage two: Uniqueness

In the second stage, we need to (i) remove duplicates, i.e., different diagrams that represent the same topological knots; (ii) remove non-prime knots (if any); and (iii) certify that all remaining knots are topologically distinct.

In hindsight, as we start stage two, our candidate set is enormous – roughly 60 times the final size of the census. Even disk space is now a serious problem: with compact representations of diagrams, the candidate set still consumes over 1 TB of space. As a result:

- We interleave tasks (i)–(iii) throughout stage two. This is because computational topology often exhibits a time/power trade-off (e.g., how fast an invariant is to compute versus how well it distinguishes different knots). How we choose to resolve this trade-off must change as our candidate set slowly shrinks but its members grow more resistant to simpler tests. Therfore we must choose the *order* of our individual tests carefully.

- Many of our algorithms work in a single read-write pass through the candidate set. We cannot hold all of our candidates in memory at once (so techniques such as union-find are completely out of reach). Even simple operations such as sorting become extremely expensive, and so must be used sparingly.

Here in section 4 we describe an initial set of operations that we perform on all candidate diagrams. The last of these initial operations splits the candidates into hyperbolic vs non-hyperbolic knots; sections 5 and 6 then outline the very different ways that we handle the remaining candidates in each category.

4.1 Exhaustive simplification via R3 moves

First we partition our candidates into equivalence classes, where the knot diagrams in each class are related by sequences of R3 moves (which do not change the number of crossings).

We do this as follows:

- For each input diagram K, we (i) use a breadth-first search to generate all possible diagrams that can be obtained through R3 moves; and then (ii) replace K with the lexicographically smallest of these diagrams. Essentially, this replaces K with a canonical representative of its equivalence class.

  This is computationally wasteful: if an equivalence class has k members then we generate the entire class k times over. We do it this way because, as described above, more sophisticated techniques such as union-find require too much memory, and this method works in a single read-write pass through the entire data set.

  In our breadth-first search we need a way to identify if a knot diagram has been seen before. For this we use *Regina*'s *knot signatures*: short strings computed in small polynomial time from a knot diagram that are the same for two diagrams if and only if the diagrams are combinatorially isomorphic. Our notion of isomorphism here treats knot diagrams as embedded in the sphere, not in the plane; see section 3.1 for details.
- We then sort the resulting list of diagrams lexicographically. This effectively groups together the members of each equivalence class by writing repeated copies of the same canonical representative, and we finish by stripping out any repeats that we see.

  With very tightly engineered data structures and a specialised large memory machine, we were able to do this sort-and-strip in memory, not on disk. However, just this sort-and-strip required 678 GB of RAM, and took a little over a day to run.

The resulting set: 7 205 537 550 candidate knot diagrams.

4.2 Attempting canonical triangulations

Next we make our first use of tools from hyperbolic geometry (despite not yet knowing which of our knots are hyperbolic). For this we work with the software *SnapPy* [13].

It is known that every hyperbolic 3-manifold with torus boundary has a *canonical cell decomposition* [15]. With appropriate subdivision this can be made into a *canonical triangulation*, and Weeks describes an algorithm for computing this triangulation that, whilst not guaranteed to terminate, works effectively in practice [32].

For each candidate knot K, we: (i) build the complement $\overline{K}$, (ii) ask *SnapPy* to find a complete hyperbolic metric on this 3-manifold $\overline{K}$, and if it is successful then we (iii) ask *SnapPy* to compute the canonical triangulation τ of $\overline{K}$.

Several things can go wrong here. First, *SnapPy* uses numerical algorithms with floating point approximations[1], and so it may incorrectly decide that $\overline{K}$ is hyperbolic and/or it may compute a triangulation τ of $\overline{K}$ that is not canonical. Nevertheless, *SnapPy* computes canonical triangulations by making local moves that preserve the topology [32], and so regardless of whether τ is canonical, it is guaranteed that τ *is* topologically the same as $\overline{K}$. Therefore, regardless of what might go wrong, we are guaranteed that if two inputs produce the same "attempted" canonical triangulation τ then the two knot complements are homeomorphic as 3-manifolds, and therefore the two knots are identical.

So: if several knots produce the same "attempted" canonical triangluation then we keep just one of these knots in our candidate set (in particular, one with the smallest number of crossings). For each knot where *SnapPy* fails to produce a hyperbolic structure or canonical triangulation, we simply keep the knot in our set.

On one triangulation at a time, *SnapPy* is extremely fast in practice; however, its algorithms are non-trivial, and for 7 billion triangulations they are far too slow to run in serial. We therefore split the candidate set into pieces which we process in parallel on a cluster, keeping both the knots and the canonical triangulations; afterwards we merge the results together using a similar sort-and-strip process as was used before.

Another point worth noting is *SnapPy* crashed ocassionally due to numerical instabilities – whilst it is enormously effective software in practice, its rare numerical instabilities become common when run through 7 billion inputs. This required specialised code that intercepted and handled crashes automatically without needing a human to babysit, restart and extract partial results from jobs on the cluster.

The resulting set: 367 000 154 candidate knot diagrams.

4.3 Separating hyperbolic from non-hyperbolic

Our next aim is to separate the hyperbolic knots from the non-hyperbolic (satellite and torus) knots, since these two classes will need very different tools going forwards.

To rigorously certify that knots are hyperbolic, we use *strict angle structures*. These are due to Casson and Rivin [28]; essentially they assign a positive internal dihedral angle to each edge of each tetrahedron of a triangulation so that (i) opposite edges in each tetrahedron have equal angles; (ii) all six angles in each tetrahedron sum to 2π; and (iii) the angles around each edge of the triangulation sum to 2π. The key fact for us is a theorem of Casson that, if an orientable 3-manifold triangulation with torus boundary has a strict angle structure, then the underlying manifold admits a complete hyperbolic metric [17].

So, for each input knot K:

- We ask *SnapPy* to find a complete hyperbolic metric on $\overline{K}$ as before, using its numerical algorithms that do not guarantee correctness. If this fails then we retriangulate $\overline{K}$ using local moves that preserve the topology and try again.
- If, after 40 retriangulations, *SnapPy* still fails to find a hyperbolic structure then we mark K as **potentially non-hyperbolic**. Note that this does not *certify* that K is non-hyperbolic: *SnapPy* might have failed, or we might be working with a "bad" triangulation that does not natively support the hyperbolic metric on $\overline{K}$.

[1] *SnapPy* is able to do verified computations, but these are too fragile to run at such an enormous scale – we return to them in later sections where the candidate set is much, much smaller.

- If *SnapPy* claims to find a hyperbolic structure on a triangulation τ of $\overline{K}$, then we attempt to *certify* that K is hyperbolic by finding a strict angle structure on τ. We do this with *Regina* using linear programming with exact arithmetic, and so the result is certified. If we do find a strict angle structure then we mark K as **certified hyperbolic**.
- If *SnapPy* finds a hyperbolic structure but *Regina* does not find a strict angle structure then we mark the knot K as **unusual**.

The resulting set: 352 160 183 certified hyperbolic knots, 14 839 971 potential non-hyperbolic knots, and no unusual knots.

5 Stage two, continued: Finishing the non-hyperbolic case

Here we finish processing the 14 839 971 potential non-hyperbolic knots. By extrapolating from smaller censuses and/or knowing what torus and satellite knots we expect to see, we would only expect a few hundred non-hyperbolic knots at most. Therefore it seems reasonable to guess that almost all of these 14 million potential non-hyperbolic knots are duplicates, and so we focus our initial efforts on stripping these duplicates out.

5.1 Exhaustive simplification, 1 extra crossing

We begin by exhaustively attempting to simplify each knot (as opposed to the randomised attempt in section 3.3). For each knot diagram K with n crossings, this involves a breadth-first search that explores all possible diagrams that can be obtained through *any* sequence of Reidemeister moves, allowing for $\leq n+1$ crossings at any stage. If we ever reach a diagram with $< n$ crossings then we know that K is not minimal, and so it must be a duplicate of another smaller knot in our set.

This is an expensive operation: the number of diagrams reachable from K is potentially exponential in n. (This is why we only run this process over our 14 million potential non-hyperbolic knots, and not our 352 million hyperbolic knots). As in section 4.1, we use knot signatures to ensure that we do not revisit the same diagram more than once.

The resulting set: 3 326 443 potential non-hyperbolic knots.

5.2 Pass moves

We next attempt simplification moves that are more complex than the standard Reidemeister moves. These are *pass moves*, which have been used with great success in knot tabulation since the late 1800s [23, 25]. A pass move involves taking a strand that passes over k consecutive crossings, and rerouting it (by moving it above the diagram) so that it passes over ℓ consecutive crossings instead; see Figure 9 for an illustration. A pass move could of course involve a similar procedure with a strand that passes *under* k consecutive crossings instead.

We are interested in finding pass moves where $\ell < k$, which reduce the total number of crossings. Our implementation is straightforward: we identify strands that pass over (or under) a maximal number of consecutive crossings, and use the Floyd-Warshall shortest path algorithm [11] to see if there is a route that crosses through fewer cells in the diagram.

The resulting set: 239 950 potential non-hyperbolic knots.

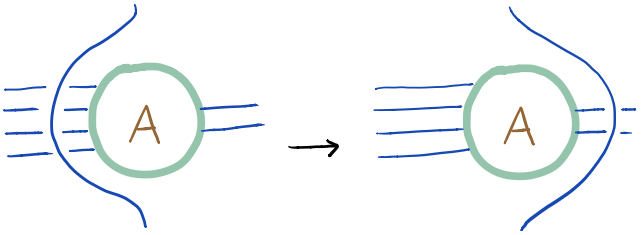


Figure 9 A pass move from $k = 4$ to $\ell = 2$ crossings.

5.3 Exhaustive simplification, 2 extra crossings

We return now to the exhaustive simplification process described in section 5.1, but this time if our input diagram has n crossings then we allow $\leq n + 2$ crossings at any intermediate stage. The number of potential knot diagrams that we can reach increases exponentially with the number of additional crossings that we allow, and so this process is significantly slower than in section 5.1. Nevertheless, we have an order of magnitude fewer knots in our set to begin with, and so the computation remains manageable.

The resulting set: 2 671 potential non-hyperbolic knots.

5.4 Exhaustive search for duplicates, 1 extra crossing

Next we perform another exhaustive search through sequences of Reidemeister moves. However, our aim now is not to simplify one diagram, but rather to find paths that connect different diagrams together.

To do this, for each n, we load all remaining diagrams with n crossings into memory at once, and perform simultaneous breadth-first searches with all of these diagrams as starting points. Whenever two searches connect, we know that the corresponding starting diagrams represent the same topological knot. We continue this process until we have exhausted all reachable diagrams with $\leq n + 1$ crossings.

As with exhaustive simplifiation, the number of diagrams that we could reach is exponential in n; given the large number of starting points, this causes a difficulties with both running time and memory. We are able to relieve the running time somewhat by using a multithreaded implementation of our breadth-first search.

Memory, however, is a serious problem that for now stops us from going beyond $n + 1$ crossings. We resolve this in the next section by splitting the knots into smaller classes.

The resulting set: 2 011 potential non-hyperbolic knots.

5.5 Separating using HOMFLY-PT polynomials

We pause now to split the remaining knots into smaller classes, where it is guaranteed that knots from different classes are distinct. We do this using the HOMFLY-PT polynomial [16, 27], a polynomial invariant that is reasonably strong, has a simple combinatorial interpretation, and can be computed in $O(2^n)$ time (which is reasonably fast in the landscape of knot invariants).[2] In fact the HOMFLY-PT can be computed in sub-exponential time [6], but Kauffman's $O(2^n)$ skein-template algorithm [24] is enough for our purposes.

The resulting set: 2 011 potential non-hyperbolic knots, split into 412 classes.

[2] For each knot K we actually compute both the HOMFLY-PT polynomial of K and the HOMFLY-PT polynomial of the reflection of K, and take whichever is lexicographically smaller. This is necessary because, unlike our census, the HOMFLY-PT polynomial is sensitive to reflection.

5.6 Exhaustive search for duplicates, 2 or 3 extra crossings

We now perform a fresh exhaustive search for duplicates, as in section 5.4, but this time exhausting all reachable diagrams with $\leq n + 3$ crossings. Each extra crossing increases our running time and memory by an order of magnitude; however, we work around this problem by performing a separate exhaustive search only within each HOMFLY-PT class. Since the classes are individually very small, the computation remains manageable.

The results are very pleasing: for *every* one of our HOMFLY-PT classes, we are able to show that *every* diagram is a duplicate of the same knot. This leaves us with just one knot per class, and therefore we have certified that all of our remaining knots are distinct.

The resulting set: 412 potential non-hyperbolic knots, all certified as distinct.

5.7 Certifying hyperbolicity, again

We do not yet know for certain that all 412 of our knots are non-hyperbolic. Since certifying *non*-hyperbolicity is an expensive process, we pass our remaining 412 knots through *SnapPy* once more in case we were unlucky with the random retriangulations the first time around. This follows exactly the same process as in section 4.3, just with different random seeds.

This time we are luckier: the hyperbolic structures and strict angle structures are able to certify 18 of our knots as hyperbolic. The other 394 knots remain potentially non-hyperbolic.

The resulting set: 394 potential non-hyperbolic knots; 18 *new* certified hyperbolic knots.

5.8 Certifying non-hyperbolicity

We now hope that our 394 *potential* non-hyperbolic knots are indeed non-hyperbolic, and so we turn our attention to certifying this.

We begin with the torus knots. These are well-understood: they are completely classified, and it is known exactly how many crossings each torus knot needs. We therefore know in advance that there should be exactly 14 torus knots in our census with ≤ 19 crossings, and we can compute their HOMFLY-PT polynomials. These 14 polynomials indeed appear in our list, and so we know that the corresponding 14 knots are non-hyperbolic.

The remaining 380 knots, if non-hyperbolic, must all be satellites. To certify that a knot K is a satellite, it is enough to find a torus τ embedded in $\overline{K}$ with the property that, if we cut $\overline{K}$ open along τ, the piece containing the boundary of $\overline{K}$ is not the product Torus $\times\, I$, and the other piece is not the solid torus [31].

We use *normal surface theory* to find candidate tori. See [21] for an overview of normal surfaces; the key fact for us is that "important" surfaces in a 3-manifold often appear in the set of *vertex normal surfaces*, which can be constructed using techniques from polytope theory and linear programming. We enumerate this set in *quad-closed coordinates*, which are optimised for working with knot complements; see [9] for details.

For each candidate torus, we use *Regina*'s implementation of solid torus recognition, which – though exponential time – is extremely efficient in practice [8]. To test for Torus $\times\, I$, we fill one of the boundaries with a solid torus in three different ways: by a theorem of Haraway [20], if none of these fillings are solid tori then the manifold is not Torus $\times\, I$.

Although there is no guarantee that the torus we seek *must* appear amongst our candidate set of vertex normal surfaces, this indeed happens for all of our 380 remaining knots.

The resulting set: 394 *certified* non-hyperbolic knots (14 torus knots, 380 satellites).

Figure 10 A sample tangle, and a tangle that includes a knot composition.

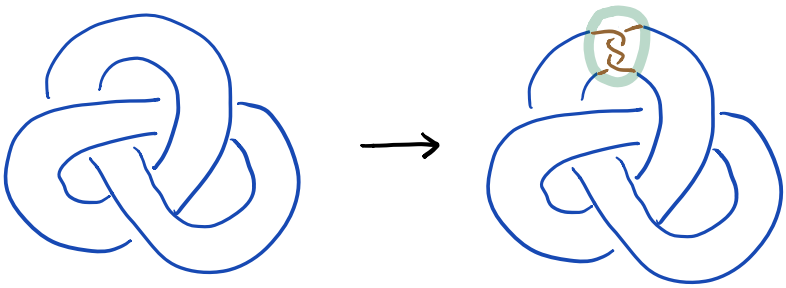


Figure 11 Inserting a tangle into the double of a trefoil.

5.9 Identifying satellites and certifying primeness

Our final step is to identify the exact structure of our 380 satellites. Our method here is simple: we predict which satellites we *expect* to see, explicitly construct them, and observe that these are the same as the 380 satellites in our census.

Our constructions are based on *tangles*. These are like knots, but instead of a closed loop of string, we have four immovable external endpoints connected by two pieces of string. See Figure 10 for some illustrations, or Adams [1] for a more thorough introduction.

The satellites we expect to see are formed by taking the double of the trefoil or figure eight knot (with 12 or 16 crossings respectively) and then inserting a tangle between two parallel strands, as shown in Figure 11.

The tangle we insert should use as few crossings as possible, should connect the two parallel copies of the trefoil or figure eight to form a single knot, and should not be the composition of a simpler tangle with another knot (as in Figure 10, right).

To build these tangles, we begin with Conway's *rational tangles*, a class of tangles that are fully classified and well understood [10]. We then hand-enumerate ten larger "templates", illustrated in Figure 12, and insert rational tangles into these templates so that the total number of crossings is ≤ 7 (thus the final satellites will have ≤ 19 crossings overall).

These tangles that we construct number 380 in total. From their constructions we know that they are prime satellites [12], and therefore must appear amongst the 380 satellites in our census. Computing their HOMFLY-PT polynomials shows that they are all distinct, and therefore (i) they are *precisely* the 380 satellites in our census, and (ii) the 380 satellites in our census are therefore prime. Since torus knots are also known to be prime, this certifies the primeness of all 394 non-hyperbolic knots.

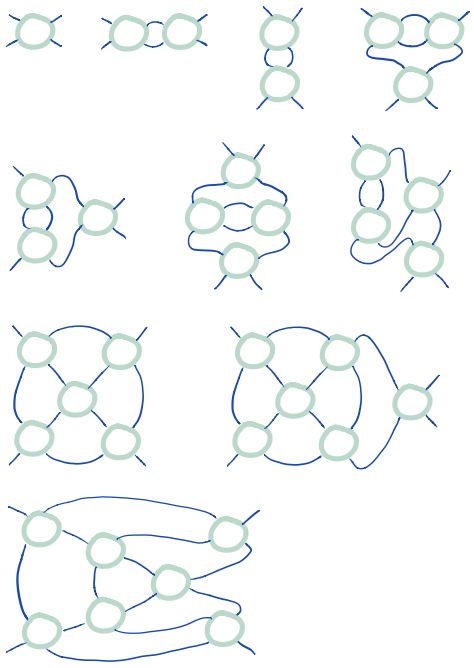


Figure 12 Templates into which we can insert rational tangles.

6 Stage two, continued: Finishing the hyperbolic case

Now we turn our attention to our 352 160 201 certified hyperbolic knots (these include the 18 extras that we picked up in section 5.7). We note that there is no need to certify primeness here, since primeness already comes as a consequence of hyperbolicity.

Extrapolating growth rates from smaller censuses suggests that our count should be close to the real number of distinct knots, and so our main task is to show that our knots are distinct. Of course we still expect to find duplicates, which we remove as they appear.

Throughout this section, we group our knots into classes, where knots in *different* classes have been certified as distinct, and knots in the *same* class might or might not be equivalent. Whenever a class shrinks down to exactly one knot, we can store that knot in the final census and forget about it in our computations here.

Throughout this section, we will count (i) the number of classes that still contain more than one knot, and (ii) the total number of knots that those classes hold.

Our intial state: 1 class and 352 160 201 knots.

6.1 Separating using HOMFLY-PT polynomials

Our first step is to compute the HOMFLY-PT polynomials of each knot, as described in section 5.6. We then sort the knots by their polynomials and use the results to split the knots into classes, one for each distinct polynomial.

As the number of crossings grows, the HOMFLY-PT polynomials become longer, with the result that simply writing all the polynomials consumes 230 GB of disk space. The subsequent sort operation was done using the GNU `sort` command, which only needed 40 GB of RAM but at the cost of doing significant work on the disk instead (which is much slower).

Our new state: 56 376 691 classes and 158 221 199 knots.

6.2 Subgroups of index 2–4

Next we turn to algebraic invariants: we can show that two knots are distinct by certifying that their complements have non-isomorphic fundamental groups.

Unfortunately, each fundamental group $\pi_1(\overline{K})$ is computed as a presentation involving generators and relations, and solving isomorphism problems on group presentations is notoriously hard (indeed, undecidable in general [26]). Instead we use the popular strategy of identifying invariants of these groups. For our application we enumerate all subgroups of index $k = 2, 3, 4$ and compute their abelian invariants, and we do the same for the *core* of $\pi_1(\overline{K})$ (the largest normal subgroup in $\pi_1(\overline{K})$). We use *GAP* [18] for our computations.

These are all computations that terminate and give exact output, though their running time increases exponentially (or worse) with the index k [30]. This is why we only use $k \leq 4$; we return to higher indices only once the number of knots remaining is substantially smaller.

Our new state: 42 091 807 classes and 115 086 342 knots.

6.3 Verified canonical triangulations

We now return to canonical triangulations, as computed in section 4.2. When used within *Sage* [29], *SnapPy* can in some cases *certify* its canonical triangulation. The underlying algorithm is based on certified interval arithmetic methods as originally used by *HIKMOT* [22] for similar topological applications.

Whether *SnapPy* and *Sage* are able to certify the canonical triangulation depends on many factors, such as numerical precision, the choice of underlying triangulation, and the geometric properties of the canonical cell decomposition. Nevertheless, if they *can* certify the canonical triangulations for two knots, we can just test the canonical triangulations for combinatorial isomorphism (a polynomial-time operation [5]): if they are isomorphic then the knots are duplicates, and if they are non-isomorphic then the knots are distinct.

When applied to our classes of potentially-equivalent knots, however, this process becomes more fragile. If *SnapPy* and *Sage* are able to certify canonical triangulations for *all* knots in a class, then we can completely resolve the class into duplicates and distinct knots. However, if the certification fails for just one knot K in the class, we cannot prove uniqueness of *any* of the knots in the class. We are still able to strip out duplicates where the canonical triangulations are isomorphic, but otherwise all knots in the class must stay – even if some of them are known to be pairwise distinct – because our problematic knot K could still be equivalent to any of them.

A technical problem with this process was a memory leak when using *SnapPy* with *Sage* (known to the *SnapPy* authors, but difficult to fix). As a result, the inputs had to be processed in small batches so the memory leaks did not accumulate too badly – an inconvenience for 115 million inputs, but nevertheless manageable.

Our new state: 7 086 classes and 21 221 knots, after removing 2 939 duplicates.

6.4 Exhaustive search for duplicates, 1 or 2 extra crossings

Next, within each class, we perform an exhaustive search for duplicates by trying all possible sequences of Reidemeister moves, allowing for at most two extra crossings at any stage. This is the same procedure seen in section 5.4 for non-hyperbolic knots.

Our new state: 3 727 classes and 12 715 knots, after removing 5 147 duplicates.[3]

[3] Note that each duplicate could potentially remove *two* knots from consideration, since removing the duplicate could leave a class with just one knot that is then moved into the final census.

6.5 Subgroups of index 5

We return again to algebraic invariants, but this time we process subgroups of index $k = 5$. Although the computations are enormously slower, we also have several orders of magnitude fewer knots to process than for indices $k \leq 4$, and so the computations remain managable.

Our new state: 479 classes and 1 040 knots.

6.6 Verified canonical triangulations, again

We return now to certified canonical triangulations with *SnapPy* and *Sage*, as seen before in section 6.3. Since our classes now contain fewer knots than they did before, it is reasonable to hope that more classes can now be resolved completely (i.e., they are not hampered by one problematic knot whose canonical triangulation cannot be certified).

Moreover, our total number of knots remaining is orders of magnitude smaller than before. We therefore make several attempts at certifying the canonical triangulation for each knot, by starting from many different retriangulations of the same knot complement.

The *SnapPy*/*Sage* memory leak remains a problem, and since we are making many attempts for each knot, our inputs need to be processed in much smaller batches. Again, since the total number of knots is much smaller than before, this is manageable.

We made several more runs through this process, beginning with 10 attempts per knot, and finishing with 300 attempts per knot.

Our new state: 45 classes and 105 knots, after removing 255 duplicates.

6.7 Subgroups of index 6

We return to algebraic invariants again, this time with index $k = 6$. As before, the computations are much slower again, but we have orders of magnitude fewer knots to process.

Our new state: 3 classes and 6 knots, all drawn in Figure 13 (each line shows one class).

6.8 Exhaustive search for duplicates, 3 extra crossings

As before, within each class we perform an exhaustive search for duplicates by trying all possible sequences of Reidemeister moves, this time allowing $\leq n + 3$ crossings at any stage.

This is enormously slow, but with only three classes remaining and a multithreaded implementation of the underlying breadth-first search, it remains (just) within feasibility.

Our new state: 1 class and 2 knots, after removing 2 duplicates. The remaining two knots are the green pair at the bottom of Figure 13.

6.9 Subgroups of index 7

For index $k = 6$, *GAP* was barely able to finish the computations and so for index $k = 7$ there is little hope. We therefore switch to *Magma* [2] for $k = 7$, since (in the author's experience) *Magma*'s running times are faster and more consistent. We did not use *Magma* until now because it is commercial software and we only had a license for a single machine, and so we could not parallelise the computations across a cluster.

Happily *Magma* was able to distinguish the final pair of knots, though even this depended on starting with the "right" group presentations. We tried three different presentations of the two groups (obtained by starting from different triangulations of the knot complements): for one pair the computations took an hour, for one pair they took closer to a day, and for one pair they did not finish after several days of running time.

Nevertheless, our new state: 0 classes and 0 knots. The census is complete!

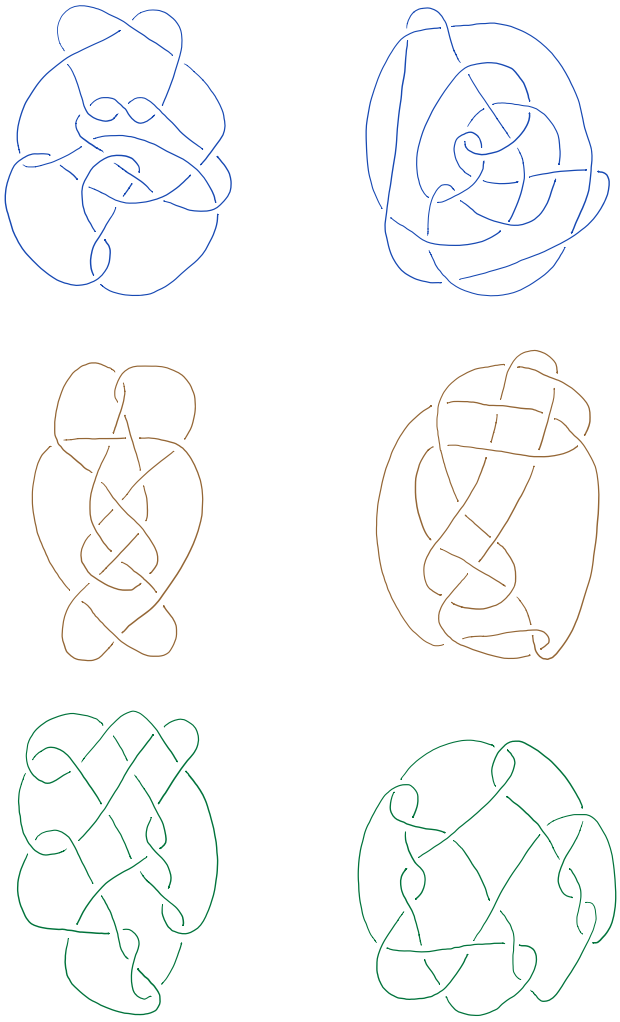


Figure 13 The three pairs of knots that remain after subgroups of index 6.

References

1. Colin C. Adams. *The Knot Book: An Elementary Introduction to the Mathematical Theory of Knots*. W. H. Freeman & Co., New York, 1994.
2. Wieb Bosma, John Cannon, and Catherine Playoust. The Magma algebra system. I. The user language. *J. Symbolic Comput.*, 24(3-4):235–265, 1997. Computational algebra and number theory (London, 1993). doi:10.1006/jsco.1996.0125.
3. Gunnar Brinkmann and Brendan McKay. plantri, 2018. URL: http://users.cecs.anu.edu.au/~bdm/plantri/.
4. Benjamin A. Burton. Introducing Regina, the 3-manifold topology software. *Experiment. Math.*, 13(3):267–272, 2004.
5. Benjamin A. Burton. Simplification paths in the Pachner graphs of closed orientable 3-manifold triangulations. Preprint, arXiv:1110.6080, October 2011.
6. Benjamin A. Burton. The HOMFLY-PT polynomial is fixed-parameter tractable. In Bettina Speckmann and Csaba D. Tóth, editors, *34th International Symposium on Computational Geometry*, volume 99 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages 18:1–18:14, Dagstuhl, Germany, 2018. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern. doi:10.4230/LIPIcs.SoCG.2018.18.

7 Benjamin A. Burton, Ryan Budney, William Pettersson, et al. Regina: Software for low-dimensional topology, 1999–2019. URL: http://regina-normal.github.io/.

8 Benjamin A. Burton and Melih Ozlen. A fast branching algorithm for unknot recognition with experimental polynomial-time behaviour. To appear in Math. Program., arXiv:1211.1079, November 2012.

9 Benjamin A. Burton and Stephan Tillmann. Computing closed essential surfaces in 3-manifolds. Preprint, arXiv:1812.11686, December 2018.

10 J. H. Conway. An enumeration of knots and links, and some of their algebraic properties. In *Computational Problems in Abstract Algebra (Proc. Conf., Oxford, 1967)*, pages 329–358. Pergamon, Oxford, 1970.

11 Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT Press, Cambridge, MA, 3rd edition, 2009.

12 Peter R. Cromwell. *Knots and links*. Cambridge University Press, Cambridge, 2004. doi:10.1017/CBO9780511809767.

13 Marc Culler, Nathan M. Dunfield, and Jeffrey R. Weeks. SnapPy, a computer program for studying the geometry and topology of 3-manifolds, 1991–2013. URL: http://snappy.computop.org/.

14 C. H. Dowker and Morwen B. Thistlethwaite. Classification of knot projections. *Topology Appl.*, 16(1):19–31, 1983. doi:10.1016/0166-8641(83)90004-4.

15 D. B. A. Epstein and R. C. Penner. Euclidean decompositions of noncompact hyperbolic manifolds. *J. Differential Geom.*, 27(1):67–80, 1988.

16 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. doi:10.1090/S0273-0979-1985-15361-3.

17 David Futer and François Guéritaud. From angled triangulations to hyperbolic structures. In *Interactions Between Hyperbolic Geometry, Quantum Topology and Number Theory*, volume 541 of *Contemp. Math.*, pages 159–182. Amer. Math. Soc., Providence, RI, 2011.

18 The GAP Group. *GAP – Groups, Algorithms, and Programming*, 2019. URL: https://www.gap-system.org.

19 C. McA. Gordon and J. Luecke. Knots are determined by their complements. *J. Amer. Math. Soc.*, 2(2):371–415, 1989.

20 Robert C. Haraway, III. Determining hyperbolicity of compact orientable 3-manifolds. Preprint, arXiv:1410.7115, October 2014.

21 Joel Hass, Jeffrey C. Lagarias, and Nicholas Pippenger. The computational complexity of knot and link problems. *J. Assoc. Comput. Mach.*, 46(2):185–211, 1999.

22 Neil Hoffman, Kazuhiro Ichihara, Masahide Kashiwagi, Hidetoshi Masai, Shin'ichi Oishi, and Akitoshi Takayasu. Verified computations for hyperbolic 3-manifolds. *Exp. Math.*, 25(1):66–78, 2016. doi:10.1080/10586458.2015.1029599.

23 Jim Hoste, Morwen Thistlethwaite, and Jeff Weeks. The first 1,701,936 knots. *Math. Intelligencer*, 20(4):33–48, 1998.

24 Louis H. Kauffman. State models for link polynomials. *Enseign. Math. (2)*, 36(1-2):1–37, 1990.

25 C. N. Little. Non-alternate ± knots. *Trans. Royal Soc. Edinburgh*, 39:771–778, 1900.

26 A. A. Markov. Insolubility of the problem of homeomorphy. In *Proc. Internat. Congress Math. 1958*, pages 300–306. Cambridge Univ. Press, New York, 1960.

27 Józef H. Przytycki and Paweł Traczyk. Invariants of links of Conway type. *Kobe J. Math.*, 4(2):115–139, 1988.

28 Igor Rivin. Euclidean structures on simplicial surfaces and hyperbolic volume. *Ann. of Math. (2)*, 139(3):553–580, 1994.

29 The Sage Developers. *SageMath, the Sage Mathematics Software System*, 2018. URL: https://www.sagemath.org/.

30 Charles C. Sims. *Computation with Finitely Presented Groups*, volume 48 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, Cambridge, 1994. doi:10.1017/CBO9780511574702.

31 William P. Thurston. Three-dimensional manifolds, Kleinian groups and hyperbolic geometry. *Bull. Amer. Math. Soc. (N.S.)*, 6(3):357–381, 1982.

32 Jeffrey R. Weeks. Convex hulls and isometries of cusped hyperbolic 3-manifolds. *Topology Appl.*, 52(2):127–149, 1993.

Elder-Rule-Staircodes for Augmented Metric Spaces

Chen Cai
Department of Computer Science and Engineering, The Ohio State University, Columbus, OH, USA
cai.507@osu.edu

Woojin Kim
Department of Mathematics, The Ohio State University, Columbus, OH, USA
kim.5235@osu.edu

Facundo Mémoli
Department of Mathematics and Department of Computer Science and Engineering,
The Ohio State University, Columbus, OH, USA
memoli@math.osu.edu

Yusu Wang
Department of Computer Science and Engineering, The Ohio State University, Columbus, OH, USA
yusu@cse.ohio-state.edu

Abstract

An augmented metric space (X, d_X, f_X) is a metric space (X, d_X) equipped with a function $f_X : X \to \mathbb{R}$. It arises commonly in practice, e.g, a point cloud X in $\mathbb{R}^d$ where each point $x \in X$ has a density function value $f_X(x)$ associated to it. Such an augmented metric space naturally gives rise to a 2-parameter filtration. However, the resulting 2-parameter persistence module could still be of wild representation type, and may not have simple indecomposables.

In this paper, motivated by the *elder-rule* for the zeroth homology of a 1-parameter filtration, we propose a barcode-like summary, called the *elder-rule-staircode*, as a way to encode the zeroth homology of the 2-parameter filtration induced by a finite augmented metric space. Specifically, given a finite (X, d_X, f_X), its elder-rule-staircode consists of $n = |X|$ number of staircase-like blocks in the plane. We show that the fibered barcode, the fibered merge tree, and the graded Betti numbers associated to the zeroth homology of the 2-parameter filtration induced by (X, d_X, f_X) can all be efficiently computed once the elder-rule-staircode is given. Furthermore, for certain special cases, this staircode corresponds exactly to the set of indecomposables of the zeroth homology of the 2-parameter filtration. Finally, we develop and implement an efficient algorithm to compute the elder-rule-staircode in $O(n^2 \log n)$ time, which can be improved to $O(n^2\alpha(n))$ if X is from a fixed dimensional Euclidean space $\mathbb{R}^d$, where $\alpha(n)$ is the inverse Ackermann function.

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

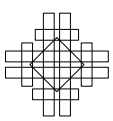
Keywords and phrases Persistent homology, Multiparameter persistence, Barcodes, Elder rule, Hierarchical clustering, Graded Betti numbers

Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.26

Related Version A full version of this paper is available at `https://arxiv.org/abs/2003.04523`.

Funding This work is supported by NSF grants DMS-1723003, CCF-1740761, DMS-1547357, and IIS-1815697.

Acknowledgements The authors thank to the anonymous reviewers who made a number of helpful comments to improve the paper. Also, CC and WK thank Cheng Xin for helpful discussions.



1 Introduction

An augmented metric space (X, d_X, f_X) is a metric space (X, d_X) equipped with a function $f_X : X \to \mathbb{R}$. It arises commonly in practice: e.g, a point cloud X in $\mathbb{R}^d$ where each point has a density function value f_X associated to it. Studying the hierarchical clustering induced in this setting has attracted much attention recently [2, 8]. Another example is where $X = V$ equals to the vertex set of a graph $G = (V, E)$, d_X represents certain graph-induced metric on X (e.g, the diffusion distance induced by G), and f_X is some descriptor function (e.g, discrete Ricci curvature) at graph nodes. This graph setting occurs often in practice for graph analysis applications, where G can be viewed as a skeleton of a hidden domain. When summarizing or characterizing G, one wishes to take into consideration both the metric structure of this domain and node attributes. Given that persistence-based summaries from only the edge weights or from only node attributes have already shown promise in graph classification (e.g, [5, 9, 18, 30]), it would be highly desirable to incorporate (potentially more informative) summaries encoding both types of information to tackle such tasks. In brief, we wish to develop topological invariants induced from such augmented metric spaces.

On the other hand, an augmented metric space naturally gives rise to a 2-parameter filtration (by filtering both via f_X and via distance d_X; see Definition 4). However, while a standard (1-parameter) filtration and its induced persistence module has persistence diagram as a complete discrete invariant, multi-parameter persistence modules do not have such complete discrete invariant [6, 13]. The 2-parameter persistence module induced from an augmented metric space may still be of wild representation type, and may not have simple indecomposables [2]. Several recent work instead consider informative (but not necessarily complete) invariants for multiparameter persistence modules [15, 19, 24, 26]. In particular, RIVET [24] provides an interactive visualization of the barcodes of 1-dimensional slices of an input 2-parameter persistence module M, called the *fibered barcode*. This interactivity uses the *graded Betti numbers* of M, another invariant for the 2-parameter persistence module.

New work. We propose a barcode-like summary, called the *elder-rule-staircode*, as a way to encode the zeroth homology of the 2-parameter filtration induced by a finite augmented metric space. Specifically, given a finite (X, d_X, f_X), its elder-rule-staircode consists of $n = |X|$ number of staircase-like blocks of $O(n)$ descriptive complexity in the plane. The development of the elder-rule-staircode is motivated by the elder-rule behind the construction of persistence pairing for a 1-parameter filtration [16]. For the 1-parameter case, *barcodes* [31] can be obtained by the decomposition of persistence modules in the realm of commutative algebra, or equivalently, by applying the elder-rule which is flavored with combinatorics or order theory. As we describe in Section 4, our elder-rule-staircodes are obtained by adapting the elder-rule for treegrams arisen from 1-parameter filtration.

Interestingly, we show that our elder-rule-staircode encodes much of topological information of the 2-parameter filtration induced by (X, d_X, f_X). In particular, the fibered barcodes, the fibered treegrams, and the graded Betti numbers associated to the zeroth homology of the 2-parameter filtration induced by (X, d_X, f_X) can all be efficiently computed from the elder-rule-staircodes (see Theorems 8, 19 and 23). Furthermore, for certain special cases, these staircodes **correspond exactly** to the set of indecomposables of the zeroth order 2-parameter persistence module induced by (X, d_X, f_X); see Theorem 17.

Finally, in Section 6, we show that the elder-rule-staircode can be computed in $O(n^2 \log n)$ time for a finite augmented metric space (X, d_X, f_X) where $n = |X|$, and $O(n^2\alpha(n))$ time if X is from a fixed dimensional Euclidean space and d_X is Euclidean distance. We have software to compute elder-rule-staircodes and to explore / retrieve information such as fibered barcodes interactively, which is available at `https://github.com/Chen-Cai-OSU/ER-staircode`.

More on related work. The *elder-rule* is an underlying principle for extracting the persistence diagram from a persistence module induced by a nested family of simplicial complexes [16, Chapter 7]. Recently this rule has come into the spotlight again for generalizing persistence diagrams [19, 26, 27] and for addressing inverse problems in TDA [14].

The software RIVET and work of [25] can also be used to recover fibered barcodes and bigraded Betti numbers. However, for the special case of zeroth 2-parameter persistence modules induced from augmented metric spaces, our elder-rule-staircodes are simpler and more efficient to achieve these goals: In particular, given an augmented metric space containing n points, the algorithm of [25] computes the zeroth bigraded Betti numbers in $\Omega(n^3)$ time, while it takes $O(n^2 \log n)$ time using elder-rule-staircode via Theorem 24. For zeroth fibered barcodes, RIVET takes $O(n^8)$ time to compute a data structure of size $O(n^6)$ so as to support efficient query time of $O(\log n + |B^L|)$ where $|B^L|$ is the size of the fibered barcode B^L for a query line L of positive slope. Our algorithm computes elder-rule-staircode of size $O(n^2)$ in $O(n^2 \log n)$ time, after which B^L can be computed in $O(|B^L| \log n)$ time for any query line L. See the full version of this paper [4] for more detailed comparison. However, it is important to note that RIVET allows much broader inputs and can work beyond zeroth homology.

2 Persistence modules and their decompositions

First we briefly review the definition of persistence modules. Let $\mathbb{P}$ be a poset. We regard $\mathbb{P}$ as the category that has elements of $\mathbb{P}$ as objects. Also, for any $\mathbf{a}, \mathbf{b} \in \mathbb{P}$, there exists a unique morphism $\mathbf{a} \to \mathbf{b}$ if and only if $\mathbf{a} \leq \mathbf{b}$. For $d \in \mathbb{N}$, let $\mathbb{Z}^d$ be the set of d-tuples of integers equipped with the partial order defined as $(a_1, a_2, \ldots, a_d) \leq (b_1, b_2, \ldots, b_d)$ if and only if $a_i \leq b_i$ for each $i = 1, 2, \ldots, d$. The poset structure on $\mathbb{R}^d$ is defined in the same way.

We fix a certain field $\mathbb{F}$ and every vector space in this paper is over $\mathbb{F}$. Let **Vec** denote the category of *finite dimensional* vector spaces over $\mathbb{F}$.

A ($\mathbb{P}$-indexed) *persistence module* is a functor $M : \mathbb{P} \to \mathbf{Vec}$. In other words, to each $\mathbf{a} \in \mathbb{P}$, a vector space $M(\mathbf{a})$ is associated, and to each pair $\mathbf{a} \leq \mathbf{b}$ in $\mathbb{P}$, a linear map $\varphi_M(\mathbf{a}, \mathbf{b}) : M(\mathbf{a}) \to M(\mathbf{b})$ is associated. When $\mathbb{P} = \mathbb{R}^d$ or $\mathbb{Z}^d$, M is said to be a *d-parameter persistence module*. A *morphism* between $M, N : \mathbb{P} \to \mathbf{Vec}$ is a natural transformation $f : M \to N$ between M and N. That is, f is a collection $\{f_{\mathbf{a}}\}_{\mathbf{a} \in \mathbb{P}}$ of linear maps such that for every pair $\mathbf{a} \leq \mathbf{b}$ in $\mathbb{P}$, the following diagram commutes:

$$\begin{array}{ccc} M(\mathbf{a}) & \xrightarrow{\varphi_M(\mathbf{a},\mathbf{b})} & M(\mathbf{b}) \\ \downarrow{\scriptstyle f_{\mathbf{a}}} & & \downarrow{\scriptstyle f_{\mathbf{b}}} \\ N(\mathbf{a}) & \xrightarrow{\varphi_N(\mathbf{a},\mathbf{b})} & N(\mathbf{b}). \end{array}$$

Two persistence modules M and N are *isomorphic*, denoted by $M \cong N$, if there exists a natural transformation $\{f_{\mathbf{a}}\}_{\mathbf{a} \in \mathbb{P}}$ from M to N where each $f_{\mathbf{a}}$ is an isomorphism.

We now review the standard definition of barcodes, following the notations from [3].

▶ **Definition 1** (Intervals). *Let $\mathbb{P}$ be a poset. An* interval *$\mathcal{J}$ of $\mathbb{P}$ is a subset $\mathcal{J} \subset \mathbb{P}$ such that: (1) $\mathcal{J}$ is non-empty. (2) If $\mathbf{a}, \mathbf{b} \in \mathcal{J}$ and $\mathbf{a} \leq \mathbf{c} \leq \mathbf{b}$, then $\mathbf{c} \in \mathcal{J}$. (3) For any $\mathbf{a}, \mathbf{b} \in \mathcal{J}$, there is a sequence $\mathbf{a} = \mathbf{a}_0, \mathbf{a}_1, \cdots, \mathbf{a}_l = \mathbf{b}$ of elements of $\mathcal{J}$ with $\mathbf{a}_i$ and $\mathbf{a}_{i+1}$ comparable for $0 \leq i \leq l - 1$.*

For $\mathcal{J}$ an interval of $\mathbb{P}$, the *interval module* $I^{\mathcal{J}} : \mathbb{P} \to \mathbf{Vec}$ is defined as

$$I^{\mathcal{J}}(\mathbf{a}) = \begin{cases} \mathbb{F} & \text{if } \mathbf{a} \in \mathcal{J}, \\ 0 & \text{otherwise,} \end{cases} \qquad \varphi_{I^{\mathcal{J}}}(\mathbf{a}, \mathbf{b}) = \begin{cases} \mathrm{id}_{\mathbb{F}} & \text{if } \mathbf{a}, \mathbf{b} \in \mathcal{J},\ \mathbf{a} \leq \mathbf{b}, \\ 0 & \text{otherwise.} \end{cases}$$

Recall that *a multiset* is a collection in which elements may occur more than once.

▶ **Definition 2** (Interval decomposability and barcodes). *A functor $M : \mathbb{P} \to \mathbf{Vec}$ is* interval decomposable *if there exists a multiset* $\mathbf{barc}(M)$ *of intervals (Definition 1) of* $\mathbb{P}$ *such that* $M \cong \bigoplus_{\mathcal{J} \in \mathbf{barc}(M)} I^{\mathcal{J}}$. *We call* $\mathbf{barc}(M)$ *the* barcode *of* M.

By the theorem of Azumaya-Krull-Remak-Schmidt [1], such a decomposition is unique up to a permutation of the terms in the direct sum. Therefore, the multiset $\mathbf{barc}(M)$ is unique if M is interval decomposable. For $d = 1$, any $M : \mathbb{R}^d$ (or $\mathbb{Z}^d$) $\to \mathbf{Vec}$ is interval decomposable and thus $\mathbf{barc}(M)$ exists. However, for $d \geq 2$, M may not be interval decomposable.

3 Elder-rule-staircodes for augmented metric spaces

Rips bifiltration for an aug-MS. Let (X, d_X) be a metric space. For $\varepsilon \in \mathbb{R}$, the *Rips complex* $\mathcal{R}_\varepsilon(X, d_X)$ is the abstract simplicial complex defined as

$$\mathcal{R}_\varepsilon(X, d_X) = \{A \subseteq X : \text{for all } x, x' \in A,\ d_X(x, x') \leq \varepsilon\}.$$

Let **Simp** be the category of abstract simplicial complexes and simplicial maps. The *Rips filtration* is the functor $\mathcal{R}_\bullet(X, d_X) : \mathbb{R} \to \mathbf{Simp}$ defined as

$$\varepsilon \mapsto \mathcal{R}_\varepsilon(X, d_X), \text{ and } \varepsilon \leq \varepsilon' \mapsto \mathcal{R}_\varepsilon(X, d_X) \hookrightarrow \mathcal{R}_{\varepsilon'}(X, d_X).$$

▶ **Definition 3** (Augmented metric spaces). *Let (X, d_X) be a metric space and $f_X : X \to \mathbb{R}$ a function. We call the triple $\mathcal{X} = (X, d_X, f_X)$* an augmented metric space *(abbrev.* aug-MS*).*
We say that $\mathcal{X}$ is injective *if $f_X : X \to \mathbb{R}$ is an injective function.*

Throughout this paper, every (augmented) metric space will be assumed to be finite. Let $\mathcal{X} = (X, d_X, f_X)$ be an aug-MS. For $\sigma \in \mathbb{R}$, let X_σ denote the sublevel set $f_X^{-1}(-\infty, \sigma] \subseteq X$. Let (X_σ, d_X) denote the restriction of the metric space (X, d_X) to the subset $X_\sigma \subseteq X$. Similarly, (X_σ, d_X, f_X) is the aug-MS obtained by restricting d_X to $X_\sigma \times X_\sigma$ and f_X to X_σ. The following 2-parameter filtration is considered in [2, 8].

▶ **Definition 4** (Rips bifiltration of an aug-MS). *Let $\mathcal{X} = (X, d_X, f_X)$ be an aug-MS. We define the* Rips bifiltration $\mathcal{R}^{\mathrm{bi}}_\bullet(\mathcal{X}) : \mathbb{R}^2 \to \mathbf{Simp}$ *of $\mathcal{X}$ as $(\varepsilon, \sigma) \mapsto \mathcal{R}_\varepsilon(X_\sigma, d_X)$.*

Applying the k-th homology functor to the Rips bifiltration $\mathcal{R}^{\mathrm{bi}}_\bullet(\mathcal{X})$, we have the persistence module $M := \mathrm{H}_k(\mathcal{R}^{\mathrm{bi}}_\bullet(\mathcal{X})) : \mathbb{R}^2 \to \mathbf{Vec}$. Let $\mathcal{L}$ denote the set of lines of positive slopes in $\mathbb{R}^2$. Given $L \in \mathcal{L}$, the restriction $M|_L : L \to \mathbf{Vec}$ can be decomposed into the unique direct sum of interval modules over L and thus we have the barcode $\mathbf{barc}(M|_L)$ of $M|_L$. The *k-th fibered barcode* of $\mathcal{X}$ is the $\mathcal{L}$-parametrized collection $\{\mathbf{barc}(M|_L)\}_{L \in \mathcal{L}}$ [10, 22, 24].

Elder-rule-staircode for an aug-MS. Let (X, d_X) be a finite metric space. For $\varepsilon \in [0, \infty)$, an *$\varepsilon$-chain* between $x, x' \in X$ stands for a sequence $x = x_1, x_2, \ldots, x_\ell = x'$ of points in X such that $d_X(x_i, x_{i+1}) \leq \varepsilon$ for $i = 1, \ldots, \ell - 1$. Now given $\mathcal{X} = (X, d_X, f_X)$ and $\sigma \in \mathbb{R}_{\geq 0}$, consider a point $x \in X_\sigma$. Then for any $\varepsilon \geq 0$, set $[x]_{(\sigma,\varepsilon)}$ as the collection of all points $x' \in X_\sigma$ that can be connected to x through an ε-chain in X_σ. The function $f_X : X \to \mathbb{R}$ induces an order on X: Given $x, x' \in X$, we say that x is *older than* x' if and only if $f_X(x) < f_X(x')$.

▶ **Definition 5** (Elder-rule-staircode for an aug-MS). *Let $\mathcal{X} = (X, d_X, f_X)$ be an* injective *aug-MS. For each $x \in X$, we define* its staircode *as:*

$$I_x := \{(\sigma, \varepsilon) \in \mathbb{R}^2 : x \in X_\sigma \text{ and } x \text{ is the oldest in } [x]_{(\sigma,\varepsilon)}\ \} \tag{1}$$

The collection $\mathcal{I}_\mathcal{X} := \{I_x\}_{x \in X}$ is called the elder-rule-staircode (ER-staircode for short) *of $\mathcal{X}$.*

See Figure 1 for an example. The relationship between the ER-staircode and the classic elder-rule will become clear in Section 4.1. An interval I of $\mathbb{R}^2$ (Definition 1) is a *staircase interval (or simply staircase)* if there exists $(\sigma_0, \varepsilon_0) \in \mathbb{R}^2$ such that either $I = \{(\sigma, \varepsilon) \in \mathbb{R}^2 : (\sigma_0, \varepsilon_0) \leq (\sigma, \varepsilon)\}$ (i.e. a quadrant) or there is also a stair-like upper boundary – there exists a non-increasing piecewise constant function $u : \mathbb{R} \to (\varepsilon_0, \infty)$ such that $I = \{(\sigma, \varepsilon) \in \mathbb{R}^2 : \sigma \in [\sigma_0, \infty) \text{ and } \varepsilon \in [\varepsilon_0, u(\sigma))\}$ (see Figure 4). It turns out that each $I_x \in \mathcal{I}_{\mathcal{X}}$ is in the form of a staircase interval (proof in the full version of this paper [4]):

▶ **Proposition 6.** *Each I_x in Definition 5 is a staircase interval of $\mathbb{R}^2$.*

Staircodes for non-injective case. Even if f_X is not injective, we still have the concept of the ER-staircode. Consider an aug-MS $\mathcal{X} = (X, d_X, f_X)$ such that f_X is not injective. To induce the ER-staircode of $\mathcal{X}$, we pick any order on X which is *compatible* with f_X: An order $<$ on X is compatible with f_X if $f_X(x) < f_X(x')$ implies $x < x'$ for all $x, x' \in X$. Now we define $\mathcal{I}^{<}_{\mathcal{X}} = \{\!\{I^{<}_x : x \in X\}\!\}$ where

$$I^{<}_x := \{(\sigma, \varepsilon) \in \mathbb{R}^2 : x \in X_\sigma \text{ and } x = \min([x]_{(\sigma,\varepsilon)}, <)\} \tag{2}$$

(we use double-curly-brackets $\{\!\{-\}\!\}$ to denote multisets). Regardless of the choice of $<$, the collection $\mathcal{I}^{<}_{\mathcal{X}} = \{\!\{I^{<}_x : x \in X\}\!\}$ satisfies all properties / theorems we prove later. Hence, for any possible compatible order $<$ we will refer to $\mathcal{I}^{<}_{\mathcal{X}}$ as an *ER-staircode* of $\mathcal{X}$.

▶ **Example 7** (Constant function case)**.** Let (X, d_X) be a metric space of n points. Then, the barcode of $\mathrm{H}_0(\mathcal{R}_\bullet(X, d_X)) : \mathbb{R} \to \mathbf{Vec}$ consists of n intervals J_i, $i = 1, \ldots, n$. Let $\mathcal{X} = (X, d_X, f_X)$ be the aug-MS where f_X is constant at $c \in \mathbb{R}$. Then, all possible total orders on X are compatible with f_X and all induce the same ER-staircode $\mathcal{I}_{\mathcal{X}} = \{\!\{[c, \infty) \times J_i : i = 1, \ldots, n\}\!\}$.

In contrast to Example 7, different orders on X in general induce different ER-staircodes of $\mathcal{X} = (X, d_X, f_X)$; see Example 9. Therefore, a single ER-staircode of $\mathcal{X}$ is not necessarily an *invariant* of $\mathcal{X}$, whereas the collection of all possible ER-staircodes of $\mathcal{X}$ can be seen so (see item 4 in Section 7). This collection, however, is *not a complete invariant of* $\mathcal{X}$ by the following reasoning: It is not difficult to find two non-isometric metric spaces (X, d_X) and (Y, d_Y) such that $\mathrm{H}_0(\mathcal{R}_\bullet(X, d_X))$ and $\mathrm{H}_0(\mathcal{R}_\bullet(Y, d_Y))$ have the same barcode. Let $f_X : X \to \mathbb{R}$ and $f_Y : Y \to \mathbb{R}$ be constant at $c \in \mathbb{R}$. Then, by Example 7, all the ER-staircodes of (X, d_X, f_X) and (Y, d_Y, f_Y) (induced by all possible total orders on X and Y) are the same.

We can recover the zeroth fibered barcode of an aug-MS $\mathcal{X}$ from its ER-staircode: The proof of the following theorem will be given in Section 4.1.

▶ **Theorem 8.** *Let $\mathcal{X}$ be an aug-MS and let $M := \mathrm{H}_0(\mathcal{R}^{\mathrm{bi}}_\bullet(\mathcal{X}))$. Let $\mathcal{I}_{\mathcal{X}} = \{\!\{I_x : x \in X\}\!\}$ be an ER-staircode of $\mathcal{X}$. For each $L \in \mathcal{L}$, the barcode $\mathbf{barc}(M|_L)$ coincides with the multiset $\{\!\{L \cap I_x : x \in X\}\!\}$ (up to removal of empty sets, see Figure 2).*

▶ **Example 9.** If an aug-MS is not injective, then there can be different ER-staircodes w.r.t. different compatible orders. However, each of them will still be valid to produce the fibered barcodes. For example, let (X, d_X) be the metric space in Figure 1 (A). Define $g_X : X \to \mathbb{R}$ by sending x_1, x_2, x_3, x_4 to $1, 2, 2, 4$, respectively. Two orders $(x_1 < x_2 < x_3 < x_4)$ and $(x_1 <' x_3 <' x_2 <' x_4)$ are compatible with g_X, giving two ER-staircodes $\mathcal{I}^{<}_{\mathcal{X}} = \{\!\{I^{<}_{x_i} : i = 1, 2, 3, 4\}\!\}$ and $\mathcal{I}^{<'}_{\mathcal{X}} = \left\{\!\!\left\{I^{<'}_{x_i} : i = 1, 2, 3, 4\right\}\!\!\right\}$. While $I^{<}_{x_i} = I^{<'}_{x_i}$ for $i = 1, 4$, the equality does not hold for $i = 2, 3$. However, both $\mathcal{I}^{<}_{\mathcal{X}}$ and $\mathcal{I}^{<'}_{\mathcal{X}}$ satisfy the statement in Theorem 8. See Figure 3.

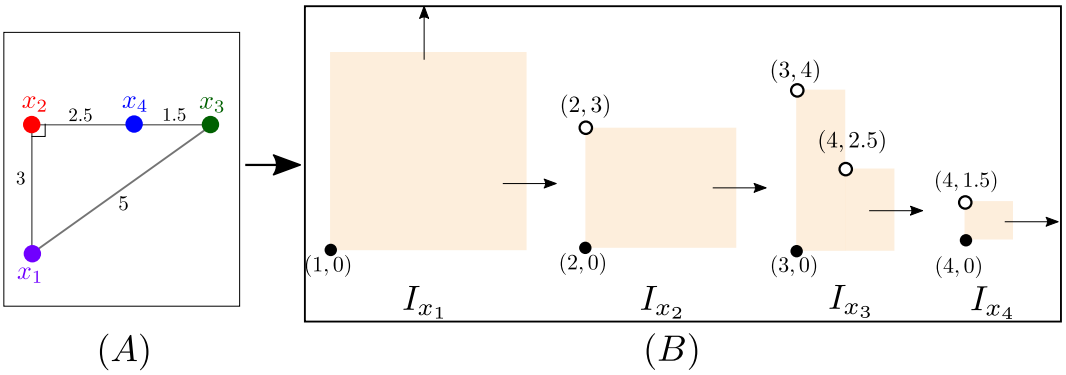


Figure 1 (A) Consider the triangle with edge lengths 3,4 and 5. Consider the aug-MS $\mathcal{X} = (X, d_X, f_X)$ where $X := \{x_1, x_2, x_3, x_4\}$, d_X is the Euclidean metric on the plane, and f_X is given as $f_X(x_i) = i$ for $i = 1, 2, 3, 4$. (B) The ER-staircode of $\mathcal{X}$.

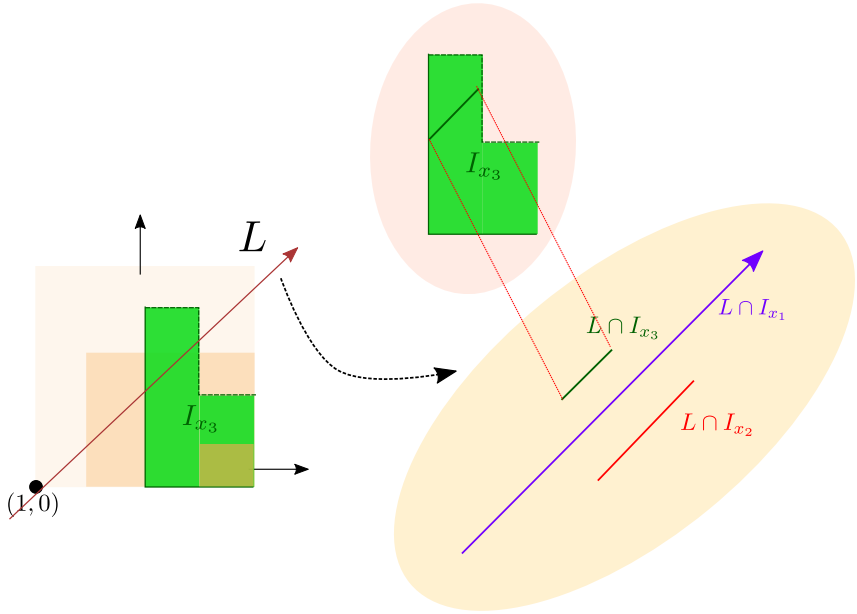


Figure 2 Left: The stack of I_{x_i}, $i = 1, 2, 3, 4$ from Figure 1 and a line $L \in \mathcal{L}$. Right: The barcode of $M|_L$. Since L does not intersect I_{x_4}, only three intervals of $L \subset \mathbb{R}^2$ appear in the barcode.

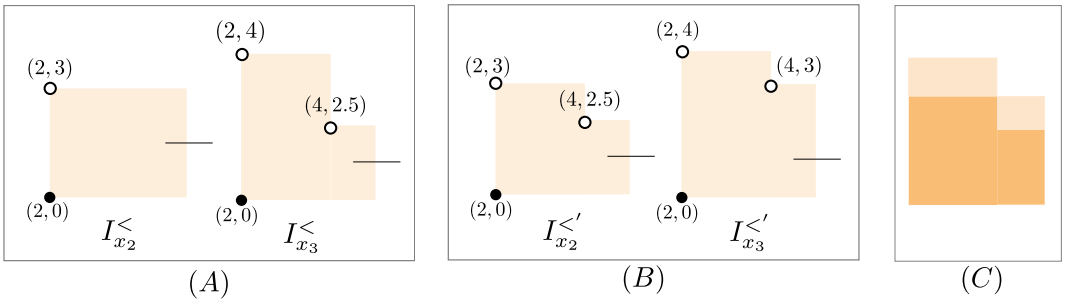


Figure 3 Example 9: (A) $I^{\leq}_{x_2}$ and $I^{\leq}_{x_3}$. (B) $I^{<'}_{x_2}$ and $I^{<'}_{x_3}$. (C) Stack of $I^{\leq}_{x_2}$ and $I^{\leq}_{x_3}$. Stack of $I^{<'}_{x_2}$ and $I^{<'}_{x_3}$ look the same. Observe that for any $L \in \mathcal{L}$, $\{\{L \cap I^{\leq}_{x_2}, L \cap I^{\leq}_{x_3}\}\} = \left\{\left\{L \cap I^{<'}_{x_2}, L \cap I^{<'}_{x_3}\right\}\right\}$.

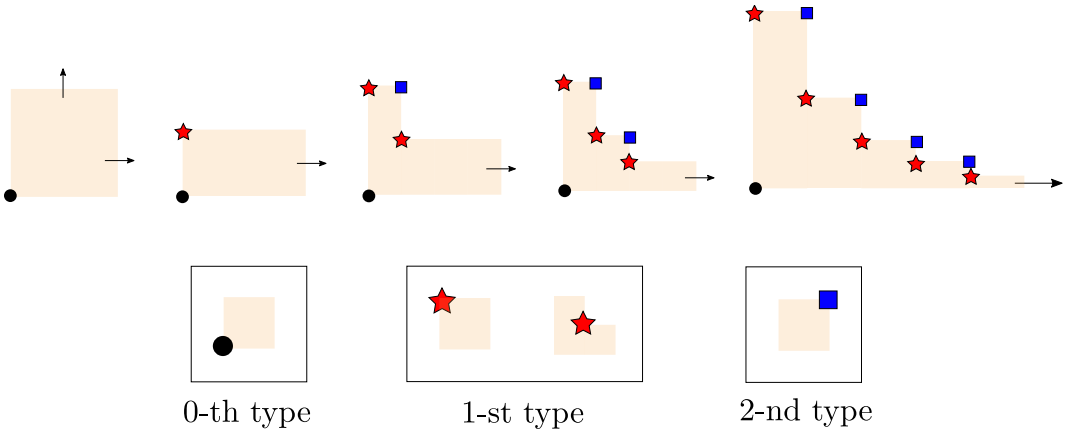


Figure 4 Every corner point of a staircase interval falls into three different types depending on its neighborhood information, as the pictures above illustrate. Staircase intervals in the first row are decorated by their corner points (a precise description is in Definition A.2 of the full version [4]).

We close this section with some definitions that will be useful later. Let I be a staircase interval of $\mathbb{R}^2$. We define the three types of corner points as in Figure 4 (rigorous definition of these corner points is in Definition A.2 in the full version [4]): Roughly speaking, for each staircase I_x, type-0 is the left-bottom point; type-1 corners are those where the boundary transitions from a vertical segment to a horizontal one, while type-2 are those transitions from a horizontal one to vertical one. For each $j = 0, 1, 2$ we define the function $\gamma_j(I) : \mathbb{R}^2 \to \mathbb{Z}_{\geq 0}$ as $\gamma_j(I)(\mathbf{a}) = \begin{cases} 1, & \mathbf{a} \text{ is a } j\text{-th type corner point of } I \\ 0, & \text{otherwise.} \end{cases}$

Elder-rule feature functions defined below will be useful in Section 5.

▶ **Definition 10.** *Let $\mathcal{X}$ be an aug-MS and let $I_{\mathcal{X}} = \{\!\{I_x : x \in X\}\!\}$ be an ER-staircode of $\mathcal{X}$. For $j = 0, 1, 2$, we define the j-th* elder-rule feature function *as the sum $\gamma_j^{\mathcal{X}} = \sum_{x \in X} \gamma_j(I_x)$.*

4 Decorated elder-rule-staircodes and treegrams

In Section 4.1 we prove Theorem 8 and introduce *bipersistence treegrams* to encode multi-scale clustering information of aug-MSs. In Section 4.2 we show that an "enriched" ER-staircode of an aug-MS $\mathcal{X}$ can recover the so-called *fibered treegram* of $\mathcal{X}$, i.e. 1-dimensional slices of the aforementioned bipersistence treegram. Also, we identify a sufficient condition on $\mathcal{X}$ for its ER-staircode to be the barcode of the 2-parameter persistence module $\mathrm{H}_0(\mathcal{R}_\bullet^{\mathrm{bi}}(\mathcal{X}))$.

4.1 Bipersistence treegrams

Let X be a non-empty finite set. Any partition P of a subset X' of X is a *sub-partition* of X; and we refer to X' as the *underlying set of* P. Elements of a sub-partition of X are called *blocks*. A partition of the empty set is defined as the empty set. By $\mathbf{Subpart}(X)$, we denote the set of *all sub-partitions of* X, i.e. $\mathbf{Subpart}(X) := \{P : \exists X' \subseteq X \ , \ P \text{ is a partition of } X'\}$. Given $P, Q \in \mathbf{Subpart}(X)$, $P \leq Q$ means that P refines Q, i.e. for all $B \in P$, there exists $C \in Q$ s.t. $B \subseteq C$. For example, let $X = \{x_1, x_2, x_3\}$; then $P \leq Q$ for sub-partitions $P := \{\{x_1\}, \{x_2\}\}$ and $Q := \{\{x_1, x_2\}, \{x_3\}\}$. Treegrams are a generalized notion of dendrograms [29].

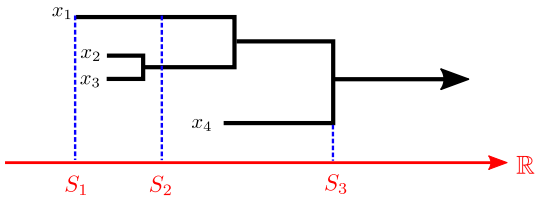


Figure 5 A (1D) treegram θ_X over the set X. Notice that $\theta_X(t) = \emptyset$ for $t \in (-\infty, S_1)$. Also, $\theta_X(S_1) = \{\{x_1\}\}$, $\theta_X(S_2) = \{\{x_1\}, \{x_2, x_3\}\}$, and $\theta_X(t) = \{X\}$ for all $t \in [S_3, \infty)$.

▶ **Definition 11** (Treegrams [29]). *A* treegram *over a finite set X is any function $\theta_X : \mathbb{R} \to$ **Subpart**(X) such that the following properties hold: (1) if $t_1 \leq t_2$, then $\theta_X(t_1) \leq \theta_X(t_2)$, (2) there exists $T > 0$ such that $\theta_X(t) = \{X\}$ for $t \geq T$ and $\theta_X(t)$ is empty for $t \leq -T$, and (3) for all t there exists $\epsilon > 0$ s.t. $\theta_X(s) = \theta_X(t)$ for $s \in [t, t + \epsilon]$. See Figure 5 for an example. Also, even when the domain $\mathbb{R}$ is replaced by any totally ordered set L isomorphic to $\mathbb{R}$, θ_X is said to be a (1-parameter) treegram.*

Given a simplicial complex K, let $K^{(0)}$ be the vertex set of K. Let $\pi_0(K)$ be the partition of the vertex set $K^{(0)}$ according to the connected components of K. A functor $\mathcal{K} : \mathbb{P} \to$ **Simp** is said to be a *filtration of* K if $\mathcal{K}(\mathbf{a}) \subseteq K$ for all $\mathbf{a} \in \mathbb{P}$, every internal map is an inclusion, and there exists $\mathbf{a}_0 \in \mathbb{P}$ such that for all $\mathbf{a} \in \mathbb{P}$ with $\mathbf{a}_0 \leq \mathbf{a}$, $\mathcal{K}(\mathbf{a}) = K$.

▶ Remark 12 (Treegrams induced by simplicial filtrations). Let K be a simplicial complex on the vertex set $X = \{x_1, x_2, \ldots, x_n\}$ and let $\mathcal{K} : \mathbb{R} \to$ **Simp** be a filtration of K. Assume that K consists solely of one connected component, i.e. $\pi_0(K) = \{X\}$. Then, the function $\pi_0(\mathcal{K}) : \mathbb{R} \to$ **Subpart**(X) defined as $\varepsilon \mapsto \pi_0(\mathcal{K}(\varepsilon))$ is a treegram over X.

The zeroth elder rule for a 1-parameter filtration. Let θ_X be a treegram over X. We define the *birth time* of x as $b(x) := \min\{\varepsilon \in \mathbb{R} : x \text{ is in the underlying set of } \theta_X(\varepsilon)\}$ (by Definition 11 (2), every $x \in X$ has the birth time $b(x)$). Pick any order $<$ on X such that $b(x) < b(x')$ implies $x < x'$ for all $x, x' \in X$. For $\varepsilon \in [b(x), \infty)$, we denote the block to which x belongs in the sub-partition $\theta_X(\varepsilon)$ by $[x]_\varepsilon$. We define the *death time* of x as $d^<(x) = \sup\{\varepsilon \in [b(x), \infty] : x = \min([x]_\varepsilon, <)\}$. As long as $<$ is compatible with the birth times, the *elder-rule-barcode* is uniquely defined (which is proved in the full version [4]):

▶ **Definition 13** (Elder-rule-barcode of a treegram). *Let $\theta_X : \mathbb{R} \to$ **Subpart**(X) be a treegram over X. For any order $<$ on X compatible with the birth times, let $J_x := [b(x), d^<(x))$. The elder-rule-barcode of θ_X is defined as the multiset* **barc**$(\theta_X) := \{\!\{J_x : x \in X\}\!\}$.

For the **1-parameter case**, the elder-rule-barcode of a treegram can be obtained by dismantling the treegram into linear pieces w.r.t. the elder rule – see the theorem below. Even though this result is well-known (e.g, [14]), we include a proof in the full version [4].

▶ **Theorem 14** (Compatibility between the elder rule and algebraic decomposition). *Let $\mathcal{K}$ and θ_X be the filtration and the treegram in Remark 12, respectively. Let* **barc**$(\theta_X) = \{\!\{J_x : x \in X\}\!\}$ *be the elder-rule-barcode of θ_X. Then, $\mathrm{H}_0(\mathcal{K}) \cong \bigoplus_{x \in X} \mathcal{I}^{J_x}$ (see Figure 6).*

Proof of Theorem 8. We are now ready to prove Theorem 8. Fix $L \in \mathcal{L}$. Since L is isomorphic to $\mathbb{R}$ as a totally ordered set, $\mathcal{K} = \mathcal{R}_\bullet^{\mathrm{bi}}(\mathcal{X})|_L : L \to$ **Simp** can be viewed as a 1-parameter filtration. Consider the treegram $\theta_X := \pi_0(\mathcal{K}) : L \to$ **Subpart**(X). By the definition of I_xs, it is clear that $\{\!\{L \cap I_x : x \in X\}\!\}$ is the elder-rule-barcode of the treegram θ_X (Definition 13). Hence, by Theorem 14, the multiset $\{\!\{L \cap I_x : x \in X\}\!\}$ is equal to the barcode of $\mathrm{H}_0(\mathcal{K})$. Since $\mathrm{H}_0(\mathcal{K}) = M|_L$, we have $\{\!\{L \cap I_x : x \in X\}\!\} =$ **barc**$(M|_L)$. ◀

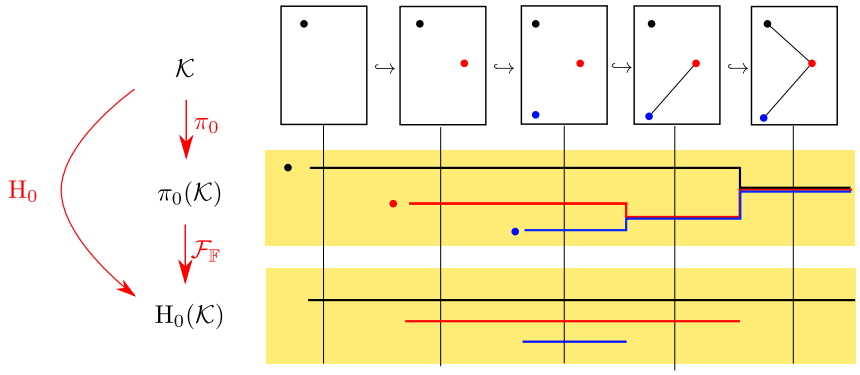


Figure 6 The first row represents a simplicial filtration $\mathcal{K}$. The second row stands for the treegram $\pi_0(\mathcal{K})$ which encodes the evolution of clusters in $\mathcal{K}$ (Remark 12). The third row is the barcode of $\mathrm{H}_0(\mathcal{K})$. The persistence module $\mathrm{H}_0(\mathcal{K})$ can be obtained by applying the linearization functor (Definition B.2 in the full version [4]) to $\pi_0(\mathcal{K})$. Alternatively, the barcode of $\mathrm{H}_0(\mathcal{K})$ can also be obtained by applying the elder rule to $\pi_0(\mathcal{K})$ (Definition 13).

Bipersistence treegrams. We now extend the notion of treegrams to encode the evolution of clusters of a 2-parameter filtration (similar ideas appear in [20]). A *bipersistence treegram* over a finite set X is any function $\theta_X^{\mathrm{bi}} : \mathbb{R}^2 \to \mathbf{Subpart}(X)$ such that if $\mathbf{a} \leq \mathbf{b}$ in $\mathbb{R}^2$, then $\theta_X^{\mathrm{bi}}(\mathbf{a}) \leq \theta_X^{\mathrm{bi}}(\mathbf{b})$.

▶ **Definition 15** (Rips bipersistence treegrams). *Given an aug-MS $\mathcal{X} = (X, d_X, f_X)$, the* Rips bipersistence treegram *of $\mathcal{X}$ is $\theta_{\mathcal{X}}^{\mathrm{bi}} : \mathbb{R}^2 \to \mathbf{Subpart}(X)$ such that $(\sigma, \varepsilon) \mapsto \pi_0\left(\mathcal{R}_\varepsilon(X_\sigma, d_X)\right)$.*

Observe that $x \in X$ belongs to the underlying set of $\theta_{\mathcal{X}}^{\mathrm{bi}}(\mathbf{a})$ if and only if $(f_X(x), 0) \leq \mathbf{a}$, i.e. $(f_X(x), 0)$ is the *birth grade* of x in $\theta_{\mathcal{X}}^{\mathrm{bi}}$. Assume that $\mathcal{X}$ is injective. Then the birth grades of elements in X is totally ordered. The ER-staircode of $\mathcal{X}$ can be extracted from $\theta_{\mathcal{X}}^{\mathrm{bi}}$: Indeed, I_x in equation (1) can be rephrased as $I_x = \{(\sigma, \varepsilon) \in \mathbb{R}^2 : x \text{ is in the underlying set of } \theta_{\mathcal{X}}^{\mathrm{bi}}(\sigma, \varepsilon) \text{ and } x \text{ has the smallest birth grade in its block of } \theta_{\mathcal{X}}^{\mathrm{bi}}(\sigma, \varepsilon)\}$. See Figure 7.

▶ **Definition 16** (Fibered treegrams). *Let $\theta_{\mathcal{X}}^{\mathrm{bi}}$ be a Rips bipersistence treegram of an aug-MS $\mathcal{X}$. The* fibered treegram *of $\theta_{\mathcal{X}}^{\mathrm{bi}}$ refers to the collection $\{\theta_{\mathcal{X}}^{\mathrm{bi}}|_L\}_{L \in \mathcal{L}}$ of treegrams obtained by restricting $\theta_{\mathcal{X}}^{\mathrm{bi}}$ to positive-slope lines (see Figure 8 for an example).*

4.2 Elder-rule-staircodes and fibered treegrams

In this section we identify a sufficient condition on an aug-MS $\mathcal{X}$ for its ER-staircode to coincide with the barcode of the 2-parameter persistence module $\mathrm{H}_0(\mathcal{R}_\bullet^{\mathrm{bi}}(\mathcal{X}))$ (Theorem 17). Also, in general, all fibered treegrams can be recovered from ER-staircodes (Theorem 19).

Let (X, d_X) be a metric space and fix $x, x' \in X$. Recall that an *ε-chain* between x and x' is a finite sequence $x = x_1, x_2, \ldots, x_\ell = x'$ in X where each consecutive pair x_i, x_{i+1} is within distance ε. Define (in fact an ultrametric) $u_X : X \times X \to \mathbb{R}_{\geq 0}$ as

$$u_X(x, x') := \min\{\varepsilon \in [0, \infty) : \text{there exists an } \varepsilon\text{-chain between } x \text{ and } x'\} \text{ (see [7])}.$$

For a metric space (X, d_X) and pick any total order $<$ on X. Let $x \in X$ be a non-minimal element of $(X, <)$. A *$<$-conqueror* of x is an element $x' \in X$ such that (1) $x' < x$, and (2) for any $x'' \in X$ with $x'' < x$, it holds that $u_X(x, x') \leq u_X(x, x'')$.

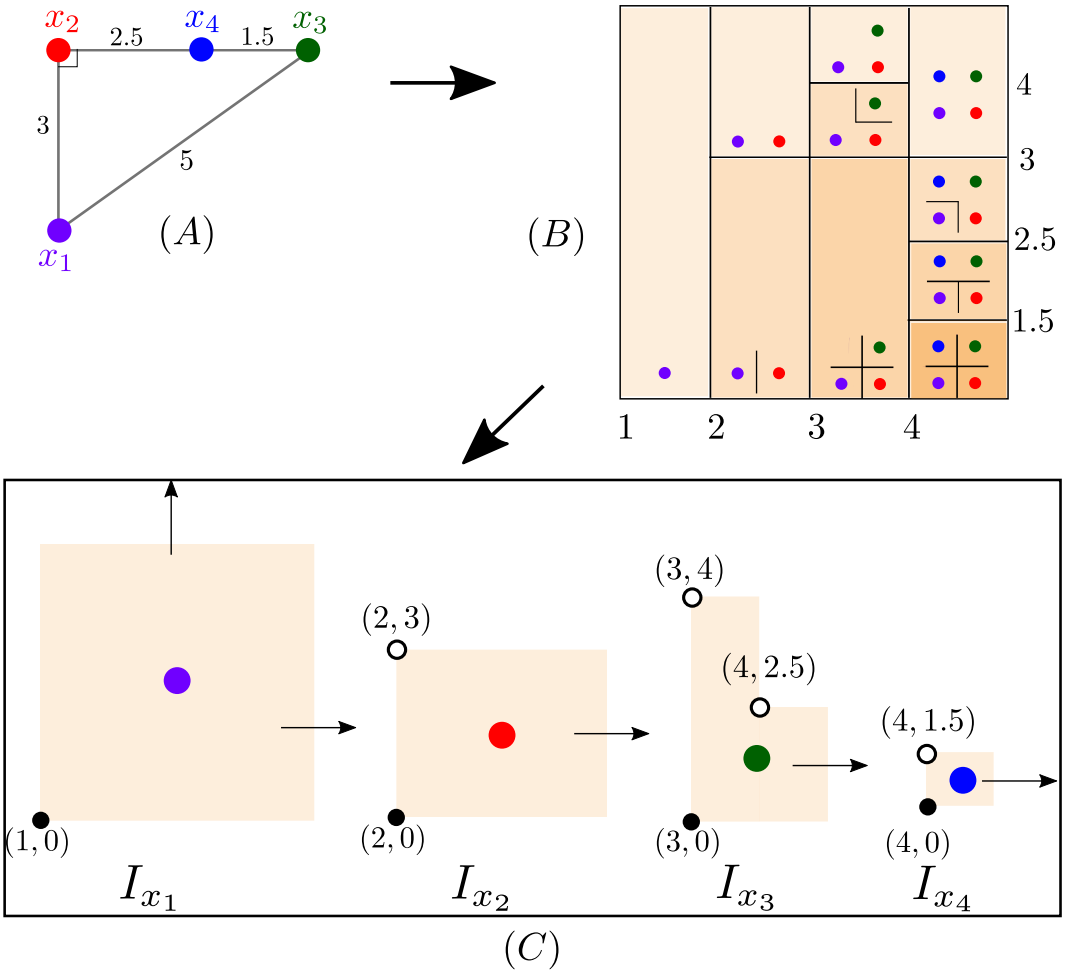


Figure 7 Consider the aug-MS $\mathcal{X}$ defined in Figure 1. Figure (A) and (C) above are identical to Figure 1 (A) and (B), respectively. (B) The Rips bipersistence treegram of $\mathcal{X}$ (Definition 15). The summarization processes (A)→(B)→(C) are analogous to the processes depicted in Figure 6.

Now consider an aug-MS $\mathcal{X} = (X, d_X, f_X)$. A $<$-*conqueror* function $c_x : [f_X(x), \infty) \to X$ of a non-minimal $x \in X$ sends $\sigma \in [f_X(x), \infty)$ to a conqueror of x in (X_σ, d_X). For the minimum $x' \in (X, <)$, define $c_{x'} : [f_X(x'), \infty) \to X$ to be the constant function at x'.

We generalize Theorem 14 and at the same time strengthen Theorem 8 for 2-parameter persistence modules induced by a special type of aug-MSs:

▶ **Theorem 17** (Compatibility between the ER-staircodes and algebraic decomposition)**.** *Let $\mathcal{X} = (X, d_X, f_X)$ be an aug-MS and fix any order $<$ on X compatible with f_X. Assume that there exists a* constant $<$*-conqueror function for* each $x \in X$.[1] *Then,* $\mathrm{H}_0\left(\mathcal{R}^{\mathrm{bi}}_\bullet(\mathcal{X})\right)$ *is interval decomposable and its barcode coincides with the ER-staircode $\mathcal{I}^{<}_{\mathcal{X}}$.*

The proof of Theorem 17 is similar to that of Theorem 14, and is in the full version [4]. Consider the aug-MS $\mathcal{X}$ in Figure 1, which satisfies the assumption in Theorem 17. Therefore, $\mathrm{H}_0\left(\mathcal{R}^{\mathrm{bi}}_\bullet(\mathcal{X})\right)$ is interval decomposable. The following corollary (proof in the full version [4]) gives an example of a class of aug-MSs to which Theorem 17 applies.

[1] Observe that if this property holds for the order $<$, then the same property holds for any other order $<'$ that is compatible with f_X, and $\mathcal{I}^{<}_{\mathcal{X}} = \mathcal{I}^{<'}_{\mathcal{X}}$.

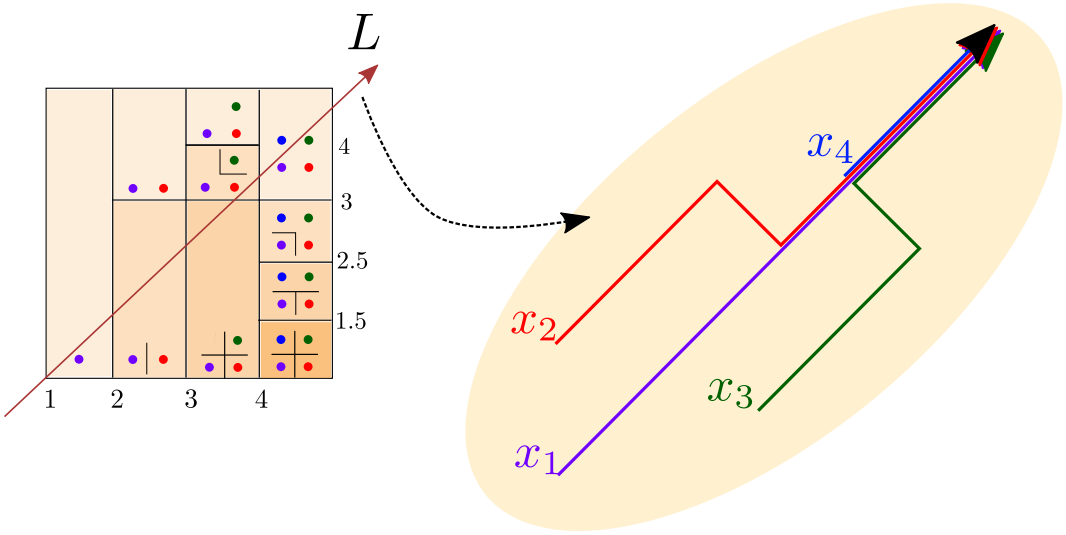


Figure 8 Consider the bipersistence treegram in Figure 7 (B) and pick a line L of positive slope. Then, we obtain a treegram over L.

▶ **Corollary 18.** *Let $\mathcal{X} = (X, d_X, f_X)$ be any aug-MS where d_X is an ultrametric, i.e. $d_X(x, x'') \leq \max(d_X(x, x'), d_X(x', x''))$ for all $x, x', x'' \in X$. Then, $\mathrm{H}_0\left(\mathcal{R}^{\mathrm{bi}}_\bullet(\mathcal{X})\right)$ is interval decomposable (in fact, its barcode consists solely of infinite rectangular intervals).*

We enrich the ER-staircode in order to query the fibered treegram: Given an aug-MS $\mathcal{X} = (X, d_X, f_X)$, let $<$ be any order on X compatible with f_X. For each x, define I^*_x as the pair (I_x, c_x) of the set I_x and the $<$-conqueror function c_x. The collection $\mathcal{I}^*_{\mathcal{X}} := \{I^*_x\}_{x \in X}$ is said to be the *decorated ER-staircode* of $\mathcal{X}$. See Figure 9. The following result holds for general aug-MSs.

▶ **Theorem 19.** *Given any $L \in \mathcal{L}$, the fibered treegram $\theta^{\mathrm{bi}}_{\mathcal{X}}|_L$ can be recovered from the decorated ER-staircode $\mathcal{I}^*_{\mathcal{X}}$ of the aug-MS $\mathcal{X} = (X, d_X, f_X)$.*

5 Elder-rule-staircodes and graded Betti numbers

We now show that we can retrieve the graded Betti numbers of $\mathrm{H}_0(\mathcal{R}^{\mathrm{bi}}_\bullet(\mathcal{X}))$ from the ER-staircode of an aug-MS $\mathcal{X}$ (Theorem 23). Along the way, we obtain a characterization result for the graded Betti number of $\mathrm{H}_0(\mathcal{R}^{\mathrm{bi}}_\bullet(\mathcal{X}))$ (Theorem 22), which is of independent interest.

Graded Betti numbers. We briefly review the concept of *graded Betti numbers* [6, 21, 24, 25, 28, 31]. Since our interests are in studying finite aug-MSs, we restrict ourselves to *finite* persistence modules – the k-th homology of a filtration of a finite simplicial complex for some $k \in \mathbb{Z}_{\geq 0}$ [8].

$$Q^{\mathbf{a}}_{\mathbf{x}} = \begin{cases} \mathbb{F}, & \text{if } \mathbf{a} \leq \mathbf{x} \\ 0, & \text{otherwise,} \end{cases} \qquad \varphi_{Q^{\mathbf{a}}}(\mathbf{x}, \mathbf{y}) = \begin{cases} \mathrm{id}_{\mathbb{F}}, & \text{if } \mathbf{a} \leq \mathbf{x} \\ 0, & \text{otherwise.} \end{cases}$$

Any $F : \mathbb{Z}^d \to \mathbf{Vec}$ is said to be *free* if there exists a multiset $\mathcal{A}$ of elements of $\mathbb{Z}^d$ such that $F \cong \bigoplus_{\mathbf{a} \in \mathcal{A}} Q^{\mathbf{a}}$. For simplicity, we refer to free persistence modules as free modules.Let M be a persistence module. An element $m \in M_{\mathbf{a}}$ for some $\mathbf{a} \in \mathbb{Z}^d$ is called a *homogeneous*

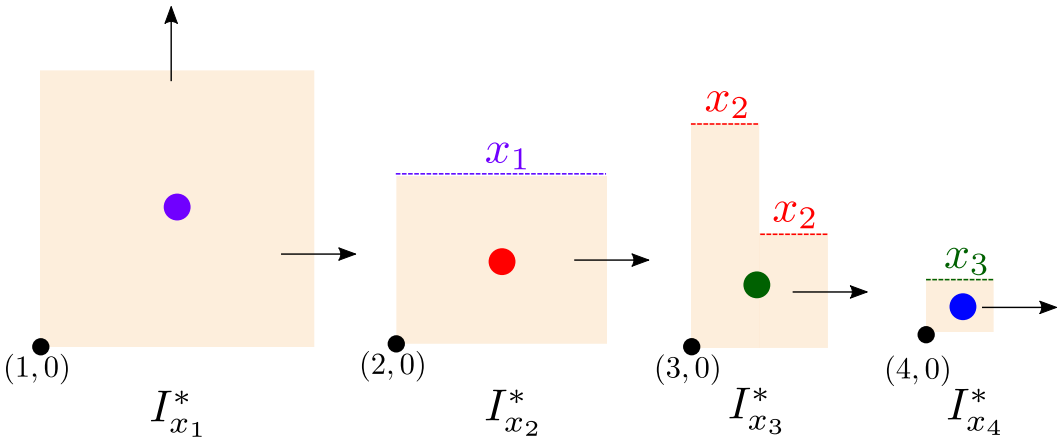


Figure 9 Decorated intervals corresponding to the four intervals in Figure 1 (C). For each $i = 2, 3, 4$, the upper boundary of I_{x_i} is decorated by the conqueror of x_i.

element of M, denoted by $\mathrm{gr}(m) = \mathbf{a}$. Let F be a free module. A *basis* B of F is a minimal homogeneous set of generators of F (see full version [4] for details). There can exist multiple bases of F, but the number of elements at each grade $\mathbf{a} \in \mathbb{Z}^d$ in a basis of F is an isomorphism invariant. For a finite M, let IM denote the submodule of M generated by the images of all linear maps $\varphi_M(\mathbf{a}, \mathbf{b})$, with $\mathbf{a} < \mathbf{b}$ in $\mathbb{Z}^d$. Assume that there is a chain of modules

$$F^\bullet : \cdots \xrightarrow{\partial_3} F^2 \xrightarrow{\partial_2} F^1 \xrightarrow{\partial_1} F^0 \xrightarrow{\partial_0} M \xrightarrow{0(=:\partial_{-1})} 0 \tag{3}$$

such that (1) each F^i is a free module, and (2) $\mathrm{im}(\partial^i) = \ker(\partial^{i-1})$, $i = 0, 1, 2, \cdots$. Then we call $F^\bullet$ a *resolution* of M. The condition (2) is referred to as *exactness* of $F^\bullet$. We call the resolution $F^\bullet$ *minimal* if $\mathrm{im}(\partial^i) \subseteq IF^{i-1}$ for $i = 1, 2, \cdots$. It is a standard fact that a minimal resolution of M always exists and is unique up to isomorphism [28, Chapter I].

▶ **Definition 20** (Graded Betti numbers). *Let $M : \mathbb{Z}^d \to \mathbf{Vec}$ be finite. Assume that a minimal free resolution of M is $F^\bullet$ in (3). For $i \in \mathbb{Z}_{\geq 0}$, the i-th graded Betti number $\beta_i^M : \mathbb{Z}^d \to \mathbb{Z}_{\geq 0}$ is defined as $\beta_i^M(\mathbf{a}) =$ (number of elements at grade $\mathbf{a}$ in any basis of F^i).*

We remark that $\beta_i^M : \mathbb{Z}^d \to \mathbb{Z}_{\geq 0}$ is the zero function for every $i > d$ [17, Theorem 1.13].

The graded Betti numbers of $\mathbf{H_0}(\mathcal{R}_\bullet^{\mathrm{bi}}(\mathcal{X}))$. Henceforth, every aug-MS $\mathcal{X} = (X, d_X, f_X)$ is assumed to be *generic*: f_X is injective and each pair of elements in X has different distance. Non-generic aug-MSs can also be easily handled; see the full version [4]. Since $\mathcal{X}$ is finite, we consider $\mathbb{Z}^2$-indexed filtration described subsequently as a substitute of $\mathcal{R}_\bullet^{\mathrm{bi}}(\mathcal{X})$:

▶ **Definition 21.** *Consider an aug-MS $\mathcal{X} = (X, d_X, f_X)$ with $X := \{x_1, \ldots, x_n\}$ and assume that $f_X(x_1) < \ldots < f_X(x_n)$. Define $f_X^{\mathbb{Z}} : X \to \mathbb{N}$ as $x_i \mapsto i$. Define $d_X^{\mathbb{Z}} : X \times X \to \mathbb{N}$ by sending each non-trivial pair (x_i, x_j) $(i \neq j)$ to $\ell \in \{1, \ldots, \binom{n}{2}\}$, where $d_X(x_i, x_j)$ is the ℓ-th smallest distance (among non-zero distance values). The restriction of $\mathcal{R}_\bullet^{\mathrm{bi}}(X, d_X^{\mathbb{Z}}, f_X^{\mathbb{Z}}) : \mathbb{R}^2 \to \mathbf{Simp}$ to $\mathbb{Z}^2$ is the $\mathbb{Z}^2$-indexed Rips filtration of $\mathcal{X}$. Also, let $\gamma_j^{\mathcal{X}}$ denote the j-th elder-rule feature function of $(X, d_X^{\mathbb{Z}}, f_X^{\mathbb{Z}})$ for $j = 0, 1, 2$ in this section.*

For Theorem 22, we introduce relevant terminology and notation. Let $\mathcal{S}$ be the $\mathbb{Z}^2$-indexed Rips filtration of an aug-MS $\mathcal{X}$ and let $\mathcal{K}$ be the *1-skeleton of* $\mathcal{S}$, i.e. $\mathcal{K}$ is another $\mathbb{Z}^2$-indexed filtration where $\mathcal{K}(\mathbf{a})$ is the 1-skeleton of $\mathcal{S}(\mathbf{a})$ for every $\mathbf{a} \in \mathbb{P}$.

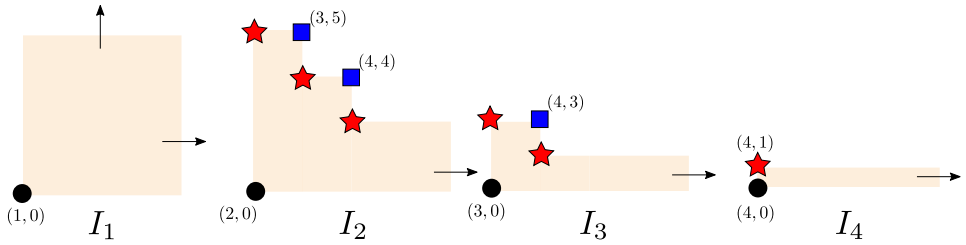


Figure 10 Assume that an aug-MS $\mathcal{X}$ consists of four staircase intervals as above with types of corners marked. From these corner types, we can obtain the graded Betti numbers of $M := \mathrm{H}_0(\mathcal{R}_\bullet^{\mathrm{bi}}(\mathcal{X}))$ via Theorem 23: Let $\mathrm{supp}(\beta_i^M) := \{\mathbf{a} \in \mathbb{Z}^2 : \beta_i^M(\mathbf{a}) \neq 0\}$ for $i = 0, 1, 2$ (the *support* of β_i^M). By Theorem 23, we have $\mathrm{supp}(\beta_0^M) = \{(i, 0) : i = 1, 2, 3, 4\}$, $\mathrm{supp}(\beta_1^M) = \{(2, 5), (3, 4), (4, 3), (3, 3), (4, 2), (4, 1)\} \setminus \{(3, 5), (4, 4), (4, 3)\} = \{(2, 5), (3, 4), (3, 3), (4, 2), (4, 1)\}$, and $\mathrm{supp}(\beta_2^M) = \{\{(3, 5), (4, 4), (4, 3)\}\} \setminus \{(2, 5), (3, 4), (4, 3), (3, 3), (4, 2), (4, 1)\} = \{(3, 5), (4, 4)\}$. All graded Betti numbers are 1 on their supports and 0 otherwise. In particular, note that the grade $\mathbf{a} = (4, 3)$ receives both a 1-st type mark (in I_2) and a 2-nd type mark (in I_3). Thus it contributes value 1 both to $\gamma_1^{\mathcal{X}}(\mathbf{a})$ and $\gamma_2^{\mathcal{X}}(\mathbf{a})$, and as a result, it does not appear in the support for β_1^M nor β_2^M.

- Note that $\mathcal{K}$ is 1-*critical*: every simplex that appears in $\mathcal{K}$ has a unique birth index. Let e be an edge that appears in $\mathcal{K}$ whose birth index is $\mathbf{b}(e) = (b_1, b_2) \in \mathbb{Z}^2$. We say that the edge e is *negative* if the number of connected components in $\mathcal{K}(b_1, b_2)$ is strictly less than that of $K(b_1, b_2 - 1)$. Otherwise, the edge e is *positive*.
- Given a simplicial complex K and $k \in \mathbb{Z}_{\geq 0}$, let $C_k(K)$ be the k-th chain group of K. For $k \in \mathbb{Z}_{\geq 0}$, let $\partial_k : C_k(K) \to C_{k-1}(K)$ be the boundary map, and $Z_k(K) := \ker(\partial_k)$ the k-th cycle group of K.
- Let $\mathcal{K} : \mathbb{Z}^2 \to \mathbf{Simp}$ be a filtration. For each $k \in \mathbb{Z}_{\geq 0}$, let $C_k(\mathcal{K}) : \mathbb{Z}^2 \to \mathbf{Vec}$ be the module defined as $C_k(\mathcal{K})(\mathbf{a}) := C_k(\mathcal{K}(\mathbf{a}))$, where the internal maps $\varphi_{\mathcal{K}}(\mathbf{a}, \mathbf{b})$ are the canonical inclusion maps $C_k(\mathcal{K}(\mathbf{a})) \hookrightarrow C_k(\mathcal{K}(\mathbf{b}))$. In particular, if $\mathcal{K}$ is 1-critical, then $C_k(\mathcal{K})$ is the free module whose basis elements one-to-one correspond to all the k-th simplices in S. More specifically, the birth of a simplex $\sigma \in S$ in $\mathcal{K}$ at $\mathbf{a} \in \mathbb{Z}^d$ corresponds to a generator of $C_k(\mathcal{K})$ at $\mathbf{a}$.

▶ **Theorem 22.** *Let $\mathcal{K}$ be the 1-skeleton of the $\mathbb{Z}^2$-indexed Rips filtration of an aug-MS. Let $\mathcal{K}^-$ be the filtration of $\mathcal{K}$ that is obtained by removing all positive edges in $\mathcal{K}$. Then,*

(i) *The following sequence of persistence modules is exact:*

$$0 \to Z_1(\mathcal{K}^-) \xrightarrow{i} C_1(\mathcal{K}^-) \xrightarrow{\partial_1} C_0(\mathcal{K}^-) \xrightarrow{p} \mathrm{H}_0(\mathcal{K}) \to 0, \tag{4}$$

where i is the canonical inclusion, ∂_1 is the boundary map, p is the canonical projection.

(ii) *The sequence in (4) is a minimal free resolution of* $\mathrm{H}_0(\mathcal{K})$.[2]

Theorem 22 is proved in the full version [4].

Given any two functions $\alpha, \alpha' : \mathbb{Z}^2 \to \mathbb{Z}_{\geq 0}$, we define $\alpha - \alpha' : \mathbb{Z}^2 \to \mathbb{Z}_{\geq 0}$ as

$$(\alpha - \alpha')(\mathbf{x}) = \max(\alpha(\mathbf{x}) - \alpha'(\mathbf{x}), 0), \text{ for } \mathbf{x} \in \mathbb{Z}^2.$$

For any aug-MS $\mathcal{X}$, we can compute the graded Betti numbers of the zeroth homology of $\mathcal{R}_\bullet^{\mathrm{bi}}(\mathcal{X})$ from the ER-staircode of $\mathcal{X}$, as specified by the following result.

[2] This means that $F^0 = C_0(\mathcal{K}^-)$, $F^1 = C_1(\mathcal{K}^-)$, $F^2 = Z_1(\mathcal{K}^-)$ and $F^i = 0$ for $i > 2$ in the chain of (3).

▶ **Theorem 23.** *Let $\mathcal{K}$ be the $\mathbb{Z}^2$-indexed Rips filtration of an aug-MS $\mathcal{X}$ and let $M := \mathrm{H}_0(\mathcal{K})$. Let β_i^M be the i-th grade Betti number of M. Then,*

$$\beta_0^M = \gamma_0^{\mathcal{X}}, \quad \beta_1^M = \gamma_1^{\mathcal{X}} - \gamma_2^{\mathcal{X}}, \quad \beta_2^M = \gamma_2^{\mathcal{X}} - \gamma_1^{\mathcal{X}}. \tag{5}$$

In particular, we note that the elder-rule feature functions $\gamma_j^{\mathcal{X}}$ are easy to compute, as one only needs to compute and aggregate the type of each corner in staircase intervals in the ER-staircode of $\mathcal{X}$. Once $\gamma_j^{\mathcal{X}}$s are known, one can easily compute the graded Betti number of $\mathrm{H}_0(\mathcal{R}_\bullet^{\mathrm{bi}}(\mathcal{X}))$ by Theorem 23. See Figure 10 for an example. We also remark that *Koszul homology formulae* [25, Proposition 5.1] are in a similar form to those in (5). However, Koszul homology formulae do not directly imply those in (5) nor vice versa.

Sketch of proof of Theorem 23. Let $\mathcal{X} := (X, d_X, f_X)$ with $X = \{x_1, \dots, x_n\}$, and assume that $f_X(x_1) < \dots < f_X(x_n)$. By the construction of $\mathcal{K}$ and $\gamma_i^{\mathcal{X}}$, it suffices to show the equalities in (5) hold on $\mathcal{A} := \{1, 2, \dots, n\} \times \{0, 1, \dots, \binom{n}{2}\} \subset \mathbb{Z}^2$ (β_i^M and $\gamma_i^{\mathcal{X}}$ vanish outside $\mathcal{A}$ for $i = 0, 1, 2$). By Theorem 22 and the construction of $\gamma_0^{\mathcal{X}}$, both of β_0^M and $\gamma_0^{\mathcal{X}}$ have values 1 on $\mathcal{A}|_{y=0} = \{(1,0), (2,0), (3,0) \dots, (n,0)\}$ and zero outside $\mathcal{A}|_{y=0}$, implying that $\beta_0^M = \gamma_0^{\mathcal{X}}$. Note that when $i = 1, 2$, the supports of β_i^M and $\gamma_i^{\mathcal{X}}$ are contained in $\mathcal{A}|_{y>0} = \{1, 2, \dots, n\} \times \{1, \dots, \binom{n}{2}\}$. Using induction on x-coordinate of $\mathbb{Z}^2$, we will prove that $\beta_1^M = \gamma_1^{\mathcal{X}} - \gamma_2^{\mathcal{X}}$ and $\beta_2^M = \gamma_2^{\mathcal{X}} - \gamma_1^{\mathcal{X}}$ *on the horizontal line* $\mathcal{A}|_{y=1} = \{1, 2, \dots, n\} \times \{1\}$. Note that $\mathcal{K}(1, b) = \{\{x_1\}\}$ for all $1 \le b \le \binom{n}{2}$, and thus again by Theorem 22 and the construction of $\gamma_i^{\mathcal{X}}$, $i = 1, 2$,

$$\text{for } 1 \le b \le \tbinom{n}{2}, \quad \beta_1^M(1, b) = \gamma_1^{\mathcal{X}}(1, b) = 0, \text{ and } \beta_2^M(1, b) = \gamma_2^{\mathcal{X}}(1, b) = 0. \tag{6}$$

Specifically, we have $\beta_1^M(1,1) = \gamma_1^{\mathcal{X}}(1,1) = \gamma_1^{\mathcal{X}}(1,1) - \gamma_2^{\mathcal{X}}(1,1)$ and $\beta_2^M(1,1) = \gamma_2^{\mathcal{X}}(1,1) = \gamma_2^{\mathcal{X}}(1,1) - \gamma_1^{\mathcal{X}}(1,1)$. Fix a natural number $m > 2$ and assume that $\beta_1^M(a,1) = \gamma_1^{\mathcal{X}}(a,1) - \gamma_2^{\mathcal{X}}(a,1)$ and $\beta_2^M(a,1) = \gamma_2^{\mathcal{X}}(a,1) - \gamma_1^{\mathcal{X}}(a,1)$ for $1 \le a \le m-1$. By Theorem A.5 and Theorem C.1 in the full version [4], we have: $\sum_{\mathbf{x} \le (m,1)} \sum_{i=0}^{2} (-1)^i \beta_i^M(\mathbf{x}) \overset{(*)}{=} \sum_{\mathbf{x} \le (m,1)} \sum_{i=0}^{2} (-1)^i \gamma_i^{\mathcal{X}}(\mathbf{x})$. Since (1) $\beta_0^M = \gamma_0^{\mathcal{X}}$ on the entire $\mathbb{Z}^2$, and (2) $\beta_i^M, \gamma_i^{\mathcal{X}}$ vanish outside $\mathcal{A}$ for $i = 1, 2$, the induction hypothesis reduces equality $(*)$ to

$$-\beta_1^M(m,1) + \beta_2^M(m,1) = -\gamma_1^{\mathcal{X}}(m,1) + \gamma_2^{\mathcal{X}}(m,1).$$

By Lemma C.4 in the full version [4], three cases are possible: (Case 1) $\beta_1^M(m,1) = 1$ and $\beta_2^M(m,1) = 0$, (Case 2) $\beta_1^M(m,1) = 0$ and $\beta_2^M(m,1) = 1$, or (Case 3) $\beta_1^M(m,1) = 0$ and $\beta_2^M(m,1) = 0$. Invoking that $\gamma_1^{\mathcal{X}}(m,1)$ and $\gamma_2^{\mathcal{X}}(m,1)$ are non-negative, in all cases, we have

$$\beta_1^M(m,1) = \gamma_1^{\mathcal{X}}(m,1) - \gamma_2^{\mathcal{X}}(m,1), \quad \beta_2^M(m,1) = \gamma_2^{\mathcal{X}}(m,1) - \gamma_1^{\mathcal{X}}(m,1),$$

completing the proof of $\beta_1^M = \gamma_1^{\mathcal{X}} - \gamma_2^{\mathcal{X}}$ and $\beta_2^M = \gamma_2^{\mathcal{X}} - \gamma_1^{\mathcal{X}}$ on $\mathcal{A}|_{y=1}$. We next apply the same strategy to the horizontal lines $y = 2, \dots,$ $y = \binom{n}{2}$ in order, completing the proof. ◀

6 Computation and Algorithms

▶ **Theorem 24.** *Let (X, d_X, f_X) be a finite aug-MS with $n = |X|$.*

(a) *We can compute the ER-staircode $I_{\mathcal{X}} = \{\!\{I_x : x \in X\}\!\}$ in $O(n^2 \log n)$ time. If $X \subset \mathbb{R}^d$ for a fixed d and d_X the Euclidean distance, the time can be improved to $O(n^2 \alpha(n))$, where $\alpha(n)$ is the inverse Ackermann function.*

(b) *Each $I_x \in I_{\mathcal{X}}$ has complexity $O(n)$. Given $I_{\mathcal{X}}$, we can compute zeroth fibered barcode B^L for any line L with positive slope in $O(|B^L| \log n)$ time where $|B^L|$ is the size of B^L.*

(c) *Given $I_{\mathcal{X}}$, we can compute the zeroth graded Betti numbers in $O(n^2)$ time.*

Below we sketch the proof of the above theorem, with missing details in [4].

Consider a function value $\sigma \in \mathbb{R}$, and recall that X_σ consists of all points in X with f_X value at most σ. Let $\mathcal{K}_\sigma = \mathcal{R}_\bullet(X_\sigma, d_X)$ denote the Rips filtration of (X_σ, d_X) (recall Remark 12). The corresponding 1-parameter treegram (dendrogram) is $\theta_\sigma := \pi_0(\mathcal{K}_\sigma)$. On the other hand, for any σ, we can consider the *complete weighted graph* $G_\sigma = (V_\sigma = X_\sigma, E_\sigma)$ with edge weight $w(x, x') = d_X(x, x')$ for any $x, x' \in X_\sigma$. It is folklore that the treegram θ_σ can be computed from the minimum spanning tree (MST) T_σ of G_σ.

Assume all points in X are ordered $x_1, x_2, \ldots, x_n$ such that $f_X(x_i) \leq f_X(x_j)$ whenever $i < j$, and set $\sigma_i = f(x_i)$ for $i \in [1, n]$. Note that as σ varies, X_σ only changes at σ_i. For simplicity, we set $\theta_i := \theta_{\sigma_i} = \pi_0(\mathcal{K}_{\sigma_i})$, $G_i := G_{\sigma_i}$ and $T_i := MST(G_i)$ is the minimum spanning tree (MST) for the weighted graph G_i. Our algorithm depends on the following lemma, the proof of which is in the full version [4].

▶ **Lemma 25.** *A decorated ER-staircode for the finite aug-MS (X, d_X, f_X) can be computed from the collection of treegrams $\{\theta_i, i \in [1, n]\}$ in $O(n^2)$ time.*

In light of the above result, the algorithm to compute ER-staircode is rather simple:

(Step 1): We start with $T_0 =$ empty tree. At the i-th iteration,

(Step 1-a) we update T_{i-1} (already computed) to obtain T_i; and

(Step 1-b) compute θ_i from T_i and θ_{i-1}.

(Step 2): We use the approach described in the proof of Lemma 25 to compute the ER-staircode in $O(n^2)$ time.

For (Step 1-a), note that G_i is obtained by inserting vertex x_i, as well as all $i-1$ edges between (x_i, x_j), $j \in [1, i-1]$, into graph G_{i-1}. By [12], one can update the minimum spanning tree T_{i-1} of G_{i-1} to obtain the MST T_i of G_i in $O(n)$ time.

For (Step 1-b), once all $i-1$ edges spanning i vertices in T_i are sorted, then we can easily build the treegram θ_i in $O(i\alpha(i)) = O(n\alpha(n))$ time, by using union-find data structure (see Figure 14 in the full version [4]). Sorting edges in T_i takes $O(i \log i) = O(n \log n)$ time. Hence the total time spent on (Step 1-b) for all $i \in [1, n]$ is $O(n^2 \log n)$.

Knowing the order of all edges in T_{i-1} does not appear to help, as compared to T_{i-1}, T_i may have $\Omega(i)$ different edges newly introduced, and these new edges still need to be sorted. Nevertheless, we show in the full version [4] that if $X \subset \mathbb{R}^d$ for a fixed dimension d, then each T_i will only have constant number of different edges compared to T_{i-1}, and we can sort all edges in T_i in $O(n)$ time by inserting the new edges to the sorted list of edges in T_{i-1}. Hence θ_i can be computed in $O(n\alpha(n)) + O(n) = O(n\alpha(n))$ time for this case. Putting everything together, Theorem 24 (a) follows. See the full version [4] for the proofs of (b) and (c).

7 Discussion

Some open questions and conjectures are as follows:

1. **Barcodes and elder-rule-staircodes.** We conjecture that if the zeroth homology of the Rips bifiltration of an augmented metric space is interval decomposable, then the barcode must coincide with the elder-rule-staircode. Also, we suspect the sufficient condition for $\mathcal{X}$ to be interval decomposable given in Theorem 17 is actually also a necessary condition. Note that Theorem 17 and these conjectures are closely related to questions raised in [2].

2. **Extension to d-augmented metric spaces.** Can we generalize our results to the setting of more than two parameters? Namely, for d-augmented metric spaces $\mathcal{X}^d := (X, d_X, f_1, f_2, \ldots, f_d)$, $f_i : X \to \mathbb{R}$, $i = 1, \ldots, d$, can we recover the zeroth homological information of the $d+1$-parameter filtration induced by $\mathcal{X}^d$ by devising "an elder-rule-staircode" of $\mathcal{X}^d$? Note that, under the assumption the set $\{(f_i(x))_{i=1}^d \in \mathbb{R}^d : x \in X\}$ is totally ordered in the poset $\mathbb{R}^d$, a straightforward generalization of the elder-rule staircode is conceivable. But it is not clear how to define elder-rule staircode without the assumption.
3. **Extension to higher-order homology.** The ambiguity mentioned in the previous paragraph also arises when trying to devise an "elder-rule-staircode" for higher-order homology of a multiparameter filtration; namely, when $k \geq 1$, the birth indices of k-cycles are not necessarily totally ordered in the multiparameter setting, and thus determining which cycle is older than another is not clear in general.
4. **Metrics and stability.** Recall that the collection $E(\mathcal{X})$ of all possible ER-staircodes of an aug-MS $\mathcal{X}$ is an invariant of $\mathcal{X}$ (the paragraph after Example 7). One possible metric between two collections of ER-staircodes is the Hausdorff distance d_{H}^b in the metric space of barcodes over $\mathbb{R}^2$ with the generalized bottleneck distance d_b [3]. On the other hand, there exists a metric d_{GH}^1 which measures the difference between aug-MSs [11] and let d_{I} be the interleaving distance between 2-parameter persistence modules [23]. Are there constants $\alpha, \beta > 0$ such that for all aug-MSs $\mathcal{X}$ and $\mathcal{Y}$, the inequalities below hold?

$$\alpha \cdot d_{\mathrm{I}}\left(\mathrm{H}_0\left(\mathcal{R}_{\bullet}^{\mathrm{bi}}(\mathcal{X})\right), \mathrm{H}_0\left(\mathcal{R}_{\bullet}^{\mathrm{bi}}(\mathcal{Y})\right)\right) \leq d_{\mathrm{H}}^b(E(\mathcal{X}), E(\mathcal{Y})) \leq \beta \cdot d_{\mathrm{GH}}^1(\mathcal{X}, \mathcal{Y}).$$

5. **Completeness.** Recall that the collection $E(\mathcal{X})$ of all the elder-rule-staircodes of an aug-MS $\mathcal{X}$ is not a complete invariant (the paragraph after Example 7). How faithful is this collection in general? Is there any class of aug-MSs $\mathcal{X}$ such that $E(\mathcal{X})$ completely characterizes $\mathcal{X}$?

References

1 Gorô Azumaya et al. Corrections and supplementaries to my paper concerning krull-remak-schmidt's theorem. *Nagoya Mathematical Journal*, 1:117–124, 1950.

2 Ulrich Bauer, Magnus B Botnan, Steffen Oppermann, and Johan Steen. Cotorsion torsion triples and the representation theory of filtered hierarchical clustering. *arXiv preprint*, 2019. arXiv:1904.07322.

3 Magnus Botnan and Michael Lesnick. Algebraic stability of zigzag persistence modules. *Algebraic & geometric topology*, 18(6):3133–3204, 2018.

4 Chen Cai, Woojin Kim, Mémoli Facundo, and Yusu Wang. Elder-rule-staircodes for augmented metric spaces. *arXiv preprint*, 2020. arXiv:2003.04523.

5 Chen Cai and Yusu Wang. Understanding the power of persistence pairing via permutation test. *arXiv preprint*, 2020. arXiv:2001.06058.

6 G. Carlsson and A. Zomorodian. The theory of multidimensional persistence. *Discrete & Computational Geometry*, 42(1):71–93, 2009. doi:doi:10.1007/s00454-009-9176-0.

7 Gunnar Carlsson and Facundo Mémoli. Characterization, stability and convergence of hierarchical clustering methods. *Journal of machine learning research*, 11(Apr):1425–1470, 2010.

8 Gunnar Carlsson and Facundo Mémoli. Multiparameter hierarchical clustering methods. In *Classification as a Tool for Research*, pages 63–70. Springer, 2010.

9 Mathieu Carriere, Frédéric Chazal, Yuichi Ike, Théo Lacombe, Martin Royer, and Yuhei Umeda. A general neural network architecture for persistence diagrams and graph classification. *arXiv preprint*, 2019. arXiv:1904.09378.

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

12 Francis Chin and David Houck. Algorithms for updating minimal spanning trees. *Journal of Computer and System Sciences*, 16(3):333–344, 1978.

13 David Cohen-Steiner, Herbert Edelsbrunner, and John Harer. Stability of persistence diagrams. *Discrete & Computational Geometry*, 37(1):103–120, 2007.

14 Justin Curry. The fiber of the persistence map for functions on the interval. *Journal of Applied and Computational Topology*, 2(3-4):301–321, 2018.

15 Tamal K Dey and Cheng Xin. Generalized persistence algorithm for decomposing multi-parameter persistence modules. *arXiv preprint*, 2019. `arXiv:1904.03766`.

16 Herbert Edelsbrunner and John Harer. *Computational topology: an introduction*. American Mathematical Soc., 2010.

17 David Eisenbud. *Commutative Algebra: with a view toward algebraic geometry*, volume 150. Springer Science & Business Media, 2013.

18 Christoph Hofer, Roland Kwitt, Marc Niethammer, and Andreas Uhl. Deep learning with topological signatures. In *Advances in Neural Information Processing Systems*, pages 1634–1644, 2017.

19 Woojin Kim and Facundo Mémoli. Generalized persistence diagrams for persistence modules over posets. *arXiv preprint*, 2018. `arXiv:1810.11517`.

20 Woojin Kim and Facundo Mémoli. Spatiotemporal persistent homology for dynamic metric spaces. *Discrete & Computational Geometry*, pages 1–45, 2020. `doi:10.1007/s00454-019-00168-w`.

21 Kevin P. Knudson. A refinement of multi-dimensional persistence. *Homology, Homotopy and Applications*, 10(1):259–281, 2008.

22 Claudia Landi. The rank invariant stability via interleavings. In *Research in Computational Topology*, pages 1–10. Springer, 2018.

23 Michael Lesnick. The theory of the interleaving distance on multidimensional persistence modules. *Found. Comput. Math.*, 15(3):613–650, June 2015. `doi:10.1007/s10208-015-9255-y`.

24 Michael Lesnick and Matthew Wright. Interactive visualization of 2-d persistence modules. *arXiv preprint*, 2015. `arXiv:1512.00180`.

25 Michael Lesnick and Matthew Wright. Computing minimal presentations and betti numbers of 2-parameter persistent homology. *arXiv preprint*, 2019. `arXiv:1902.05708`.

26 Alex McCleary and Amit Patel. Multiparameter persistence diagrams. *arXiv preprint*, 2019. `arXiv:1905.13220v3`.

27 Amit Patel. Generalized persistence diagrams. *Journal of Applied and Computational Topology*, 1(3-4):397–419, 2018.

28 Irena Peeva. *Graded syzygies*, volume 14. Springer Science & Business Media, 2010.

29 Zane Smith, Samir Chowdhury, and Facundo Mémoli. Hierarchical representations of network data with optimal distortion bounds. In *Signals, Systems and Computers, 2016 50th Asilomar Conference on*, pages 1834–1838. IEEE, 2016.

30 Qi Zhao and Yusu Wang. Learning metrics for persistence-based summaries and applications for graph classification. In *33rd Annu. Conf. Neural Inf. Processing Systems (NeuRIPS)*, 2019. to appear.

31 Afra Zomorodian and Gunnar Carlsson. Computing persistent homology. *Discrete & Computational Geometry*, 33(2):249–274, 2005.

Faster Approximation Algorithms for Geometric Set Cover

Timothy M. Chan
Department of Computer Science, University of Illinois at Urbana-Champaign, IL, USA
tmc@illinois.edu

Qizheng He
Department of Computer Science, University of Illinois at Urbana-Champaign, IL, USA
qizheng6@illinois.edu

Abstract

We improve the running times of $O(1)$-approximation algorithms for the set cover problem in geometric settings, specifically, covering points by disks in the plane, or covering points by halfspaces in three dimensions. In the unweighted case, Agarwal and Pan [SoCG 2014] gave a randomized $O(n \log^4 n)$-time, $O(1)$-approximation algorithm, by using variants of the multiplicative weight update (MWU) method combined with geometric data structures. We simplify the data structure requirement in one of their methods and obtain a *deterministic* $O(n \log^3 n \log\log n)$-time algorithm. With further new ideas, we obtain a still faster randomized $O(n \log n (\log\log n)^{O(1)})$-time algorithm.

For the weighted problem, we also give a randomized $O(n \log^4 n \log\log n)$-time, $O(1)$-approximation algorithm, by simple modifications to the MWU method and the quasi-uniform sampling technique.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Set cover, approximation algorithms, multiplicate weight update method, random sampling, shallow cuttings

Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.27

Related Version A full version of this paper is available at `http://arxiv.org/abs/2003.13420`.

Funding *Timothy M. Chan*: Supported in part by NSF Grant CCF-1814026.

1 Introduction

Unweighted geometric set cover. In this paper we study one of the most fundamental classes of geometric optimization problems: *geometric set cover*. Given a set X of $O(n)$ points and a set S of $O(n)$ geometric objects, find the smallest subset of objects from S to cover all points in X. In the dual set system, the problem corresponds to *geometric hitting set* (finding the smallest number of points from X that hit all objects in S).

This class of problems has been *extensively* investigated in the computational geometry literature. Since they are NP-hard in most scenarios, attention is turned towards approximation algorithms. Different types of objects give rise to different results. Typically, approximation algorithms fall into the following categories:

1. Simple heuristics, e.g., greedy algorithms.
2. Approaches based on solving the linear programming (LP) relaxation (i.e., fractional set cover) and rounding the LP solution.
3. Polynomial-time approximation schemes (PTASs), e.g., via local search, shifted grids/quadtrees (sometimes with dynamic programming), or separator-based divide-and-conquer.

Generally, greedy algorithms achieve only logarithmic approximation factors (there are some easy cases where they give $O(1)$ approximation factors, e.g., hitting set for fat objects such as disks/balls in the "continuous" setting with $X = \mathbb{R}^d$ [20]). The LP-based approaches give better approximation factors in many cases, e.g., $O(1)$ approximation for set cover and hitting set for disks in 2D and halfspaces in 3D, set cover for objects in 2D with linear "union complexity", and hitting set for pseudodisks in 2D [7, 19, 5, 32, 29]. Subsequently, local-search PTASs have been found by Mustafa and Ray [27] in some cases, including set cover and hitting set for disks in 2D and halfspaces in 3D (earlier, PTASs were known for hitting set only in the continuous setting for unit disks/balls [21], and for arbitrary disks/balls and fat objects [11]).

Historically, the focus has been on obtaining good approximation factors. Here, we are interested in obtaining approximation algorithms with good – ideally, near linear – running time. Concerning the efficiency of known approximation algorithms:

1. Certain simple heuristics can lead to fast $O(1)$-approximation algorithms in some easy cases (e.g., continuous hitting set for unit disks or unit balls by using grids), but generally, even simple greedy algorithms may be difficult to implement in near linear time (as they may require nontrivial dynamic geometric data structures).
2. LP-based approaches initially may not seem to be the most efficient, because of the need to solve an LP. However, a general-purpose LP solver can be avoided. The set-cover LP can alternatively be solved (approximately) by the *multiplicative weight update* (MWU) method. In the computational geometry literature, the technique has been called *iterative reweighting*, and its use in geometric set cover was explored by Brönnimann and Goodrich [7] (one specific application appeared in an earlier work by Clarkson [18]), although the technique was known even earlier outside of geometry. On the other hand, the LP-rounding part corresponds to the well-known geometric problem of constructing *ε-nets*, for which efficient algorithms are known [15, 25].
3. PTAS approaches generally have large polynomial running time, even when specialized to specific approximation factors. For example, see [8] for efforts in improving the degree of the polynomial.

In this paper we design faster approximation algorithms for geometric set cover via the LP/MWU-based approaches. There has been a series of work on speeding up MWU methods for covering or packing LPs (e.g., see [17, 22, 33]). In geometric settings, we would like more efficient algorithms (as generating the entire LP explicitly would already require quadratic time), by somehow exploiting geometric data structures. The main previous work was by Agarwal and Pan [4] from SoCG 2014, who showed how to compute an $O(1)$-approximation for set cover for 2D disks or 3D halfspaces, in $O(n \log^4 n)$ randomized time.

Agarwal and Pan actually proposed two MWU-based algorithms: The first is a simple variant of the standard MWU algorithm of Brönnimann and Goodrich, which proceeds in logarithmically many rounds. The second views the problem as a 2-player zero-sum game, works quite differently (with weight updates to both points and objects), and uses randomization; the analysis is more complicated. Because the first algorithm requires stronger data structures – notably, for approximate weighted range counting with dynamic changes to the weights – Agarwal and Pan chose to implement their second algorithm instead, to get their $O(n \log^4 n)$ result for 3D halfspaces.

New results. In this paper we give:

- a *deterministic* near-linear $O(1)$-approximation algorithm for set cover for 3D halfspaces. Its running time is $O(n \log^3 n \log\log n)$, which besides eliminating randomization is also a little faster than Agarwal and Pan's;

- a still faster randomized near-linear $O(1)$-approximation algorithm for set cover for 3D halfspaces. Its running time is $O(n \log n \log^{O(1)} \log n)$, which is essentially optimal[1] ignoring minor $\log \log n$ factors.

Although generally shaving logarithmic factors may not be the most important endeavor, the problem is fundamental enough that we feel it worthwhile to find the most efficient algorithm possible.

Our approach interestingly is to go back to Agarwal and Pan's first MWU algorithm. We show that with one simple modification, the data structure requirement can actually be relaxed: namely, for the approximate counting structure, there is no need for weights, and the only update operation is insertion. By standard techniques, insertion-only data structures reduce to static data structures. This simple idea immediately yields our deterministic result. (Before, Bus et al. [9] also aimed to find variants of Agarwal and Pan's first algorithm with simpler data structures, but they did not achieve improved theoretical time bounds.) Our best randomized result requires a more sophisticated combination of several additional ideas. In particular, we incorporate random sampling in the MWU algorithm, and extensively use *shallow cuttings*, in both primal and dual space.

We have stated our results for set cover for 3D halfspaces. This case is arguably the most central. It is equivalent to hitting set for 3D halfspaces, by duality, and also includes set cover and hitting set for 2D disks as special cases, by the standard lifting transformation. The case of 3D dominance ranges is another special case, by a known transformation [14, 28] (although for the dominance case, word-RAM techniques can speed up the algorithms further). The ideas here are likely useful also in the other cases considered in Agarwal and Pan's paper (e.g., hitting set for rectangles, set cover for fat triangles, etc.), but in the interest of keeping the paper focused, we will not discuss these implications.

Weighted geometric set cover. Finally, we consider the weighted version of set cover: assuming that each object is given a weight, we now want a subset of the objects of R with the minimum total weight that covers all points in X. The weighted problem has also received considerable attention: Varadarajan [31] and Chan et al. [13] used the LP-based approach to obtain $O(1)$-approximation algorithms for weighted set cover for 3D halfspaces (or for objects in 2D with linear union complexity); the difficult part is in constructing ε-nets with small weights, which they solved by the *quasi-random sampling* technique. Later, Mustafa, Raman, and Ray [26] discovered a quasi-PTAS for 3D halfspaces by using geometric separators; the running time is very high ($n^{\log^{O(1)} n}$).

Very recently, Chekuri, Har-Peled, and Quanrud [16] described new randomized MWU methods which can efficiently solve the LP corresponding to various generalizations of geometric set cover, by using appropriate geometric data structures. In particular, for weighted set cover for 3D halfspaces, they obtained a randomized $O(n \log^{O(1)} n)$-time algorithm to solve the LP but with an unspecified number of logarithmic factors. They did not address the LP-rounding part, i.e., construction of an ε-net of small weight – a direct implementation of the quasi-uniform sampling technique would not lead to a near-linear time bound.

We observe that a simple direct modification of the standard MWU algorithm of Brönnimann and Goodrich, or Agarwal and Pan's first algorithm, can also solve the LP for weighted geometric set cover, with arguably simpler data structures than Chekuri et al.'s. Secondly,

[1] Just deciding whether a solution exists requires $\Omega(n \log n)$ time in the algebraic decision-tree model, even for 1D intervals.

we observe that an ε-net of small weight can be constructed in near-linear time, by using quasi-uniform sampling more carefully. This leads to a randomized $O(n \log^4 n \log\log n)$-time, $O(1)$-approximation algorithm for weighted set cover for 3D halfspaces (and thus for 2D disks).

2 Preliminaries

Let X be a set of points and S be a set of objects. For a point p, its *depth* in S refers to the number of objects in S containing p. A point p is said to be *ε-light* in S, if it has depth $\leq \varepsilon|S|$ in S; otherwise it is *ε-heavy*. A subset of objects $T \subseteq S$ is an *ε-net* of S if T covers all points that are ε-heavy in S.

It is known that there exists an ε-net with size $O(\frac{1}{\varepsilon})$ for any set of halfspaces in 3D or disks in 2D [24] (or more generally for objects in the plane with linear union complexity [19]).

2.1 The Basic MWU Algorithm

We first review the standard multiplicative weight[2] update (MWU) algorithm for geometric set cover, as described by Brönnimann and Goodrich [7] (which generalizes an earlier algorithm by Clarkson [18], and is also well known outside of computational geometry).

Let X be the set of input points and S be the set of input objects, with $n = |X| + |S|$. Let OPT denote the size of the minimum set cover. We assume that a value $t = \Theta(\mathrm{OPT})$ is known; this assumption will be removed later by a binary search for t. In the following pseudocode, we work with a multiset $\hat{S}$; in measuring size or counting depth, we include multiplicities (e.g., $|\hat{S}|$ is the sum of the multiplicities of all its elements).

1: Guess a value $t \in [\mathrm{OPT}, 2\,\mathrm{OPT}]$ and set $\varepsilon = \frac{1}{2t}$.
2: Define a multiset $\hat{S}$ where each object i in S initially has multiplicity $m_i = 1$.
3: **while** we can find a point $p \in X$ which is ε-light in $\hat{S}$ **do**
4: **for** each object i containing p **do** ▷ call lines 4–5 a *multiplicity-doubling step*
5: Double its multiplicity m_i.
6: Return an ε-net of the multiset $\hat{S}$.

Since at the end all points in X are ε-heavy in $\hat{S}$, the returned subset is a valid set cover of X. For halfspaces in 3D or disks in 2D, its size is $O(\frac{1}{\varepsilon}) = O(t) = O(\mathrm{OPT})$.

A standard analysis shows that the algorithm always terminates after $O(t \log \frac{n}{t})$ multiplicity-doubling steps. We include a quick proof: Each multiplicity-doubling step increases $|\hat{S}|$ by a factor of at most $1 + \varepsilon$, due to the ε-lightness of p. Thus, after z doubling steps, $|\hat{S}| \leq n(1+\varepsilon)^z \leq ne^{\varepsilon z} = ne^{z/(2t)}$. On the other hand, consider a set cover T^* of size t. In each multiplicity-doubling step, at least one of the objects in T^* has its multiplicity doubled. So, after z multiplicity-doubling steps, the total multiplicity in T^* is at least $t2^{z/t}$. We conclude that $t2^{z/t} \leq |\hat{S}| \leq ne^{z/(2t)}$, implying that $z = O(t \log \frac{n}{t})$.

[2] In our algorithm description, we prefer to use the term "multiplicity" instead of "weight", to avoid confusion with the weighted set cover problem later.

2.2 Agarwal and Pan's (First) MWU Algorithm

Next, we review Agarwal and Pan's first variant of the MWU algorithm [4]. One issue in implementing the original algorithm lies in the test in line 3: searching for one light point by scanning all points in X from scratch every time seems inefficient. In Agarwal and Pan's refined approach, we proceed in a small number of rounds, where in each round, we examine the points in X in a fixed order and test for lightness in that order.

1: Guess a value $t \in [\mathrm{OPT}, 2\,\mathrm{OPT}]$ and set $\varepsilon = \frac{1}{2t}$.
2: Define a multiset $\hat{S}$ where each object i in S initially has multiplicity $m_i = 1$.
3: **loop** ▷ call this the start of a new *round*
4: **for** each point $p \in X$ in any fixed order **do**
5: **while** p is ε-light in $\hat{S}$ **do**
6: **for** each object i containing p **do** ▷ call lines 6–7 a *multiplicity-doubling step*
7: Double its multiplicity m_i.
8: **if** the number of multiplicity-doubling steps in this round exceeds t **then**
9: Go to line 3 and start a new round.
10: Terminate and return an $\frac{\varepsilon}{2}$-net of the multiset $\hat{S}$.

To justify correctness, observe that since each round performs at most t multiplicity-doubling steps, $|\hat{S}|$ increases by a factor of at most $(1+\varepsilon)^t \le e^{\varepsilon t} \le e^{1/2} < 2$. Thus, a point p that is checked to be ε-heavy in $\hat{S}$ at any moment during the round will remain $\frac{\varepsilon}{2}$-heavy in $\hat{S}$ at the end of the round.

Since all but the last round performs t multiplicity-doubling steps and we have already shown that the total number of such steps is $O(t \log \frac{n}{t})$, the number of rounds is $O(\log \frac{n}{t})$.

3 "New" MWU Algorithm

Agarwal and Pan's algorithm still requires an efficient data structure to test whether a given point is light, and the data structure needs to support dynamic changes to the multiplicities. We propose a new variant that requires simpler data structures.

Our new algorithm is almost identical to Agarwal and Pan's, but with just one very simple change! Namely, after line 3, at the beginning of each round, we add the following line, to readjust all multiplicities:

3.5: for each object i, reset its multiplicity $m_i \leftarrow \lceil m_i \frac{10n}{|\hat{S}|} \rceil$.

To analyze the new algorithm, consider modifying the multiplicity m_i instead to $\lceil m_i \frac{10n}{|\hat{S}|} \rceil \cdot \frac{|\hat{S}|}{10n}$. The algorithm behaves identically (since the multiplicities are identical except for a common rescaling factor), but is more convenient to analyze. In this version, multiplicities are nondecreasing over time (though they may be non-integers). After the modified line 3.5, the new $|\hat{S}|$ is at most $\sum_i \left(m_i \frac{10n}{|\hat{S}|} + 1 \right) \cdot \frac{|\hat{S}|}{10n} \le 11n \cdot \frac{|\hat{S}|}{10n} = 1.1|\hat{S}|$. If the algorithm makes z multiplicity-doubling steps, then it performs line 3.5 at most z/t times and we now have $|\hat{S}| \le n(1+\varepsilon)^z \cdot 1.1^{z/t} \le n e^{z/(2t)} \cdot 1.1^{z/t}$. This is still sufficient to imply that $z = O(t \log \frac{n}{t})$, and so the number of rounds remains $O(\log \frac{n}{t})$.

Now, let's go back to line 3.5 as written. The advantage of this multiplicity readjustment step is that it decreases $|\hat{S}|$ to $\sum_i \left(m_i \frac{10n}{|\hat{S}|} + 1\right) = O(n)$. At the end of the round, $|\hat{S}|$ increases by a factor of at most $(1+\varepsilon)^t < 2$ and so remains $O(n)$. Thus, in line 7, instead of doubling the multiplicity of an object, we can just repeatedly increment the multiplicity (i.e., insert one copy of an object) to reach the desired value. The total number of increments per round is $O(n)$.

Note that in testing for ε-lightness in line 5, a constant-factor approximation of the depth is sufficient, with appropriate adjustments of constants in the algorithm. Also, although the algorithm as described may test the same point p for lightness several times in a round, this can be easily avoided: we just keep track of the increase D in the depth of the current point p; the new depth of p can be 2-approximated by the maximum of the old depth and D.

To summarize, an efficient implementation of each round of the new algorithm requires solving the following geometric data structure problems (Report for line 6, and Approx-Count-Decision for line 5):

Problem Report: Design a data structure to store a static set S of size $O(n)$ so that given a query point $p \in X$, we can report all objects in S containing the query point p. Here, the output size of a query is guaranteed to be at most $O(k)$ where $k := \frac{n}{t}$ (since ε-lightness of p implies that its depth is at most $\varepsilon|\hat{S}| = \Theta(\frac{n}{t})$ even including multiplicities).

Problem Approx-Count-Decision: Design a data structure to store a multiset $\hat{S}$ of size $O(n)$ so that given a query point $p \in X$, we can either declare that the number of objects in $\hat{S}$ containing p is less than a fixed threshold value k, or that the number is more than $\frac{k}{c}$, for some constant $c > 1$. Here, the threshold again is $k := \frac{n}{t}$ (since $\varepsilon|\hat{S}| = \Theta(\frac{n}{t})$). The data structure should support the following type of updates: insert one copy of an object to $\hat{S}$. (Deletions are not required.) Each point in X is queried once.

To bound the cost of the algorithm:

- Let T_{report} denote the total time for $O(t)$ queries in Problem Report.
- Let T_{count} denote the total time for $O(n)$ queries and $O(n)$ insertions in Problem Approx-Count-Decision. (Note that the initialization of $\hat{S}$ at the beginning of the round can be done by $O(n)$ insertions.)
- Let T_{net} denote the time for computing an ε-net of size $O(\frac{1}{\varepsilon})$ for a given multiset $\hat{S}$ of size $O(n)$.

The total running time over all $O(\log \frac{n}{t})$ rounds is

$$O((T_{\text{report}} + T_{\text{count}}) \log \tfrac{n}{t} + T_{\text{net}}). \tag{1}$$

4 Implementations

In this section, we describe specific implementations of our MWU algorithm when the objects are halfspaces in 3D (which include disks in 2D as a special case by the standard lifting transformation). We first consider deterministic algorithms.

4.1 Deterministic Version

Shallow cuttings. We begin by reviewing an important tool that we will use several times later. For a set of n planes in $\mathbb{R}^3$, a *k-shallow ε-cutting* is a collection of interior-disjoint polyhedral cells, such that each cell intersects at most εn planes, and the union of the cells cover all points of level at most k (the *level* of a point refers to the number of planes below it). The list of all planes intersecting a cell Δ is called the *conflict list* of Δ. Matoušek [23]

proved the existence of a k-shallow $(\frac{ck}{n})$-cutting with $O(\frac{n}{k})$ cells for any constant c. Chan and Tsakalidis [15] gave an $O(n \log \frac{n}{k})$-time deterministic algorithm to construct such a cutting, along with all its conflict lists (an earlier randomized algorithm was given by Ramos [30]). If c is sufficiently large, the cells may be made "downward", i.e., they all contain $(0, 0, -\infty)$.

Constructing ε-nets. The best known deterministic algorithm for constructing ε-nets for 3D halfspaces is by Chan and Tsakalidis [15] and runs in $T_{\text{net}} = O(n \log \frac{1}{\varepsilon}) = O(n \log n)$ time.

The result follows directly from their shallow cutting algorithm (using a simple argument of Matoušek [23]): Without loss of generality, assume that all halfspaces are upper halfspaces, so depth corresponds to level with respect to the bounding planes (we can compute a net for lower halfspaces separately and take the union, with readjustment of ε by a factor of 2). We construct an (εn)-shallow $\frac{\varepsilon}{2}$-cutting with $O(\frac{1}{\varepsilon})$ cells, and for each cell, add a plane completely below the cell (if it exists) to the net. To see correctness, for a point p with level εn, consider the cell Δ containing p; at least $\varepsilon n - \frac{\varepsilon n}{2} > 0$ planes are completely below Δ, and so the net contains at least one plane below p.

Solving Problem Report. This problem corresponds to 3D *halfspace range reporting* in dual space, and by known data structures [10, 2, 15], the total time to answer $O(t)$ queries is $T_{\text{report}} = O(t \cdot (\log n + k)) = O(t \log n + n)$, assuming an initial preprocessing of $O(n \log n)$ time (which is done only once).

This result also follows directly from shallow cuttings (since space is not our concern, the solution is much simplified): Without loss of generality, assume that all halfspaces are upper halfspaces. We construct a k-shallow $O(\frac{k}{n})$-cutting with $O(\frac{n}{k})$ downward cells. Given a query point $p \in X$, we find the cell containing p, which can be done in $O(\log n)$ time by planar point location; we then do a linear search over its conflict list, which has size $O(k)$.

Note that the point location operations can be actually be done during preprocessing in $O(n \log n)$ time since X is known in advance. This lowers the time bound for $O(t)$ queries to $T_{\text{report}} = O(tk) = O(n)$.

Solving Problem Approx-Count-Decision. This problem corresponds to the decision version of 3D *halfspace approximate range counting* in dual space, and several deterministic and randomized data structures have already been given in the static case [1, 3], achieving $O(\log n)$ query time and $O(n \log n)$ preprocessing time.

This result also follows directly from shallow cuttings: Without loss of generality, assume that all halfspaces are upper halfspaces. We construct a $\frac{n}{b^i}$-shallow $O(\frac{1}{b^i})$-cutting with $O(b^i)$ downward cells for every $i = 1, \ldots, \log_b n$ for some constant b. Chan and Tsakalidis's algorithm can actually construct all $O(\log n)$ such cuttings in $O(n \log n)$ total time. With these cuttings, we can compute an $O(1)$-approximation to the depth/level of a query point p by simply finding the largest i such that p is contained in a cell of the $\frac{n}{b^i}$-shallow cutting (the level of p would then be $O(\frac{n}{b^i})$ and at least $\frac{n}{b^{i+1}}$). In Chan and Tsakalidis's construction, each cell in one cutting intersects $O(1)$ cells in the next cutting, and so we can locate the cells containing p in $O(1)$ time per i, for a total of $O(\log n)$ time.

To solve Problem Approx-Count-Decision, we still need to support insertion. Although the approximate decision problem is not decomposable, the above solution solves the approximate counting problem, which is decomposable, so we can apply the standard *logarithmic method* [6] to transform the static data structure into a semi-dynamic, insertion-only data structure. The transformation causes a logarithmic factor increase, yielding in our case $O(\log^2 n)$ query time and $O(\log^2 n)$ insertion time. Thus, the total time for $O(n)$ queries and insertions is $T_{\text{count}} = O(n \log^2 n)$.

Conclusion. By (1), the complete algorithm has running time $O((T_{\text{report}} + T_{\text{count}}) \log \frac{n}{t} + T_{\text{net}}) = O((n + n \log^2 n) \log \frac{n}{t} + n \log n) = O(n \log^3 n)$.

One final issue remains: we have assumed that a value $t \in [\text{OPT}, 2\,\text{OPT}]$ is given. In general, either the algorithm produces a solution of size $O(t)$, or (if it fails to complete within $O(\log \frac{n}{t})$ rounds) the algorithm may conclude that $\text{OPT} > t$. We can thus find an $O(1)$-approximation to OPT by a binary search over t among the $O(\log n)$ possible powers of 2, with $O(\log \log n)$ calls to the algorithm. The final time bound is $O(n \log^3 n \log \log n)$.

▶ **Theorem 1.** *Given $O(n)$ points and $O(n)$ halfspaces in $\mathbb{R}^3$, we can find a subset of halfspaces covering all points, of size within $O(1)$ factor of the minimum, in deterministic $O(n \log^3 n \log \log n)$ time.*

4.2 Randomized Version 1

We now describe a better solution to Problem Approx-Count-Decision, by using randomization and the fact that all query points (namely, X) are given in advance.

Reducing the number of insertions in Problem Approx-Count-Decision. In solving Problem Approx-Count-Decision, one simple way to speed up insertions is to work with a random sample R of $\hat{S}$. When we insert an object to $\hat{S}$, we independently decide to insert it to the sample R with probability $\rho := \frac{c_0 \log n}{k}$, or ignore it with probability $1 - \rho$, for a sufficiently large constant c_0. (Different copies of an object are treated as different objects here.) It suffices to solve the problem for the sample R with the new threshold around ρk.

To justify correctness, consider a fixed query point $p \in X$. Let $x_1, x_2, \ldots$ be the sequence of objects in $\hat{S}$ that contain p, in the order in which they are inserted (extend the sequence arbitrarily to make its length greater than k). Let $y_i = 1$ if object x_i is chosen to be in the sample R, or 0 otherwise. Note that the x_i's may not be independent (since the object we insert could depend on random choices made before); however, the y_i's are independent. By the Chernoff bound, $\sum_{i=1}^{k/c} y_i \le \frac{(1+\delta)\rho k}{c}$ and $\sum_{i=1}^{k} y_i \ge (1-\delta)\rho k$ with probability $1 - e^{-\Omega(\rho k)} = 1 - n^{-\Omega(c_0)}$ for any fixed constant $\delta > 0$. Thus, with high probability, at any time, if the number of objects in R containing p is more than $\frac{(1+\delta)\rho k}{c}$, then the number of objects in $\hat{S}$ containing p is more than $\frac{k}{c}$; if the former number is less than $(1-\delta)\rho k$, then the later number is less than k. Since there are $O(n)$ possible query points, all queries are correct with high probability.

By this strategy, the number of insertions is reduced to $O(\rho n) = O(\frac{n}{k} \log n) = O(t \log n)$ with high probability.

Preprocessing step. Next we use a known preprocessing step to ensure that each object contains at most $\frac{n}{t}$ points, in the case of 3D halfspaces. This subproblem was addressed in Agarwal and Pan's paper [4] (where it was called "(P5)" – curiously, they used it to implement their second algorithm but not their first MWU-based algorithm.) We state a better running time:

▶ **Lemma 2.** *In $O(n \log t)$ time, we can find a subset $T_0 \subseteq S$ of $O(t)$ halfspaces, such that after removing all points in X covered by T_0, each halfspace of S contains at most $\frac{n}{t}$ points.*

Proof. We may assume that all halfspaces are upper halfspaces. We work in dual space, where S is now a set of points and X is a set of planes. The goal is to find a subset $T_0 \subseteq S$ of $O(t)$ points such that after removing all planes of X that are below some points of T_0, each point of S has depth/level at most $\frac{n}{t}$.

We proceed in rounds. Let b be a constant. In the i-th round, assume that all points of S have level $\leq \frac{n}{b^i}$. Compute a $\frac{n}{b^i}$-shallow $\frac{1}{b^{i+1}}$-cutting with $O(b^i)$ cells. In each cell, add an arbitrary point of S (if exists) to the set T_0. In total $O(b^i)$ points are added. Remove all planes that are below these added points from X.

Consider a point p of S. Let Δ be the cell containing p, and let q be the point in Δ that was added to T_0. Any plane that is below p but not removed (and thus above q) must intersect Δ, so there can be at most $\frac{n}{b^{i+1}}$ such planes. Thus, after the round, the level of p is at most $\frac{n}{b^{i+1}}$. We terminate when b^i reaches t. The total size of T_0 is $O(\sum_{i=1}^{\log_b t} b^i) = O(t)$.

Naively computing each shallow cutting from scratch by Chan and Tsakalidis's algorithm would require $O(n \log n \cdot \log n) = O(n \log^2 n)$ total time. But Chan and Tsakalidis's approach can compute multiple shallow cuttings more quickly: given a $\frac{n}{b^i}$-shallow cutting along with its conflict lists, we can compute the next $\frac{n}{b^{i+1}}$-shallow cutting along with its conflict lists in $O(n + b^i \log b^i)$ time. However, in our application, before computing the next cutting, we also remove some of the input planes. Fortunately, this type of scenario has been examined in a recent paper by Chan [12], who shows that the approach still works, provided that the next cutting is relaxed to cover only points covered by the previous cutting (see Lemma 8 in his paper); this is sufficient in our application. In our application, we also need to locate the cell containing each point of S. This can still be done in $O(n)$ time given the locations in the previous cutting. Thus, the total time is $O(\sum_{i=1}^{\log_b t}(n + b^i \log b^i)) = O(n \log t)$. ◀

At the end, we add T_0 back to the solution, which still has $O(t)$ total size.

Solving Problem Approx-Count-Decision. We now propose a very simple approach to solve Problem Approx-Count-Decision: just explicitly maintain the depth of all points in X. Each query then trivially takes $O(1)$ time. When inserting an object, we find all points contained in the object and increment their depths.

Due to the above preprocessing step, the number of points contained in the object is $O(\frac{n}{t})$. For the case of 3D halfspaces, we can find these points by halfspace range reporting; as explained before for Problem Report, this can be done in $O(\frac{n}{t})$ time by using shallow cuttings, after an initial preprocessing in $O(n \log n)$ time. Thus, each insertion takes $O(\frac{n}{t})$ time. Since the number of insertions has been reduced to $O(t \log n)$ by sampling, the total time for Problem Approx-Count-Decision is $T_{\text{count}} = O((t \log n) \cdot \frac{n}{t}) = O(n \log n)$.

Conclusion. By (1), the complete randomized algorithm has running time $O((T_{\text{report}} + T_{\text{count}}) \log \frac{n}{t} + T_{\text{net}}) = O((n + n \log n) \log \frac{n}{t} + n \log n) = O(n \log n \log \frac{n}{t}) = O(n \log^2 n)$ (even including the $O(n \log n)$-time preprocessing step). Including the binary search for t, the time bound is $O(n \log^2 n \log \log n)$.

4.3 Randomized Version 2

Finally, we combine the ideas from both the deterministic and randomized implementations, to get our fastest randomized algorithm for 3D halfspaces.

Solving Problem Approx-Count-Decision. We may assume that all halfspaces are upper halfspaces. We work in dual space, where $\hat{S}$ is now a multiset of points and X is a set of planes. In a query, we want to approximately count the number of points in $\hat{S}$ that are above a query plane in X. By the sampling reduction from Section 4.2, we may assume that the number of insertions to $\hat{S}$ is $O(t \log n)$. By the preprocessing step from Section 4.2, we may assume that all points in $\hat{S}$ have level at most $\frac{n}{t}$.

Compute a $\frac{n}{t}$-shallow $O(\frac{1}{t})$-cutting with $O(t)$ downward cells, along with its conflict lists. For each point $p \in R$, locate the cell containing p. All this can be done during a (one-time) preprocessing in $O(n \log n)$ time.

For each cell Δ, we maintain $\hat{S} \cap \Delta$ in a semi-dynamic data structure for 3D approximate halfspace range counting. As described in Section 4.1, we get $O(\log^2 n_\Delta)$ query and insertion time, where $n_\Delta = |\hat{S} \cap \Delta|$.

In an insertion of a point p to $\hat{S}$, we look up the cell Δ containing p and insert the point to the approximate counting structure in Δ.

In a query for a plane $h \in X$, we look up the cells Δ whose conflict lists contain h, answer approximate counting queries in these cells, and sum the answers.

We bound the total time for all insertions and queries. For each cell Δ, the number of insertions in its approximate counting structure is n_Δ and the number of queries is $O(\frac{n}{t})$ (since each plane $h \in X$ is queried once). The total time is

$$O\left(\sum_\Delta (\tfrac{n}{t} + n_\Delta) \log^2 n_\Delta\right).$$

Since there are $O(t)$ terms and $\sum_\Delta n_\Delta = O(t \log n)$, we have $n_\Delta = O(\log n)$ "on average"; applying Jensen's inequality to the first term, we can bound the sum by $O(n \log^2 \log n + t \log^3 n)$. Thus, $T_{\text{count}} = O(n \log^2 \log n + t \log^3 n)$.

Conclusion. By (1), the complete randomized algorithm has running time $O((T_{\text{report}} + T_{\text{count}}) \log \frac{n}{t} + T_{\text{net}}) = O((n + n \log^2 \log n + t \log^3 n) \log \frac{n}{t} + n \log n) = O(n \log n \log^2 \log n + t \log^4 n)$. If $t \le n/\log^3 n$, the first term dominates. On the other hand, if $t > n/\log^3 n$, our earlier randomized algorithm has running time $O(n \log n \log \frac{n}{t}) = O(n \log n \log \log n)$. In any case, the time bound is at most $O(n \log n \log^2 \log n)$. Including the binary search for t, the time bound is $O(n \log n \log^3 \log n)$.

▶ **Theorem 3.** *Given $O(n)$ points and $O(n)$ halfspaces in $\mathbb{R}^3$, we can find a subset of halfspaces covering all points, of size within $O(1)$ factor of the minimum, in $O(n \log n \log^3 \log n)$ time by a randomized Monte-Carlo algorithm with error probability $O(n^{-c_0})$ for any constant c_0.*

Remark. The number of the $\log \log n$ factors is improvable with still more effort, but we feel it is of minor significance.

5 Weighted Set Cover

In this final section, we consider the weighted set cover problem. We define ε-lightness and ε-nets as before, ignoring the weights. It is known that there exists an ε-net of S with total weight $O(\frac{1}{\varepsilon} \cdot \frac{w(S)}{|S|})$, for any set of 3D halfspaces or 2D disks (or objects in 2D with linear union complexity) [13]. Here, the weight $w(S)$ of a set S refers to the sum of the weights of the objects in S.

5.1 MWU Algorithm in the Weighted Case

Let X be the set of input points and S be the set of weighted input objects, where object i has weight w_i, with $n = |X| + |S|$. Let OPT be the weight of the minimum-weight set cover. We assume that a value $t \in [\text{OPT}, 2\,\text{OPT}]$ is given; this assumption can be removed by a binary search for t.

We may delete objects with weights $> t$. We may automatically include all objects with weights $< \frac{1}{n}t$ in the solution, and delete them and all points covered by them, since the total weight of the solution increases by only $O(n \cdot \frac{1}{n}t) = O(t)$. Thus, all remaining objects have weights in $[\frac{1}{n}t, t]$. By rescaling, we may now assume that all objects have weights in $[1, n]$ and that $t = \Theta(n)$.

In the following, for a multiset $\hat{S}$ where object i has multiplicity m_i, the weight of the multiset is defined as $w(\hat{S}) = \sum_i m_i w_i$.

We describe a simple variant of the basic MWU algorithm to solve the weighted set cover problem. (A more general, randomized MWU algorithm for geometric set cover was given recently by Chekuri, Har-Peled, and Quanrud [16], but our algorithm is simpler to describe and analyze.) The key innovation is to replace doubling with multiplication by a factor $1 + \frac{1}{w_i}$, where w_i is the weight of the concerned object i. (Note that multiplicities may now be non-integers.)

1: Guess a value $t \in [\mathrm{OPT}, 2\,\mathrm{OPT}]$.
2: Define a multiset $\hat{S}$ where each object i in S initially has multiplicity $m_i = 1$.
3: **repeat**
4: Find a point p which is ε-light in $\hat{S}$ with $\varepsilon = \frac{1}{2t} \cdot \frac{w(\hat{S})}{|\hat{S}|}$.
5: **for** each object i containing p **do** ▷ call lines 5–6 a "multiplicity-increasing step"
6: Multiply its multiplicity m_i by $1 + \frac{1}{w_i}$.
7: **until** all points are ε-heavy in $\hat{S}$.
8: Return an ε-net of the multiset $\hat{S}$.

Since at the end all points are ε-heavy in $\hat{S}$, the returned subset is a valid set cover of X. For halfspaces in 3D or disks in 2D, its weight is $O(\frac{1}{\varepsilon} \cdot \frac{w(\hat{S})}{|\hat{S}|}) = O(\mathrm{OPT})$.

We now prove that the algorithm terminates in $O(t \log n) = O(n \log n)$ multiplicity-increasing steps.

In each multiplicity-increasing step, $w(\hat{S})$ increases by

$$\sum_{\text{object } i \text{ containing } p} m_i \cdot \frac{1}{w_i} \cdot w_i \;=\; \sum_{\text{object } i \text{ containing } p} m_i \;\le\; \frac{w(\hat{S})}{2t},$$

i.e., $w(\hat{S})$ increases by a factor of at most $1 + \frac{1}{2t}$. Initially, $w(\hat{S}) \le n^2$. Thus, after z multiplicity-increasing steps, $w(\hat{S}) \le n^2(1 + \frac{1}{2t})^z \le n^2 e^{z/(2t)}$.

On the other hand, consider the optimal set cover T^*. Suppose that object i has its multiplicity increased z_i times. In each multiplicity-increasing step, at least one object in T^* has its multiplicity increased. So, after z multiplicity-increasing steps, $\sum_{i \in T^*} z_i \ge z$ and $\sum_{i \in T^*} w_i \le t$. In particular, $z_i/w_i \ge z/t$ for some $i \in T^*$. Therefore, $w(\hat{S}) \ge (1 + \frac{1}{w_i})^{z_i} w_i \ge (1 + \frac{1}{w_i})^{z_i} \ge 2^{z_i/w_i} \ge 2^{z/t}$ (since $w_i \ge 1$). We conclude that $2^{z/t} \le w(\hat{S}) \le n^2 e^{z/(2t)}$, implying that $z = O(t \log n)$.

Similar to Agarwal and Pan's first MWU algorithm, we can also divide the multiplicity-increasing steps into rounds, with each round performing up to t multiplicity-increasing steps. Within each round, the total weight $w(\hat{S})$ increases by at most $(1 + \frac{1}{2t})^t = O(1)$. Also if $|\hat{S}|$ increases by a constant factor, we immediately start a new round: because $|\hat{S}| \le w(\hat{S})$ and $w(\hat{S})$ may be doubled at most $O(\log n)$ times, this case can happen at most $O(\log n)$ times. This ensures that if a point is checked to be ε-heavy at any moment during a round, it will remain $\Omega(\varepsilon)$-heavy at the end of the round. There are only $O(\log n)$ rounds.

Additional ideas are needed to speed up implementation (in particular, our modified MWU algorithm with multiplicity-readjustment steps does not work as well now). First, we work with an approximation $\tilde{m}_i$ to the multiplicity m_i of each object i. By rounding, we may assume all weights w_i are powers of 2. In the original algorithm, $m_i = (1 + \frac{1}{w_i})^{z_i}$, where z_i is the number of points $p \in Z$ that are contained in object i, and Z be the multiset consisting of all points p that have undergone multiplicity-increasing steps so far. Note that since the total multiplicity is $n^{O(1)}$, we have $z_i = O(w_i \log n)$. Let $Y^{(w_i)}$ be a random sample of Z where each point $p \in Z$ is included independently with probability $\frac{\log^2 n}{w_i}$ (if $w_i = O(\log^2 n)$, we can just set $Y^{(w_i)} = Z$). Let y_i be the number of points $p \in Y^{(w_i)}$ that are contained in object i. By the Chernoff bound, since $\frac{\log^2 n}{w_i} z_i = O(\log^3 n)$, we have $|y_i - \frac{\log^2 n}{w_i} z_i| \le O(\log^2 n)$ with high probability. By letting $\tilde{m}_i = (1 + \frac{1}{w_i})^{y_i w_i / \log^2 n}$, it follows that $\tilde{m}_i$ and m_i are within a factor of $O(1)$ of each other, with high probability, at all times, for all i. Thus, our earlier analysis still holds when working with $\tilde{m}_i$ instead of m_i. Since $z_i = O(w_i \log n)$, we have $y_i = O(\log^3 n)$ with high probability. So, the total number of increments to all y_i and updates to all $\tilde{m}_i$ is $O(n \log^3 n)$. In lines 5–6, we flip a biased coin to decide whether p should be placed in the sample $Y^{(2^j)}$ (with probability $\frac{\log^2 n}{2^j}$) for each j, and if so, we use halfspace range reporting in the dual to find all objects i of weight 2^j containing p, and increment y_i and update $\tilde{m}_i$. Over all $O(n \log n)$ executions of lines 5–6 and all $O(\log n)$ indices j, the cost of these halfspace range reporting queries is $O(n \log n \cdot \log n \cdot \log n)$ plus the output size. As the total output size for the queries is $O(n \log^3 n)$, the total cost is $O(n \log^3 n)$.

We also need to redesign a data structure for lightness testing subject to multiplicity updates: For each j, we maintain a subset $S^{(j)}$ containing all objects i with multiplicity at least 2^j, in a data structure to support approximate depth (without multiplicity). The depth of a point p in $\hat{S}$ can be $O(1)$-approximated by $\sum_j 2^j \cdot$ (depth of p in $S^{(j)}$). Each subset $S^{(j)}$ undergoes insertion only, and the logarithmic method can be applied to each $S^{(j)}$. Since $|\hat{S}| \le w(\hat{S}) \le n^{O(1)}$, there are $O(\log n)$ values of j. This slows down lightness testing by a logarithmic factor, and so in the case of 3D halfspaces, the overall time bound is $O(n \log^4 n \log\log n)$, excluding the ε-net construction time.

We can efficiently construct an ε-net of the desired weight for 3D halfspaces in $O(n \log n)$ randomized time, by using the quasi-random sampling technique of Varadarajan [31] and Chan et al. [13] in a more careful way. Due to lack of space, we defer the description to the full paper. We conclude:

▶ **Theorem 4.** *Given $O(n)$ points and $O(n)$ weighted halfspaces in $\mathbb{R}^3$, we can find a subset of halfspaces covering all points, of total weight within $O(1)$ factor of the minimum, in $O(n \log^4 n \log\log n)$ expected time by a randomized Las Vegas algorithm.*

Remark. A remaining open problem is to find efficient deterministic algorithms for the weighted problem. Chan et al. [13] noted that the quasi-uniform sampling technique can be derandomized via the method of conditional probabilities, but the running time is high.

References

1 Peyman Afshani and Timothy M. Chan. On approximate range counting and depth. *Discrete & Computational Geometry*, 42(1):3–21, 2009. doi:10.1007/s00454-009-9177-z.

2 Peyman Afshani and Timothy M. Chan. Optimal halfspace range reporting in three dimensions. In *Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 180–186, 2009.

3 Peyman Afshani, Chris Hamilton, and Norbert Zeh. A general approach for cache-oblivious range reporting and approximate range counting. *Computational Geometry*, 43(8):700–712, 2010.

4 Pankaj K. Agarwal and Jiangwei Pan. Near-linear algorithms for geometric hitting sets and set covers. In *Proceedings of the 30th Symposium on Computational Geometry (SoCG)*, page 271, 2014.

5 Boris Aronov, Esther Ezra, and Micha Sharir. Small-size ε-nets for axis-parallel rectangles and boxes. *SIAM Journal on Computing*, 39(7):3248–3282, 2010. doi:10.1137/090762968.

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 Hervé Brönnimann and Michael T. Goodrich. Almost optimal set covers in finite VC-dimension. *Discrete & Computational Geometry*, 14(4):463–479, 1995.

8 Norbert Bus, Shashwat Garg, Nabil H. Mustafa, and Saurabh Ray. Limits of local search: Quality and efficiency. *Discrete & Computational Geometry*, 57(3):607–624, 2017. doi:10.1007/s00454-016-9819-x.

9 Norbert Bus, Nabil H. Mustafa, and Saurabh Ray. Practical and efficient algorithms for the geometric hitting set problem. *Discrete Applied Mathematics*, 240:25–32, 2018.

10 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. doi:10.1137/S0097539798349188.

11 Timothy M. Chan. Polynomial-time approximation schemes for packing and piercing fat objects. *Journal of Algorithms*, 46(2):178–189, 2003. doi:10.1016/S0196-6774(02)00294-8.

12 Timothy M. Chan. Dynamic geometric data structures via shallow cuttings. In *Proceedings of the 35th Symposium on Computational Geometry (SoCG)*, pages 24:1–24:13, 2019.

13 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 23rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1576–1585, 2012.

14 Timothy M. Chan, Kasper Green Larsen, and Mihai Pătraşcu. Orthogonal range searching on the RAM, revisited. In *Proceedings of the 27th Symposium on Computational Geometry (SoCG)*, pages 1–10, 2011.

15 Timothy M. Chan and Konstantinos Tsakalidis. Optimal deterministic algorithms for 2-d and 3-d shallow cuttings. *Discrete & Computational Geometry*, 56(4):866–881, 2016.

16 Chandra Chekuri, Sariel Har-Peled, and Kent Quanrud. Fast LP solving and approximation algorithms for geometric packing and covering problems. In *Proceedings of the 31st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1019–1038, 2020.

17 Chandra Chekuri and Kent Quanrud. Randomized MWU for positive LPs. In *Proceedings of the 29th annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 358–377, 2018.

18 Kenneth L. Clarkson. Algorithms for polytope covering and approximation. In *Workshop on Algorithms and Data Structures*, pages 246–252, 1993.

19 Kenneth L. Clarkson and Kasturi Varadarajan. Improved approximation algorithms for geometric set cover. *Discrete & Computational Geometry*, 37(1):43–58, 2007.

20 Alon Efrat, Matthew J. Katz, Frank Nielsen, and Micha Sharir. Dynamic data structures for fat objects and their applications. *Computational Geometry*, 15(4):215–227, 2000. doi:10.1016/S0925-7721(99)00059-0.

21 Dorit S. Hochbaum and Wolfgang Maass. Approximation schemes for covering and packing problems in image processing and VLSI. *Journal of the ACM (JACM)*, 32(1):130–136, 1985. doi:10.1145/2455.214106.

22 Christos Koufogiannakis and Neal E. Young. A nearly linear-time PTAS for explicit fractional packing and covering linear programs. *Algorithmica*, 70(4):648–674, 2014. doi:10.1007/s00453-013-9771-6.

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

24 Jiří Matoušek, Raimund Seidel, and Emo Welzl. How to net a lot with little: Small ε-nets for disks and halfspaces. In *Proceedings of the 6th Symposium on Computational Geometry (SoCG)*, pages 16–22, 1990.

25 Nabil H. Mustafa. Computing optimal epsilon-nets is as easy as finding an unhit set. In *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 87:1–87:12, 2019. doi:10.4230/LIPIcs.ICALP.2019.87.

26 Nabil H. Mustafa, Rajiv Raman, and Saurabh Ray. Quasi-polynomial time approximation scheme for weighted geometric set cover on pseudodisks and halfspaces. *SIAM J. Comput.*, 44(6):1650–1669, 2015. doi:10.1137/14099317X.

27 Nabil H. Mustafa and Saurabh Ray. PTAS for geometric hitting set problems via local search. In *Proceedings of the 25th Symposium on Computational Geometry (SoCG)*, pages 17–22, 2009.

28 János Pach and Gábor Tardos. Tight lower bounds for the size of epsilon-nets. In *Proceedings of the 27th Symposium on Computational Geometry (SoCG)*, pages 458–463, 2011. doi:10.1145/1998196.1998271.

29 Evangelia Pyrga and Saurabh Ray. New existence proofs for ϵ-nets. In *Proceedings of the 24th Symposium on Computational Geometry (SoCG)*, pages 199–207, 2008. doi:10.1145/1377676.1377708.

30 Edgar A. Ramos. On range reporting, ray shooting and k-level construction. In *Proceedings of the 15th Symposium on Computational Geometry (SoCG)*, pages 390–399, 1999.

31 Kasturi Varadarajan. Weighted geometric set cover via quasi-uniform sampling. In *Proceedings of the 42nd ACM Symposium on Theory of Computing (STOC)*, pages 641–648, 2010.

32 Kasturi R. Varadarajan. Epsilon nets and union complexity. In *Proceedings of the 25th Symposium on Computational Geometry (SoCG)*, pages 11–16, 2009. doi:10.1145/1542362.1542366.

33 Neal E. Young. Sequential and parallel algorithms for mixed packing and covering. In *Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 538–546, 2001. doi:10.1109/SFCS.2001.959930.

Further Results on Colored Range Searching

Timothy M. Chan
Department of Computer Science, University of Illinois at Urbana-Champaign, IL, USA
tmc@illinois.edu

Qizheng He
Department of Computer Science, University of Illinois at Urbana-Champaign, IL, USA
qizheng6@illinois.edu

Yakov Nekrich
Department of Computer Science, Michigan Technological University, Houghton, MI, USA
yakov.nekrich@googlemail.com

Abstract

We present a number of new results about range searching for colored (or "categorical") data:

1. For a set of n colored points in three dimensions, we describe randomized data structures with $O(n \operatorname{polylog} n)$ space that can report the distinct colors in any query orthogonal range (axis-aligned box) in $O(k \operatorname{polyloglog} n)$ expected time, where k is the number of distinct colors in the range, assuming that coordinates are in $\{1, \ldots, n\}$. Previous data structures require $O(\frac{\log n}{\log \log n} + k)$ query time. Our result also implies improvements in higher constant dimensions.
2. Our data structures can be adapted to halfspace ranges in three dimensions (or circular ranges in two dimensions), achieving $O(k \log n)$ expected query time. Previous data structures require $O(k \log^2 n)$ query time.
3. For a set of n colored points in two dimensions, we describe a data structure with $O(n \operatorname{polylog} n)$ space that can answer colored "type-2" range counting queries: report the number of occurrences of every distinct color in a query orthogonal range. The query time is $O(\frac{\log n}{\log \log n} + k \log \log n)$, where k is the number of distinct colors in the range. Naively performing k uncolored range counting queries would require $O(k \frac{\log n}{\log \log n})$ time.

Our data structures are designed using a variety of techniques, including colored variants of randomized incremental construction (which may be of independent interest), colored variants of shallow cuttings, and bit-packing tricks.

2012 ACM Subject Classification Theory of computation → Computational geometry; Theory of computation → Data structures design and analysis

Keywords and phrases Range searching, geometric data structures, randomized incremental construction, random sampling, word RAM

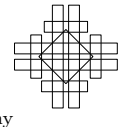
Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.28

Related Version A full version of this paper is available at `http://arxiv.org/abs/2003.11604`.

Funding *Timothy M. Chan*: Supported in part by NSF Grant CCF-1814026.

1 Introduction

Colored range searching (also known as "categorical range searching", or "generalized range searching") have been extensively studied in computational geometry since the 1990s. For example, see the papers [5, 11, 18, 19, 20, 21, 23, 24, 25, 26, 28, 29, 30, 31, 32, 34, 36, 37, 38, 40, 46] and the survey by Gupta et al. [22]. Given a set of n colored data points (where the color of a point represents its "category"), the objective is to build data structures that can provide statistics or some kind of summary about the colors of the points inside a query range. The most basic types of queries include:



- *colored range reporting*: report all the distinct colors in the query range.
- *colored "type-1" range counting*: find the number of distinct colors in the query range.
- *colored "type-2" range counting*: report the number of points of color χ in the query range, for *every* color χ in the range.

In this paper, we focus on colored range reporting and type-2 colored range counting. Note that the output size in both instances is equal to the number k of distinct colors in the range, and we aim for query time bounds that depend linearly on k, of the form $O(f(n) + kg(n))$. Naively using an uncolored range reporting data structure and looping through all points in the range would be too costly, since the number of points in the range can be significantly larger than k.

1.1 Colored orthogonal range reporting

The most basic version of the problem is perhaps *colored orthogonal range reporting*: report the k distinct colors inside an orthogonal range (an axis-aligned box). It is not difficult to obtain an $O(n \operatorname{polylog} n)$-space data structure with $O(k \operatorname{polylog} n)$ query time [22] for any constant dimension d: one approach is to directly modify the d-dimensional range tree [15, 43], and another approach is to reduce colored range reporting to uncolored range emptiness [26] (by building a one-dimensional range tree over the colors and storing a range emptiness structure at each node). Both approaches require $O(k \operatorname{polylog} n)$ query time rather than $O(\operatorname{polylog} n + k)$ as in traditional (uncolored) orthogonal range searching: the reason is that in the first approach, each color may be discovered polylogarithmically many times, whereas in the second approach, each discovered color costs us $O(\log n)$ range emptiness queries, each of which requires polylogarithmic time.

Even the 2D case remains open, if one is interested in optimizing logarithmic factors. For example, Larsen and van Walderveen [32] and Nekrich [37] independently presented data structures with $O(n \log n)$ space and $O(\log \log U + k)$ query time in the standard word-RAM model, assuming that coordinates are integers bounded by U. The query bound is optimal, but the space bound is not. Recently, Chan and Nekrich [11] have improved the space bound to $O(n \log^{3/4+\varepsilon} n)$ for an arbitrarily small constant $\varepsilon > 0$, while keeping $O(\log \log U + k)$ query time.

In 3D, the best result to date is by Chan and Nekrich [11], who obtained a data structure with $O(n \log^{9/5+\varepsilon} n)$ space and $O(\frac{\log n}{\log \log n} + k)$ query time. The first step is a data structure for the case of 3D dominance (i.e., 3-sided) ranges: as they noted, this case can be solved in $O(n)$ space and $O(\frac{\log n}{\log \log n} + k)$ time by a known reduction [44, Section 3.1] to 3D 5-sided box stabbing [12]. For 3D 5-sided box stabbing (or more simply, 2D 4-sided rectangle stabbing), a matching lower bound of $\Omega(\frac{\log n}{\log \log n} + k)$ is known for $O(n \operatorname{polylog} n)$-space structures, due to Pătraşcu [42]. A natural question then arises: is $O(\frac{\log n}{\log \log n} + k)$ query time also tight for 3D colored dominance range reporting?

We show that the answer is no – the $O(\frac{\log n}{\log \log n})$ term can in fact be improved when k is small. Specifically, we present a randomized data structure for 3D colored dominance range reporting with $O(n \log n)$ space and $O(\log \log U + k \log \log n)$ expected time in the standard word-RAM model. (We use only Las Vegas randomization, i.e., the query algorithm is always correct; an oblivious adversary is assumed, i.e., the query range should be independent of the random choices made by the preprocessing algorithm.) Combining with Chan and Nekrich's method [11], we can then obtain a data structure for 3D colored orthogonal range reporting with $O(n \log^{2+\varepsilon} n)$ space and $O(\log \log U + k \log \log n)$ expected query time.

An improved solution in 3D automatically implies improvements in any constant dimension $d > 3$, by using standard range trees [15, 43] to reduce the dimension, at a cost of about one logarithmic factor (ignoring $\log\log$ factors) per dimension. This way, we obtain a data structure in d dimensions with $O(n \log^{d-1+\varepsilon} n)$ space and $O(k(\frac{\log n}{\log\log n})^{d-3} \log\log n)$ query time.[1] (Note that $O((\frac{\log n}{\log\log n})^{d-3} \log\log n)$ is the current best query time bound for standard (uncolored) range emptiness [9] for $O(n \operatorname{polylog} n)$-space structures on the word RAM.)

1.2 Colored 3D halfspace range reporting

An equally fundamental problem is *colored halfspace range reporting*. In 2D, an $O(n)$-space data structure with $O(\log n + k)$ query time is known [2, 22]. In 3D, the current best result is obtained by applying a general reduction of colored range reporting to uncolored range emptiness [26], which yields $O(n \log n)$ space and $O(k \log^2 n)$ query time [22]. (An alternative solution with $O(n)$ space and $O(n^{2/3+\varepsilon} + k)$ time is also known, by reduction to simplex range searching.) The 3D case is especially important, as 2D colored circular range reporting reduces to 3D colored halfspace range reporting by the standard lifting transformation.

We describe a randomized data structure with $O(n \log n)$ space and $O(k \log n)$ expected query time for 3D halfspace ranges (and thus 2D circular ranges). This is a logarithmic-factor improvement over the previous query time bound.

1.3 Colored 2D orthogonal type-2 range counting

Finally, we consider colored orthogonal "type-2" range counting: compute the number of occurrences of every color in a given orthogonal range. Despite the nondescript name, colored type-2 counting is quite natural, providing more information than colored reporting, as we are generating an entire histogram. The problem was introduced by Gupta et al. [23] (and more recently revisited by Ganguly et al. [20] in external memory). An old paper by Bozanis et al. [5] gave a solution in the 1D case with $O(n)$ space and $O(\log n + k)$ query time, which implies a solution in 2D with $O(n \log n)$ space and $O(\log^2 n + k \log n)$ query time. Alternatively, to answer a colored type-2 counting query, we can first answer a colored range reporting query, followed by k standard (uncolored) range counting queries, if we store each color class in a standard range counting data structure; by known results on colored range reporting [11] and standard range counting [27], this then yields $O(n \log^{3/4+\varepsilon} n)$ space and $O(k \frac{\log n}{\log\log n})$ query time. Thus, in some sense, a type-2 counting query corresponds to "simultaneous" range counting queries on multiple point sets.[2]

We present a data structure for the problem in 2D with $O(n \log^{1+\varepsilon} n)$ space and $O(\frac{\log n}{\log\log n} + k \log\log n)$ query time in the standard word-RAM model. As 2D standard (uncolored) range counting has an $\Omega(\frac{\log n}{\log\log n})$ time lower bound for $O(n \operatorname{polylog} n)$-space structures [41], our result shows, surprisingly, that answering multiple range counting queries "simultaneously" are cheaper than answering one by one – we only have to pay $O(\log\log n)$ cost per color!

1.4 Techniques

Our solutions for colored 3D dominance range reporting and 3D halfspace range reporting are based on similar ideas. We in fact propose two different methods.

---

[1] In all reported bounds, we implicitly assume $k > 0$. The $k = 0$ case can be handled by answering one initial uncolored range emptiness query.

[2] See [1] for a different notion of "concurrent" range reporting.

In the first method (Section 2), we solve the $k = 1$ case (testing whether a range contains only one color) by introducing a colored variant of *randomized incremental construction*; we then extend the solution to the general case by a randomized one-dimensional range tree over the colors. Along the way, we prove a combinatorial lemma which may be of independent interest: in a colored point set in 3D, if we randomly permute the color classes and randomly permute the points within each color classes, and if we insert the points in the resulting order, then the convex hull undergoes $O(n \log n)$ structural changes in expectation. (It is a well known fact, by Clarkson and Shor [14], that in the uncolored setting, if we insert points in a random order, the 3D convex hull undergoes $O(n)$ structural changes in expectation.)

In the second method (Section 3), which is slightly more efficient, we solve the $k = 1$ case differently, by adapting known techniques for uncolored 3D halfspace range reporting [6] based on *random sampling* (namely, conflict lists of lower envelopes of random subsets). The approach guarantees only $\Omega(1)$ success probability per query (in the uncolored setting, *shallow cuttings* can fix the problem, but they do not seem easily generalizable to the 3D colored setting). Fortunately, we show that a solution for the $k = 1$ case with constant success probability is sufficient to complete the solution for the general case.

Our method for colored 2D type-2 orthogonal range counting (Section 5) is technically the most involved. It is obtained by a nontrivial combination of several techniques, including the *recursive grid* approach of Alstrup, Brodal, and Rauhe [3], bit packing tricks, and 2D shallow cuttings. Our work demonstrates yet again the power of the recursive grid approach (see [9, 11, 12] for other recent examples).

2 Colored 3D Halfspace Range Reporting: First Method

In this and the next section, we describe our two methods for 3D halfspace ranges. The case of 3D dominance ranges is similar and will be addressed later in Section 4.

2.1 Combinatorial lemmas on colored randomized incremental construction

Our first method relies on a simple combinatorial lemma related to a colored version of randomized incremental construction of 3D convex hulls (the uncolored version of the lemma, where all points are assigned different colors, is well known in computational geometry, from the seminal work by Clarkson and Shor [14]):

▶ **Lemma 1.** *Given a set S of n colored points in $\mathbb{R}^3$, if we first randomly permute the color classes, then for each color according to this order we simultaneously insert all points with that color, then the expected total number of structural changes to the convex hull is $O(n)$.*

Proof. Consider a random permutation of the colors. Let C_i be the i-th color class, i.e., the set of all points with the i-th color in the permutation. Let m be the number of color classes. Let $V_i = \bigcup_{j=1}^{i} C_j$ contain all points with the first i colors. Let $\mathrm{CH}(V_i)$ denote the convex hull of V_i. Let Δ_i^+ be the set of all facets in $\mathrm{CH}(V_i)$ that are not in $\mathrm{CH}(V_{i-1})$, i.e., all hull facets created when we insert the i-th color class C_i.

For each i, we have $\mathbb{E}[|C_i|] = \frac{n}{m}$ and $\mathbb{E}[|V_i|] = \frac{in}{m}$. We use backwards analysis [45]. Observe that $|\Delta_i^+|$ is bounded by the total degree of all points of C_i in $\mathrm{CH}(V_i)$. The total degree over all points in $\mathrm{CH}(V_i)$ is $O(|V_i|)$. Conditioned on a fixed V_i, we have $\mathbb{E}[|\Delta_i^+|] = O(\frac{|V_i|}{i})$. So, unconditionally, $\mathbb{E}[|\Delta_i^+|] = O(\frac{n}{m})$. Therefore, the expected total number of hull facets created is $\mathbb{E}[\sum_{i=1}^{m} |\Delta_i^+|] = O(n)$. ◀

The following refinement of the lemma further bounds the total amount of changes to the convex hull when we additionally insert points one by one in a random order within each color class. The proof is slightly trickier. (The first lemma is already sufficient to bound the space of our new data structure, but the refined lemma will be useful in bounding the preprocessing time.)

▶ **Lemma 2.** *Given a set S of n colored points in $\mathbb{R}^3$, if we first randomly permute the color classes, then randomly permute the points in each color class, and insert the points one by one according to this order, then the expected total number of structural changes of the convex hull is $O(n \log n)$.*

Proof. Continuing the earlier proof, let Δ_i^- be the set of all facets in $\mathrm{CH}(V_{i-1})$ that are not in $\mathrm{CH}(V_i)$, i.e., all hull facets destroyed when we insert the i-th color class C_i. Since the total number of facets destroyed is at most the total number of facets created, $\mathbb{E}[\sum_{i=1}^m |\Delta_i^-|] \le \mathbb{E}[\sum_{i=1}^m |\Delta_i^+|] = O(n)$.

Now, consider a random permutation of the points in C_i. Let $V_{i,j}$ contain all points in V_{i-1} and also the first j points of C_i. Let $G_{i,j}$ be the subgraph formed by all edges of $\mathrm{CH}(V_{i,j})$ that are incident to the vertices of C_i. Then every vertex v in $G_{i,j}$ is either in C_i or is incident to a facet of Δ_i^- (because if $v \notin C_i$, then v must be a vertex of $\mathrm{CH}(V_{i-1})$, and at least one of its incident facets in $\mathrm{CH}(V_{i-1})$ will be destroyed when C_i is inserted). Thus, $G_{i,j}$ has $O(|C_i| + |\Delta_i^-|)$ vertices, and since $G_{i,j}$ is a planar graph, it has $O(|C_i| + |\Delta_i^-|)$ edges.

Let $\Delta_{i,j}^+$ be the set of all facets in $\mathrm{CH}(V_{i,j})$ that are not in $\mathrm{CH}(V_{i,j-1})$, i.e., all hull facets created when we insert the j-th point in C_i. We use backwards analysis again. Observe that $|\Delta_{i,j}^+|$ is bounded by the degree of the j-th point in C_i in $\mathrm{CH}(V_{i,j})$. The total degree over all points of C_i in $\mathrm{CH}(V_{i,j})$ is at most twice the number of edges in $G_{i,j}$. Conditioned on a fixed C_i and a fixed $V_{i,j}$, we thus have $\mathbb{E}[|\Delta_{i,j}^+|] = O\left(\frac{|C_i|+|\Delta_i^-|}{j}\right)$. As the right-hand side does not depend on the local permutation of the color class C_i, the expectation holds conditioned only on the global permutation of the colors. Unconditionally, the expected total number of hull facets created is

$$O\left(\mathbb{E}\left[\sum_{i=1}^{m}\sum_{j=1}^{|C_i|}\frac{|C_i|+|\Delta_i^-|}{j}\right]\right) = O\left(\mathbb{E}\left[\sum_{i=1}^{m}(|C_i|+|\Delta_i^-|)\log n\right]\right) = O(n\log n). \qquad \blacktriangleleft$$

Remarks.

1. The $O(n \log n)$ bound in the refined lemma is tight: Consider $\frac{n}{2}$ points lying on the xy-plane in convex position, each assigned a different color. In addition, add $\frac{n}{2}$ points on the z-axis above the xy-plane, all with a common color χ_0. When we insert the color class for χ_0, there are already $\Omega(n)$ points on the xy-plane with probability $\Omega(1)$. In an iteration where the next point we insert with color χ_0 has larger z-coordinate than all previous points, the insertion would create $\Omega(n)$ new hull edges in expectation. By a well known analysis, the expected number of such iterations is given by the Harmonic number, which is $\Theta(\log n)$. This shows an $\Omega(n \log n)$ lower bound.
2. The same argument holds for other geometric structures besides 3D convex hulls, e.g., Voronoi diagrams of 2D points and trapezoidal decompositions of 2D disjoint line segments.
3. We can generalize the refined lemma to the setting when we have a hierarchy of color classes with ℓ levels, and we randomly permute the child subclasses of each color class. (The refined lemma corresponds to the $\ell = 2$ case.) The bound becomes $O(n \log^{\ell-1} n)$. This result seems potentially relevant to implementing randomized incremental constructions in a hierarchical external-memory model.

2.2 The $k = 1$ case

We now reveal how colored randomized incremental construction can help solve the colored range reporting problem. We start with the case $k = 1$, i.e., we want to test whether there is only one color in the query range. By an uncolored range search, we can find one point in the range (in $O(\log n)$ time for 3D halfspace ranges) and identify its color χ. Thus, the problem is to verify that all points in the range have the same color χ.

Fix a total ordering of the colors. It is easy to see that the problem reduces to two subproblems: for a given query color χ, (i) decide whether there exists a point in the range with color $< \chi$, and (ii) decide whether there exists a point in the range with color $> \chi$. By symmetry, it suffices to solve subproblem (i). To this end, we imagine inserting the points in increasing order of color, and maintaining a data structure for (uncolored) range emptiness for the points. We can make this semi-dynamic data structure (which supports insertions only) *persistent*. Then we can solve subproblem (i) by querying a past version of the range emptiness data structure, right after all points with color $< \chi$ were inserted.

In the case of 3D upper halfspaces (lower halfspaces can be handled symmetrically), a range emptiness query reduces to finding an extreme point on the upper hull along a query direction, or equivalently, intersecting the lower envelope of the dual planes at a query vertical line. By projection, this reduces to a planar point location query, answerable in $O(\log n)$ time by a linear-space data structure [15, 43]. However, we need a data structure that supports insertions, and in general this increases the query time (by an extra logarithmic factor via the standard "logarithmic method" [4]).

The key is to observe that the above approach works regardless of which total ordering of the colors we use. Our idea is simply to use a *random ordering* of the colors! (For (ii), note that the reverse of a uniformly random ordering is still uniformly random.) By Lemma 1, the upper hull undergoes $O(n)$ expected number of structural changes. So is the dual lower envelope. We can then apply a known dynamic planar point location method; for example, the method by Chan and Nekrich [10] achieves $O(\log n(\log\log n)^2)$ query time and $O(\log n \log\log n)$ amortized update time per change to the envelope. The data structure can be made persistent, for example, by applying Dietz's technique [17], with a $\log\log n$ factor penalty (the space usage is related to the total update time). The final data structure supports queries in $O(\log n(\log\log n)^3)$ (worst-case) time and uses $O(n \log n(\log\log n)^2)$ expected space. (Note that the space bound can be made worst-case, by repeating $O(1)$ expected number of times until a "good" ordering is found.)

Remark on preprocessing time. It isn't obvious how to efficiently insert an entire color class to the 3D convex hull, even knowing that the total number of structural changes is small. To get good preprocessing time, we propose inserting points one by one within each color class, since Lemma 2 ensures that the number of changes to the convex hull is still near linear ($O(n \log n)$). Several implementation options can then yield $O(n \operatorname{polylog} n)$ expected preprocessing time: (i) we can use a general-purpose dynamic convex hull data structure [7] (in the insertion-only case, the cost per update is $O(f \log^2 n)$ where f is the amount of structural changes); (ii) we can adapt standard randomized incremental algorithms, e.g., handling the point location steps by using history DAGs [35] (this requires further randomized analysis); or (iii) we can adapt standard randomized incremental algorithms, but handling the point location steps by using a known dynamic planar point location method [10].

▶ **Theorem 3.** *For n colored points in $\mathbb{R}^3$, there is a data structure with $O(n \operatorname{polylog} n)$ expected preprocessing time and $O(n \log n(\log\log n)^2)$ space that can test whether the number of colors in a query halfspace is exactly 1 in $O(\log n(\log\log n)^3)$ time.*

2.3 The general case

Previous papers [22, 26] (see also [13] in the uncolored case) have noted a straightforward black-box reduction of colored range reporting to the $k = 0$ case (range emptiness), essentially by using a one-dimensional range tree over the colors: More precisely, we split the color classes into two halves. We build a data structure for $k = 0$, and recursively build a data structure for the two halves. Space usage increases by a logarithmic factor. If the $k = 0$ structure has $Q_0(n)$ query time, the overall query time is $O(kQ_0(n) \log n)$, since at each of the $O(\log n)$ levels of recursion tree, $O(k)$ nodes are examined.

We present a new black-box reduction of colored range reporting to the $k \leq 1$ case, which saves a logarithmic factor, by using a similar idea but with randomization.

▶ **Theorem 4.** *Suppose that for n colored points, there is a data structure with $P(n)$ (expected) preprocessing time and $S(n)$ space that can decide whether the number of colors in a query range is exactly 1 in $Q_1(n)$ time. In addition, the data structure can decide whether the range is empty, and if not, report one point, in $Q_0(n)$ time. Then there is a randomized Las Vegas data structure with $O(P(n) \log n)$ expected preprocessing time and $O(S(n) \log n)$ space that can report all k distinct colors in a query range in $O(k(Q_0(n) + Q_1(n)))$ expected time, assuming that $P(n)/n$ and $S(n)/n$ are nondecreasing.*

Proof. We split the color classes into two parts, where each color is *randomly* assigned to one of the two parts. We build the given $k = 1$ structure and range emptiness structure, and recursively build a data structure for the two parts. Space usage increases by a logarithmic factor (with high probability).

To answer a query, we test whether the range is empty or whether $k = 1$. If so, we are done. Otherwise, we recursively query both parts.

Consider a query range that is independent of the random choices made by the data structure. At the i-th level of the recursion tree, how many nodes are examined (in expectation)? This question is analogous to the following: place k balls randomly (independently) into 2^i bins; how many bins contain two or more balls? The number is upper-bounded by the number of pairs of balls that are in the same bin. Since the probability that a fixed pair of balls are placed in the same bin is $1/2^i$, the expected number of pairs is at most $k^2/2^i$.

Thus, the expected number of nodes examined at the i-th level is at most $\min\{2^i, k^2/2^i\}$. The overall expected number of nodes examined is

$$O\left(\sum_i \min\{2^i, k^2/2^i\}\right) = O\left(\sum_{i:\, 2^i \leq k} 2^i + \sum_{i:\, 2^i > k} k^2/2^i\right) = O(k). \qquad \blacktriangleleft$$

Combining Theorems 3 and 4 yields:

▶ **Theorem 5.** *For n colored points in $\mathbb{R}^3$, there is a randomized Las Vegas data structure with $O(n \operatorname{polylog} n)$ expected preprocessing time and $O(n \log^2 n (\log\log n)^2)$ space that can report all k distinct colors in a query halfspace in $O(k \log n (\log\log n)^3)$ expected time.*

3 Colored 3D Halfspace Range Reporting: Second Method

We next describe a slightly better (and simpler) method for colored 3D halfspace range reporting.

3.1 The $k = 1$ case

The idea is to relax the $k = 1$ subproblem and allow the query algorithm to occasionally be wrong (since we will be using randomization anyways for the general case). The algorithm has constant error probability and can only make one-sided errors: if it returns "yes", we must have $k = 1$. We work in dual space: given a set of colored planes in $\mathbb{R}^3$, we want to decide whether the number of colors among the planes below a query point is exactly 1.

Preprocessing. Take a random sample R of the planes, where each color class is included independently with probability $\frac{1}{2}$. Take the lower envelope $\mathrm{LE}(R)$ of R, and consider the vertical decomposition $\mathrm{VD}(R)$ of the region underneath $\mathrm{LE}(R)$. (The vertical decomposition is defined as follows: we triangulate each face of $\mathrm{LE}(R)$ by joining each vertex to the bottom vertex of the face; for each triangle, we form the unbounded prism containing all points underneath the triangle.) For each cell $\Delta \in \mathrm{VD}(R)$, let L_Δ denote the set of distinct colors among all planes intersecting Δ (the "color conflict list" of Δ). We store the list L_Δ if $|L_\Delta| \leq c$ for a sufficiently large constant c; otherwise, we mark Δ as "bad".

Clearly, the space usage is $O(n)$, since there are $O(n)$ cells in $\mathrm{VD}(R)$ and each list stored has constant size. To bound the preprocessing time, we can generate (up to c elements of) each list L_Δ by answering colored range reporting queries at the three vertices of Δ, since a plane intersects Δ iff it is below at least one of the vertices of Δ. By previous results, these $O(n)$ colored range reporting queries take $O(n \operatorname{polylog} n)$ time.

In addition, for each color class, we store an (uncolored) range emptiness structure (i.e., a planar point location structure for the xy-projection of the lower envelope of the color class). This takes $O(n)$ space in total.

Querying. Given a query point q, we find the cell $\Delta(q)$ of $\mathrm{VD}(R)$ containing q in $O(\log n)$ time by planar point location (on the xy-projection of $\mathrm{VD}(R)$). If the cell does not exist (i.e., q lies above $\mathrm{LE}(R)$), or if the cell is bad, we return "no". Otherwise, for each of the at most c colors in the conflict list $L_{\Delta(q)}$, we test whether any plane below q has that color by querying the corresponding range emptiness structure in $O(\log n)$ time. We return "yes" iff exactly one color passes the test. The overall query time is $O(\log n)$.

The algorithm is clearly correct if it returns "yes". Consider a fixed query point q, such that there is just one color χ among all planes below q. The algorithm would erroneously return "no" in two scenarios: (i) when q lies above $\mathrm{LE}(R)$, or (ii) when $|L_{\Delta(q)}| > c$. The probability of (i) is the probability that the color χ is chosen in the random sample R, which is $\frac{1}{2}$. By the following lemma, and Markov's inequality, the probability of (ii) is at most 0.1 (say) for a sufficiently large constant c. This lemma directly follows from Clarkson and Shor's technique [14] (see the full paper for a quick proof).

▶ **Lemma 6.** *For a fixed point q, we have $\mathbb{E}[|L_{\Delta(q)}|] = O(1)$.*

We conclude:

▶ **Theorem 7.** *For n colored points in $\mathbb{R}^3$, there is a randomized Monte Carlo data structure with $O(n \operatorname{polylog} n)$ preprocessing time and $O(n)$ space that decides whether the number of colors in a query halfspace is exactly 1 in $O(\log n)$ time; if the actual answer is true, the algorithm returns "yes" with probability $\Omega(1)$, else it always returns "no".*

Remarks. The method can be viewed as a variant of Chan's random-sampling-based method for uncolored 3D halfspace range reporting [6]. In the uncolored setting, errors can be completely avoided by replacing lower envelopes of samples with *shallow cuttings* [33], but it is unclear how to do so in the colored setting.

3.2 The general case

Finally, to solve the general problem, we use a variant of Theorem 4 that tolerates one-sided errors in the given $k = 1$ data structure.

▶ **Theorem 8.** *Suppose that for n colored points, there is a randomized Monte Carlo data structure with $P(n)$ (expected) preprocessing time and $S(n)$ space that decides whether the number of colors in a query range is exactly 1 in $Q_1(n)$ time; if the actual answer is true, the algorithm returns "yes" with probability $\Omega(1)$, else it always returns "no". In addition, the data structure can decide whether the range is empty, and if not, report one point, in $Q_0(n)$ time (without errors). Then there is a randomized Las Vegas data structure with $O(P(n)\log n)$ expected preprocessing time and $O(S(n)\log n)$ space that can report all k distinct colors in a query range in $O(k(Q_0(n) + Q_1(n)))$ expected time, assuming that $P(n)/n$ and $S(n)/n$ are nondecreasing.*

Proof. We use the same approach as in the proof of Theorem 4. In the query algorithm, if the range is empty or the $k = 1$ structure returns "yes", we are done; otherwise, we recursively query both parts.

To analyze the query time, we say that a node in the recursion tree is *bad* if the number of colors in the query range at the node is exactly 1. Our earlier analysis shows that the expected total number of non-bad nodes visited is $O(k)$. However, because of the possibility of one-sided errors, the query algorithm may examine some bad nodes. For each bad node v visited by the query algorithm, we charge v to its lowest ancestor u that is not bad. Then for a fixed node u, we may have up to two paths of nodes charged to u. The expected number of nodes charged to a fixed node u is at most $O(\sum_i (1 - \Omega(1))^i) = O(1)$. We conclude that the expected total number of nodes visited is $O(k)$. ◀

Combining Theorems 7 and 8 yields:

▶ **Theorem 9.** *For n colored points in $\mathbb{R}^3$, there is a randomized Las Vegas data structure with $O(n \operatorname{polylog} n)$ expected preprocessing time and $O(n \log n)$ space that can report all k distinct colors in a query halfspace in $O(k \log n)$ expected time.*

4 Colored 3D Dominance Range Reporting

Both methods can be adapted to solve the colored 3D dominance range reporting problem: here, we want to report the distinct colors of all points inside a 3-sided range of the form $(-\infty, q_1] \times (-\infty, q_2] \times (-\infty, q_3]$. Equivalently, we can map input points (p_1, p_2, p_3) to orthants $[p_1, \infty) \times [p_2, \infty) \times [p_3, \infty)$, and the problem becomes reporting the distinct colors among all orthants containing a query point $q = (q_1, q_2, q_3)$. By replacing values with their ranks, we may assume that all coordinates are in $\{1, \ldots, n\}$ (in a query, an initial predecessor search to reduce to rank space requires an additional $O(\log \log U)$ cost by van Emde Boas trees). We assume the standard word-RAM model.

In the first method, the combinatorial lemmas on colored randomized incremental constructions can be extended to the union of the orthants (a "staircase polyhedron"). In fact, by a known transformation involving an exponentially spaced grid [9, 39], orthants can be

mapped to halfspaces and a union of orthants can be mapped to a halfspace intersection, or in the dual, a 3D convex hull. For the $k = 1$ structure, we not only randomly permute the color classes but also randomly permute the points inside each color class, and maintain the union of the orthants as points are inserted one by one. Instead of using persistence, we reduce to static 3D point location: we insert in reverse order, and as a new orthant is inserted, we create a region for the newly added portion of the union (i.e., the new orthant minus the old union). Identifying the smallest color of the orthants containing q (to solve subproblem (i)) reduces to locating the region containing q. The expected total size of these regions is $O(n \log n)$ by Lemma 2; we can further subdivide each of these regions into boxes (by taking a vertical decomposition), without asymptotically increasing the total size. Known results on orthogonal point location in a 3D subdivision of (space-filling) boxes [16, 12] then give $O((\log \log n)^2)$ query time and space linear in the size of the subdivision. Thus, the final data structure for the general case has $O(n \log^2 n)$ space and $O(\log \log U + k(\log \log n)^2)$ expected query time.

In the second method, we replace lower envelopes with unions of orthants. The only main change is that planar point location queries for orthogonal subdivisions now cost $O(\log \log n)$ time by Chan's result [8] instead of $O(\log n)$. Thus, the final data structure has $O(n \log n)$ space and $O(\log \log U + k \log \log n)$ expected time.

▶ **Theorem 10.** *For n colored points in $\mathbb{R}^3$, there is a randomized Las Vegas data structure with $O(n \operatorname{polylog} n)$ expected preprocessing time and $O(n \log n)$ space that can report all k distinct colors in a query dominance range in $O(\log \log U + k \log \log n)$ expected time.*

We can extend the result of Theorem 10 to orthogonal ranges with more sides or to $d > 3$ dimensions. See the full paper for more details.

5 Colored 2D Orthogonal Type-2 Range Counting

Our solution for orthogonal type-2 range counting is described in stages. First we consider the *capped* variant of type-2 range counting. A capped query returns the correct answer if the number of colors k in the query range does not exceed $\log^3 n$. If $k > \log^3 n$, the answer to the capped query is *NULL*. Capped queries in the case when the query range is bounded on 2 sides are considered in Section 5.1. We extend the solution to 3-sided and 4-sided queries, as well as for the case when the number of colors can be arbitrarily large, in the full paper.

5.1 Capped 2-Sided Queries

With foresight, we will solve the more general weighted version of this problem. Each point in S is also assigned a positive integer *weight*. For a 2-sided query range Q, we want to identify all colors that occur in Q; for each color we report the total weight of all its occurrences in Q.

We will denote by n the total weight of all points in S; we will denote by m the total number of points in S. We prove the following result:

▶ **Lemma 11.** *Let S be the set of m points in $\mathbb{R}^2$ with total weight $n \geq m$. There exists a data structure that uses $O(m(\log \log n)^2)$ words of space and supports 2-sided capped type-2 counting queries in $O(\log n / \log \log n + k \log \log n)$ time.*

Our data structure is based on the recursive grid approach [3]. The set of points is recursively sub-divided into vertical slabs (or columns) and horizontal slabs (or rows).

Data Structure. Let $\tau = \log^3 n_0$ where n_0 is the total weight of all points in the global data set (thus τ remains unchanged on all recursion levels). We divide the set of points into $\sqrt{n/\tau}$ columns so that either the total weight of all points in a column is bounded by $O(\sqrt{n\tau})$ or a column contains only one point. This division can be obtained by scanning the set of points in the left-to-right order. We add points to a column C_i for $i = 1, 2, \ldots$ by repeating the following steps: (1) if the weight of the next point p exceeds $\sqrt{n\tau}$, we increment i, (2) we add p to C_i, and (3) if the total weight of C_i exceeds $\sqrt{n\tau}$, we increment i.

Thus either the total weight of a column exceeds $\sqrt{n\tau}$ or the next column contains a point of weight at least $\sqrt{n\tau}$. Hence the number of columns is $O(\sqrt{n/\tau})$. We also divide the set of points into rows satisfying the same conditions. Let $p_{ij} = (x_i, y_j)$ denote the point where the upper boundary of the j-th row intersects the right boundary of the i-th column. Let $Dom(i, j) = [0, x_i] \times [0, y_j]$, denote the range dominated by p_{ij}. If $Dom(i, j)$ contains at most τ distinct colors, we store the list L_{ij} of colors that occur in $Dom(i, j)$. For every color in L_{ij} we also keep the number of its occurrences in $Dom(i, j)$. If the range $Dom(i, j)$ contains more than τ different colors, we set $L_{ij} = NULL$. Thus L_{ij} provides the answer to a capped type-2 counting query on $[0, x_i] \times [0, y_j]$.

Every row/column of weight at least τ^2 that contains more than one point is recursively divided in the same way as explained above. If the total weight of all points is smaller than τ^2, we can answer a type-2 range counting query in $O(k)$ time.

Slow Queries. A query $[0, a] \times [0, b]$ is answered as follows. We identify the column C_{i+1} containing a and the row R_{j+1} containing b. The query is then divided into the middle part $[0, x_i] \times [0, y_j]$, the upper part $[0, a] \times [y_j, b]$ and the right part $[x_i, a] \times [0, y_j]$. The answer to the middle query is stored in the pre-computed list L_{ij}. The upper query is contained in the row R_{j+1} and the right query is contained in the column C_{i+1}. Hence we can answer the upper and the right query using data structures on R_{j+1} and C_{i+1} respectively. If $L_{ij} = NULL$, we return $NULL$ because the number of colors in the query range exceeds $\log^2 n$; if the answer to a query on C_{i+1} or R_{j+1} is $NULL$, we also return $NULL$. Otherwise, we merge the answers to the three queries. The resulting list L can contain up to three items of the same color because the same color can occur in the left, right, and middle query. Since the items in L are sorted by color, we can scan L and compute the total number of occurrences for each color in time proportional to the length of L.

The total query time is given by the formula $Q(n, k) = O(k) + Q(\sqrt{n\tau}, k_1) + Q(\sqrt{n\tau}, k_2)$ where k is the number of colors in the query range and n is the total weight of all points. We denote by k_1 (resp. k_2) the total number of colors reported by the query on R_{j+1} (resp. C_{i+1}). There are at most 2^i recursive calls at level i of recursion. The total weight of points at recursion level i is bounded by $n^{1/2^i} \log^{3(1-1/2^i)} n$. Hence the number of recursion levels is bounded by $\ell = \log\log n - 2\log\log\log n$ and the total query time is $\sum_{i=1}^{\ell} 2^i \cdot k = O(k \cdot (\log n / \log\log n))$.

Fast Queries. We can significantly speed-up queries using the following approach. We keep colors of all points in a column/row in the rank space. Thus each point column or row on the l-th level of recursion contains $O(n^{1/2^l})$ points. Hence for any list L_{ij} on the l-th recursion level we can keep each color and the number of its occurrences in $Dom(i, j)$ using $O((1/2^l) \log n)$ bits.

As explained above, the query on recursion level l is answered by merging three lists: the list L_{ij} that contains the pre-computed answer to the middle query, the list of colors that occur in the right query, and the list of colors that occur in the upper query. Every list occupies $O(k/2^l)$ words of $\log n$ bits. Hence we can merge these lists in $O(k/2^l)$ time using table look-ups. Hence the total query time is $\sum_{i=1}^{\ell} 2^i \cdot \lceil k/2^i \rceil = O(\log n / \log\log n + k \cdot \log\log n)$.

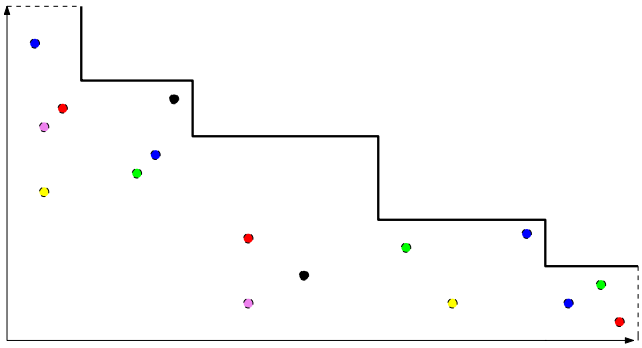


Figure 1 Example of colored t-shallow cutting for $t = 3$.

Color Encoding. In order to merge lists efficiently we must be able to convert the color encoding for the slab V^l into color encoding for the slab V^{l-1} that contains V^l. Moreover the conversion should be performed in $O(k/2^l)$ time, i.e., in sub-constant time per color. For this purpose we introduce the concept of *colored t-shallow cutting* that adapts the concept of shallow cutting to the muti-color scenario. A colored t-shallow cutting for a set of points S is the set of $O(|S|/t)$ cells. Each cell is a rectangle with one corner in the point $(0, 0)$. Each cell contains points of at most $2t$ different colors. If some point q is not contained in any cell of the t-shallow cutting, then q dominates points of at least t different colors.

A colored t-shallow cutting can be constructed using the staircase approach, see e.g., [47]. We start in the point $(0, x_{\max} + 1)$ where $x_{\max}$ is the largest x-coordinate of a point in S. We move p in the $+y$ direction until p dominates $2t$ different colors. Then we move p in the $-x$ direction until p dominates t different colors. We alternatingly move p in $+y$ and $-x$ directions until the x-coordinate of p is 0 or the y-coordinate of p is $y_{\max} + 1$ where $y_{\max}$ is the largest y-coordinate of any point in S. Each point where we stopped moving p in $+y$ direction and started moving p in the $-x$ direction is the upper right corner of some cell. We can show that the number of cell does not exceed $O(|S|/t)$: Let $c_i = (x_i, y_i)$ and $c_{i+1} = (x_{i+1}, y_{i+1})$ denote two consecutive corners (in the left-to-right order) of a t-shallow cutting. Consider all points $p = (x_p, y_p)$ such that $x_i \leq x_p \leq x_{i+1}$ and $y_p \leq y_{i+1}$. By our construction, points p that satisfy these conditions have t different colors. Hence there are at least t such points p and we can assign t unique points to every corner of a colored t-shallow cutting. Hence the number of corners is $O(n/t)$.

For each slab V^l on any recursion level l, we construct colored t-shallow cuttings for $t = 2, 4, \ldots, \tau$. For every cell c_j of each shallow cutting we create the list $\text{clist}(c_j)$ of colors that occur in c_j. Colors in $\text{clist}(c_j)$ are stored in increasing order. For each color we store its rank in V^l and its rank in the slab V^{l-1} that contains V^l. Consider a 2-sided query to a slab V^l on recursion level l. The answer to this query is a sorted list $LIST(q)$ of t colors in the rank space of V^l. If $t < \log^2 n$, then the 2-sided query range is contained in some cell c_j of the colored $2^{\lceil \log t \rceil}$ shallow cutting. Using $\text{clist}(c_j)$ we can convert colors in $LIST(q)$ into the rank space of V^{l-1} where V^{l-1} is the slab that contains V^l. The conversion is based on a universal look-up table and takes $O(t/2^l)$ time.

This result can be extended to 3-sided and 4-sided capped queries using divide-and-conquer on range trees and the recursive grid approach. The space usage of the data structure for capped 4-sided queries is $O(m \log^{\varepsilon} n)$. Finally the range tree on colors enables us to get rid of the restriction on the number of colors, but the space usage of the data structure is increased by $O(\log n)$ factor. The complete description can be found in the full paper.

▶ **Theorem 12.** *Let S be the set of m points in $\mathbb{R}^2$ with total weight $n \geq m$. There exists a data structure that uses $O(m \log m \log^{\varepsilon} n)$ words of space and supports 4-sided type-2 range counting queries in $O(\log n / \log\log n + k \log\log n)$ time.*

References

1. Peyman Afshani, Cheng Sheng, Yufei Tao, and Bryan T. Wilkinson. Concurrent range reporting in two-dimensional space. In *Proc. 25th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 983–994, 2014. doi:10.1137/1.9781611973402.73.
2. Pankaj K. Agarwal, Siu-Wing Cheng, Yufei Tao, and Ke Yi. Indexing uncertain data. In *Proc. 28th ACM Symposium on Principles of Database Systems (PODS)*, pages 137–146, 2009. doi:10.1145/1559795.1559816.
3. Stephen Alstrup, Gerth Stølting Brodal, and Theis Rauhe. New data structures for orthogonal range searching. In *Proc. 41st IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 198–207, 2000. doi:10.1109/SFCS.2000.892088.
4. Jon Louis Bentley and James B. Saxe. Decomposable searching problems I: static-to-dynamic transformation. *J. Algorithms*, 1(4):301–358, 1980. doi:10.1016/0196-6774(80)90015-2.
5. Panayiotis Bozanis, Nectarios Kitsios, Christos Makris, and Athanasios K. Tsakalidis. New upper bounds for generalized intersection searching problems. In *Proc. 22nd International Colloquium on Automata, Languages and Programming (ICALP)*, pages 464–474, 1995. doi:10.1007/3-540-60084-1_97.
6. Timothy M. Chan. Random sampling, halfspace range reporting, and construction of ($\leq k$)-levels in three dimensions. *SIAM J. Comput.*, 30(2):561–575, 2000. doi:10.1137/S0097539798349188.
7. Timothy M. Chan. A dynamic data structure for 3-d convex hulls and 2-d nearest neighbor queries. *J. ACM*, 57(3):16:1–16:15, 2010. doi:10.1145/1706591.1706596.
8. Timothy M. Chan. Persistent predecessor search and orthogonal point location on the word RAM. *ACM Transactions on Algorithms*, 9(3):22, 2013.
9. Timothy M. Chan, Kasper Green Larsen, and Mihai Pătraşcu. Orthogonal range searching on the RAM, revisited. In *Proc. 27th ACM Symposium on Computational Geometry (SoCG)*, pages 1–10, 2011.
10. Timothy M. Chan and Yakov Nekrich. Towards an optimal method for dynamic planar point location. *SIAM Journal on Computing*, 47(6):2337–2361, 2018.
11. Timothy M. Chan and Yakov Nekrich. Better data structures for colored orthogonal range reporting. In *Proc. 31st ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 627–636, 2020.
12. Timothy M. Chan, Yakov Nekrich, Saladi Rahul, and Konstantinos Tsakalidis. Orthogonal point location and rectangle stabbing queries in 3-d. In *Proc. 45th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 31:1–31:14, 2018. doi:10.4230/LIPIcs.ICALP.2018.31.
13. Bernard Chazelle, Richard Cole, Franco P. Preparata, and Chee-Keng Yap. New upper bounds for neighbor searching. *Information and Control*, 68(1-3):105–124, 1986. doi:10.1016/S0019-9958(86)80030-4.
14. Kenneth L Clarkson and Peter W Shor. Applications of random sampling in computational geometry, ii. *Discrete & Computational Geometry*, 4(5):387–421, 1989.
15. Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark Overmars. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, 3rd edition, 2008.
16. Mark de Berg, Marc van Kreveld, and Jack Snoeyink. Two-dimensional and three-dimensional point location in rectangular subdivisions. *Journal of Algorithms*, 18(2):256–277, 1995.
17. Paul F. Dietz. Fully persistent arrays. In *Proc. 1st Workshop on Algorithms and Data Structures (WADS)*, pages 67–74, 1989. doi:10.1007/3-540-51542-9_8.

18 Hicham El-Zein, J. Ian Munro, and Yakov Nekrich. Succinct color searching in one dimension. In *Proc. 28th International Symposium on Algorithms and Computation (ISAAC)*, pages 30:1–30:11, 2017. doi:10.4230/LIPIcs.ISAAC.2017.30.

19 Travis Gagie, Juha Kärkkäinen, Gonzalo Navarro, and Simon J. Puglisi. Colored range queries and document retrieval. *Theor. Comput. Sci.*, 483:36–50, 2013. doi:10.1016/j.tcs.2012.08.004.

20 Arnab Ganguly, J. Ian Munro, Yakov Nekrich, Rahul Shah, and Sharma V. Thankachan. Categorical range reporting with frequencies. In *Proc. 22nd International Conference on Database Theory (ICDT)*, pages 9:1–9:19, 2019. doi:10.4230/LIPIcs.ICDT.2019.9.

21 Roberto Grossi and Søren Vind. Colored range searching in linear space. In *Proc. 14th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT)*, pages 229–240, 2014. doi:10.1007/978-3-319-08404-6_20.

22 Prosenjit Gupta, Ravi Janardan, Saladi Rahul, and Michiel H. M. Smid. Computational geometry: Generalized (or colored) intersection searching. In *Handbook of Data Structures and Applications*, chapter 67, pages 1042–1057. CRC Press, 2nd edition, 2018. URL: https://www.csa.iisc.ac.in/~saladi/Papers/ds2-handbook.pdf.

23 Prosenjit Gupta, Ravi Janardan, and Michiel H. M. Smid. Further results on generalized intersection searching problems: Counting, reporting, and dynamization. *J. Algorithms*, 19(2):282–317, 1995. doi:10.1006/jagm.1995.1038.

24 Prosenjit Gupta, Ravi Janardan, and Michiel H. M. Smid. Algorithms for generalized halfspace range searching and other intersection searching problems. *Comput. Geom.*, 6:1–19, 1996. doi:10.1016/0925-7721(95)00012-7.

25 Prosenjit Gupta, Ravi Janardan, and Michiel H. M. Smid. A technique for adding range restrictions to generalized searching problems. *Inf. Process. Lett.*, 64(5):263–269, 1997. doi:10.1016/S0020-0190(97)00183-X.

26 Prosenjit Gupta, Ravi Janardan, and Michiel H. M. Smid. Algorithms for some intersection searching problems involving circular objects. *International Journal of Mathematical Algorithms*, 1:35–52, 1999.

27 Joseph JáJá, Christian Worm Mortensen, and Qingmin Shi. Space-efficient and fast algorithms for multidimensional dominance reporting and counting. In *Proc. 15th International Symposium on Algorithms and Computation (ISAAC)*, pages 558–568, 2004. doi:10.1007/978-3-540-30551-4_49.

28 Ravi Janardan and Mario A. Lopez. Generalized intersection searching problems. *International Journal of Computational Geometry and Applications*, 3(1):39–69, 1993.

29 Haim Kaplan, Natan Rubin, Micha Sharir, and Elad Verbin. Efficient colored orthogonal range counting. *SIAM J. Comput.*, 38(3):982–1011, 2008. doi:10.1137/070684483.

30 Haim Kaplan, Micha Sharir, and Elad Verbin. Colored intersection searching via sparse rectangular matrix multiplication. In *Proc. 22nd ACM Symposium on Computational Geometry (SoCG)*, pages 52–60, 2006. doi:10.1145/1137856.1137866.

31 Kasper Green Larsen and Rasmus Pagh. I/O-efficient data structures for colored range and prefix reporting. In *Proc. 23rd ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 583–592, 2012. doi:10.1137/1.9781611973099.49.

32 Kasper Green Larsen and Freek van Walderveen. Near-optimal range reporting structures for categorical data. In *Proc. 24th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 256–276, 2013.

33 Jiří Matoušek. Reporting points in halfspaces. *Comput. Geom.*, 2:169–186, 1992. doi:10.1016/0925-7721(92)90006-E.

34 Christian Worm Mortensen. Generalized static orthogonal range searching in less space. Technical report, IT University Technical Report Series 2003-33, 2003.

35 Ketan Mulmuley. *Computational Geometry: An Introduction Through Randomized Algorithms*. Prentice-Hall, 1994.

36 S. Muthukrishnan. Efficient algorithms for document retrieval problems. In *Proc. 13th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 657–666, 2002. doi:10.1145/545381.545469.

37 Yakov Nekrich. Efficient range searching for categorical and plain data. *ACM Trans. Database Syst.*, 39(1):9, 2014. doi:10.1145/2543924.

38 Yakov Nekrich and Jeffrey Scott Vitter. Optimal color range reporting in one dimension. In *Proc. 21st European Symposium on Algorithms (ESA)*, pages 743–754, 2013. doi:10.1007/978-3-642-40450-4_63.

39 János Pach and Gábor Tardos. Tight lower bounds for the size of epsilon-nets. In *Proc. 27th ACM Symposium on Computational Geometry (SoCG)*, pages 458–463, 2011. doi:10.1145/1998196.1998271.

40 Manish Patil, Sharma V. Thankachan, Rahul Shah, Yakov Nekrich, and Jeffrey Scott Vitter. Categorical range maxima queries. In *Proc. 33rd ACM Symposium on Principles of Database Systems (PODS)*, pages 266–277, 2014. doi:10.1145/2594538.2594557.

41 Mihai Patrascu. Lower bounds for 2-dimensional range counting. In *Proc. 39th ACM Symposium on Theory of Computing (STOC)*, pages 40–46, 2007. doi:10.1145/1250790.1250797.

42 Mihai Patrascu. Unifying the landscape of cell-probe lower bounds. *SIAM J. Comput.*, 40(3):827–847, 2011. doi:10.1137/09075336X.

43 F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction*. Springer–Verlag, 1985.

44 Saladi Rahul. Approximate range counting revisited. In *Proc. 33rd International Symposium on Computational Geometry (SoCG)*, pages 55:1–55:15, 2017. doi:10.4230/LIPIcs.SoCG.2017.55.

45 Raimund Seidel. Backwards analysis of randomized geometric algorithms. In J. Pach, editor, *New Trends in Discrete and Computational Geometry*, pages 37–67. Springer-Verlag, 1993.

46 Qingmin Shi and Joseph JáJá. Optimal and near-optimal algorithms for generalized intersection reporting on pointer machines. *Inf. Process. Lett.*, 95(3):382–388, 2005. doi:10.1016/j.ipl.2005.04.008.

47 Darren Erik Vengroff and Jeffrey Scott Vitter. Efficient 3-d range searching in external memory. In *Proc. 28th ACM Symposium on Theory of Computing (STOC)*, pages 192–201, 1996.

A Generalization of Self-Improving Algorithms

Siu-Wing Cheng
HKUST, Hong Kong, China
scheng@cse.ust.hk

Man-Kwun Chiu
Institut für Informatik, Freie Universität Berlin, Germany
chiumk@zedat.fu-berlin.de

Kai Jin
HKUST, Hong Kong, China
cscjjk@gmail.com

Man Ting Wong
HKUST, Hong Kong, China
mtwongaf@connect.ust.hk

Abstract

Ailon et al. [SICOMP'11] proposed self-improving algorithms for sorting and Delaunay triangulation (DT) when the input instances $x_1, \cdots, x_n$ follow some unknown *product distribution*. That is, x_i comes from a fixed unknown distribution $\mathcal{D}_i$, and the x_i's are drawn independently. After spending $O(n^{1+\varepsilon})$ time in a learning phase, the subsequent expected running time is $O((n+H)/\varepsilon)$, where $H \in \{H_{\mathrm{S}}, H_{\mathrm{DT}}\}$, and H_{S} and H_{DT} are the entropies of the distributions of the sorting and DT output, respectively. In this paper, we allow dependence among the x_i's under the *group product distribution*. There is a hidden partition of $[1, n]$ into groups; the x_i's in the k-th group are fixed unknown functions of the same hidden variable u_k; and the u_k's are drawn from an unknown product distribution. We describe self-improving algorithms for sorting and DT under this model when the functions that map u_k to x_i's are well-behaved. After an $O(\mathrm{poly}(n))$-time training phase, we achieve $O(n + H_{\mathrm{S}})$ and $O(n\alpha(n) + H_{\mathrm{DT}})$ expected running times for sorting and DT, respectively, where $\alpha(\cdot)$ is the inverse Ackermann function.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases expected running time, entropy, sorting, Delaunay triangulation

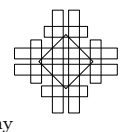
Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.29

Related Version A full version of this paper is available at `http://arxiv.org/abs/2003.08329`.

Funding *Siu-Wing Cheng*: Research of Cheng is supported by Research Grants Council, Hong Kong, China (project no. 16200317).
Man-Kwun Chiu: Research of Chiu is supported by ERC StG 757609.
Kai Jin: Research of Jin is supported by Research Grants Council, Hong Kong, China (project no. 16200317).
Man Ting Wong: Research of Wong is supported by Research Grants Council, Hong Kong, China (project no. 16200317).

1 Introduction

Ailon et al. [1] proposed *self-improving algorithms* for sorting and Delaunay triangulation (DT). The setting is that the input is drawn from an unknown but fixed distribution $\mathcal{D}$. The goal is to automatically compute some auxiliary structures in a *training phase*, so that these auxiliary structures allow an algorithm to achieve an expected running time better than the worst-case optimum in the subsequent *operation phase*. The expected running time in the operation phase is known as the *limiting complexity*.



This model is attractive for two reasons. First, it addresses the criticism that worst-case time complexity alone may not be relevant because worst-case input may occur rarely, if at all. Second, it is more general than some previous average-case analyses that deal with distributions that have simple, compact formulations such as the uniform, Poisson, and Gaussian distributions. There is still a constraint in the work of Ailon et al. [1], that is, $\mathcal{D}$ must be a *product distribution*, meaning that the i-th input item follows a particular distribution and two distinct input items are independently drawn from their respective distributions.

Self-improving algorithms under product distributions have been proposed for sorting, DT, 2D coordinatewise maxima, and 2D convex hull. For sorting, Ailon et al. showed that a limiting complexity of $O((n+H_{\mathrm{S}})/\varepsilon)$ can be achieved for any $\varepsilon \in (0,1)$, where H_{S} denotes the entropy of the output permutations. This limiting complexity is optimal in the comparison-based model by Shannon's theory [11]. The training phase uses $O(n^{\varepsilon})$ input instances and runs in $O(n^{1+\varepsilon})$ time. The probability of achieving the stated limiting complexity is at least $1-1/n$. Ailon et al. [1] also proposed a self-improving algorithm for DT. The performance of the training phase is the same. The limiting complexity is $O((n+H_{\mathrm{DT}})/\varepsilon)$, where H_{DT} denotes the entropy of the output Delaunay triangulations. Self-improving algorithms for 2D coordinatewise maxima and convex hulls have been developed by Clarkson et al. [9]. The limiting complexities for the 2D maxima and 2D convex hull problems are $O(\mathrm{OptM}+n)$ and $O(\mathrm{OptC}+n\log\log n)$, where OptM and OptC are the expected depths of optimal linear decision trees for the maxima and convex hull problems, respectively.

It is natural to allow dependence among input items. However, some restriction is necessary because Ailon et al. showed that $\Omega(2^{n\log n})$ bits of storage are necessary for optimally sorting n numbers if there is no restriction on the input distribution. In [8], two extensions are considered for sorting. The first extension assumes that there is a hidden partition of $[1,n]$ into groups G_k's. The input items with indices in G_k are unknown linear functions of a common parameter u_k. The parameters $u_1, u_2, \cdots$ follow a product distribution. A limiting complexity of $O((n+H_{\mathrm{S}})/\varepsilon)$ can be achieved after a training phase that processes $O(n^{\varepsilon})$ instances in $O(n^2\log^3 n)$ time and $O(n^2)$ space. The second extension assumes that the input is a hidden mixture of product distributions, and that an upper bound m is given on the number of distributions in the mixture. A limiting complexity of $O((n\log m+H_{\mathrm{S}})/\varepsilon)$ can be achieved after a training phase that processes $O(mn\log(mn))$ instances in $O(mn\log^2(mn)+m^{\varepsilon}n^{1+\varepsilon}\log(mn))$ time and $O(mn\log(mn)+m^{\varepsilon}n^{1+\varepsilon})$ space.

In this paper, we revisit the problems of sorting and DT when there is a hidden partition of $[1,n]$ into $G_1, G_2, \cdots$ such that for each k, there is a hidden parameter u_k such that for all $i \in G_k$, $x_i = h_{i,k}(u_k)$ for some unknown function $h_{i,k}$. For sorting, u_k belongs to $\mathbb{R}$; for DT, u_k belongs to $\mathbb{R}^2$; and $u_1, u_2, \cdots$ follow a product distribution. We call such an input distribution a *group product distribution*. The groups G_k's are not given and they have to be learned in the training phase. Our generalization have the following features.

- For sorting, we do not assume any specific formulation of the functions $h_{i,k}$'s. Neither is any oracle given for evaluating them. We only assume that the graph of each $h_{i,k}$ has at most c_0 extrema, where c_0 is a known value, and the graphs of two distinct $h_{i,k}$ and $h_{j,k}$ intersect in $O(1)$ points. Our algorithm does not reconstruct or approximate the $h_{i,k}$'s.
- For DT, we assume that there are bivariate polynomials $h^x_{i,k}$ and $h^y_{i,k}$ in u_k for $i \in G_k$ that give the x- and y-coordinates of the i-th input point. The degrees of the $h^x_{i,k}$'s and $h^y_{i,k}$'s are no more than a fixed constant. No further information about these bivariate polynomials are given. Depending on the distribution of u_k, it may be impossible to reconstruct the equations of $h^x_{i,k}$ and $h^y_{i,k}$ using the input data.

Let $\alpha(\cdot)$ be the inverse Ackermann function. We prove that an optimal $O(n + H_S)$ limiting complexity for sorting and a nearly optimal $O(n\alpha(n) + H_{DT})$ limiting complexity for DT can be achieved with probability at least $1 - O(1/n)$ after a polynomial-time training phase. The training takes $\tilde{O}(c_0 n^3 + c_0^2 n^2)$ time for sorting and $\tilde{O}(n^{10})$ time for DT.

We use several new techniques to obtain our results. To learn the hidden partition for sorting, we need to test if two indices i and j are in the same group. By collecting x_i's and x_j's from some instances, we can reduce the test to finding the longest monotonic subsequence (LMS) among the points (x_i, x_j)'s. We establish a threshold such that i and j are in the same group with very high probability if and only if the LMS has length greater than or equal to the threshold. In the operation phase, there are $O(n)$ ordered intervals from left to right, and we need to sort the subset of an input instance I within an interval. Under the group product distribution, there can be a large subset $I' \subset I$ from the same group that reside in an interval, which does not happen in the case of product distribution. We do not have enough time to sort I' from scratch. Instead, we need to recognize that I' gives a result similar to what we have seen in the training phase. To this end, we must be able to "read off" the answer from some precomputed information in the training phase in order to beat the worst-case bound. We use a trie to compute and store *Lehmer codes* [18] in the training phase. This trie structure is essential for achieving the optimal limiting complexity.

The hidden partition for DT seems harder to learn given the 2D nature of the problem. We employ tools from algebraic geometry to do so, which is the reason for requiring the $h^x_{i,k}$'s and $h^y_{i,k}$'s to be bivariate polynomials of fixed degree. In the operation phase, we need to compute the Voronoi diagram of the subsets of I inside the triangles of a canonical Delaunay triangulation. Under the group product distribution, a large subset $I' \subset I$ may fall in to the same triangle t, which does not happen in the case of product distribution. The big hurdle is to decide what information to compute and store in the training phase so that we can "read off" the Voronoi diagram needed. We need a structure that is equivalent to a Delaunay triangulation or Voronoi diagram. Yet it should be "more combinatorial" in nature so that it is not as sensitive to geometric perturbations. The split tree of a point set fits this role nicely because the Delaunay triangulation can be computed from it in linear expected time [2]. We can now expand the trie structure used for sorting to record different split trees that are generated in the training phase to facilitate the DT computation in the operation phase.

2 Self-improving sorter

Let $u_k \in \mathbb{R}$ denote the parameter that governs the group G_k. Let $h_{i,k}$ denote the function that determines $x_i = h_{i,k}(u_k)$ for $i \in G_k$. We do not impose any particular formulation of the $h_{i,k}$'s as long as they satisfy the following properties:

- $\forall k\ \forall i$, the graph of $h_{i,k}$ has at most c_0 extrema for a known value c_0.
- $\forall k\ \forall i \neq j$, the graphs of $h_{i,k}$ and $h_{j,k}$ intersect at $O(1)$ points.
- $\forall k\ \forall i\ \forall c \in \mathbb{R}$, $\Pr\big[h_{i,k}(u_k) = c\big] = 0$.

2.1 Hidden partition and V-list

We first learn the hidden partition of $[1, n]$. Given a sequence σ of real numbers, let $\mathrm{LMS}(\sigma)$ be the length of the *longest monotone subsequence* of σ (either increasing or decreasing), and let $\mathrm{LIS}(\sigma)$ be the length of the *longest increasing subsequence* of σ.

We describe how to test if the indices 1 and 2 belong to the same group. The other index pairs can be handled in the same way. Let $N = \max\{100^3, (90\ln(4n^3))^2, (6c_0 + 3)^2\}$, where c_0 is the upper bound on the number of extrema of $h_{i,k}$. Take N instances. Let $I_1, I_2, \cdots, I_N$

be these instances in increasing order of their first items. That is, $x_1^{(1)} < \cdots < x_1^{(N)}$, where $x_1^{(i)}$ denotes the first item in I_i. Similarly, $x_2^{(i)}$ denotes the second item in I_i. Then, compute $\mathrm{LIS}\big(x_2^{(1)}, \ldots, x_2^{(N)}\big)$ and $\mathrm{LIS}\big(x_2^{(N)}, \ldots, x_2^{(1)}\big)$ in $O(N \log N)$ time. The larger of the two is $\mathrm{LMS}\big(x_2^{(1)}, \ldots, x_2^{(N)}\big)$. If $\mathrm{LMS}\big(x_2^{(1)}, \ldots, x_2^{(N)}\big) \geq N/(2c_0+1)$, report that 1 and 2 are in the same group. Otherwise, report that 1 and 2 are in different groups.

We show that the above test works correctly with very high probability. The following result is obtained by applying the first theorem in [17] and the setting that $N \geq 100^3$.

▶ **Lemma 1** ([17]). *If σ is a permutation of $[1, N]$ drawn uniformly at random, then* $\Pr\Big[\mathrm{LIS}(\sigma) \geq 3\sqrt{N}\Big] \leq \exp\big(-\sqrt{N}/90\big) = O(1/n^3)$.

We are ready to show the correctness of our test procedure.

▶ **Lemma 2.** *If the indices 1 and 2 belong to the same group, then* $\mathrm{LMS}\big(x_2^{(1)}, \ldots, x_2^{(N)}\big) \geq N/(2c_0+1)$*; otherwise,* $\Pr\Big[\mathrm{LMS}\big(x_2^{(1)}, \ldots, x_2^{(N)}\big) \geq N/(2c_0+1)\Big] = O(1/n^3)$.

Proof. Suppose that 1 and 2 belong to group G_k. Then, $x_1 = h_{1,k}(u_k)$ and $x_2 = h_{2,k}(u_k)$. Let $t_1, \ldots, t_m$, where $m \leq 2c_0$, be the values of u_k at the extrema of $h_{1,k}$ and $h_{2,k}$. Let $t_0 = -\infty$ and let $t_{m+1} = +\infty$. By the pigeonhole principle, there exists $j \in [0, m]$ such that $\big|\{x_1^{(1)}, \ldots x_1^{(N)}\} \cap [t_j, t_{j+1})\big| \geq N/(m+1) \geq N/(2c_0+1)$, and both $h_{1,k}$ and $h_{2,k}$ are monotonic in $[t_j, t_{j+1})$. It follows that $\mathrm{LMS}\big(x_2^{(1)}, \ldots, x_2^{(N)}\big) \geq N/(2c_0+1)$.

Suppose that 1 and 2 belong to different groups. The distribution of $x_2^{(1)}, \cdots, x_2^{(N)}$ is the same as the uniform distribution of the permutations of $[1, N]$. Lemma 1 implies that

$$\Pr\big[\mathrm{LIS}\big(x_2^{(1)}, \ldots, x_2^{(N)}\big) \geq 3\sqrt{N}\big] \leq O(1/n^3).$$

Symmetrically,

$$\Pr\big[\mathrm{LIS}\big(x_2^{(N)}, \ldots, x_2^{(1)}\big) \geq 3\sqrt{N}\big] \leq O(1/n^3).$$

As $\mathrm{LMS}\big(x_2^{(1)}, \ldots, x_2^{(N)}\big) = \max\big\{\mathrm{LIS}\big(x_2^{(1)}, \ldots, x_2^{(N)}\big), \mathrm{LIS}\big(x_2^{(N)}, \ldots, x_2^{(1)}\big)\big\}$, by the union bound, we get $\Pr\big[\mathrm{LMS}\big(x_2^{(1)}, \ldots, x_2^{(N)}\big) \geq 3\sqrt{N}\big] \leq O(1/n^3)$. Since $N \geq (6c_0+3)^2$, we have $3\sqrt{N} \leq N/(2c_0+1)$. Hence, $\Pr\big[\mathrm{LMS}\big(x_2^{(1)}, \ldots, x_2^{(N)}\big) \geq N/(2c_0+1)\big] = O(1/n^3)$. ◀

There are $O(n^2)$ index pairs to check, each taking $O(N \log N)$ time. We conclude that:

▶ **Corollary 3.** *The partition of $[1, n]$ into groups can be learned in $\tilde{O}(c_0^2 n^2)$ time using $\tilde{O}(c_0^2 n^2)$ instances. The probability of success is at least $1 - O(1/n)$.*

Following [1], we define a *V-list*, $(v_0, v_1, \cdots, v_{n+1})$, in the training phase as follows. Take $\lambda = n^2 \ln n$ instances. Let $y_1 < \cdots < y_{\lambda n}$ be these numbers sorted in increasing order. Define $v_0 = -\infty$, $v_r = y_{\lambda r}$ for all $r \in [1, n]$, and $v_{n+1} = +\infty$. We take $[v_0, v_1)$ to be $(-\infty, v_1)$.

▶ **Lemma 4.** *It holds with probability at least $1 - 1/n^{192}$ that for every $r \in [0, n]$,*

$$\mathrm{E}_{I \sim \mathcal{D}}[|I \cap [v_r, v_{r+1})|] = O(1).$$

Proof. Recall that $y_1 < \cdots < y_{\lambda n}$ is the sorted list of the λ instances used to define the V-list. Let $y_0 = -\infty$ and let $y_{\lambda n+1} = \infty$. Fix a pair (i, j) for some distinct $i, j \in [0, \lambda n + 1]$ such that $y_i < y_j$. Let m_{ij} be the number of instances among the λ instances that contain neither y_i nor y_j. Denote them by $I_1, \cdots, I_{m_{ij}}$. Note that $m_{ij} = \lambda - 2$ or $\lambda - 1$.

For $a \in [1, m_{ij}]$, define the random variable $Y_a^{(i,j)} = \big|I_a \cap [y_i, y_j]\big|$. Define $Y^{(i,j)} = Y_1^{(i,j)} + \ldots + Y_{m_{ij}}^{(i,j)}$. We call (i, j) a *good pair* if $\mathrm{E}\big[Y^{(i,j)}\big] \leq 11\lambda$ or $Y^{(i,j)} > \lambda$. We prove in the following that (i, j) is a good pair with high probability.

For $a \in [1, m_{ij}]$, let $X_a = \frac{1}{n} Y_a^{(i,j)}$. Let $X = X_1 + \ldots + X_{m_{ij}} = \frac{1}{n} Y^{(i,j)}$. Note that $X_1, \ldots, X_{m_{ij}}$ are independent and each lies in the range $[0, 1]$. By Hoeffding's inequality [15], $\Pr\big[\big|X - \mathrm{E}[X]\big| \geq \beta m_{ij}\big] < 2e^{-2m_{ij}\beta^2}$ for any $\beta > 0$. Setting $\beta = 10\mathrm{E}[X]/(11m_{ij})$ gives $\Pr\big[\big|X - \mathrm{E}[X]\big| \geq 10\mathrm{E}[X]/11\big] < 2e^{-2m_{ij}(10\mathrm{E}[X])^2/(11m_{ij})^2}$. Thus, $\Pr\big[X < \mathrm{E}[X]/11\big] < 2e^{-200\mathrm{E}[X]^2/(121m_{ij})}$. When $\mathrm{E}[X] > 11\lambda/n$, we have $\Pr[X < \lambda/n] \leq \Pr\big[X < \mathrm{E}[X]/11\big] < 2e^{-200\times 121\lambda^2/(121n^2 m_{ij})} = 2e^{-200\lambda^2/(n^2 m_{ij})} < 2e^{-200\lambda/n^2} = 2n^{-200}$. In other words, it holds that $\Pr\big[Y^{(i,j)} < \lambda\big] < 2n^{-200}$ when $\mathrm{E}\big[Y^{(i,j)}\big] > 11\lambda$. Hence, for a fixed pair (i, j), it holds with probability at least $1 - n^{-199}$ that (i, j) is a good pair.

There are at most $(\lambda n + 2)^2 < n^7$ pairs of distinct indices. By the union bound, it holds with probability at least $1 - n^{-192}$ that all pairs of distinct indices from $[0, \lambda n + 1]$ are good.

Assume that all distinct pairs of indices from $[0, \lambda n + 1]$ are indeed good. Consider two consecutive elements v_r and v_{r+1} in the V-list. They correspond to y_i and y_j, respectively, for some distinct $i, j \in [0, \lambda n + 1]$. By the construction of the V-list, the range (y_i, y_j) contains fewer than λ points among $y_1 < \cdots < y_{\lambda n}$. There are m_{ij} instances used in defining V that contain neither y_i and y_j. Therefore, fewer than λ points from these m_{ij} instances fall in $[y_i, y_j]$, that is, $Y^{(i,j)} < \lambda$. Then, $\mathrm{E}\big[Y^{(i,j)}\big] \leq 11\lambda$ because (i, j) is a good pair by assumption. Hence, $\mathrm{E}\big[|I \cap [v_r, v_{r+1}]|\big] \leq \mathrm{E}\big[Y^{(i,j)}\big]/m_{ij} \leq \mathrm{E}\big[Y^{(i,j)}\big]/(\lambda - 2) = O(1)$. ◀

Lemma 4 is more general than its counterparts in [1, 8] in that it does not make any assumption on the distribution that generates the instances. The price to pay is that $O(n^2 \log n)$ instances are needed instead of $O(\log n)$ in [1, 8], but this cost will be dominated by the trie construction cost in the training phase to be discussed in the next section.

2.2 Trie

In the operation phase, we distribute the numbers in an instance I to the intervals $[v_r, v_{r+1})$'s, sort numbers in each interval, and concatenate the results. We need an efficient method to distribute the numbers. In [8], the functions $h_{i,k}$'s are linear, so we can reformulate each $h_{i,k}$ as a linear function in another input number, say x_1. This gives an arrangement of lines for each G_k (including the horizontal lines at each v_r). Cutting vertically through each arrangement vertex gives a sorted order of the x_i's and the v_r's within a range on x_1.

We cannot compute such arrangements here because there is no assumption on the formulations of the $h_{i,k}$'s. We use a different method. Define:

> $b_k : \mathbb{R}^{|G_k|} \to [0, n]^{|G_k|}$ such that for all $|G_k|$-tuple of numbers $(z_1, \cdots, z_{|G_k|})$, we have $b_k(z_1, \cdots, z_{|G_k|}) = (r_1, \cdots, r_{|G_k|})$ such that for all $m \in \big[1, |G_k|\big]$, $z_m \in [v_{r_m}, v_{r_m+1})$.

The output of b_k tells us to which interval a number in I with index from G_k belongs. In order to distribute these numbers quickly, we need another function defined as follows:

> $\pi_k : \mathbb{R}^{|G_k|} \to [0, n]^{|G_k|}$ such that for all $|G_k|$-tuple of numbers $(z_1, \cdots, z_{|G_k|})$, we have $\pi_k(z_1, \cdots, z_{|G_k|}) = (j_1, \cdots, j_{|G_k|})$ such that for all $m \in \big[1, |G_k|\big]$, $j_m = 0$ if $z_m = \min_{a \in [1,m]} z_a$; otherwise, j_m is the index of the largest element in $\{z_1, \cdots, z_{m-1}\}$ that is less than z_m.

The output of π_k is the Lehmer code of $z_1, \cdots, z_{|G_k|}$ [18]. For our purposes, $(z_1, \cdots, z_{|G_k|})$ is given as a doubly linked list L, and the output of π_k is a list of pointers to entries in L. Given $\pi_k(z_1, \cdots, z_{|G_k|})$, it is easy to sort $z_1, \cdots, z_{|G_k|}$ in increasing order in $O(|G_k|)$ time.

Let $I|_{G_k}$ denote the subsequence of I with indices from G_k. For any set $S \subseteq \mathbb{R}^{|G_k|}$, let $b_k(S) = \{b_k(x) : x \in S\}$ and let $\pi_k(S) = \{\pi_k(x) : x \in S\}$.

▶ **Lemma 5.** *Let $S = \{I|_{G_k} : I \sim \mathcal{D}\}$. Then, $|b_k(S)| = O(c_0 n|G_k|)$ and $|\pi_k(S)| = O(|G_k|^2)$.*

Proof. Assume that $G_k = [1, m]$. The graph of each $h_{i,k}(u_k)$ is a curve in $\mathbb{R}^2$. By assumption, there are $O(m^2)$ intersections among the $h_{i,k}$'s for $i \in [1, m]$. The vertical lines through these intersections divide $\mathbb{R}^2$ into $O(m^2)$ slabs. Clearly, $\pi_k(x_1, \ldots, x_m)$ is invariant when u_k is restricted to any one of these slabs. So there are $O(m^2)$ possible outcomes for $\pi_k(x_1, \ldots, x_m)$.

Add horizontal lines $y = v_r$ for each $r \in [1, n]$. There are at most $(c_0 + 1)nm$ intersections between these horizontal lines and the graphs of $h_{i,k}$'s mentioned above. The vertical lines through the intersections in the overlay of the graphs of $h_{i,k}$'s and these horizontal lines define $O(c_0 nm)$ slabs. Clearly, $b_k(x_1, \ldots, x_m)$ is invariant when u_k is restricted to any one of these slabs. So there are $O(c_0 nm)$ possible outcomes for $b_k(x_1, \ldots, x_m)$. ◀

We prove the main tool for achieving our results on sorting.

▶ **Theorem 6.** *Let $S = \{I|_{G_k} : I \sim \mathcal{D}\}$. Let $f \in \{b_k, \pi_k\}$. Let $n_0 = \max\{n, |f(S)|\}$. Using $n_0 \ln n_0(\ln n + \ln c_0)$ instances in the training phase, with probability at least $1 - n_0^{-2}$, we can compute a data structure such that given $I \sim \mathcal{D}$, it returns $f(I|_{G_k})$ in $O(|G_k| + H_f)$ expected time, where H_f is the entropy of $f(S)$. The data structure uses $O(n_0|G_k|)$ space, and it can be constructed in $\tilde{O}(n_0 n)$ time.*

Proof. Let $N = n_0 \ln n_0(\ln n + \ln c_0)$. Take instances $I_1, \cdots, I_N$ in the training phase. Let $\{\beta_1, \cdots, \beta_M\}$ be the set of distinct outcomes among $f(I_1|_{G_k}), \cdots, f(I_N|_{G_k})$. Let $\tilde{\rho}_i$ be the frequency of β_i divided by N.

Store the β_j's in a trie T. There is one leaf in T for each β_j. Each edge in T has a label from $[0, n]$ so that for $j \in [1, M]$, β_j is equal to the string of symbols on the path from the root to the leaf for β_j.

Given an instance I in the operation phase, we show how to return $f(I|_{G_k})$. For simplicity, let $I|_{G_k} = (x_1, \cdots, x_{|G_k|})$. Let x_0 be a dummy symbol to the left of x_1. Then, repeat the following starting at the root of T: when we are at x_{i-1} and a node u, find the child w of u such that the label on uw is consistent with x_i, and then move to x_i and w. The existence of a particular child w of u depends on whether there is an input instance in the training phase that prompts the creation of w. If we reach a leaf, we return the corresponding β_j. If we cannot find an appropriate child to proceed at any point, we abort and compute $f(I|_{G_k})$ in $O(|G_k| \log n)$ time using plain algorithms. We discuss how to find the correct child quickly.

If $f = b_k$, we seek an edge label r such that $x_i \in [v_r, v_{r+1})$. We use a nearly optimal binary search tree A_u to store the labels of edges to the children of u [19], which can be constructed in linear time [13]. The weight of w in A_u is the node weight that we assign to all nodes of T recursively as follows. The weight of a leaf of T for β_i is $\tilde{\rho}_i$. The weight of an internal node of T is the sum of the weights of its children. The search time of A_u is $O(\log(\text{weight}(u)/\text{weight}(w)))$. To build A_u, in the training phase, we grow a balanced binary search tree L_u from being initially empty to the final set of elements in A_u. That is, as new elements of A_u are discovered, they are inserted into L_u in $O(\log n)$ each. Also, L_u provides access to children of u in the trie in $O(\log n)$ time in the training phase. At the end of the training phase, we build A_u as a nearly optimal binary search tree of the elements in L_u. So the construction of T for b_k takes $O(n_0 n \log n)$ time.

Consider the case of $f = \pi_k$. At the node u, inductively, we know $\gamma : [1, i-1] \to [1, i-1]$ such that $x_{\gamma(a)}$ is the a-th smallest number among $x_1, \cdots, x_{i-1}$. We saw the same permutation γ at u in the training phase. Thus, we organize a search tree A_u with i leaves corresponding to

the i intervals $\omega_0, \omega_1, \ldots, \omega_{i-1}$, where ω_0 denotes the interval between $-\infty$ and the smallest number, ω_a denotes the interval between the a-th and $(a+1)$-th smallest numbers for $a \in [1, i-2]$, and ω_{i-1} denotes the interval between the largest number and ∞. We store $\gamma(a)$ and a pointer to $x_{\gamma(a)}$ at the internal node in A_u that separates the a-th smallest number from the $(a+1)$-th smallest number, so that we can decide in $O(1)$ time whether $x_i < x_{\gamma(a)}$ or $x_i > x_{\gamma(a)}$. The search of A_u terminates at a leaf representing the interval between $x_{\gamma(c)}$ and $x_{\gamma(c+1)}$ for some c. Thus, $\gamma(c)$ is the index of the largest element in $\{x_1, \cdots, x_{i-1}\}$ that is less than x_i. That leaf of A_u also stores a pointer to the corresponding child of u.

We store each map γ as a persistent search tree [12] to facilitate an efficient construction in the training phase. The a-th node in the symmetric order stores $\gamma(a)$. When we create a child w of u in the trie and label the trie edge uw by $\gamma(c)$, we need to extend γ to a map $\gamma' : [1, i] \to [1, i]$ at w such that $\gamma'(a) = \gamma(a)$ for $a \in [1, c]$, $\gamma'(c+1) = i$, and $\gamma'(a+1) = \gamma(a)$ for $a \in [c+1, i-1]$. This can be done by a persistent insertion of a new node between the c-th and $(c+1)$-th nodes in γ. The new version of γ produced is γ'. This takes $O(\log n)$ amortized time and $O(1)$ amortized space.

We store A_u as a nearly optimal binary search tree as in the case of $f = b_k$. In the operation phase, accessing a child w in A_u takes $O(\log(\text{weight}(u)/\text{weight}(w)))$ time.

As a result, querying T takes $O(|G_k| + \log(1/\tilde{\rho}_i))$ time if the search terminates at the leaf of T for β_i. Although the true probability of this event is not $\tilde{\rho}_i$, the entropy of H_f is approximated well by $\sum_i \tilde{\rho}_i \log(1/\tilde{\rho}_i)$, giving an expected time of $O(|G_k| + H_f)$. If we are stuck at some internal node of T, the query time is $O(|G_k| \log n)$ due to the total access time of the A_u's before getting stuck and computing $f(I|_{G_k})$ (by plain algorithms) afterwards. Nevertheless, the probability of this event is very low thanks to the training phase. Hence, the overall expected query time is $O(|G_k| + H_f)$. The details are given in the full version [7]. ◀

Fredman [14] obtained a special case of Theorem 6 when $f = \pi_k$, $\pi_k(S)$ is given, and every outcome in $\pi_k(S)$ is equally likely.

2.3 Operation phase

1. For $r \in [0, n]$, initialize $Z_r := \emptyset$.
2. Repeat the following steps for each group G_k.
   a. Let $I|_{G_k} = (x_{i_1}, \cdots, x_{i_{|G_k|}})$. Compute $\pi_k(x_{i_1}, \cdots, x_{i_{|G_k|}})$ and $b_k(x_{i_1}, \cdots, x_{i_{|G_k|}})$ using Theorem 6.
   b. Use $\pi_k(x_{i_1}, \cdots, x_{i_{|G_k|}})$ to sort $(x_{i_1}, \cdots, x_{i_{|G_k|}})$. Let $x_{s_1} < \cdots < x_{s_{|G_k|}}$ denote the sorting output. Using $b_k(x_{i_1}, \cdots, x_{i_{|G_k|}})$, for all $j \in [1, |G_k|]$, we can read off the interval $[v_{r_j}, v_{r_j+1})$ to which x_{s_j} belongs.
   c. By a left-to-right scan, break $(x_{s_1}, \cdots, x_{s_{|G_k|}})$ at the boundaries of the intervals $[v_r, v_{r+1})$'s into contiguous subsequences. Insert each contiguous subsequence σ as a new element into Z_r, where $[v_r, v_{r+1})$ is the interval that contains the numbers in σ.
3. For each $r \in [0, n]$, merge the subsequences in Z_r into one sorted list.
4. Concatenate the sorted lists of step 3 in the left-to-right order of the intervals. Return the output.

▶ **Theorem 7.** *Under the group product distribution setting, there is a self-improving sorter with a limiting complexity of $O(n + H_S)$. The storage needed by the operation phase is $O(c_0 n^3)$. The training phase processes $\tilde{O}(c_0^2 n^2)$ instances in $\tilde{O}(c_0 n^3 + c_0^2 n^2)$ time using $\tilde{O}(c_0 n^3 + c_0^2 n^2)$ space. The success probability is at least $1 - O(1/n)$.*

Proof. Correctness is obvious. The training complexities and space used in the operation phase follow from Corollary 3, Lemma 5, and Theorem 6. By Theorem 6, step 2(a) takes $O(n+\sum_k H_{b_k}+\sum_k H_{\pi_k})$ time. By the result in [1, Lemma 2.3], $\sum_k H_{b_k} = O(n+H_S)$ because one can merge the sorted order of I with the V-list to obtain the outputs of all b_k's in $O(n)$ time. As there are $O(|G_k|^2)$ different outputs of π_k by Lemma 5, $H_{\pi_k} = O(\ln |G_k|) = O(|G_k|)$ and so $\sum_k H_{\pi_k} = O(n)$. Step 3 runs in $O(\sum_r \sum_k |\sigma_{k,r}| \log |Z_r|) = O\big(\sum_r \sum_k |Z_r||\sigma_{k,r}|\big)$ time, where $\sigma_{k,r} = I|_{G_k} \cap [v_r, v_{r+1})$. Let $y_{k,r}$ be number of groups other than G_k that have elements in Z_r. Then, $|Z_r| \leq y_{k,r} + 1$. So $\mathrm{E}\big[|Z_r||\sigma_{k,r}|\big] \leq \mathrm{E}\big[(y_{k,r}+1)|\sigma_{k,r}|\big] = \mathrm{E}\big[|\sigma_{k,r}|\big] + \mathrm{E}\big[y_{k,r}|\sigma_{k,r}|\big] = \mathrm{E}\big[|\sigma_{k,r}|\big] + \mathrm{E}\big[y_{k,r}\big]\mathrm{E}\big[|\sigma_{k,r}|\big] = (1+\mathrm{E}\big[y_{k,r}\big])\mathrm{E}\big[|\sigma_{k,r}|\big]$, which is $O\big(\mathrm{E}\big[|\sigma_{k,r}|\big]\big)$ as Lemma 4 implies that $\mathrm{E}\big[y_{k,r}\big] = O(1)$. Finally, $\sum_k \sum_r \mathrm{E}\big[|\sigma_{k,r}|\big] = O(n)$. ◀

3 Self-improving Delaunay triangulator

An input instance I consists of n points, $(p_1, \cdots, p_n)$, where $p_i = (p_{i,x}, p_{i,y})$. We assume that there is a hidden partition of $[1, n]$ into disjoint groups $G_1, G_2, \ldots$. For all $k \geq 1$, G_k is governed by a random variable $u_k \in \mathbb{R}^2$. That is, there exist $h^x_{i,k}, h^y_{i,k} : \mathbb{R}^2 \to \mathbb{R}$ such that $p_{i,x} = h^x_{i,k}(u_k)$ and $p_{i,y} = h^y_{i,k}(u_k)$. The functions $h^x_{i,k}$ and $h^y_{i,k}$ are bivariate polynomials with degree at most some known constant d_0. We assume that the following properties hold.

- For all non-zero bivariate polynomial f of degree at most $d_0 d_1$, $\Pr[f(u_k) = 0] = 0$, where $d_1 = 2(d_0^2/2 + d_0)^{16}$.
- $\forall\, i \forall\, k \forall c \in \mathbb{R}$, both $\Pr[h^x_{i,k}(u_k) = c]$ and $\Pr[h^y_{i,k}(u_k) = c]$ are zero.

We show in the full version [7] that $G_1, G_2, \cdots$ can be learned in $O(n^2)$ time almost surely using $O(n^2)$ instances.

3.1 Auxiliary structures

Take a set S of $\lambda = n^2 \ln n$ instances. Take a $(1/n)$-net V' of S with respect to disks, that is, for any disk C, $|C \cap S| \geq |S|/n \Rightarrow C \cap V' \neq \emptyset$. It is known that $|V'| = O(n)$ [10]. Add to V' three special points that form a huge triangle τ such that any input point lies inside τ. Let V be the union of V' and these three special points. The canonical Delaunay triangulation $\mathrm{Del}(V)$ satisfies the following property with a proof analogous to that of Lemma 4.

▶ **Lemma 8.** *It holds with probability at least $1 - 1/n^{189}$ that for every triangle $t \in \mathrm{Del}(V)$, $\mathrm{E}_{I\sim\mathcal{D}}[|I \cap C_t|] = O(1)$, where C_t is the circumscribing disk of t.*

We will need the following function which is the counterpart of b_k for self-improving sorters. Let $|\mathrm{Del}(V)|$ denote the number of triangles in $\mathrm{Del}(V)$. Arbitrarily assign indices from 1 to $|\mathrm{Del}(V)|$ to the triangles in $\mathrm{Del}(V)$; the triangle with index r is denoted by t_r.

$B_k : \mathbb{R}^{2|G_k|} \to \big[1, |\mathrm{Del}(V)|\big]^{|G_k|}$ such that for all $|G_k|$-tuple of points $(q_1, \cdots, q_{|G_k|})$, $B_k(q_1, \cdots, q_{|G_k|}) = (r_1, \cdots, r_{|G_k|})$ such that q_i lies in the triangle $t_{r_i} \in \mathrm{Del}(V)$.

We use a variant of the *fair split tree* in [3]. Let $\hat{R}$ and R be the smallest axis-aligned bounding square and the smallest axis-aligned bounding rectangle, respectively, of the given input instance I. We initialize the split tree to be a single node u with $R(u) = R$ and $\hat{R}(u) = \hat{R}$. In general, for any internal node w, $R(w)$ is the smallest axis-aligned rectangle of the subset of points represented by w, and $\hat{R}(w)$ is an outer rectangle that encloses $R(w)$. To split w, take the bisecting line of $R(w)$ that is perpendicular to a longest side of $R(w)$. This line splits the point set at w into two non-empty subsets, and it also splits $\hat{R}(w)$ into the outer rectangles of the children of w. The expansion bottoms out at nodes that represent only one point in I. This gives a split tree I which is a full binary tree with n leaves. Since we bisect $R(w)$ at each internal node w, we call the output split tree a *halving split tree*.

For every rectangle r, let $\ell_{\min}(r)$ and $\ell_{\max}(r)$ be the minimum and maximum side lengths of r, respectively. The following property is satisfied [3, equation (1) in Lemma 4.1]:

$$\text{For each node } u \text{ of the split tree, } \ell_{\min}(\hat{R}(u)) \geq \frac{1}{3}\ell_{\max}(R(\text{parent}(u))). \tag{1}$$

There are non-halving split trees that also satisfy (1), which are called fair split trees in [3]. We use $SplitT(I)$ to denote a fair split tree of I. Using (1), one can show that a fair split tree can be used to produce a well-separated pair decomposition of $O(n)$ size, which can then be used to produce a Delaunay triangulation in $O(n)$ expected time (see Lemma 12 below).

We will require the following function which is the counterpart of π_k for self-improving sorters. Let HST denote the set of halving split trees of all possible $|G_k|$ points in $\mathbb{R}^2$.

$\Pi_k : \mathbb{R}^{2|G_k|} \to [0,n]^{|G_k|} \times [0,n]^{|G_k|} \times HST$ such that for all $|G_k|$-tuple of points q, $\Pi_k(\mathsf{q}) = (\pi_k(\mathsf{q}_x), \pi_k(\mathsf{q}_y),$ the halving split tree of $\mathsf{q})$, where $\mathsf{q} = (q_1, \cdots, q_{|G_k|})$, $\mathsf{q}_x = (q_{1,x}, \cdots, q_{|G_k|,x})$, and $\mathsf{q}_y = (q_{1,y}, \cdots, q_{|G_k|,y})$.

We call the first output of Π_k the x-order and the second output the y-order.

▶ **Lemma 9.** *Given $S = \{I|_{G_k} : I \sim \mathcal{D}\}$, $|B_k(S)| = O(n^2|G_k|^2)$ and $|\Pi_k(S)| = O(|G_k|^8)$.*

Proof. Assume that $G_k = [1,m]$. Recall that the functions $h^x_{i,k}$'s and $h^y_{i,k}$'s have degrees at most d_0. Consider the following equations for some possibly non-distinct indices $i_1, i_2, i_3, i_4 \in [1,m]$: $h^x_{i_1,k} = h^x_{i_2,k}$, $h^y_{i_1,k} = h^y_{i_2,k}$, $h^x_{i_1,k} - h^x_{i_2,k} = h^y_{i_3,k} - h^y_{i_4,k}$, $h^x_{i_1,k} + h^x_{i_2,k} = 2h^x_{i_3,k}$, and $h^y_{i_1,k} + h^y_{i_2,k} = 2h^y_{i_3,k}$. Each of these equations is an algebraic curve in $\mathbb{R}^2$ of degree at most d_0, so the arrangement $\mathcal{A}$ of these $O(m^4)$ curves has complexity $O(m^8)$. We argue that there are no more combinatorially different halving split trees than the cells in $\mathcal{A}$.

Take the coarser arrangement $\mathcal{A}_0$ formed by the curves $h^x_{i_1,k} = h^x_{i_2,k}$ and $h^y_{i_1,k} = h^y_{i_2,k}$ for distinct indices $i_1, i_2 \in [1,m]$. Each cell of $\mathcal{A}_0$ gives a distinct combination of x-order and y-order, so each cell may lead to a different split tree. Take a cell C of $\mathcal{A}_0$. Suppose that u is the split tree root, $\hat{R}(u)$ has p_{i_1} and p_{i_2} on its vertical sides, and we split $\hat{R}(u)$ vertically. The curves $\{h^x_{i_1,k} + h^x_{i_2,k} = 2h^x_{i_3,k} : i_3 \in [1,m] \setminus \{i_1, i_2\}\}$ divide C into interior-disjoint regions. Each region corresponds to a distinct partition of points into the children w_1 and w_2 of u. Take one such region C'. Suppose that the smallest bounding rectangle of points in w_1 have $p_{i'_1}$ and $p_{i'_2}$ on its vertical sides and $p_{i'_3}$, and $p_{i'_4}$ on its horizontal sides. The sign of $(h^x_{i'_1,k} - h^x_{i'_2,k}) - (h^y_{i'_3,k} - h^y_{i'_4,k})$ determines whether $\hat{R}(w_1)$ is split vertically or horizontally. Similarly, the sign of $(h^x_{i'_5,k} - h^x_{i'_6,k}) - (h^y_{i'_7,k} - h^y_{i'_8,k})$ for some i'_5, i'_6, i'_7, i'_8 tells us if $\hat{R}(w_2)$ is split vertically or horizontally. So the two curves $(h^x_{i'_1,k} - h^x_{i'_2,k}) = (h^y_{i'_3,k} - h^y_{i'_4,k})$ and $(h^x_{i'_5,k} - h^x_{i'_6,k}) = (h^y_{i'_7,k} - h^y_{i'_8,k})$ divide C' to subregions such that each subregion corresponds to a combinatorial distinct splitting of w_1 and w_2. Continuing with the above argument shows that there are no more halving split trees than the cells in $\mathcal{A}$. So $|\Pi_k(S)| = O(m^8)$.

Let $\mathcal{L}$ be the set of support lines of all edges in $\text{Del}(V)$. Let the equations of these lines be $\alpha_j x + \beta_j y + \gamma_j = 0$ for $j \in [1, |\mathcal{L}|]$. Consider the following algebraic curves in $\mathbb{R}^2$.

$$\alpha_j h^x_{i,k}(u_k) + \beta_j h^y_{i,k}(u_k) + \gamma_j = 0 \quad \text{for } i \in [1,m] \text{ and } j \in [1, |\mathcal{L}|].$$

Let $\mathcal{A}'$ be the arrangement of these $O(nm)$ curves. Since these curves have degrees no more than d_0, the complexity of $\mathcal{A}'$ is $O(n^2m^2)$. Inside any cell of $\mathcal{A}'$, the signs of

$$\alpha_j h^x_{i,k}(u_k) + \beta_j h^y_{i,k}(u_k) + \gamma_j \quad \text{for } i \in [1,m] \text{ and } j \in [1, |\mathcal{L}|]$$

are invariant. These signs determine $B_k(p_1, \ldots, p_m)$. Hence, $|B_k(S)| = O(n^2m^2)$. ◀

We can generalize Theorem 6 to work for B_k and Π_k.

▶ **Theorem 10.** *Theorem 6 holds for $f \in \{B_k, \Pi_k\}$.*

Proof. The input to B_k and Π_k is the tuple $\mathsf{q} = I|_{G_k}$. W.l.o.g., let $\mathsf{q} = (q_1, \cdots, q_{|G_k|})$.

For B_k, we build a trie T like the case of $f = b_k$ in the proof of Theorem 6. If we are at q_{i-1} and a node u of T, we seek an edge uw for some child w of u such that q_i lies in $t_r \in \mathrm{Del}(V)$, where r is the label of uw. Therefore, we build a distribution-sensitive planar point location structure for the subset of triangles of $\mathrm{Del}(V)$ represented by the labels of edges from u to its children [16]. For an edge uw with label r, the weight of t_r in the point location structure is equal to the weight of w in T, which is defined recursively as in the proof of Theorem 6. So finding w takes $O(\log(\mathrm{weight}(u)/\mathrm{weight}(w)))$ time [16] as before.

Consider Π_k. The top $|G_k|$ levels of T is a trie T_1 for determining $\pi_k(q_{1,x}, \cdots, q_{|G_k|,x})$. So T_1 is a copy of the trie for π_k in the proof of Theorem 6. We expand each leaf u of T_1 into a trie $T_{2,u}$ for determining $\pi_k(q_{1,y}, \cdots, q_{|G_k|,y})$. Let T_2 be the composition of T_1 and all $T_{2,u}$'s. Given an instance I, we know the x-order and y-order of I at a leaf in T_2.

We expand each leaf v of T_2 to a trie $T_{3,v}$ as follows. Let $P_v = I$. For $i \in [1, n-1]$, v may have a child corresponding to a vertical cut between the i-th and $(i+1)$-th points in the x-order of P_v. Similarly, for $i \in [1, n-1]$, v may have a child corresponding to a horizontal cut between the i-th and $(i+1)$-th points in the y-order of P_v. So v has at most $2n-2$ children. The existence of a particular child w of v depends on whether some input instance in the training phase prompts the creation of w. Each child w of v represents a subset $P_w \subset P_v$. We recursively expand the children of v, and the recursion bottoms out when we reach a node that represents only a single point. The trie $T_{3,v}$ models the sequence of possible cuts deployed in the training phase in constructing a split tree for a point set that has the same x-order and y-order represented by v. Hence, each leaf of $T_{3,v}$ represents a split tree. Let T_3 be the composition of T_2 and all $T_{3,v}$'s.

We need a fast way to locate a child in T_3. We build a nearly optimal binary search trees at each internal node of T_1 and all $T_{2,u}$'s as in the case of π_k in the proof of Theorem 6. At each internal node u of $T_{3,v}$, we keep two nearly optimal binary search trees $A_{u,0}$ and $A_{u,1}$ organized as follows. At the node u, inductively, we know two functions $\gamma_x : [1, |P_u|] \to [1, n]$ and $\gamma_y : [1, |P_u|] \to [1, n]$ such that $q_{\gamma_x(a)}$ and $q_{\gamma_y(a)}$ has the a-th smallest x- and y-coordinates in P_u, respectively. We discovered the same functions γ_x and γ_y at u in the training phase. We organize a nearly optimal binary search tree $A_{u,0}$ with $|P_u| + 1$ leaves corresponding to the $|P_u| + 1$ gaps among $-\infty$, the x-coordinates of points in P_u in increasing order, and ∞. We store $\gamma_x(a)$ at the internal node in $A_{u,0}$ that separates the a-th and the $(a+1)$-th smallest x-coordinates. By comparing $q_{\gamma_x(|P_u|),x} - q_{\gamma_x(1),x}$ and $q_{\gamma_y(|P_u|),y} - q_{\gamma_y(1),y}$ in $O(1)$ time, we can determine whether u should be split vertically or horizontally.

If the cutting line at u is vertical and has x-coordinate X, we can decide in $O(1)$ time whether $X < q_{\gamma_x(a),x}$ or $X > q_{\gamma_x(a),x}$. The search of $A_{u,0}$ terminates at a leaf representing the gap between $q_{\gamma_x(c),x}$ and $q_{\gamma_x(c+1),x}$ for some c. It means that the cut at X should lead us to the child w of u such that the label uw represents a vertical cut between the c-th and $(c+1)$-th smallest x-coordinates. The weight of the leaf $A_{u,0}$ corresponding to child w is $\mathrm{weight}(w)$. The search time of $A_{u,0}$ is thus $O(\log(\mathrm{weight}(u)/\mathrm{weight}(w)))$.

The search tree $A_{u,1}$ is symmetrically organized for the horizontal cuts using γ_y.

Since we preserve the time to descend from a node of the trie to its appropriate child as in the proof of Theorem 6, the overall expected search time of the trie is still $O(|G_k| + H_f)$.

The construction of T_2 has been described in Theorem 6. The construction of $T_{3,v}$ for each leaf v of T_2 follows the procedure to construct a split tree [3]. The construction of the auxiliary structures γ_x', γ_y's, $A_{u,0}$'s. and $A_{u,1}$'s in the training phase is also similar to the construction of analogous structures in Theorem 6. ◀

Given a point set P and $Q \subseteq P$, we can construct $SplitT(Q)$ from $SplitT(P)$ as follows. Make a copy T of $SplitT(P)$. Remove all leaves of T that represent points in $P \setminus Q$. Repeatedly remove nodes in T with only one child until there is none. For each node u of $SplitT(P)$ that is inherited by $SplitT(Q)$, the same cut that splits u in $SplitT(P)$ is also used in splitting u in $SplitT(Q)$. So $SplitT(Q)$ may not be a halving split tree. For every surviving node u in $SplitT(Q)$, $R(u)$ may shrink due to point deletions, but $\hat{R}(u)$ may remain the same or expand. For example, if parent of u is deleted but the grandparent of u survives, then $\hat{R}(u)$ in $SplitT(Q)$ is equal to $\hat{R}(\text{parent}(u))$ in $SplitT(P)$. Hence, (1) is still satisfied by $SplitT(Q)$.

▶ **Lemma 11.** *Suppose that we have constructed $SplitT(P)$ for a point set P.*

(i) *For any $Q \subseteq P$, $SplitT(Q)$ can be computed from $SplitT(P)$ in $O(|P|)$ time.*

(ii) *For any subsets $Q_1, \cdots, Q_m$ of P, if each Q_i is ordered as in the preorder traversal of $SplitT(P)$, then $SplitT(Q_1), \cdots, SplitT(Q_m)$ can be computed from $SplitT(P)$ in $O\big(\alpha(|P|) \cdot \big(|P| + \sum_{i=1}^{m} |Q_i|\big)\big)$ time, where $\alpha(\cdot)$ is the inverse Ackermann function.*

Proof. The correctness of (i) follows from our previous discussion. For (ii), the construction of $SplitT(Q_i)$ boils down to $O(|Q_i|)$ nearest common ancestor queries in $SplitT(P)$, which can be solved in the time stated [20]. There are solutions without the factor $\alpha(\cdot)$, but they require table lookup which is incompatible with the comparison-based model here. ◀

We also need the following result that follows from the works in [3, 2]. The proof is in the full version [7].

▶ **Lemma 12.** *There is a randomized algorithm that constructs $\mathrm{Del}(P)$ from $SplitT(P)$ in $O(|P|)$ expected time.*

3.2 Operation phase

Let $I = (p_1, \cdots, p_n)$ be an input instance. The construction of $\mathrm{Del}(I)$ proceeds as follows.

1. For each k, compute $B_k(I|_{G_k})$ and $\Pi_k(I|_{G_k})$ using Theorem 10.
2. For $i \in G_k$, $B_k(I|_{G_k})$ gives the triangle $t' \in \mathrm{Del}(V)$ that contains p_i, and a BFS in $\mathrm{Del}(V)$ from t' gives $\Delta_i = \{t \in \mathrm{Del}(V) : p_i \in C_t\}$, where C_t is the circumscribing disk of t.
3. For each k,
   a. $\Pi_k(I|_{G_k})$ gives $SplitT(I|_{G_k})$ (note that the halving split tree is also a fair split tree);
   b. traverse $SplitT(I|_{G_k})$ in preorder to produce an ordered list Q_k of points in $I|_{G_k}$;
   c. initialize $Q_{k,t} = \emptyset$ for all $t \in \bigcup_{i \in G_k} \Delta_i$;
   d. for all $p_i \in Q_k$ (in order) and $t \in \Delta_i$, append p_i to $Q_{k,t}$;
4. Compute $SplitT(Q_{k,t})$ for all k and $t \in \bigcup_{i \in G_k} \Delta_i$ from $SplitT(I|_{G_k})$ using Lemma 11(ii).
5. For all k and $t \in \bigcup_{i \in G_k} \Delta_i$, compute $\mathrm{Del}(Q_{k,t})$ from $SplitT(Q_{k,t})$ using Lemma 12.
6. Compute $\mathrm{Vor}(V \cup I)$ and hence $\mathrm{Del}(V \cup I)$ from the $\mathrm{Del}(Q_{k,t})$'s over all k and t.
7. Split $\mathrm{Del}(V \cup I)$ to produce $\mathrm{Del}(I)$ and $\mathrm{Del}(V)$. Return $\mathrm{Del}(I)$.

By Lemma 9 and Theorem 10, the training phase takes $\tilde{O}(n^{10})$ time. Most of time is spent on constructing the tries for the groups $G_1, G_2, \ldots$ to support the retrieval of the $B_k(I|_{G_k})$'s and $\Pi_k(I|_{G_k})$'s.

By Theorem 10, step 1 takes $O(n + \sum_k H_{B_k} + \sum_k H_{\Pi_k})$ expected time. Observe that $|Q_{k,t}|$ is the total number of pairs (p_i, t), where $p_i \in I$ and $t \in \mathrm{Del}(V)$, such that $i \in G_k$ and $p_i \in C_t$. Therefore, $\sum_{k,t} |Q_{k,t}| = \sum_{i=1}^{n} |\Delta_i|$. By Lemma 8, $\mathrm{E}\big[\sum_{i=1}^{n} |\Delta_i|\big] = O(n)$, and therefore, $\mathrm{E}\big[\sum_k |Q_{k,t}|\big] = O(1)$. So steps 2 and 3 run in $O(n)$ expected time. By Lemma 11(ii), the expected running time of step 4 is $O\big(\sum_k |G_k|\alpha(n) + \mathrm{E}\big[\sum_{k,t} |Q_{k,t}|\big]\alpha(n)\big) =$

$O\big(n\alpha(n) + \mathrm{E}\big[\sum_{i=1}^{n} |\Delta_i|\big]\alpha(n)\big) = O(n\alpha(n))$. By Lemma 12, the expected running time of step 5 is $O\big(\mathrm{E}\big[\sum_{k,t} |Q_{k,t}|\big]\big) = O(n)$. For step 7, a randomized algorithm is given in [6] that splits $\mathrm{Del}(V \cup I')$ in $O(|V| + |I'|) = O(n)$ expected time. It remains to discuss the implementation and running time of step 6.

Step 6 is decomposed into two tasks. Let P_t be subset of I that lie in C_t, i.e., $P_t = \bigcup_k Q_{k,t}$. First, for all $t \in \mathrm{Del}(V)$, compute $\mathrm{Del}(P_t)$ by merging the non-empty $\mathrm{Del}(Q_{k,t})$'s over all k. Second, construct $\mathrm{Vor}(V \cup I)$ by merging the $\mathrm{Vor}(P_t)$'s over all $t \in \mathrm{Del}(V)$.

The first task is equivalent to computing $\mathrm{Vor}(P_t)$ for all $t \in \mathrm{Del}(V)$. It is known how to merge two Voronoi diagrams in linear time [4, 5]. So we can merge the non-empty $\mathrm{Vor}(Q_{k,t})$'s in $O\big(\sum_k z_t |Q_{k,t}|\big)$ time, where z_t is the number of groups G_k's with points in C_t. The expected running time of the first task is $O\big(\mathrm{E}\big[\sum_t \sum_k z_t |Q_{k,t}|\big]\big)$. We have seen the analysis of the similar quantity $O\big(\mathrm{E}\big[\sum_r \sum_k |Z_r||\sigma_{k,r}|\big]\big)$ in the proof of Theorem 7, and the same analysis gives $\mathrm{E}\big[\sum_t z_t \sum_k |Q_{k,t}|\big] = O\big(\sum_t \sum_k \mathrm{E}\big[|Q_{k,t}|\big]\big) = O(n)$.

Consider the second task of merging the $\mathrm{Vor}(P_t)$'s over all $t \in \mathrm{Del}(V)$. We use the same strategy in [1]. For each $t \in \mathrm{Del}(V)$, let ν_t be the Voronoi vertex dual to t in $\mathrm{Vor}(V)$. For each Voronoi cell C of $\mathrm{Vor}(V)$, pick the vertex ν_t of C such that $|P_t|$ is smallest, breaking ties by selecting t with the smallest id in $\mathrm{Del}(V)$, and then triangulate C by connecting ν_t to other vertices of C. This gives the *geode triangulation* of $\mathrm{Vor}(V)$ with respect to I.

▶ **Lemma 13** ([1]). *For any geode triangle $\tau = \nu_{t_1}\nu_{t_2}\nu_{t_3}$, $\mathrm{Vor}(V \cup I) \cap \tau = \mathrm{Vor}\big(\{v_\tau\} \cup \bigcup_{i=1}^{3} P_{t_i}\big) \cap \tau$, where v_τ is the point in V whose Voronoi cell contains τ.*

By Lemma 13, we can compute $\mathrm{Vor}\big(\{v_\tau\} \cup \bigcup_{i=1}^{3} P_{t_i}\big) \cap \tau$ for all geode triangles τ, and stitch these fragments to form $\mathrm{Vor}(V \cup I)$. The expected running time is dominated by the expected construction time of $\mathrm{Vor}\big(\{v_\tau\} \cup \bigcup_{i=1}^{3} P_{t_i}\big)$ for all τ's. We merge $\mathrm{Vor}(P_{t_1}), \mathrm{Vor}(P_{t_1}), \mathrm{Vor}(P_{t_3}), v_\tau$ to form $\mathrm{Vor}\big(\{v_\tau\} \cup \bigcup_{i=1}^{3} P_{t_i}\big)$ in linear time [4, 5]. The expected merging time is $O\big(\mathrm{E}\big[1 + \sum_{i=1}^{3} |P_{t_i}|\big]\big) = O(1)$ because $\mathrm{E}\big[|P_{t_i}|\big] = O(1)$ by Lemma 8.

In summary, we conclude that step 6 runs in $O(n)$ expected time.

▶ **Theorem 14.** *Under the group product distribution setting, there is a self-improving Delaunay triangulator with a limiting complexity of $O(n\alpha(n) + H_{\mathrm{DT}})$. The storage needed by the operation phase is $O(n^9)$. The training phase processes $\tilde{O}(n^9)$ instances in $\tilde{O}(n^{10})$ time using $O(n^9)$ space. The success probability is at least $1 - O(1/n)$.*

Proof. The training time complexity and the space complexities of the training and operation phases follow from previous discussion. Moreover, as discussed earlier, the expected running time in the operation phase is $O(n\alpha(n) + \sum_k H_{B_k} + \sum_k H_{\Pi_k})$. Given I and $\mathrm{Del}(I)$, an algorithm is given in [1, Section 4.2] for finding the triangles in $\mathrm{Del}(V)$ that contain the points in I. The same algorithm also works in the group product distribution setting. The expected running time of this algorithm is dominated by $\mathrm{E}\big[\sum_{i=1}^{n} |\Delta_i|\big]$, which is $O(n)$ as we argued previously. We can then apply the result in [1, Lemma 2.3] to conclude that $\sum_k H_{B_k} = O(n + H_{\mathrm{DT}})$. There are $O(|G_k|^8)$ different outputs of Π_k, so $\sum_k H_{\Pi_k} = O(\sum_k \ln |G_k|) = O(n)$. ◀

References

1 N. Ailon, B. Chazelle, K. Clarkson, D. Liu, W. Mulzer, and C. Seshadhri. Self-improving algorithms. *SIAM Journal on Computing*, 40(2):350–375, 2011. doi:10.1137/090766437.

2 K. Buchin and W. Mulzer. Delaunay triangulations in O(sort(N)) time and more. *Journal of the ACM*, 58(2):6:1–6:27, 2011. doi:10.1145/1944345.1944347.

3 P.B. Callahan and S.R. 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.

4 T.M. Chan. A simpler linear-time algorithm for intersecting two convex polyhedra in three dimensions. *Discrete & Computational Geometry*, 56(4):860–865, 2016. doi:10.1007/s00454-016-9785-3.

5 B. Chazelle. An optimal algorithm for intersecting three-dimensional convex polyhedra. *SIAM Journal on Computing*, 21(4):671–696, 1992. doi:10.1137/0221041.

6 B. Chazelle, O. Devillers, F. Hurtado, M. Mora, V. Sacristan, and M. Teillaud. Splitting a delaunay triangulation in linear time. *Algorithmica*, 34(1):39–46, 2002. doi:10.1007/s00453-002-0939-8.

7 S.-W. Cheng, M.-K. Chiu, K. Jin, and M.T. Wong. A generalization of self-improving algorithms. *CoRR*, abs/2003.08329, 2020. arXiv:2003.08329.

8 S.-W. Cheng, K. Jin, and L. Yan. Extensions of self-improving sorters. *Algorithmica*, 82:88–106, 2020. doi:10.1007/s00453-019-00604-6.

9 K.L. Clarkson, W. Mulzer, and C. Seshadhri. Self-improving algorithms for coordinatewise maxima and convex hulls. *SIAM Journal on Computing*, 43(2):617–653, 2014. doi:10.1137/12089702X.

10 K.L. Clarkson and K. Varadarajan. Improved approximation algorithms for geometric set cover. *Discrete and Computational Geometry*, 37:43–58, 2007. doi:10.1007/s00454-006-1273-8.

11 T.M. Cover and J.A. Thomas. *Elements of Information Theory*. Wiley-Interscience, New York, 2 edition, 2006.

12 J.R. Driscoll, N. Sarnak, D.D. Sleator, and R.E. Tarjan. Making data structures persistent. *Journal of Computer System and Sciences*, 38:86–124, 1989. doi:10.1016/0022-0000(89)90034-2.

13 M.L. Fredman. Two applications of a probabilistic search technique: sorting $x+y$ and building balanced search trees. In *Proceedings of the 7th ACM Symposium on Theory of Computing*, pages 240–244, 1975. doi:10.1145/800116.803774.

14 M.L. Fredman. How good is the information theory bound in sorting? *Theoretical Computer Science*, 1(4):355–361, 1976. doi:10.1016/0304-3975(76)90078-5.

15 W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963. URL: http://www.jstor.org/stable/2282952.

16 J. Iacono. Expected asymptotically optimal planar point location. *Computational Geometry: Theory and Applications*, 29:19–22, 2004. doi:10.1016/j.comgeo.2004.03.010.

17 J.H. Kim. On increasing subsequences of random permutations. *Journal of Combinatorial Theory, Series A*, 76(1):148–155, 1996. doi:10.1006/jcta.1996.0095.

18 D.H. Lehmer. Teaching combinatorial tricks to a computer. In *Proceedings of Symposia in Applied Mathematics, Combinatorial Analysis*, volume 10, pages 179–193. American Mathematics Society, 1960.

19 K. Mehlhorn. Nearly optimal binary search trees. *Acta Informatica*, 5:287–295, 1975. doi:10.1007/BF00264563.

20 R.E. Tarjan. Applications of path compression on balanced trees. *Journal of the ACM*, 26(4):690–715, 1979. doi:10.1145/322154.322161.

Dynamic Distribution-Sensitive Point Location

Siu-Wing Cheng
Department of Computer Science and Engineering, HKUST, Hong Kong, China
scheng@cse.ust.hk

Man-Kit Lau
Department of Computer Science and Engineering, HKUST, Hong Kong, China
lmkaa@connect.ust.hk

Abstract

We propose a dynamic data structure for the distribution-sensitive point location problem. Suppose that there is a fixed query distribution in $\mathbb{R}^2$, and we are given an oracle that can return in $O(1)$ time the probability of a query point falling into a polygonal region of constant complexity. We can maintain a convex subdivision $\mathcal{S}$ with n vertices such that each query is answered in $O(\mathrm{OPT})$ expected time, where OPT is the minimum expected time of the best linear decision tree for point location in $\mathcal{S}$. The space and construction time are $O(n \log^2 n)$. An update of $\mathcal{S}$ as a mixed sequence of k edge insertions and deletions takes $O(k \log^5 n)$ amortized time. As a corollary, the randomized incremental construction of the Voronoi diagram of n sites can be performed in $O(n \log^5 n)$ expected time so that, during the incremental construction, a nearest neighbor query at any time can be answered optimally with respect to the intermediate Voronoi diagram at that time.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases dynamic planar point location, convex subdivision, linear decision tree

Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.30

Related Version `https://arxiv.org/abs/2003.08288`

Funding Supported by Research Grants Council, Hong Kong, China (project no. 16201116).

1 Introduction

Planar point location is a classical problem in computational geometry. In the static case, a subdivision is preprocessed into a data structure so that, given a query point, the face containing it can be reported efficiently. In the dynamic case, the data structure needs to accommodate edge insertions and deletions. It is assumed that every new edge inserted does not cross any existing edge. There are well-known worst-case optimal results in the static case [1, 19, 25, 29]. There has been a long series of results in the dynamic case [3, 5, 8, 9, 14, 15, 21, 26, 27]. For a dynamic connected subdivision of n vertices, an $O(\log n)$ query time and an $O(\log^{1+\varepsilon} n)$ update time for any $\varepsilon > 0$ can be achieved [8].

When the faces have different probabilities of containing the query point, it is appropriate to minimize the expected query time. Assume that these probabilities are given or accessible via an oracle. Arya et al. [4] and Iacono [23] obtained optimal expected query time when the faces have constant complexities. Later, Collete et al. [16] obtained the same result for connected subdivisions. So did Afshani et al. [2] and Bose et al. [7] for general subdivisions.

In the case that no prior information about the queries is available, Iacono and Mulzer [24] designed a method for triangulations that can process an online query sequence σ in time proportional to n plus the entropy of σ. We developed solutions for convex and connected subdivisions in a series of work [11, 10, 12]. For convex subdivisions, the processing time is $O(T_{\mathrm{opt}} + n)$, where T_{opt} is the minimum time needed by a linear decision tree to process σ [10]. For connected subdivisions, the processing time is $O(T_{\mathrm{opt}} + n + |\sigma| \log(\log^* n))$ [12].

In this paper, we are interested in dynamic distribution-sensitive planar point location. Such a problem arises when there are online demands for servers that open and close over time, and a nearest server needs to be located for a demand. For example, walking tourists may look for a facility nearby (e.g. convenience store) and search on their mobile phones. The query distribution can be characterized using historical data. New convenience store may open and existing ones may go out of business. If we use the Euclidean metric, then we are locating a query point in a dynamic convex subdivision which is a Voronoi diagram. We are interested in solutions with optimal expected query time.

We assume that there is an oracle that can return in $O(1)$ time the probability of a query point falling inside a polygonal region of constant complexity. We propose a data structure for maintaining a convex subdivision $\mathcal{S}$ with n vertices such that each query is answered in $O(\text{OPT})$ expected time, where OPT is the minimum expected time of the best *point location decision tree for* $\mathcal{S}$, i.e., the best linear decision tree for answering point location queries in $\mathcal{S}$. An update of $\mathcal{S}$ as a mixed sequence of k edge insertions and deletions can be performed in $O(k \log^5 n)$ amortized time. The space and construction time are $O(n \log^2 n)$. As a corollary, we can carry out the randomized incremental construction of the Voronoi diagram of n sites so that, during the incremental construction, a nearest neighbor query at any time can be answered optimally with respect to the intermediate Voronoi diagram at that time. The expected total construction time is $O(n \log^5 n)$ because each site insertion incurs $O(1)$ expected structural changes to the Voronoi diagram. A key ingredient in our solution is a new data structure, *slab tree*, for maintaining a triangulation with a nearly optimal expected point location time and polylogarithmic amortized update time. This data structuring technique may find other applications. Omitted proofs and details are in [13].

2 Dynamic convex subdivision

Let $\mathcal{S}$ be a convex subdivision. Let $\partial\mathcal{S}$ be the outer boundary of $\mathcal{S}$, which bounds a convex polygon. A *general-update sequence* Φ is a mixed sequence of edge insertions and deletions in $\mathcal{S}$ that produces a convex subdivision. The intermediate subdivision after each edge update is only required to be connected, not necessarily convex. Vertices may be inserted into or deleted from $\partial\mathcal{S}$, but the shape of $\partial\mathcal{S}$ is never altered. We will present in Sections 3-5 a dynamic point location structure for a DK-triangulation of $\mathcal{S}$ (to be defined below). Theorem 8 in Section 5 summarizes the performance of this data structure. We show how to apply Theorem 8 to obtain a dynamic distribution-sensitive point location structure for $\mathcal{S}$.

2.1 Dynamic DK-triangulation

Let P be a convex polygon. Find three vertices x, y and z that roughly trisect the boundary of P. This gives a triangle xyz. Next, find a vertex w that roughly bisects the chain delimited by x and y. This gives a triangle xyw adjacent to xyz. We recurse on the other chains to produce a DK-triangulation of P [17]. It has the property that any line segment inside P intersects $O(\log |P|)$ triangles. A DK-triangulation of $\mathcal{S}$ is obtained by computing the DK-triangulations of its bounded faces. Goodrich and Tamassia [20] proposed a method to maintain a *balanced geodesic triangulation* of a connected subdivision. We can use it to maintain a DK-triangulation of $\mathcal{S}$ because a DK-triangulation is a balanced geodesic triangulation. By their method, each edge insertion/deletion in $\mathcal{S}$ is transformed into $O(\log n)$ edge insertions and deletions in the DK-triangulation of $\mathcal{S}$, where n is the number of vertices of $\mathcal{S}$. Consequently, each edge insertion/deletion in $\mathcal{S}$ takes $O(\log^2 n)$ time.

2.2 Point location

We modify our adaptive point location structure for static convex subdivisions [10] to make it work for the distribution-sensitive setting. Compute a DK-triangulation Δ_1 of $\mathcal{S}$. For each triangle $t \in \Delta_1$, use the oracle to compute the probability $\Pr(t)$ of a query point falling into t. This probability is the weight of that triangle. We call the triangles in Δ_1 *non-dummy* because we will introduce some *dummy* triangles later.

Construct a data structure D_1 for Δ_1 with two parts. The first part of D_1 is a new dynamic distribution-sensitve point location structure for triangulations (Theorem 8). The query time of the first part of D_1 is $O(\mathrm{OPT} + \log\log n)$, where OPT is the minimum expected time of the best point location decision tree for Δ_1. The second part can be any dynamic point location structure with $O(\log n)$ query time, provided that its update time is $O(\log^2 n)$ and space is $O(n \log^2 n)$ [3, 8, 15, 28].

For $i \geq 2$, define $n_i = (\log_2 n_{i-1})^4$ inductively, where $n_1 = n$. To construct Δ_i from Δ_{i-1}, extract the non-dummy triangles in Δ_{i-1} whose probabilities of containing a query point are among the top $(\log_2 n_{i-1})^4$. For each subset of extracted triangles that lie inside the same bounded face of $\mathcal{S}$, compute their convex hull and its DK-triangulation. These convex hulls are holes in the polygon H_i with $\partial\mathcal{S}$ as its outer boundary. Triangulate H_i. We call the triangles used in triangulating H_i *dummy* and the triangles in the DK-triangulations of the holes of H_i *non-dummy*. The dummy and non-dummy triangles form the triangulation Δ_i. The size of Δ_i is $O(n_i)$. For each non-dummy triangle $t \in \Delta_i$, set its weight to be $\max\{\Pr(t), W_i^*/n_i\}$, where W_i^* is the sum of $\Pr(t)$ over the non-dummy triangles t in Δ_i. Dummy triangles are given weight W_i^*/n_i. The total weight W_i of all triangles in Δ_i is $\Theta(W_i^*)$. Construct D_i as the point location structure of Iacono [23] for Δ_i, which can answer a query in $O\big(\log \frac{W_i}{w_i}\big)$ time, where w_i is the weight of the triangle containing the query point. The query time of D_i is no worse than $O(\log n_i)$ in the worst case as $w_i \geq W_i^*/n_i = \Theta(W_i/n_i)$.

A hierarchy $(\Delta_1, D_1), \ldots, (\Delta_m, D_m)$ is obtained in the end, where the size of Δ_m is less than some predefined constant. So $m = O(\log^* n)$.

For $i \geq 2$, label every non-dummy triangle $t \in \Delta_i$ with the id of the bounded face of $\mathcal{S}$ that contains it. If t is located by a query, we can report the corresponding face of $\mathcal{S}$. The labelling of triangles in Δ_1 is done differently in order to allow updates in Δ_1 to be performed efficiently. For each vertex p of $\mathcal{S}$, its incident triangles in Δ_1 are divided into circularly consecutive groups by the incident edges of p in $\mathcal{S}$. Thus, each group lies in a distinct face of $\mathcal{S}$ incident to p. We store these groups in clockwise order in a biased search tree T_p [6] associated with p. Each group is labelled by the bounded face of $\mathcal{S}$ that contains it. The group weight is the maximum of $1/n$ and the total probability of a query point falling into triangles in that group. The threshold of $1/n$ prevents the group weight from being too small, allowing T_p to be updated in $O(\log n)$ time. The query time to locate a group is $O\big(\log \frac{W}{w}\big)$, where w is the weight of that group and W is the total weight in T_p. Suppose that D_1 returns a triangle $t \in \Delta_1$ incident to p. We find the group containing t which tells us the face of $\mathcal{S}$ that contains t. If p is a boundary vertex of $\mathcal{S}$, there are two edges in $\partial\mathcal{S}$ incident to p, so we can check in $O(1)$ time whether t lies in the exterior face. Otherwise, we search T_p to find the group containing t in $O\big(\log \frac{W}{w}\big) = O\big(\log \frac{1}{\Pr(t)}\big)$ time.

Given a query point q, we first query D_m with q. If a non-dummy triangle is reported by D_m, we are done. Otherwise, we query D_{m-1} and so on.

▶ **Lemma 1.** *Let $\mathcal{D} = ((\Delta_1, D_1), \ldots, (\Delta_m, D_m))$ be the data structure maintained for $\mathcal{S}$. The expected query time of $\mathcal{D}$ is $O(\mathrm{OPT})$, where* OPT *is the minimum expected time of the best point location decision tree for $\mathcal{S}$.*

2.3 General-update sequence

Let Φ be a general-update sequence with $k \leq n/2$ edge updates. We call k the *size* of Φ. As discussed in Section 2.1, each edge update in $\mathcal{S}$ is transformed into $O(\log n)$ edge updates in Δ_1. Updating Δ_1 takes $O(k \log^2 n)$ time. We also update the biased search tree T_p at each vertex p of $\mathcal{S}$ affected by the structural changes in Δ_1. This step also takes $O(k \log^2 n)$ time.

For $i \geq 2$, we recompute Δ_i from Δ_{i-1} and then D_i from Δ_i. By keeping the triangles of Δ_1 in a max-heap according to the triangle probabilities, which can be updated in $O(k \log^2 n)$ time, we can extract the $n_2 = \log_2^4 n$ triangles to form Δ_2 in $O(n_2 \log n_2)$ time. For $i \geq 3$, we scan Δ_{i-1} to extract the $n_i = \log_2^4 n_{i-1}$ triangles to form Δ_i in $O(n_{i-1} + n_i \log n_i)$ time. For $i \geq 2$, constructing D_i takes $O(n_i)$ time [23]. The total update time of Δ_i and D_i for $i \geq 2$ is $O\left(\sum_{i \geq 2} \log^4 n_{i-1} \log\log n_{i-1}\right)$, which telescopes to $O(\log^4 n \log\log n)$.

Consider D_1. The second part of D_1 is a dynamic point location structure that admits an edge insertion/deletion in Δ_1 in $O(\log^2 n)$ time, giving $O(k \log^3 n)$ total time. By Theorem 8 in Section 5, the update time of the first part of D_1 is $O(k \log^5 n)$ amortized.

In the biased search tree T_p's at the vertices p of $\mathcal{S}$, there are different weight thresholds of $1/n$ depending on when a threshold was computed. To keep these thresholds within a constant factor of each other, we rebuild the entire data structure periodically. Let n' be the number of vertices in the last rebuild. Let $c < 1/2$ be a constant. We rebuild when the total number of edge updates in $\mathcal{S}$ in all general-update sequences exceeds cn' since the last rebuild. Rebuilding the first part of D_1 takes $O(n \log^2 n)$ time by Theorem 8. The second part of D_1 can also be constructed in $O(n \log^2 n)$ time. This results in an extra $O(\log^2 n)$ amortized time per edge update in S.

▶ **Theorem 2.** *Suppose that there is a fixed but unknown query point distribution in $\mathbb{R}^2$, and there is an oracle that returns in $O(1)$ time the probability of a query point falling into a polygonal region of constant complexity. There exists a dynamic point location structure for maintaining a convex subdivision $\mathcal{S}$ of n vertices with the following guarantees.*

- *Any query can be answered in $O(\text{OPT})$ expected time, where OPT is the minimum expected query time of the best point location linear decision tree for $\mathcal{S}$.*
- *The data structure uses $O(n \log^2 n)$ space, and it can be constructed in $O(n \log^2 n)$ time.*
- *A general-update sequence with size $k \leq n/2$ takes $O(k \log^5 n)$ amortized time.*

3 Slab tree: fixed vertical lines

In this section, we present a static data structure for distribution-sensitive point location in a triangulation. Its dynamization will be discussed in Sections 4 and 5.

For any region $R \subset \mathbb{R}^2$, let $\Pr(R)$ denote the probability of a query point falling into R. Let Δ be a triangulation with a convex outer boundary. The vertices of Δ lie on a given set $\mathcal{L}$ of vertical lines, but some line in $\mathcal{L}$ may not pass through any vertex of Δ. For simplicity, we assume that no two vertices of Δ lie on the same vertical line at any time.

Enclose Δ with an axis-aligned bounding box B such that no vertex of Δ lies on the boundary of B. We assume that the left and right sides of B lie on the leftmost and rightmost lines in $\mathcal{L}$. Connect the highest vertex of Δ to the upper left and upper right corners of B, and then connect the lowest vertex of Δ to the lower left and lower right corners of B. This splits $B \setminus \Delta$ into two triangles and two simple polygons. The two simple polygons are triangulated using the method of Hershberger and Suri [22]. Let Δ_B denote the triangle tiling of B formed by Δ and the triangulation of $B \setminus \Delta$. Let n denote the number of triangles in Δ_B. Any line segment in $B \setminus \Delta$ intersects $O(\log n)$ triangles in Δ_B [22]. When we discuss updates in Δ later, the portion $\Delta_B \setminus \Delta$ of the tiling will not change although new vertices may be inserted into the outer boundary of Δ.

3.1 Structure definition

Let $(l_1, l_2, \cdots, l_{|\mathcal{L}|})$ be the vertical lines in $\mathcal{L}$ in left-to-right order. We build the *slab tree* $\mathcal{T}$ as follows. The root of $\mathcal{T}$ represents the slab bounded by l_1 and $l_{|\mathcal{L}|}$. The rest of $\mathcal{T}$ is recursively defined by constructing at most three children for every node v of $\mathcal{T}$.

We use $slab(v)$ to denote the slab represented by v. Let $(l_i, \cdots, l_k)$ be the subsequence of lines that intersect $slab(v)$. Choose $j \in [i, k)$ such that both the probabilties of a query point falling between l_i and l_j and between l_{j+1} and l_k are at most $\Pr(slab(v))/2$. Create the nodes v_L, v_M, and v_R as the left, middle, and right children of v, respectively, where $slab(v_L)$ is bounded by l_i and l_j, $slab(v_M)$ is bounded by l_j and l_{j+1}, and $slab(v_R)$ is bounded by l_{j+1} and l_k. No vertex of Δ_B lies in the interior of v_M.

The recursive expansion of $\mathcal{T}$ bottoms out at a node v if v is at depth $\log_2 n$ or $slab(v)$ contains no vertex of Δ_B in its interior. So the middle child of a node is always a leaf.

Every node v of $\mathcal{T}$ stores several secondary structures. A connected region $R \subset \mathbb{R}^2$ *spans* v if there is a path $\rho \subset R \cap slab(v)$ that intersects both bounding lines of $slab(v)$. The triangulation Δ_B induces a partition of $slab(v)$ into three types of regions:

- Free Gap: For all triangle t that spans v but not $parent(v)$, $t \cap slab(v)$ is a *free gap* of v.
- Blocked Gap: Let E be the set of all edges and triangles in Δ_B that intersect $slab(v)$ but do not span v. Every connected component in the intersection between $slab(v)$ and the union of edges and triangles in E is a *blocked gap* of v.
- Shadow Gap: Take the union of the free gaps of all proper ancestors of v. Each connected component in the intersection between this union and $slab(v)$ is a *shadow gap* of v.

The upper boundary of a blocked gap g has at most two edges, and so does the lower boundary of g. If not, there would be a triangle t outside g that touches g, intersects $slab(v)$, and does not span v. But then t should have been included in g, a contradiction.[1]

Two gaps of v are *adjacent* if the lower boundary of one is the other's upper boundary.

The list of free and blocked gaps of v are stored in vertical order in a balanced search tree, denoted by $gaplist(v)$. Group the gaps in $gaplist(v)$ into maximal contiguous subsequences. Store each such subsequence in a biased search tree [6] which allows an item with weight w to be accessed in $O\big(\log \frac{W}{w}\big)$ time, where W is the total weight of all items. The weight of a gap g set to be $\Pr(g)$. We call each such biased search tree a *gap tree* of v.

For every internal node v of $\mathcal{T}$, we set up some pointers from the gaps of v to the gap trees of the children of v as follows. Let w be a child of v. The free gaps of v only give rise to shadow gaps of w, so they do not induce any item in $gaplist(w)$. Every blocked gap g of v gives rise to a contiguous sequence σ of free and blocked gaps of w. Moreover, σ is maximal in $gaplist(w)$ because g is not adjacent to any other blocked gap of v. Therefore, σ is stored as one gap tree T_σ of w. We keep a pointer from g to the root of T_σ.

Since we truncate the recursive expansion of the slab tree $\mathcal{T}$ at depth $\log_2 n$, we may not be able to answer every query using $\mathcal{T}$. We need a backup which is a dynamic point location structure $\mathcal{T}^*$ [3, 8, 15, 28]. Any worst-case dynamic point location structure with $O(\log n)$ query time suffices, provided that its update time is $O(\log^2 n)$ and its space is $O(n \log n)$.

[1] There is one exception: when a blocked gap boundary contains a boundary edge e of Δ, updates may insert new vertices in the interior of e, splitting e into collinear boundary edges. However, the portion $\Delta_B \setminus \Delta$ of the triangle tiling remains fixed. We ignore this exception to simplify the presentation.

3.2 Querying

Given a query point q, we start at the root r of $\mathcal{T}$, and q must lie in a gap stored in the only gap tree of r. In general, when we visit a node v of $\mathcal{T}$, we also know a gap tree T_v of v such that q lies in one of the gaps in T_v. We search T_v to locate the gap, say g, that contains q. If g is a free gap, the search terminates because we have located a triangle in Δ_B that contains q. Suppose that g is a blocked gap. Then, we check in $O(1)$ time which child w of v satisfies $q \in \mathit{slab}(w)$. By construction, g contains a pointer to the gap tree T_w of w that stores the free and blocked gaps of w in $g \cap \mathit{slab}(w)$. We jump to T_w to continue the search. If the search reaches a leaf of $\mathcal{T}$ without locating a triangle of Δ_B, we answer the query using $\mathcal{T}^*$.

▶ **Lemma 3.** *The expected query time of $\mathcal{T}$ is $O(OPT + \log\log n)$, where OPT is the expected query time of the best point location decision tree for Δ.*

3.3 Construction

The children of a node v of $\mathcal{T}$ can be created in time linear in the number of lines in $\mathcal{L}$ that intersect $\mathit{slab}(v)$. Thus, constructing the primary tree of $\mathcal{T}$ takes $O(|\mathcal{L}| \log n)$ time.

The gap lists and gap trees are constructed via a recursive traversal of $\mathcal{T}$. In general, when we come to a node v of $\mathcal{T}$ from $\mathit{parent}(v)$, we maintain the following preconditions.

- We have only those triangles in Δ_B such that each intersects $\mathit{slab}(v)$ and does not span $\mathit{parent}(v)$. These triangles form a directed acyclic graph G_v: triangles are graph vertices, and two triangles sharing a side are connected by a graph edge directed from the triangle above to the one below.[2]
- The connected components of G_v are sorted in order from top to bottom. Note that each connected component intersects both bounding lines of $\mathit{slab}(v)$.

Each connected component C in G_v corresponds to a maximum contiguous subsequence of free and blocked gaps in $\mathit{gaplist}(v)$ (to be computed), so for each C, we will construct a gap tree T_C. We will return the roots of all such T_C's to $\mathit{parent}(v)$ in order to set up pointers from the blocked gaps of $\mathit{parent}(v)$ to the corresponding T_C's.

Gap list. We construct $\mathit{gaplist}(v)$ first. Process the connected components of G_v in vertical order. Let C be the next one. The restriction of the upper boundary of C to $\mathit{slab}(v)$ is the upper gap boundary induced by C. Perform a topological sort of the triangles in C. We pause whenever we visit a triangle $t \in C$ that spans v. Let t' denote the last triangle in C encountered that spans v, or in the absence of such a triangle, the upper boundary of C. If $t \cap t' = \emptyset$ or $t \cap t'$ does not span v, the region in $\mathit{slab}(v)$ between t' and t is a blocked gap, and we append it to $\mathit{gaplist}(v)$. Then, we append $\mathit{slab}(v) \cap t$ as a newly discovered free gap to $\mathit{gaplist}(v)$. The construction of $\mathit{gaplist}(v)$ takes $O(|G_v|)$ time.

Recurse at the children. Let v_L, v_M and v_R denote the left, middle and right children of v. We scan the connected components of G_v in the vertical order to extract G_{v_L}. A connected component C in G_v may yield multiple components in G_{v_L} because the triangles that span v are omitted. The components in G_{v_L} are ordered vertically by a topological sort of C. Thus, G_{v_L} and the vertical ordering of its connected components are produced in $O(|G_v|)$ time. The generation of G_{v_M}, G_{v_R} and the vertical orderings of their connected components is similar. Then, we recurse at v_L, v_M and v_R.

[2] Refer to [19, Section 4] for a proof that this ordering is acyclic.

Gap trees. After we have recursively handled the children of v, we construct a gap tree for each maximal contiguous subsequence of gaps in $gaplist(v)$. The construction takes linear time [6]. The recursive call at v_L returns a list, say X, of the roots of gap trees at v_L, and X is sorted in vertical order. There is a one-to-one correspondence between X and the blocked gaps of v in vertical order. Therefore, in $O(|gaplist(v)|)$ time, we can set up pointers from the blocked gaps of v to the corresponding gap tree roots in X. The pointers from the blocked gaps of v to the gap tree roots at v_M and v_R are set up in the same manner. Afterwards, if v is not the root of $\mathcal{T}$, we return the list of gap tree roots at v in vertical order.

Running time. We spend $O(|G_v|)$ time at each node v. If a triangle t contributes to G_v for some node v, then either $slab(v) \cap t$ is a free gap of v, or $slab(v) \cap t$ is incident to the leftmost or rightmost vertex of t. Like storing segments in a segment tree, t contributes $O(\log n)$ free gaps. The nodes of $\mathcal{T}$ whose slabs contain the leftmost (resp. rightmost) vertex of t form a root-to-leaf path. Therefore, t contributes $O(\log n)$ triangles in the G_v's over all nodes v in $\mathcal{T}$. The sum of $|G_v|$ over all nodes v of $\mathcal{T}$ is $O(n \log n)$.

▶ **Lemma 4.** *Given Δ_B and $\mathcal{L}$, the slab tree and its auxiliary structures, including gap lists and gap trees, can be constructed in $O(|\mathcal{L}| \log n)$ time and $O(n \log n)$ space.*

4 Handling triangulation-updates: fixed vertical lines

We discuss how to update the slab tree when Δ_B is updated such that every new vertex lies on a vertical line in the given set $\mathcal{L}$. This restriction will be removed later in Section 5. A *triangulation-update* U has the following features:

- It specifies some triangles in Δ whose union is a polygon R_U possibly with holes.
- It specifies a new triangulation T_U of R_U. T_U may contain vertices in the interior of R_U. T_U does not have any new vertex in the boundary of R_U, except possibly for the boundary edges of R_U that lie on the outer boundary of Δ.
- The construction of T_U takes $O(|T_U| \log |T_U|)$ time.
- The *size* of U is the total number of triangles in $\Delta \cap R_U$ and T_U.

Our update algorithm is a localized version of the construction algorithm in Section 3.3. It is also based on a recursive traversal of the slab tree $\mathcal{T}$. When we visit a node v of $\mathcal{T}$, we have a directed acyclic graph H_v that represents *legal* and *illegal* regions in $T_U \cap slab(v)$:

- For each triangle $t \in T_U$ that intersects the interior of $slab(v)$ and does not span $parent(v)$, $t \cap slab(v)$ is a legal region in H_v.
- Take the triangles in T_U that span $parent(v)$. Intersect their union with $slab(v)$. Each resulting connected component that has a boundary vertex in the interior of $slab(v)$ is an *illegal region*. Its upper and lower boundaries contain at most two edges each. Requiring a boundary vertex inside $slab(v)$ keeps the complexity of illegal regions low.
- Store H_v as a directed acyclic graph: regions are graph vertices, and two regions sharing a side are connected by an edge directed from the region above to the one below.

To update $gaplist(v)$, we essentially merge it with a topologically sorted order of regions in each component of H_v.

▶ **Lemma 5.** *Updating $gaplist(v)$ and the gap trees of v takes $O(|H_v| \log n)$ amortized time.*

Let $c < 1/2$ be a constant. We rebuild $\mathcal{T}$ and its auxiliary structures with respect to $\mathcal{L}$ and the current Δ_B when the total size of triangulation-updates exceeds cn' since the initial construction or the last rebuild, where n' was the number of triangles in Δ_B then.

▶ **Lemma 6.** *Let n denote the number of triangles in Δ_B.*

- $n = \Theta(n')$.
- *Any query can be answered in $O(\mathrm{OPT} + \log\log n)$ expected time, where* OPT *is the minimum expected query time of the best point location decision tree for Δ.*
- *The data structure uses $O(n \log n)$ space and can be constructed in $O(|\mathcal{L}| \log n)$ time.*
- *A triangulation-update of size $k \leq n/2$ takes $O(k \log^2 n + (|\mathcal{L}| \log n)/n)$ amortized time.*

5 Allowing arbitrary vertex location

In this section, we discuss how to allow a new vertex to appear anywhere instead of on one of the fixed lines in $\mathcal{L}$. This requires revising the slab tree structure. The main issue is how to preserve the geometric decrease in the probability of a query point falling into the slabs of internal nodes on every root-to-leaf path in $\mathcal{T}$.

Initialize $\mathcal{L}$ to be the set of vertical lines through the vertices of the initial Δ_B. Construct the initial slab tree $\mathcal{T}$ for Δ_B and $\mathcal{L}$ using the algorithm in Section 3.3. Whenever $\mathcal{T}$ is rebuilt, we also rebuild $\mathcal{L}$ to be the set of vertical lines through the vertices of the current Δ_B. Between two successive rebuilds, we grow $\mathcal{L}$ monotonically as triangulation-updates are processed. Although every vertex of Δ_B lies on a line in $\mathcal{L}$, some line in $\mathcal{L}$ may not pass through any vertex of Δ_B between two rebuilds.

The free, blocked, and shadow gaps of a slab tree node are defined as in Section 3. So are the gap trees of a slab tree node. However, gap weights are redefined in Section 5.1 in order that they are robust against small geometric changes.

When a triangulation-update U is processed, we first process the vertical lines through the vertices of T_U before we process T_U as specified in Section 4. For each vertical line ℓ through the vertices of T_U, if $\ell \notin \mathcal{L}$, we insert ℓ into $\mathcal{L}$ and then into $\mathcal{T}$. Sections 5.2 and 5.3 provide the details of this step. The processing of T_U is discussed in Section 5.4.

Querying is essentially the same as in Section 3.2 except that we need a fast way to descend the slab tree as some nodes have $O(\log n)$ children. This is described in Section 5.2.

5.1 Weights of gaps and more

Let n' be the number of triangles in Δ_B in the initial construction or the last rebuild, whichever is more recent. Let $N = 2(c+1)n'$, where c is the constant in the threshold cn' for triggering a rebuild of $\mathcal{T}$.

For every free gap g, let t_g denote the triangle in the current Δ_B that contains g, and the weight of g is $wt(g) = \max\{\Pr(t_g), 1/N\}$. The alternative $1/N$ makes the access time of g in a gap tree no worse than $O(\log N) = O(\log n)$.

For every blocked gap g, every vertex p of Δ_B, and every node v of $\mathcal{T}$, define:

- $wt(p)$ = sum of $\max\left\{\frac{1}{N}, \Pr(t)\right\}$ over all triangles $t \in \Delta_B$ incident to p.
- $vert(g) = \{\text{vertex } p \text{ lying in } g : \exists \text{ triangle } pqr \in \Delta_B \text{ s.t. } interior(pqr) \cap interior(g) \neq \emptyset\}$.
- $wt(g) = \sum_{p \in vert(g)} wt(p)$.
- $blocked\text{-}gaps(p) = \{\text{blocked gap } g : p \in vert(g)\}$.
- $vert(v)$ = the subset of vertices of Δ_B that lie in $slab(v)$.
- $lines(v)$ = the subset of lines in $\mathcal{L}$ that intersect $slab(v)$.

The set $vert(g)$ is only used for notational convenience. The set $blocked\text{-}gaps(p)$ is not stored explicitly. We discuss how to retrieve $blocked\text{-}gaps(p)$ in Section 5.2. The sets $vert(v)$ and $lines(v)$ are stored as balanced search trees in increasing order of x-coordinates.

5.2 Revised slab tree structure

Node types. A vertical line *pierces* a slab if the line intersects the interior of that slab. An internal node v of $\mathcal{T}$ has children of two possible types.

- HEAVY-CHILD: A child w of v is a *heavy-child* if $\Pr(slab(w)) > \Pr(slab(v))/2$.
  - The heavy-child w may be labelled *active* or *inactive* upon its creation. This label will not change. If w was created in the initial construction or the last rebuild of $\mathcal{T}$, then w is inactive.
  - If w is inactive, *gaplist*(w) and the gap trees of w are represented as before. If w is active, then w is a leaf, and *gaplist*(w) and the gap trees of w are stored as persistent data structures using the technique of node copying [18].
- LIGHT-CHILD: There are two sequences of *light-children* of v, denoted by *left-light*(v) and *right-light*(v), which satisfy the following properties.
  - For each light child w of v, $\Pr(slab(w)) \leq \Pr(slab(v))/2$.
  - For each light child w of v, *gaplist*(w) and the gap trees of w are represented as before.
  - Let *left-light*$(v) = (w_1, w_2, \cdots, w_k)$ and let *right-light*$(v) = (w_{k+1}, w_{k+2}, \cdots, w_m)$ in the left-to-right order of the nodes.
    - For $i \in [1, k-1] \cup [k+1, m-1]$, $slab(w_i)$ and $slab(w_{i+1})$ are interior-disjoint and share a boundary.
    - If v has an active heavy-child w, then $slab(w)$ is bounded by the right and left boundaries of $slab(w_k)$ and $slab(w_{k+1})$, respectively. Otherwise, the right boundary of $slab(w_k)$ is the left boundary of $slab(w_{k+1})$.
    - If v does not have an active heavy child, v has at most $2\log_2 N + 2$ children.
    - If v has an active heavy-child, the following properties are satisfied.
      - **(i)** For $r \geq 1$, a light-child w of v has *rank* r if the number of lines in $\mathcal{L}$ that intersect $slab(w)$ is in the range $[2^r, 2^{r+1})$. So $r \leq \log_2 N$, where $N = 2(c+1)n'$. We denote r by *rank*(w).
      - **(ii)** We have *rank*$(w_1) > \cdots >$ *rank*(w_k) and *rank*$(w_{k+1}) < \cdots <$ *rank*(w_m). For $r \in [1, \log_2 N]$, there is at most one light-child of rank r in each of *left-light*(v) and *right-light*(v) .

Node access. Each node v keeps a biased search tree *children*(v). The weight of a child w in *children*(v) is $\max\{\frac{\Pr(slab(v))}{2\log_2 N+2}, \Pr(slab(w))\}$, where $N = 2(c+1)n'$. Since $n = \Theta(n')$, accessing w takes $O\big(\min\{\log \frac{\Pr(slab(v))}{\Pr(slab(w))}, \log\log n\}\big)$ time. For each blocked gap g of v, we use a biased search tree T_g to store pointers to the gap trees induced by g at the children of v. The weight of the node in T_g that represents a gap tree T at a child w is $\max\{\frac{\Pr(slab(v))}{2\log_2 N+2}, \Pr(slab(w))\}$. Accessing T via T_g takes $O\big(\min\{\log \frac{\Pr(slab(v))}{\Pr(slab(w))}, \log\log n\}\big)$ time. Given a vertex p of Δ_B, there are $O(\log n)$ blocked gaps in *blocked-gaps*(p) and we can find them as follows. Traverse the path from the root of $\mathcal{T}$ to the leaf whose slab contains p, and for each node v encountered, we search *gaplist*(v) to find the blocked gap of v that contains p. The time needed is $O(\log^2 n)$.

5.3 Insertion of a vertical line into the slab tree

Let ℓ be a new vertical line. We first insert ℓ into $\mathcal{L}$ and then insert ℓ into $\mathcal{T}$ in a recursive traversal towards the leaf whose slab is pierced by ℓ.

Internal node. Suppose that we visit an internal node v. We first insert ℓ into *lines*(v). We query *children*(v) to find the child slab pierced by ℓ. If v does not have an active heavy-child, recursively insert ℓ at the child found. Otherwise, we work on *left-light*(v) or *right-light*(v).

- Case 1: ℓ pierces *slab*(w_j) for some $w_j \in$ *left-light*(v). If *slab*(w_j) intersects fewer than $2^{rank(w_j)+1}$ lines in $\mathcal{L}$, recursively insert ℓ into w_j and no further action is needed.[3] Otherwise, *slab*(w_j) intersects $2^{rank(w_j)+1}$ lines in $\mathcal{L}$, violating the structural property of a light-child. In this case, we merge some nodes in *left-light*(v) as follows.

  Let *left-light*$(v) = (w_1, \cdots, w_j, \cdots)$. Find the largest $i \le j$ such that the number of lines in $\mathcal{L}$ that intersect *slab*$(w_i) \cup \cdots \cup$ *slab*(w_j) is in the range $[2^r, 2^{r+1})$ for some *rank*$(w_i) \le r <$ *rank*(w_{i-1}). Note that $r >$ *rank*(w_j). Let ℓ_L denote the left boundary of *slab*(w_i). Let ℓ_R denote the right boundary of *slab*(w_j). Let S denote the slab bounded by ℓ_L and ℓ_R. We rebuild the slab subtree rooted at w_i and its auxiliary structures to expand *slab*(w_i) to S as follows. It also means that *rank*(w_i) is updated to r. The children $w_{i+1}, \ldots, w_j$ and their old subtrees are deleted afterwards.

  Let $V =$ *vert*$(w_i) \cup \cdots \cup$ *vert*(w_j). Let $\Lambda =$ *lines*$(w_i) \cup \cdots \cup$ *lines*(w_j). First, we construct a new slab subtree rooted at w_i with respect to V and Λ as described in Section 3.1. No auxiliary structure is computed yet. We control the construction so that it does not produce any node at depth greater than $\log_2 N = O(\log n)$ with respect to the whole slab tree. The construction time is $O(|\Lambda| \log n)$. Afterwards, *slab*(w_i) becomes S. *Label all heavy-children in the new slab subtree rooted at w_i as inactive.*

  Mark the triangles that are incident to the vertices in V, overlap with S, and do not span S. Let G_{w_i} be the set of marked triangles. This takes $O(|G_{w_i}|)$ time, assuming that each vertex p has pointers to its incident triangle(s) intersected by a vertical line through p. The old blocked gaps of w_i will be affected by the rebuild at w_i. The old free gaps of w_i contained in some triangles in G_{w_i} will be absorbed into some blocked gaps. The other old free gaps of w_i are not affected because their containing triangles span S.

  To update *gaplist*(w_i), intersect G_{w_i} with S to generate the directed acyclic graph H_{w_i} and then update *gaplist*(w_i) as in Section 4. This takes $O(|G_{w_i}| \log n)$ amortized time. Only the blocked gaps of w_i can induce gap lists and gap trees at the descendants of w_i. Therefore, as in the construction algorithm in Section 3.3, we can take the subset of G_{w_i} that induce the blocked gaps of w_i and recursively construct the gap lists and gap trees at the descendants of w_i. This takes $O(|G_{w_i}| \log^2 n)$ time by an analysis analogous to the one for Lemma 4.[4] For each blocked gap g of w_i, we create a biased search tree of pointers to the gap trees induced by g at the children of w_i.

  The update of *gaplist*(w_i) preserves the old shadow gaps of w_i, and it does not generate any new shadow gap. Therefore, no two gap trees of w_i can be merged and no gap tree of w_i can be split, although the content of a gap tree may be updated. A gap tree of w_i is updated only when some free gaps in it are merged into some blocked gaps. Thus, updating the gap trees of w_i takes $O(|G_{w_i}| \log n)$ time.

  Finally, *vert*$(w_i) := V$, *lines*$(w_i) := \Lambda$, and the recursive insertion of ℓ terminates.
- Case 2: ℓ pierces *slab*(w_j) for some $w_j \in$ *right-light*(v). Symmetric to Case 1.
- Case 3: ℓ pierces the active heavy-child of v. An active heavy-child is a leaf of the slab tree. We discuss how to insert a vertical line at a leaf next.

---

[3] The line ℓ has already been inserted into $\mathcal{L}$.

[4] Since w_i has $O(\log n)$ instead of $O(1)$ children, the construction time has an extra log factor.

Leaf node. Suppose that we come to a leaf v. If $\text{depth}(v) = \log_2 N$, do nothing and return. Otherwise, there are two cases. Note that *gaplist*(v) consists of free gaps only. The line ℓ divides *slab*(v) into slabs S_L and S_R on the left and right of ℓ, respectively.

- Case 1: v is not an active heavy-child of *parent*(v). Turn v into an internal node by making two children w_L and w_R of v with *slab*$(w_L) = S_L$ and *slab*$(w_R) = S_R$. If $\Pr(\textit{slab}(w_L)) > \Pr(\textit{slab}(v))/2$, then w_L is the heavy-child of v, label w_L active, and set *left-light*$(v) := \emptyset$. If not, w_L is a light-child of rank one and set *left-light*$(v) := (w_L)$. The handling of w_R is symmetric. As *gaplist*(v) consists of free gaps only, *gaplist*(w_L) and *gaplist*(w_R) are empty. So w_L and w_R have no gap tree. The initializations of *vert*(w_L), *vert*(w_R), *lines*(w_L), and *lines*(w_R) are trivial.
- Case 2: v is an active heavy-child of *parent*(v). We expand *left-light*(*parent*(v)) and/or *right-light*(*parent*(v)) as follows. W.l.o.g., assume that $\Pr(S_L) \leq \Pr(\textit{slab}(v))/2$. Update *slab*$(v) := S_R$, which does not change *gaplist*(v) or any gap tree of v combinatorially. The weights of gaps in *gaplist*(v) are also unaffected.
  - Case 2.1: $\Pr(S_R) \leq \Pr(\textit{slab}(\textit{parent}(v)))/2$. Then, *parent*$(v)$ has no heavy-child afterwards. Note that *parent*(v) has at most $2\log_2 N + 1$ children before this update, where $N = 2(c+1)n'$. Create a light-child w_L of *parent*(v) with *slab*$(w_L) = S_L$. Note that *gaplist*(w_L) and the gap trees of w_L are combinatorially identical to those of v, which are stored as persistent search trees. We copy them to form *gaplist*(w_L) and the gap trees of w_L, each taking $O(1)$ amortized space and time. Append w_L to *left-light*(*parent*(v)). Add v to *right-light*(*parent*(v)) as its leftmost element. Therefore, *parent*(v) has at most $2\log_2 N + 2$ children afterwards.
  - Case 2.2: $\Pr(S_R) > \Pr(\textit{slab}(\textit{parent}(v)))/2$. Then, v remains the active heavy-child of *parent*(v). We handle S_L as follows.
    - If *left-light*(*parent*(v)) contains no light-child of rank one, then create a light-child w_L with *slab*$(w_L) = S_L$ as in Case 2.1 above, and append w_L to *left-light*(*parent*(v)).
    - Otherwise, let *left-light*(*parent*(v)) $= (w_1, \cdots, w_k)$, i.e., *rank*$(w_k) = 1$. Find the largest $i \leq k$ such that the number of lines in $\mathcal{L}$ that intersect *slab*$(w_i) \cup \cdots \cup$ *slab*$(w_k) \cup S_L$ is in the range $[2^r, 2^{r+1})$ for some *rank*$(w_i) \leq r <$ *rank*(w_{i-1}). Expand *slab*(w_i) to the slab bounded by the left boundary of *slab*(w_i) and the right boundary of S_L as in Case 1 of the insertion of a vertical line at an internal node. Rebuild the slab subtree rooted at w_i and its auxiliary structures. *Label all heavy-children in the new slab subtree rooted at w_i as inactive.*

▶ **Lemma 7.** *Let $\mathcal{T}$ be the slab tree constructed for a set $\mathcal{L}'$ of vertical lines and Δ_B in the initial construction or the last rebuild, whichever is more recent. For any $\mathcal{L} \supset \mathcal{L}'$, the insertion time of lines in $\mathcal{L} \setminus \mathcal{L}'$ into $\mathcal{T}$ is $O(|\mathcal{L} \setminus \mathcal{L}'| \log^4 n)$ plus some charges on edges of Δ_B such that every edge gains at most $O(\log^4 n)$ charge since the initial construction or the last rebuild, whichever is more recent.*

5.4 Handling triangulation-updates

Let U be a triangulation-update of size $k \leq n/2$. Let n' be the number of triangles in Δ_B in the initial construction or the last rebuild, whichever is more recent. Let c be a constant less than $1/2$. If the threshold cn' has been exceeded by the total size of triangulation-updates (including U) since the initial construction or the last rebuild, we rebuild $\mathcal{T}$ and its auxiliary structures. It takes $O(n \log^2 n)$ time and space. If U does not trigger a rebuild, we proceed as follows instead.

Step 1. Check the $O(k)$ vertical lines through the vertices of T_U. For each line that does not appear in $\mathcal{L}$, we insert it into $\mathcal{L}$ and then into $\mathcal{T}$ as discussed in Section 5.3.

Step 2. The weights of $O(k)$ vertices may change and $O(k)$ vertices may be inserted or deleted. It is straightforward to update the weights of existing vertices, set the weights of new vertices, and delete vertices in $O(k)$ time. For every vertex p of the old triangulation, let $wt'(p)$ be its weight in the old triangulation. For every vertex p of the new triangulation, let $wt(p)$ be its weight in the new triangulation. We perform the following action.

Action-I: for every vertex p of the old triangulation that lies in R_U,
- for every gap $g \in \textit{blocked-gaps}(p)$, update $wt(g) := wt(g) - wt'(p)$;
- if p does not lie in the boundary of R_U, then for each slab tree node v such that $p \in \textit{vert}(v)$, delete p from $\textit{vert}(v)$.

Action-I runs in $O(k \log^2 n)$ time.

Step 3. For every vertex p of T_U that lies strictly inside R_U, and every ancestor v of the leaf node of $\mathcal{T}$ whose slab contains p, insert p into $\textit{vert}(v)$. This step takes $O(k \log^2 n)$ time.

Step 4. To update the gap lists and the gap trees, traverse $\mathcal{T}$ as in Section 4. For each node v of $\mathcal{T}$ visited, form a directed acyclic graph H_v of regions to update v as in Section 4. This step takes $O(\sum_v |H_v| \log n)$ amortized time.

Step 5. The weight of a free gap does not change as long as its defining triangle is preserved. The weights of some blocked gaps may not be updated completely yet, and we fix them by performing Action-II below. Assume that a zero weight is assigned initially to every blocked gap that is created by the triangulation-update and contains vertices in T_U only.

Action-II: for each vertex p of T_U and every gap $g \in \textit{blocked-gaps}(p)$, update $wt(g) := wt(g) + wt(p)$.

Action-II runs in $O(k \log^2 n)$ time.

▶ **Theorem 8.** *Let n denote the number of triangles in Δ_B.*
- *Any query can be answered in $O(\mathrm{OPT} + \log\log n)$ expected time, where OPT is the minimum expected query time of the best point location decision tree for Δ.*
- *The data structure uses $O(n \log^2 n)$ space, and it can be constructed in $O(n \log^2 n)$ time.*
- *A triangulation-update of size $k \leq n/2$ takes $O(k \log^4 n)$ amortized time.*

References

1. U. Adamy and R. Seidel. On the exact worst case query complexity of planar point location. *Journal of Algorithms*, 27(1):189–217, 2000.
2. P. Afshani, J. Barbay, and T. Chan. Instance-optimal geometric algorithms. *Journal of the ACM*, 64(1):3:1–3:38, 2017.
3. L. Arge, G.S. Brodal, and L Georgiadis. Improved dynamic planar point location. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, pages 305–314, 2006.
4. S. Arya, T. Malamatos, D. Mount, and K. Wong. Optimal expected-case planar point location. *SIAM Journal on Computing*, 37(2):584–610, 2007.
5. H. Baumgarten, H. Jung, and K. Mehlhorn. Dynamic point location in general subdivisions. *Journal of Algorithms*, 17(3):342–380, 1994.

6 S.W. Bent, D.D. Sleator, and R.E. Tarjan. Biased search trees. *SIAM Journal on Computing*, 14(3):545–568, 1985.

7 Prosenjit Bose, Luc Devroye, Karim Douieb, Vida Dujmovic, James King, and Pat Morin. Odds-on trees, 2010. `arXiv:1002.1092`.

8 T. Chan and Y. Nekrich. Towards an optimal method for dynamic planar point location. *SIAM Journal on Computing*, 47(6):2337–2361, 2018.

9 S.-W. Cheng and R. Janardan. New results on dynamic planar point location. *SIAM Journal on Computing*, 21(5):972–999, 1992.

10 S.-W. Cheng and M.-K. Lau. Adaptive planar point location. In *Proceedings of the 33rd International Symposium of Computational Geometry*, pages 30:1–30:15, 2017.

11 S.-W. Cheng and M.-K. Lau. Adaptive point location in planar convex subdivisions. *International Journal of Computational Geometry and Applications*, 27(1–2):3–12, 2017.

12 S.-W. Cheng and M.-K. Lau. Adaptive planar point location, 2018. `arXiv:1810.00715`.

13 S.-W. Cheng and M.-K. Lau. Dynamic distribution-sensitive point location, 2020. `arXiv:2003.08288`.

14 Y.-J. Chiang, F.P. Preparata, and R. Tamassia. A unified approach to dynamic point location, ray shooting, and shortest paths in planar maps. *SIAM Journal on Computing*, 25(1):207–233, 1996.

15 Y.-J. Chiang and R. Tamassia. Dynamization of the trapezoid method for planar point location in monotone subdivisions. *Internatational Journal of Computational Geometry and Applications*, 2(3):311–333, 1992.

16 S. Collette, V. Dujmović, J. Iacono, S. Langerman, and P. Morin. Entropy, triangulation, and point location in planar subdivisions. *ACM Transactions on Algorithms*, 8(3):29:1–29:18, 2012.

17 D.P. Dobkin and D.G. Kirkpatrick. Determining the separation of preprocessed polyhedra—a unified approach. In *Proceedings of the 17th International Colloquium on Automata, Languages and Programming*, pages 400–413, 1990.

18 J.R. Driscoll, N. Sarnak, D.D. Sleator, and R.E. Tarjan. Making data structures persistent. *Journal of Computer and System Sciences*, 38(1):86–124, 1989.

19 H. Edelsbrunner, L. J Guibas, and J. Stolfi. Optimal point location in a monotone subdivision. *SIAM Journal on Computing*, 15(2):317–340, 1986.

20 M.T. Goodrich and R. Tamassia. Dynamic ray shooting and shortest paths in planar subdivisions via balanced geodesic triangulations. *Journal of Algorithms*, 23(1):51–73, 1997.

21 M.T. Goodrich and R. Tamassia. Dynamic trees and dynamic point location. *SIAM Journal on Computing*, 28(2):612–636, 1998.

22 J. Hershberger and S. Suri. A pedestrian approach to ray shooting: Shoot a ray, take a walk. *Journal of Algorithms*, 18(3):403–431, 1995.

23 J. Iacono. Expected asymptotically optimal planar point location. *Computational Geometry: Theory and Applications*, 29(1):19–22, 2004.

24 J. Iacono and W. Mulzer. A static optimality transformation with applications to planar point location. *International Journal of Computational Geometry and Applications*, 22(4):327–340, 2012.

25 D. G. Kirkpatrick. Optimal search in planar subdivisions. *SIAM Journal on Computing*, 12(1):28–35, 1983.

26 E. Oh. Point location in incremental planar subdivisions. In *Proceedings of the 29th International Symposium on Algorithms and Computation*, pages 51:1–51:12, 2018.

27 E. Oh and H.-K. Ahn. Point location in dynamic planar subdivision. In *Proceedings of the 34th International Symposium on Computational Geometry*, pages 63:1–53:14, 2018.

28 F.P. Preparata and R. Tamassia. Fully dynamic point location in a monotone subdivision. *SIAM Journal on Computing*, 18(4):811–830, 1989.

29 N. Sarnak and R. E. Tarjan. Planar point location using persistent search trees. *Communications of ACM*, 29(7):669–679, 1986.

No-Dimensional Tverberg Theorems and Algorithms

Aruni Choudhary
Institut für Informatik, Freie Universität Berlin, Takustraße 9, 14195 Berlin, Germany
arunich@inf.fu-berlin.de

Wolfgang Mulzer
Institut für Informatik, Freie Universität Berlin, Takustraße 9, 14195 Berlin, Germany
mulzer@inf.fu-berlin.de

Abstract

Tverberg's theorem states that for any $k \geq 2$ and any set $P \subset \mathbb{R}^d$ of at least $(d+1)(k-1)+1$ points, we can partition P into k subsets whose convex hulls have a non-empty intersection. The associated search problem lies in the complexity class PPAD $\cap$ PLS, but no hardness results are known. In the *colorful* Tverberg theorem, the points in P have colors, and under certain conditions, P can be partitioned into *colorful* sets, in which each color appears exactly once and whose convex hulls intersect. To date, the complexity of the associated search problem is unresolved. Recently, Adiprasito, Bárány, and Mustafa [SODA 2019] gave a *no-dimensional* Tverberg theorem, in which the convex hulls may intersect in an *approximate* fashion. This relaxes the requirement on the cardinality of P. The argument is constructive, but does not result in a polynomial-time algorithm.

We present a deterministic algorithm that finds for any n-point set $P \subset \mathbb{R}^d$ and any $k \in \{2, \ldots, n\}$ in $O(nd\lceil \log k \rceil)$ time a k-partition of P such that there is a ball of radius $O\left(\frac{k}{\sqrt{n}}\mathrm{diam(P)}\right)$ that intersects the convex hull of each set. Given that this problem is not known to be solvable exactly in polynomial time, and that there are no approximation algorithms that are truly polynomial in any dimension, our result provides a remarkably efficient and simple new notion of approximation.

Our main contribution is to generalize Sarkaria's method [Israel Journal Math., 1992] to reduce the Tverberg problem to the Colorful Carathéodory problem (in the simplified tensor product interpretation of Bárány and Onn) and to apply it algorithmically. It turns out that this not only leads to an alternative algorithmic proof of a no-dimensional Tverberg theorem, but it also generalizes to other settings such as the colorful variant of the problem.

2012 ACM Subject Classification Theory of computation → Computational geometry; Theory of computation → Graph algorithms analysis

Keywords and phrases Tverberg's theorem, Colorful Carathéodory Theorem, Tensor lifting

Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.31

Related Version A full version is available on the arXiv (`https://arxiv.org/abs/1907.04284`).

Funding Supported in part by ERC StG 757609.

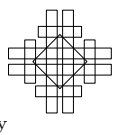
Acknowledgements We would like to thank Frédéric Meunier for stimulating discussions about the Colorful Carathéodory theorem and related problems and for hosting us during a research stay at his lab. We would also like to thank Sergey Bereg for helpful comments on a previous version of this manuscript.

1 Introduction

In 1921, Radon [19] proved a seminal theorem in convex geometry: given a set P of at least $d+2$ points in $\mathbb{R}^d$, one can always split P into two non-empty sets whose convex hulls intersect. In 1966, Tverberg [25] generalized Radon's theorem to allow for more sets

36th International Symposium on Computational Geometry (SoCG 2020).
Editors: Sergio Cabello and Danny Z. Chen; Article No. 31; pp. 31:1–31:17

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



in the partition. Specifically, he showed that any point set $P \subset \mathbb{R}^d$ of cardinality at least $(d+1)(k-1)+1$ can be split into k sets $T_1, \ldots, T_k \subset P$ whose convex hulls have a non-empty intersection, i.e., $\text{conv}(T_1) \cap \cdots \cap \text{conv}(T_k) \neq \emptyset$, where $\text{conv}(\cdot)$ denotes the convex hull.

By now, several alternative proofs of Tverberg's theorem are known, e.g., [3, 6, 15, 20, 21, 26, 27]. Perhaps the most elegant proof is due to Sarkaria [21], with simplifications by Bárány and Onn [6] and by Aroch et al. [3]. In this paper, all further references to *Sarkaria's method* refer to the simplified version. This proof proceeds by a reduction to the *Colorful Carathéodory Theorem*, another celebrated result in convex geometry: given $r \geq d+1$ point sets $P_1, \ldots, P_r \subset \mathbb{R}^d$ that have a common point y in their convex hulls $\text{conv}(P_1), \ldots, \text{conv}(P_r)$, there is a *traversal* $x_1 \in P_1, \ldots, x_r \in P_r$, such that $\text{conv}(\{x_1, \ldots, x_r\})$ contains y. Sarkaria's proof [21] uses a Tensor product to lift the original points of the Tverberg instance into higher dimensions, and then uses the Colorful Carathéodory traversal to obtain a Tverberg partition for the original point set.

From a computational point of view, a Radon partition is easy to find by solving $d+1$ linear equations. On the other hand, finding Tverberg partitions is not straightforward. Since a Tverberg partition must exist if P is large enough, finding such a partition is a total search problem. In fact, the problem of computing a Colorful Carathéodory traversal lies in the complexity class $\text{PPAD} \cap \text{PLS}$ [14, 16], but no better upper bound is known. Since Sarkaria's proof gives a polynomial-time reduction from the problem of finding a Tverberg partition to the problem of finding a colorful traversal, the same complexity applies to Tverberg partitions. Again, as of now we do not know better upper bounds for the general problem. Miller and Sheehy [15] and Mulzer and Werner [17] provided algorithms for finding *approximate* Tverberg partitions, computing a partition into fewer sets than is guaranteed by Tverberg's theorem in time that is linear in n, but quasi-polynomial in the dimension. These algorithms were motivated by applications in mesh generation and statistics that require finding a point that lies "deep" in P. A point in the common intersection of the convex hulls of a Tverberg partition has this property, with the partition serving as a certificate of depth.

Tverberg's theorem also admits a colorful variant, first conjectured by Bárány and Larman [5]. The setup consists of $d+1$ point sets $P_1, \ldots, P_{d+1} \subset \mathbb{R}^d$, each set interpreted as a different color and having size t. For a given k, the goal is to find k pairwise-disjoint *colorful* sets (i.e., each set contains at most one point from each P_i) $A_1, \ldots, A_k$ such that $\cap_{i=1}^{k} \text{conv}(A_i) \neq \emptyset$. The problem is to determine the optimal value for t such that such a colorful partition always exists. Bárány and Larman [5] conjectured that $t = k$ suffices and they proved the conjecture for $d = 2$ and arbitrary k, and for $k = 2$ and arbitrary d. The first result for the general case was given by Živaljević and Vrećica [29] through topological arguments. Using another topological argument, Blagojević, Matschke, and Ziegler [7] showed that (i) if $k+1$ is prime, then $t = k$; and (ii) if $k+1$ is not prime, then $k \leq t \leq 2k-2$. These are the best known bounds for arbitrary k. Later Matoušek, Tancer, and Wagner [13] gave a geometric proof that is inspired by the proof of Blagojević, Matschke, and Ziegler [7].

More recently, Soberón [22] showed that if more color classes are available, then the conjecture holds for any k. More precisely, for $P_1, \ldots, P_n \subset \mathbb{R}^d$ with $n = (k-1)d+1$, each of size k, there exist k colorful sets whose convex hulls intersect. Moreover, there is a point in the common intersection so that the coefficients of its convex combination are the same for each colorful set in the partition. The proof uses Sarkaria's tensor product construction.

Recently Adiprasito, Bárány, and Mustafa [1] established a relaxed version of the Colorful Carathéodory Theorem and some of its descendants [4]. For the Colorful Carathéodory theorem, this allows for a (relaxed) traversal of arbitrary size, with a guarantee that the convex hull of the traversal is close to the common point y. For the Colorful Tverberg

problem, they prove a version of the conjecture where the convex hulls of the colorful sets intersect approximately. This also gives a relaxation for Tverberg's theorem [25] that allows arbitrary-sized partitions, again with an approximation notion of intersection. Adiprasito et al. refer to these results as *no-dimensional* versions of the respective classic theorems, since the dependence on the ambient dimension is relaxed. The proofs use averaging arguments. The argument for the no-dimensional Colorful Carathéodory also gives an efficient algorithm to find a suitable traversal. However, the arguments for the no-dimensional Tverberg results do not give a polynomial-time algorithm for finding the partitions.

Our contributions. We prove no-dimensional variants of the Tverberg theorem and its colorful counterpart that allow for efficient algorithms. Our proofs are inspired by Sarkaria's method [21] and the averaging technique by Adiprasito, Bárány, and Mustafa [1]. For the colorful version, we additionally make use of ideas of Soberón [22]. Furthermore, we also give a no-dimensional generalized ham-sandwich theorem that interpolates [28] between the centerpoint theorem and the ham-sandwich theorem [24], again with an efficient algorithm.

Algorithmically, Tverberg's theorem is useful for finding centerpoints of high-dimensional point sets, which in turn has applications in statistics and mesh generation [15]. In fact, most algorithms for finding centerpoints are Monte-Carlo, returning some point p and a probabilistic guarantee that p is indeed a centerpoint [9, 11]. However, this is coNP-hard to verify. On the other hand, a (possibly approximate) Tverberg partition immediately gives a certificate of depth [15,17]. Unfortunately, there are no polynomial-time algorithms for finding optimal Tverberg partitions, and the approximation algorithms are not truly polynomial in the dimension. In this context, our result provides a fresh notion of approximation that also leads to very fast polynomial-time algorithms.

Furthermore, the Tverberg problem is intriguing from a complexity theoretic point of view, because it constitutes a total search problem that is not known to be solvable in polynomial time, but which is also unlikely to be NP-hard. So far, such problems have mostly been studied in the context of algorithmic game theory [18], and only very recently a similar line of investigation has been launched for problems in high-dimensional discrete geometry [10, 12, 14, 16]. Thus, we show that the *no-dimensional* variant of Tverberg's theorem is easy from this point of view. Our main results are as follows:

- Sarkaria's method uses a specific set of k vectors in $\mathbb{R}^{k-1}$ to lift the points in the Tverberg instance to a Colorful Carathéodory instance. We refine this method to vectors that are defined with the help of a given graph. The choice of this graph is important in proving good bounds for the partition and in the algorithm. We believe that this generalization is of independent interest and may prove useful in other scenarios that rely on the tensor product construction.
- We prove an efficient no-dimensional Tverberg result:

  ▶ **Theorem 1.1** (efficient no-dimensional Tverberg theorem). *Let $P \subset \mathbb{R}^d$ be a set of n points, and let $k \in \{2, \ldots, n\}$ be an integer. Let $\mathbb{D}(\cdot)$ denote the diameter.*
  - *For any choice of positive integers $r_1, \ldots, r_k$ that satisfy $\sum_{i=1}^{k} r_i = n$, there is a partition $T_1, \ldots, T_k$ of P with $|T_1| = r_1, |T_2| = r_2, \ldots, |T_k| = r_k$, and a ball B of radius $\frac{n\,\mathbb{D}(P)}{\min_i r_i}\sqrt{\frac{10\lceil \log_4 k\rceil}{n-1}} = O\left(\frac{\sqrt{n\log k}}{\min_i r_i}\,\mathbb{D}(P)\right)$ that intersects the convex hull of each T_i.*
  - *The bound is better for the case $n = rk$ and $r_1 = \cdots = r_k = r$. There exists a partition $T_1, \ldots, T_k$ of P with $|T_1| = \cdots = |T_k| = r$ and a d-dimensional ball of radius $\sqrt{\frac{k(k-1)}{n-1}}\mathbb{D}(P) = O\left(\frac{k}{\sqrt{n}}\mathbb{D}(P)\right)$ that intersects the convex hull of each T_i.*

  In either case, we can compute the partition in deterministic time $O(nd\lceil \log k\rceil)$.

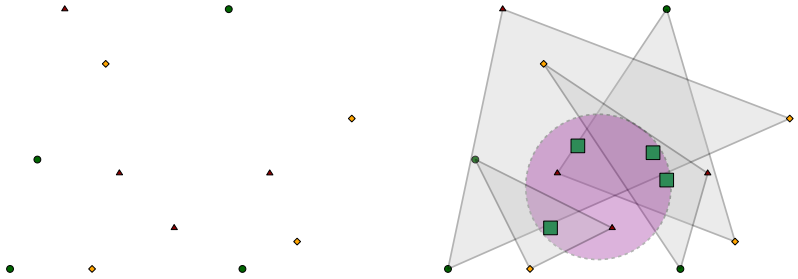


Figure 1 Left: a point set on three colors and four points of each color. Right: a colorful partition with a ball containing the centroids (squares) of the sets of the partition.

- and a colorful counterpart (for a simple example, see Figure 1):

  ▶ **Theorem 1.2** (efficient no-dimensional Colorful Tverberg). *Let $P_1, \ldots, P_n \subset \mathbb{R}^d$ be n point sets, each of size k, with k being a positive integer, so that the total number of points is $N = nk$. Then, there are k pairwise-disjoint colorful sets $A_1, \ldots, A_k$ and a ball of radius $\sqrt{\frac{2k(k-1)}{N}} \max_i \mathbb{D}(P_i) = O\left(\frac{k}{\sqrt{N}} \max_i \mathbb{D}(P_i)\right)$ that intersects $\operatorname{conv}(A_i)$ for each $i \in [k]$. We can find the A_is in deterministic time $O(Ndk)$.*

- For any sets $P, x \subset \mathbb{R}^d$, the *depth* of x with respect to P is the largest positive integer k such that every half-space that contains x also contains at least k points of P.

  ▶ **Theorem 1.3** (no-dimensional Generalized Ham-Sandwich Theorem). *Let $P_1, \ldots, P_k$ be $k \leq d$ finite point sets in $\mathbb{R}^d$. Then there is a $(d-k+1)$-dimensional ball B and $k-1$ lines $\ell_1, \ldots, \ell_{k-1}$ such that the d-dimensional Cartesian product $B \times \ell_{k-1} \times \ell_{k-2} \times \cdots \times \ell_1$ has depth at least $\left\lceil \frac{|P_i|}{m_i} \right\rceil$ with respect to P_i, for $i \in [k]$. Here, $\{2 \leq m_i \leq |P_i|, i \in [k]\}$ is any set of chosen integer parameters. The ball B has radius $(2+2\sqrt{2}) \max_i \frac{\mathbb{D}(P_i)}{\sqrt{m_i}}$ and the lines $\ell_1, \ldots, \ell_{k-1}$ can be determined in $O(dk^2 + \sum_i |P_i| d)$ time.*

The colorful Tverberg result is similar in spirit to the regular version, but from a computational viewpoint, it does not make sense to use the colorful algorithm to solve the regular Tverberg problem. Due to space constraints, the colorful Tverberg and the Generalized Ham-Sandwich results have been deferred to an extended version in [8].

Compared to the results of Adiprasito et al. [1], our radius bounds are slightly worse. More precisely, they show that both in the colorful and the non-colorful case, there is a ball of radius $O\left(\sqrt{\frac{k}{n}}\mathbb{D}(P)\right)$ that intersects the convex hulls of the sets of the partition. They also show this bound is close to optimal. In contrast, our result is off by a factor of $O(\sqrt{k})$, but derandomizing the proof of Adiprasito et al. [1] gives only a brute-force $2^{O(n)}$-time algorithm. In contrast, our approach gives almost linear time algorithms for both cases, with a linear dependence on the dimension.

Techniques. Adiprasito et al. first prove the colorful no-dimensional Tverberg theorem using an averaging argument over an exponential number of possible partitions. Then, they specialize their result for the non-colorful case, obtaining a bound that is asymptotically optimal. Unfortunately, it is not clear how to derandomize the averaging argument efficiently. The method of conditional expectations applied to their averaging argument leads to a runtime of $2^{O(n)}$. To get around this, we follow an alternate approach towards both versions of the Tverberg theorem. Instead of a direct averaging argument, we use a reduction to the Colorful Carathéodory theorem that is inspired by Sarkaria's proof, with some additional

twists. We will see that this reduction also works in the no-dimensional setting, i.e., by a reduction to the no-dimensional Colorful Carathéodory theorem of Adiprasito et al., we obtain a no-dimensional Tverberg theorem, with slightly weaker radius bounds, as stated above. This approach has the advantage that their Colorful Carathéodory theorem is based on an averaging argument that permits an efficient derandomization using the method of conditional expectations [2]. In fact, we will see that the special structure of the no-dimensional Colorful Carathéodory instance that we create allows for a very fast evaluation of the conditional expectations, as we fix the next part of the solution. This results in an algorithm whose running time is $O(nd\lceil \log k \rceil)$ instead of $O(ndk)$, as given by a naive application of the method. With a few interesting modifications, this idea also works in the colorful setting. This seems to be the first instance of using Sarkaria's method with special lifting vectors, and we hope that this will prove useful for further studies on Tverberg's theorem and related problems.

Outline of the paper. We describe our extension of Sarkaria's technique in Section 2 and then use it in combination with a result from Section 3 to prove the no-dimensional Tverberg result. In Section 3, we expand upon the details of an averaging argument that is useful for the Tverberg result. Section 4 is devoted to the algorithm for computing the Tverberg partition. We conclude in Section 5 with some observations and open questions. The results for the colorful setting and the generalized ham-sandwich theorem are presented in the extended version [8].

2 Tensor product and no-dimensional Tverberg Theorem

In this section, we prove a no-dimensional Tverberg result. Let $\mathbb{D}(\cdot)$ denote the diameter of any point set in $\mathbb{R}^d$. Let $P \subset \mathbb{R}^d$ be our given set of n points. We assume for simplicity that the centroid of P, that we denote by $c(P)$, coincides with the origin $\mathbf{0}$, so that $\sum_{x\in P} x = \mathbf{0}$. For ease of presentation, we denote the origin by $\mathbf{0}$ in all dimensions, as long as there is no danger of confusion. Also, we write $\langle \cdot, \cdot \rangle$ for the usual scalar product between two vectors in the appropriate dimension, and $[n]$ for the set $\{1, \dots, n\}$.

Tensor product. Let $x = (x_1, \dots, x_d) \in \mathbb{R}^d$ and $y = (y_1, \dots, y_m) \in \mathbb{R}^m$ be any two vectors. The *tensor product* $\otimes$ is the operation that takes x and y to the dm-dimensional vector $x \otimes y$ whose ij-th component is $x_i y_j$. Easy calculations show that for any $x, x' \in \mathbb{R}^d, y, y' \in \mathbb{R}^m$, the operator $\otimes$ satisfies: (i) $x \otimes y + x' \otimes y = (x + x') \otimes y$; (ii) $x \otimes y + x \otimes y' = x \otimes (y + y')$; and (iii) $\langle x \otimes y, x' \otimes y' \rangle = \langle x, x' \rangle \langle y, y' \rangle$. By (iii), the L_2-norm $\|x \otimes y\|$ of the tensor product $x \otimes y$ is exactly $\|x\|\|y\|$. For any set of vectors $X = \{x_1, x_2, \dots\}$ in $\mathbb{R}^d$ and any m-dimensional vector $q \in \mathbb{R}^m$, we denote by $X \otimes q$ the set of tensor products $\{x_1 \otimes q, x_2 \otimes q, \dots\} \subset \mathbb{R}^{dm}$. Throughout this paper, all distances will be in the L_2-norm.

A set of lifting vectors. We generalize the tensor construction that was used by Sarkaria to prove the Tverberg theorem [21]. For this, we provide a way to construct a set of k vectors $\{q_1, \dots, q_k\}$ that we use to create tensor products. The motivation behind the precise choice of these vectors will be explained a little later in this section. Let $\mathcal{G}$ be an (undirected) simple, connected graph of k nodes. Let $\|\mathcal{G}\|$ denote the number of edges in $\mathcal{G}$, $\Delta(\mathcal{G})$ denote the maximum degree of any node in $\mathcal{G}$, and $\mathrm{diam}(\mathcal{G})$ denote the diameter of $\mathcal{G}$, i.e., the maximum length of a shortest path between a pair of vertices in $\mathcal{G}$.

We orient the edges of $\mathcal{G}$ in an arbitrary manner to obtain a directed graph. We use this directed version of $\mathcal{G}$ to define a set of k vectors $\{q_1, \dots, q_k\}$ in $\|\mathcal{G}\|$ dimensions. This is done as follows: each vector q_i corresponds to a unique node v_i of $\mathcal{G}$. Each coordinate

position of the vectors corresponds to a unique edge of $\mathcal{G}$. If v_iv_j is a directed edge of $\mathcal{G}$, then q_i contains a 1 and q_j contains a -1 in the corresponding coordinate position. The remaining co-ordinates are zero. That means, the vectors $\{q_1, \ldots, q_k\}$ are in $\mathbb{R}^{\|\mathcal{G}\|}$. Also, $\sum_{i=1}^{k} q_i = \mathbf{0}$. It can be verified that this is the unique linear dependence (up to scaling) between the vectors for any choice of edge orientations of $\mathcal{G}$. This means that the rank of the matrix with the q_is as the rows is $k-1$. It can be verified that:

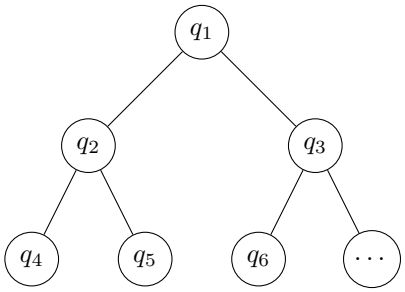
▷ Claim 2.1. For each vertex v_i, the squared norm $\|q_i\|^2$ is the degree of v_i. For $i \neq j$, the dot product $\langle q_i, q_j \rangle$ is -1 if v_iv_j is an edge in $\mathcal{G}$, and 0 otherwise.

An immediate application of Claim 2.1 and property (iii) of the tensor product is that for any set of k vectors $\{u_1, \ldots, u_k\}$, each of the same dimension, the following relation holds:

$$\begin{aligned}
&\left\| \sum_{i=1}^{k} u_i \otimes q_i \right\|^2 = \sum_{i=1}^{k}\sum_{j=1}^{k} \langle u_i \otimes q_i, u_j \otimes q_j \rangle = \sum_{i=1}^{k}\sum_{j=1}^{k} \langle u_i, u_j \rangle \langle q_i, q_j \rangle \\
&= \sum_{i=1}^{k} \langle u_i, u_i \rangle \langle q_i, q_i \rangle + 2 \sum_{1 \leq i < j \leq k} \langle u_i, u_j \rangle \langle q_i, q_j \rangle = \sum_{i=1}^{k} \|u_i\|^2 \|q_i\|^2 - 2 \sum_{v_iv_j \in E} \langle u_i, u_j \rangle \\
&= \sum_{v_iv_j \in E} \|u_i - u_j\|^2, \qquad (1)
\end{aligned}$$

where E is the set of edges of $\mathcal{G}$.[1]

As an example, such a set of vectors can be formed by taking $\mathcal{G}$ as a balanced binary tree with k nodes, and orienting the edges away from the root. Let q_1 correspond to the root. A simple instance of the vectors is shown below:



The vectors in the figure above can be represented as the matrix

$$\begin{pmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \\ \cdots \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & \ldots \\ -1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & \ldots \\ 0 & -1 & 0 & 0 & 1 & 1 & 0 & 0 & \ldots \\ 0 & 0 & -1 & 0 & 0 & 0 & 1 & 1 & \ldots \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & \ldots \\ & & & \cdots & & & & & \end{pmatrix}$$

where the i-th row of the matrix corresponds to vector q_i. As $\|\mathcal{G}\| = k-1$, each vector is in $\mathbb{R}^{k-1}$. The norm $\|q_i\|$ is one of $\sqrt{2}$, $\sqrt{3}$, or 1, depending on whether v_i is the root, an internal node with two children, or a leaf, respectively. The height of $\mathcal{G}$ is $\lceil \log k \rceil$ and the maximum degree is $\Delta(\mathcal{G}) = 3$.

[1] We note that this identity is very similar to the Laplacian quadratic form that is used in spectral graph theory; see, e.g., the lecture notes by Spielman [23] for more information.

Lifting the point set. Let $P = \{p_1, \ldots, p_n\} \subset \mathbb{R}^d$. Our goal is to find a (relaxed) Tverberg partition of P into k sets. For this, we first pick a graph $\mathcal{G}$ with k vertices, as in the previous paragraph, and we derive a set of k lifting vectors $\{q_1, \ldots, q_k\}$ from $\mathcal{G}$. Then, we lift each point of P to a set of vectors in $d\|\mathcal{G}\|$ dimensions, by taking tensor products with the vectors $\{q_1, \ldots, q_k\}$. More precisely, for $a \in [n]$ and $j = 1, \ldots, k$, let $p_{a,j} = p_a \otimes q_j \in \mathbb{R}^{d\|\mathcal{G}\|}$. For $a \in [n]$, we let $P_a = \{p_{a,1}, \ldots, p_{a,k}\}$ be the lifted points obtained from p_a. We have, $\|p_{a,j}\| = \|q_j\|\|p_a\| \leq \sqrt{\Delta(\mathcal{G})}\|p_a\|$. By the bi-linear properties of the tensor product, we have

$$c(P_a) = \frac{1}{k}\sum_{j=1}^{k}(p_a \otimes q_j) = \frac{1}{k}\left(p_a \otimes \left(\sum_{j=1}^{k} q_j\right)\right) = \frac{1}{k}(p_a \otimes \mathbf{0}) = \mathbf{0},$$

so the centroid $c(P_a)$ coincides with the origin, for $a \in [n]$.

The next lemma contains the technical core of our argument. It shows how to use the lifted point sets to derive a useful partition of P into k subsets of prescribed sizes. We defer its proof to Section 3.

▶ **Lemma 2.2.** *Let $P = \{p_1, \ldots, p_n\}$ be a set of n points in $\mathbb{R}^d$ satisfying $\sum_{i=1}^{p} p_i = \mathbf{0}$. Let $P_1, \ldots, P_n$ denote the point sets obtained by lifting each $p \in P$ using the vectors $\{q_1, \ldots, q_k\}$.*

For any choice of positive integers $r_1, \ldots, r_k$ that satisfy $\sum_{i=1}^{k} r_i = n$, there is a partition $T_1, \ldots, T_k$ of P with $|T_1| = r_1, |T_2| = r_2, \ldots, |T_k| = r_k$ such that the centroid of the set of lifted points $T := \{T_1 \otimes q_1 \cup \cdots \cup T_k \otimes q_k\}$ (this set is also a traversal of $P_1, \ldots, P_n$) has distance less than $\delta = \sqrt{\frac{\Delta(\mathcal{G})}{2(n-1)}}\mathbb{D}(P)$ from the origin $\mathbf{0}$.

The bound is better for the case $n = rk$ and $r_1 = \cdots = r_k = \frac{n}{k}$. There exists a partition $T_1, \ldots, T_k$ of P with $|T_1| = |T_2| = \cdots = |T_k| = r$ such that the centroid of $T := \{T_1 \otimes q_1 \cup \cdots \cup T_k \otimes q_k\}$ has distance less than $\gamma = \sqrt{\frac{\|\mathcal{G}\|}{k(n-1)}}\mathbb{D}(P)$ from the origin $\mathbf{0}$.

Using Lemma 2.2, we show that there is a ball of bounded radius that intersects the convex hull of each T_i. Let $\alpha_1 = \frac{r_1}{n}, \ldots, \alpha_k = \frac{r_k}{n}$ be positive real numbers. The centroid of T, $c(T)$, can be written as

$$c(T) = \frac{1}{n}\sum_{i=1}^{k}\sum_{x\in T_i} x \otimes q_i = \sum_{i=1}^{k}\frac{1}{n}\left(\sum_{x\in T_i} x\right)\otimes q_i = \sum_{i=1}^{k}\frac{r_i}{n}\left(\frac{1}{r_i}\sum_{x\in T_i} x\right)\otimes q_i = \sum_{i=1}^{k}\alpha_i c_i \otimes q_i,$$

where $c_i = c(T_i)$ denotes the centroid of T_i, for $i \in [k]$. Using Equation (1),

$$\|c(T)\|^2 = \left\|\sum_{i=1}^{k}\alpha_i c_i \otimes q_i\right\|^2 = \sum_{v_iv_j\in E}\|\alpha_i c_i - \alpha_j c_j\|^2. \tag{2}$$

Let $x_1 = \alpha_1 c_1, x_2 = \alpha_2 c_2, \ldots, x_k = \alpha_k c_k$. Then

$$\sum_{i=1}^{k} x_i = \sum_{i=1}^{k}\alpha_i c_i = \sum_{i=1}^{k}\frac{r_i}{n}\left(\frac{1}{r_i}\sum_{p\in T_i} p\right) = \frac{1}{n}\sum_{j=1}^{n} p_j = \mathbf{0},$$

so the centroid of $\{x_1, \ldots, x_k\}$ coincides with the origin. Using $\|c(T)\| < \delta$ and Equation (2),

$$\sum_{v_iv_j\in E}\|x_i - x_j\|^2 = \sum_{v_iv_j\in E}\|\alpha_i c_i - \alpha_j c_j\|^2 < \delta^2.$$

We bound the distance from x_1 to every other x_j. For each $i \in [k]$, we associate to x_i the node v_i in $\mathcal{G}$. Let the shortest path from v_1 to v_j in $\mathcal{G}$ be denoted by $(v_1, v_{i_1}, v_{i_2}, \ldots, v_{i_z}, v_j)$. This path has length at most $\operatorname{diam}(\mathcal{G})$. Using the triangle inequality and the Cauchy-Schwarz inequality,

$$\begin{aligned}\|x_1 - x_j\| &\le \|x_1 - x_{i_1}\| + \|x_{i_1} - x_{i_2}\| + \cdots + \|x_{i_z} - x_j\| \\ &\le \sqrt{\operatorname{diam}(\mathcal{G})}\sqrt{\|x_1 - x_{i_1}\|^2 + \|x_{i_1} - x_{i_2}\|^2 + \cdots + \|x_{i_z} - x_j\|^2} \\ &\le \sqrt{\operatorname{diam}(\mathcal{G})}\sqrt{\sum_{v_iv_j \in E} \|x_i - x_j\|^2} < \sqrt{\operatorname{diam}(\mathcal{G})}\delta. \end{aligned} \tag{3}$$

Therefore, the ball of radius $\beta := \sqrt{\operatorname{diam}(\mathcal{G})}\delta$ centered at x_1 covers the set $\{x_1, \ldots, x_k\}$. That means, the ball covers the convex hull of $\{x_1, \ldots, x_k\}$ and in particular contains the origin. Using triangle inequality, the ball of radius 2β centered at the origin contains $\{x_1, \ldots, x_k\}$. Then the norm of each x_i is at most 2β which implies that the norm of each c_i is at most $2\beta/\alpha_i$. Therefore, the ball of radius $\frac{2\beta}{\min_i \alpha_i} = \frac{2n\sqrt{\operatorname{diam}(\mathcal{G})}\delta}{\min_i r_i}$ centered at $\mathbf{0}$ contains the set $\{c_1, \ldots, c_k\}$. Substituting the value of δ from Lemma 2.2, the ball of radius

$$\frac{2n\sqrt{\operatorname{diam}(\mathcal{G})}}{\min_i r_i}\sqrt{\frac{\Delta(\mathcal{G})}{2(n-1)}}\mathbb{D}(P) = \frac{n\mathbb{D}(P)}{\min_i r_i}\sqrt{\frac{2\operatorname{diam}(\mathcal{G})\Delta(\mathcal{G})}{n-1}}$$

centered at $\mathbf{0}$ covers the set $\{c_1, \ldots, c_k\}$.

Optimizing the choice of $\mathcal{G}$. The radius of the ball has a term $\sqrt{\operatorname{diam}(\mathcal{G})\Delta(\mathcal{G})}$ that depends on the choice of $\mathcal{G}$. For a path graph this term has value $\sqrt{(k-1)2}$. For a star graph, that is, a tree with one root and $k-1$ children, this is $\sqrt{k-1}$. If $\mathcal{G}$ is a balanced s-ary tree, then the Cauchy-Schwarz inequality in Equation (3) can be modified to replace $\operatorname{diam}(\mathcal{G})$ by the height of the tree. Then the term is $\sqrt{\lceil \log_s k \rceil (s+1)}$ which is minimized for $s = 4$. The radius bound for this choice of $\mathcal{G}$ is $\frac{n\mathbb{D}(P)}{\min_i r_i}\sqrt{\frac{10\lceil \log_4 k \rceil}{n-1}}$ as claimed in Theorem 1.1.

Balanced partition. For the case $n = rk$ and $r_1 = \cdots = r_k = r$, we give a better bound for the radius of the ball containing the centroids $c_1, \ldots, c_k$. In this case we have $\alpha_1 = \alpha_2 = \cdots = \alpha_k = \frac{r}{n} = \frac{1}{k}$. Then Equation (2) is

$$\|c(T)\|^2 = \sum_{v_iv_j \in E} \|\alpha_i c_i - \alpha_j c_j\|^2 = \frac{1}{k^2}\sum_{v_iv_j \in E} \|c_i - c_j\|^2.$$

Since $\|c(T)\| < \gamma$, we get

$$\sum_{v_iv_j \in E} \|c_i - c_j\|^2 < k^2\gamma^2. \tag{4}$$

Similar to the general case, we bound the distance from c_1 to any other centroid c_j. For each i, we associate to c_i the node v_i in $\mathcal{G}$. There is a path of length at most $\operatorname{diam}(\mathcal{G})$ from v_1 to any other node. Using the Cauchy-Schwarz inequality and substituting the value of γ, we get

$$\begin{aligned}\|c_1 - c_j\| &\le \sqrt{\operatorname{diam}(\mathcal{G})}\sqrt{\sum_{v_iv_j \in E} \|c_i - c_j\|^2} < \sqrt{\operatorname{diam}(\mathcal{G})}k\gamma = \sqrt{\frac{\operatorname{diam}(\mathcal{G})\|\mathcal{G}\|}{k(n-1)}}k\mathbb{D}(P) \\ &= \sqrt{\frac{k}{n-1}}\sqrt{\operatorname{diam}(\mathcal{G})\|\mathcal{G}\|}\mathbb{D}(P). \end{aligned} \tag{5}$$

Therefore, a ball of radius $\sqrt{\frac{k}{n-1}}\sqrt{\operatorname{diam}(\mathcal{G})\|\mathcal{G}\|}\mathbb{D}(P)$ centered at c_1 contains the set $c_1, \ldots, c_k$. The factor $\sqrt{\operatorname{diam}(\mathcal{G})\|\mathcal{G}\|}$ is minimized when $\mathcal{G}$ is a star graph, which is a tree. We can replace the term $\operatorname{diam}(\mathcal{G})$ by the height of the tree. Then the ball containing $c_1, \ldots, c_k$ has radius $\sqrt{\frac{k(k-1)}{n-1}}\mathbb{D}(P)$, as claimed in Theorem 1.1.

As balanced as possible. When k does not divide n, but we still want a balanced partition, we take any subset of $n_0 = k\lfloor \frac{n}{k} \rfloor$ points of P and get a balanced Tverberg partition on the subset. Then we add the removed points one by one to the sets of the partition, adding at most one point to each set.

As shown above, there is a ball of radius less than $\sqrt{\frac{k(k-1)}{n_0-1}}\mathbb{D}(P)$ that intersects the convex hull of each set in the partition. Noting that $\frac{1}{\sqrt{n_0-1}} \leq \sqrt{\frac{k+2}{k}}\frac{1}{\sqrt{n-1}}$, a ball of radius less than $\sqrt{\frac{(k+2)(k-1)}{(n-1)}}\mathbb{D}(P)$ intersects the convex hull of each set of the partition.

3 Existence of a desired partition

This section is dedicated to the proof of Lemma 2.2. Like Adiprasito et al. [1], we use an averaging argument. More precisely, we bound the average norm δ of the centroid of the lifted points $\{T_1 \otimes q_1 \cup \cdots \cup T_k \otimes q_k\}$ over all partitions of P of the form $T_1, \ldots, T_k$, for which the sets in the partition have sizes $r_1, \ldots, r_k$ respectively, with $\sum_{i=1}^{k} r_i = n$.

Each such partition can be interpreted as a traversal of the lifted point sets $P_1, \ldots, P_n$ that contains r_i points lifted with q_i for $i \in [k]$. Thus, consider any traversal of this type $X = \{x_1, \ldots, x_n\}$ of $P_1, \ldots, P_n$, where $x_a \in P_a$, for $a \in [n]$. The centroid of X is $c(X) = \frac{\sum_{a=1}^{n} x_a}{n}$. We bound the expectation $n^2\mathbb{E}\left(\|c(X)\|^2\right) = \mathbb{E}\left(\left\|\sum_{a=1}^{n} x_a\right\|^2\right)$, over all possible traversals X. By the linearity of expectation, $\mathbb{E}\left(\left\|\sum_{a=1}^{n} x_a\right\|^2\right)$ can be written as

$$\mathbb{E}\left(\sum_{a=1}^{n}\|x_a\|^2 + \sum_{\substack{a,b\in[n]\\ a<b}} 2\langle x_a, x_b\rangle\right) = \mathbb{E}\left(\sum_{a=1}^{n}\|x_a\|^2\right) + 2\mathbb{E}\left(\sum_{\substack{a,b\in[n]\\ a<b}}\langle x_a, x_b\rangle\right).$$

We next find the coefficient of each term of the form $\|x_a\|^2$ and $\langle x_a, x_b\rangle$ in the expectation. Using the multinomial coefficient, the total number of traversals X is $\binom{n}{r_1, r_2, \ldots, r_k} = \frac{n!}{r_1!r_2!\cdots r_k!}$. Furthermore, for any lifted point $x_a = p_{a,j}$, the number of traversals X with $p_{a,j} \in X$ is $\binom{n-1}{r_1, \ldots, r_j - 1, \ldots, r_k} = \frac{(n-1)!}{r_1!\cdots(r_j-1)!\cdots r_k!}$. So the coefficient of $\|p_{a,j}\|^2$ is $\frac{\frac{(n-1)!}{r_1!\cdots(r_j-1)!\cdots r_k}}{\frac{n!}{r_1!\cdots r_k!}} = \frac{r_j}{n}$. Similarly, for any pair of points $(x_a, x_b) = (p_{a,i}, p_{b,j})$, there are two cases in which they appear in the same traversal: first, if $i = j$, the number of traversals is $\frac{(n-2)!}{r_1!\cdots(r_i-2)!\cdots r_k!}$. The coefficient of $\langle p_{a,i}, p_{b,j}\rangle$ in the expectation is hence $\frac{r_i(r_i-1)}{n(n-1)}$. Second, if $i \neq j$, the number of traversals is calculated to be $\frac{(n-2)!}{r_1!\cdots(r_i-1)!\cdots(r_j-1)!\cdots r_k!}$. The coefficient of $\langle p_{a,i}, p_{b,j}\rangle$ is $\frac{r_i r_j}{n(n-1)}$. Substituting the coefficients, we bound the expectation as

$$\mathbb{E}\left(\sum_{a=1}^{n}\|x_a\|^2\right)+2\mathbb{E}\left(\sum_{\substack{a,b\in[n]\\a<b}}\langle x_a,x_b\rangle\right)=\sum_{a=1}^{n}\sum_{j=1}^{k}\|p_{a,j}\|^2\frac{r_j}{n}$$
$$+2\sum_{\substack{a,b\in[n]\\a<b}}\left(\sum_{j=1}^{k}\langle p_{a,j},p_{b,j}\rangle\frac{r_j(r_j-1)}{n(n-1)}+\sum_{\substack{i,j\in[k]\\i\neq j}}\langle p_{a,i},p_{b,j}\rangle\frac{r_ir_j}{n(n-1)}\right)$$
$$=\sum_{j=1}^{k}\frac{r_j}{n}\sum_{a=1}^{n}\|p_{a,j}\|^2+\frac{2}{n(n-1)}\sum_{\substack{a,b\in[n]\\a<b}}\left(\sum_{i,j\in[k]}\langle p_{a,i},p_{b,j}\rangle r_ir_j-\sum_{j=1}^{k}\langle p_{a,j},p_{b,j}\rangle r_j\right)$$
$$=\sum_{j=1}^{k}r_j\left(\frac{1}{n}\sum_{a=1}^{n}\|p_{a,j}\|^2\right)+\sum_{\substack{a,b\in[n]\\a<b}}\sum_{i,j\in[k]}\frac{2\langle p_{a,i},p_{b,j}\rangle r_ir_j}{n(n-1)}-\sum_{\substack{a,b\in[n]\\a<b}}\sum_{j=1}^{k}\frac{2\langle p_{a,j},p_{b,j}\rangle r_j}{n(n-1)}.$$

We bound the value of each of the three terms individually to get an upper bound on the value of the expression. The first term can be bounded as

$$\sum_{j=1}^{k}r_j\left(\frac{1}{n}\sum_{a=1}^{n}\|p_{a,j}\|^2\right)=\frac{1}{n}\sum_{j=1}^{k}r_j\left(\sum_{a=1}^{n}\|p_a\|^2\|q_j\|^2\right)=\frac{1}{n}\left(\sum_{j=1}^{k}r_j\|q_j\|^2\right)\sum_{a=1}^{n}\|p_a\|^2$$
$$\leq\frac{1}{n}\left(\Delta(\mathcal{G})\sum_{j=1}^{k}r_j\right)\sum_{a=1}^{n}\|p_a\|^2=\frac{1}{n}\left(\Delta(\mathcal{G})n\right)\sum_{a=1}^{n}\|p_a\|^2<\Delta(\mathcal{G})\left(\frac{n\mathbb{D}(P)^2}{2}\right),$$

where we have made use of Claim 2.1 and the fact that $\sum_{a=1}^{n}\|p_a\|^2<\frac{n\mathbb{D}(P)^2}{2}$ (see [1, Lemma 4.1]). The second term can be re-written as

$$\sum_{\substack{a,b\in[n]\\a<b}}\sum_{i,j\in[k]}\frac{2\langle p_{a,i},p_{b,j}\rangle r_ir_j}{n(n-1)}=\sum_{i,j\in[k]}\frac{2r_ir_j}{n(n-1)}\left(\sum_{\substack{a,b\in[n]\\a<b}}\langle p_{a,i},p_{b,j}\rangle\right)$$
$$=\sum_{i,j\in[k]}\frac{2r_ir_j}{n(n-1)}\left(\sum_{\substack{a,b\in[n]\\a<b}}\langle p_a\otimes q_i,p_b\otimes q_j\rangle\right)=\sum_{i,j\in[k]}\frac{2r_ir_j}{n(n-1)}\left(\sum_{\substack{a,b\in[n]\\a<b}}\langle p_a,p_b\rangle\langle q_i,q_j\rangle\right)$$
$$=\left(\sum_{i,j\in[k]}\frac{2\langle q_i,q_j\rangle r_ir_j}{n(n-1)}\right)\cdot\sum_{\substack{a,b\in[n]\\a<b}}\langle p_a,p_b\rangle=\left(\sum_{(v_i,v_j)\in E}\frac{2(r_i-r_j)^2}{n(n-1)}\right)\cdot\sum_{\substack{a,b\in[n]\\a<b}}\langle p_a,p_b\rangle\leq 0,$$

where we have again made use of Claim 2.1. We also used $c(P)=\mathbf{0}$ to bound the term $\sum_{a,b,\in[n],a<b}\langle p_a,p_b\rangle=-\frac{1}{2}\sum_{a=1}^{n}\|p_a\|^2<0$. The second term is non-positive and therefore can be removed since the total expectation is always non-negative. The third term is

$$\sum_{\substack{a,b\in[n]\\a<b}}\sum_{j=1}^{k}\frac{-2\langle p_{a,j},p_{b,j}\rangle r_j}{n(n-1)}=\sum_{\substack{a,b\in[n]\\a<b}}\sum_{j=1}^{k}\frac{-2\left\langle p_a\otimes q_j,p_b\otimes q_j\right\rangle r_j}{n(n-1)}$$

$$=\sum_{\substack{a,b\in[n]\\a<b}}\sum_{j=1}^{k}\frac{-2\langle p_a,p_b\rangle\|q_j\|^2 r_j}{n(n-1)}=\left(\sum_{j=1}^{k}\|q_j\|^2 r_j\right)\left(\sum_{\substack{a,b\in[n]\\a<b}}\frac{-2\langle p_a,p_b\rangle}{n(n-1)}\right)$$

$$<\left(\sum_{j=1}^{k}\|q_j\|^2 r_j\right)\left(\frac{n\mathbb{D}(P)^2}{2n(n-1)}\right)=\left(\sum_{j=1}^{k}\|q_j\|^2 r_j\right)\frac{\mathbb{D}(P)^2}{2(n-1)}<\frac{n\Delta(\mathcal{G})\mathbb{D}(P)^2}{2(n-1)}.$$

Collecting the three terms, the expression is upper bounded by

$$\frac{\mathbb{D}(P)^2\Delta(\mathcal{G})n}{2}+\frac{\mathbb{D}(P)^2\Delta(\mathcal{G})n}{2(n-1)}=\frac{\mathbb{D}(P)^2\Delta(\mathcal{G})n}{2}\left(1+\frac{1}{n-1}\right)=\frac{\mathbb{D}(P)^2\Delta(\mathcal{G})n^2}{2(n-1)},$$

which bounds the expectation by $\frac{1}{n^2}\left(\frac{\mathbb{D}(P)^2\Delta(\mathcal{G})n^2}{2(n-1)}\right)=\frac{\mathbb{D}(P)^2\Delta(\mathcal{G})}{2(n-1)}$. This shows that there is a traversal such that its centroid has norm less than $\mathbb{D}(P)\sqrt{\frac{\Delta(\mathcal{G})}{2(n-1)}}$, as claimed in Lemma 2.2.

Balanced case. For the case that n is a multiple of k, and $r_1=\cdots=r_k=\frac{n}{k}=r$, the upper bound can be improved: the first term in the expectation is

$$\sum_{j=1}^{k}r_j\left(\frac{1}{n}\sum_{a=1}^{n}\|p_{a,j}\|^2\right)=\frac{r}{n}\sum_{j=1}^{k}\sum_{a=1}^{n}\|p_{a,j}\|^2=\frac{r}{n}\sum_{j=1}^{k}\sum_{a=1}^{n}\|p_a\|^2\|q_j\|^2$$

$$=\frac{r}{n}\left(\sum_{j=1}^{k}\|q_j\|^2\right)\sum_{a=1}^{n}\|p_a\|^2=\frac{r}{n}2\|\mathcal{G}\|\sum_{a=1}^{n}\|p_a\|^2<\frac{r}{n}2\|\mathcal{G}\|\left(\frac{n\mathbb{D}(P)^2}{2}\right)\leq r\|\mathcal{G}\|\mathbb{D}(P)^2,$$

The second term is zero, and the third term is less than

$$\left(\sum_{j=1}^{k}\|q_j\|^2 r_j\right)\frac{\mathbb{D}(P)^2}{2(n-1)}=r\left(\sum_{j=1}^{k}\|q_j\|^2\right)\frac{\mathbb{D}(P)^2}{2(n-1)}=2r\|\mathcal{G}\|\frac{\mathbb{D}(P)^2}{2(n-1)}=\frac{r\|\mathcal{G}\|\mathbb{D}(P)^2}{(n-1)}.$$

The expectation is upper bounded as

$$n^2\mathbb{E}\left(\|c(X)\|^2\right)<r\|\mathcal{G}\|\mathbb{D}(P)^2+\frac{r\|\mathcal{G}\|\mathbb{D}(P)^2}{(n-1)}$$

$$\implies\mathbb{E}\left(\|c(X)\|^2\right)<\frac{r\|\mathcal{G}\|\mathbb{D}(P)^2}{n^2}\left(1+\frac{1}{n-1}\right)=\frac{r\|\mathcal{G}\|\mathbb{D}(P)^2}{n(n-1)}=\frac{\|\mathcal{G}\|\mathbb{D}(P)^2}{k(n-1)},$$

which shows that there is at least one balanced traversal X whose centroid has norm less than $\sqrt{\frac{\|\mathcal{G}\|}{k(n-1)}}\mathbb{D}(P)$, as claimed in Lemma 2.2.

4 Computing the Tverberg partition

We now give a deterministic algorithm to compute no-dimensional Tverberg partitions. The algorithm is based on the method of conditional expectations. First, in Section 4.1 we give an algorithm for the general case when the sets in the partitions are constrained to have given sizes $r_1,\ldots,r_k$. The choice of $\mathcal{G}$ is crucial for the algorithm.

The balanced case of $r_1 = \cdots = r_k$ has a better radius bound and uses a different graph $\mathcal{G}$. The algorithm for the general case also extends to the balanced case with a small modification, that we discuss in Section 4.2. We get the same runtime in either case:

▶ **Theorem 4.1.** *Given a set of n points $P \subset \mathbb{R}^d$, and any choice of k positive integers $r_1, \ldots, r_k$ that satisfy $\sum_{i=1}^{k} r_i = n$, a no-dimensional Tverberg k-partition of P with the sets of the partition having sizes $r_1, \ldots, r_k$ can be computed in time $O(nd\lceil \log k \rceil)$.*

4.1 Algorithm for the general case

The input is a set of n points $P \subset \mathbb{R}^d$ and k positive integers $r_1, \ldots, r_k$ satisfying $\sum_{i=1}^{k} r_i = n$. We use the tensor product construction from Section 2 that are derived from a graph $\mathcal{G}$. Each point of P is lifted implicitly using the vectors $\{q_1, \ldots, q_k\}$ to get the set $\{P_1, \ldots, P_n\}$. We then compute a traversal of $\{P_1, \ldots, P_n\}$ using the method of conditional expectations [2], the details of which can be found below. Grouping the points of the traversal according to the lifting vectors used gives us the required partition. We remark that in our algorithm we do not explicitly lift any vector using the tensor product, thereby avoiding costs associated with working on vectors in $d\|\mathcal{G}\|$ dimensions.

We now describe a procedure to find a traversal that corresponds to a desired partition of P. We go over the points in $\{P_1, \ldots, P_n\}$ iteratively in reverse order and find the traversal $Y = (y_1 \in P_1, \ldots, y_n \in P_n)$ point by point. More precisely, we determine y_n in the first step, then y_{n-1} in the second step, and so on. In the first step, we go over all points of P_n and select any point $y_n \in P_n$ that satisfies $\mathbb{E}\left(c\|(x_1, x_2, \ldots, x_{n-1}, y_n)\|^2\right) \leq \mathbb{E}\left(c\|(x_1, x_2, \ldots, x_{n-1}, x_n)\|^2\right)$. For the general step, suppose we have already selected the points $\{y_{s+1}, y_{s+2}, \ldots, y_n\}$. To determine y_s, we choose any point from P_s that achieves

$$\mathbb{E}\left(\|c(x_1, x_2, \ldots, x_{s-1}, y_s, y_{s+1}, \ldots, y_n)\|^2\right) \leq \mathbb{E}\left(\|c(x_1, x_2, \ldots, x_s, y_{s+1}, \ldots, y_n)\|^2\right). \quad (6)$$

The last step gives the required traversal. We expand $\mathbb{E}(\|c(x_1, x_2, \ldots, x_{s-1}, y_s, \ldots, y_n)\|^2)$ to

$$\mathbb{E}\left(\left\|\frac{1}{n}\left(\sum_{i=1}^{s-1} x_i + \sum_{i=s}^{n} y_i\right)\right\|^2\right) = \frac{1}{n^2}\mathbb{E}\left(\left\|\left(\sum_{i=1}^{s-1} x_i + \sum_{i=s+1}^{n} y_i\right) + y_s\right\|^2\right)$$

$$= \frac{1}{n^2}\left(\mathbb{E}\left(\left\|\sum_{i=1}^{s-1} x_i + \sum_{i=s+1}^{n} y_i\right\|^2\right) + \|y_s\|^2 + 2\left\langle y_s, \mathbb{E}\left(\sum_{i=1}^{s-1} x_i + \sum_{i=s+1}^{n} y_i\right)\right\rangle\right)$$

$$= \frac{1}{n^2}\left(\mathbb{E}\left(\left\|\sum_{i=1}^{s-1} x_i + \sum_{i=s+1}^{n} y_i\right\|^2\right) + \|y_s\|^2 + 2\left\langle y_s, \mathbb{E}\left(\sum_{i=1}^{s-1} x_i\right) + \sum_{i=s+1}^{n} y_i\right\rangle\right).$$

We pick a y_s for which $\mathbb{E}(\|c(x_1, x_2, \ldots, x_{s-1}, y_s, \ldots, y_n)\|^2)$ is at most the average over all choices of $y_s \in P_s$. As the term $\mathbb{E}\left(\left\|\sum_{i=1}^{s-1} x_i + \sum_{i=s+1}^{n} y_i\right\|^2\right)$ is constant over all choices of y_s, and the factor $\frac{1}{n^2}$ is constant, we can remove them from consideration. We are left with

$$\|y_s\|^2 + 2\left\langle y_s, \mathbb{E}\left(\sum_{i=1}^{s-1} x_i\right) + \sum_{i=s+1}^{n} y_i\right\rangle = \|y_s\|^2 + 2\left\langle y_s, \mathbb{E}\left(\sum_{i=1}^{s-1} x_i\right)\right\rangle + 2\langle y_s, \sum_{i=s+1}^{n} y_i\rangle. \quad (7)$$

Let $y_s = p_s \otimes q_i$. The first term is $\|y_s\|^2 = \|p_s \otimes q_i\|^2 = \|p_s\|^2\|q_i\|^2$. Let $r'_1, \ldots, r'_k$ be the number of elements of $T_1, \ldots, T_k$ that are yet to be determined. In the beginning, $r'_i = r_i$ for each i. Using the coefficients from Section 3, $\mathbb{E}\left(\sum_{i=1}^{s-1} x_i\right)$ can be written as

$$\mathbb{E}\left(\sum_{i=1}^{s-1} x_i\right) = \sum_{i=1}^{s-1}\sum_{j=1}^{k} p_{i,j}\frac{r'_j}{s-1} = \sum_{j=1}^{k}\frac{r'_j}{s-1}\sum_{i=1}^{s-1} p_{i,j} = \sum_{j=1}^{k}\frac{r'_j}{s-1}\sum_{i=1}^{s-1} p_i \otimes q_j$$
$$= \frac{1}{s-1}\sum_{j=1}^{k} r'_j\left(\sum_{i=1}^{s-1} p_i\right)\otimes q_j = \left(\frac{1}{s-1}\sum_{i=1}^{s-1} p_i\right)\otimes\left(\sum_{j=1}^{k} r'_j q_j\right) = c_{s-1}\otimes\left(\sum_{j=1}^{k} r'_j q_j\right),$$

where $c_{s-1} = \frac{\sum_{i=1}^{s-1} p_i}{s-1}$ is the centroid of the first $(s-1)$ points. Using this, the second term can be simplified as

$$2\left\langle y_s, \mathbb{E}\left(\sum_{i=1}^{s-1} x_i\right)\right\rangle = 2\left\langle p_s\otimes q_i, c_{s-1}\otimes\left(\sum_{j=1}^{k} r'_j q_j\right)\right\rangle = 2\left\langle p_s, c_{s-1}\right\rangle\left\langle q_i, \sum_{j=1}^{k} r'_j q_j\right\rangle$$
$$= 2\langle p_s, c_{s-1}\rangle\left(r'_i\|q_i\|^2 - \sum_{v_iv_j\in E} r'_j\right) = \langle p_s, c_{s-1}\rangle R_i,$$

where $R_i = 2\left(r'_i\|q_i\|^2 - \sum_{v_iv_j\in E} r'_j\right)$. The third term is

$$2\left\langle y_s, \sum_{j=s+1}^{n} y_j\right\rangle = 2\sum_{j=s+1}^{n}\langle y_s, y_j\rangle = 2\sum_{j=s+1}^{n}\left\langle p_s\otimes q_i, p_j\otimes q_{i_j}\right\rangle$$
$$= 2\sum_{j=s+1}^{n}\langle p_s, p_j\rangle\langle q_i, q_{i_j}\rangle = 2\left\langle p_s, \sum_{p\in T_i} p\|q_i\|^2 - \sum_{j:v_iv_j\in E}\sum_{p\in T_j} p\right\rangle$$
$$= \left\langle p_s, 2\left(\|q_i\|^2\sum_{p\in T_i} p - \sum_{j:v_iv_j\in E}\sum_{p\in T_j} p\right)\right\rangle = \langle p_s, U_i\rangle,$$

where $U_i = 2\left(\|q_i\|^2\sum_{p\in T_i} p - \sum_{j:v_iv_j\in E}\sum_{p\in T_j} p\right)$ and T_j represents the set of points in $p_{s+1},\ldots,p_n$ that was lifted using q_j in the traversal. Collecting the three terms, we get

$$\|p_s\|^2\|q_i\|^2 + \langle p_s, c_{s-1}\rangle R_i + \langle p_s, U_i\rangle = \alpha_s N_i + \beta_s R_i + \langle p_s, U_i\rangle, \tag{8}$$

with $N_i = \|q_i\|^2$, $\alpha_s := \|p_s\|^2$, $\beta_s := \langle p_s, c_{s-1}\rangle$. The terms α_s, β_s, p_s are fixed for iteration s.

Algorithm. For each $s \in [1,n]$, we pre-compute the prefix sums $\sum_{a=1}^{s} p_a$, and α_s and β_s. With this information, it is straightforward to compute a traversal in $O(ndk)$ time by evaluating the expression for each choice of p_s. We describe a more careful method that reduces this time to $O(nd\lceil\log k\rceil)$.

We assume that $\mathcal{G}$ is a balanced μ-ary tree. Recall that each node v_i of $\mathcal{G}$ corresponds to a vector q_i. We augment $\mathcal{G}$ with the following additional information for each node v_i:

- $N_i = \|q_i\|^2$: recall that this is the degree of v_i.
- N_i^{st}: this is the average of the N_j over all elements v_j in the subtree rooted at v_i.
- r'_i: as before, this is the number of elements of the set T_i of the partition that are yet to be determined. We initialize each $r'_i := r_i$.
- $R_i = 2\left(r'_i N_i - \sum_{v_iv_j\in E} r'_j\right)$, that is, $r'_i N_i$ minus the r'_j for each node v_j that is a neighbor of v_i in $\mathcal{G}$, times two. We initialize $R_i := 0$.

- R_i^{st}: this is the average of the R_j values over all nodes v_j in the subtree rooted at v_i. We initialize this to 0.
- T_i, u_i: as before, T_i is the set of vectors of the traversal that was lifted using q_i. u_i is the sum of the vectors of T_i. We initialize $T_i = \varnothing$ and $u_i = \mathbf{0}$.
- $U_i = 2\left(\|q_i\|^2 \sum_{p\in T_i} p - \sum_{j:v_iv_j\in E}\sum_{p\in T_j} p\right) = 2\left(u_iN_i - \sum_{v_iv_j\in E} u_j\right)$, initially $\mathbf{0}$.
- U_i^{st}: this is the average of the vectors U_j for all nodes v_j in the subtree of v_i. U^{st} is initialized as $\mathbf{0}$ for each node.

Additionally, each node contains pointers to its children and parents. N^{st}, R^{st} are initialized in one pass over $\mathcal{G}$.

In step s, we find an $i \in [k]$ for which Equation (8) has a value at most the average

$$\begin{aligned} A_s &= \frac{1}{k}\left(\sum_{i=1}^{k} \alpha_s N_i + \beta_s R_i + \langle p_s, U_i\rangle\right) = \frac{\alpha_s}{k}\sum_{i=1}^{k} N_i + \frac{\beta_s}{k}\sum_{i=1}^{k} R_i + \left\langle p_s, \frac{1}{k}\sum_{i=1}^{k} U_i\right\rangle \\ &= \alpha_s N_1^{st} + \beta_s R_1^{st} + \langle p_s, U_1^{st}\rangle, \end{aligned}$$

where v_1 is the root of $\mathcal{G}$. Then y_s satisfies Equation (6).

To find such a node v_i, we start at the root $v_1 \in \mathcal{G}$. We compute the average A_s and evaluate Equation (8) at v_1. If the value is at most A_s, we report success, setting $i = 1$. If not, then for at least one child v_m of v_1, the average for the subtree is less than A_s, that is, $\alpha_s N_m^{st} + \beta_s R_m^{st} + \langle p_s, U_m^{st}\rangle < A_s$. We scan the children of v_1 and compute the expression to find such a node v_m. We recursively repeat the procedure on the subtree rooted at v_m, and so on until we find a suitable node. There is at least one node in the subtree at v_m for which Equation (8) evaluates to less than A_s, so the procedure is guaranteed to find such a node.

Let v_i be the chosen node. We update the information stored in the nodes of the tree for the next iteration. We set

- $r_i' := r_i' - 1$ and $R_i := R_i - 2N_i$. Similarly we update the R_i values for neighbors of v_i.
- We set $T_i := T_i \cup \{p_s\}$, $u_i := u_i + p_s$ and $U_i := U_i + 2N_ip_s$. Similarly we update the U_i values for the neighbors.
- For each child of v_i and each ancestor of v_i on the path to v_1, we update R^{st} and U^{st}.

After the last step of the algorithm, we get a partition $T_1, \ldots, T_k$ of P. The set of points $\{T_1 \otimes q_1, \ldots, T_k \otimes q_k\}$ is a traversal of $\{P_1, \ldots, P_n\}$, hence using Lemma 2.2 the sets $T_1, \ldots, T_k$ form the required partition of P. This completes the description of the algorithm.

Proof of Theorem 4.1 for the general case. Computing the prefix sums and α_s, β_s takes $O(nd)$ time in total. Creating and initializing the tree takes $O(k)$ time. In step s, computing the average A_s and evaluating Equation 8 takes $O(d)$ time per node. Therefore, computing Equation 8 for the children of a node takes $O(d\mu)$ time, as $\mathcal{G}$ is a μ-ary tree. In the worst case, the search for v_i starts at the root and goes to a leaf, exploring $O(\mu\lceil\log_\mu k\rceil)$ nodes in the process and hence takes $O(d\mu\lceil\log_\mu k\rceil)$ time. For updating the tree, the information local to v_i and its neighbors can be updated in $O(d\mu)$ time. To update R^{st} and U^{st} we travel on the path to the root, which can be of length $O(\lceil\log_\mu k\rceil)$ in the worst case, and hence takes $O(d\mu\lceil\log_\mu k\rceil)$ time. There are n steps in the algorithm, each taking $O(d\mu\lceil\log_\mu k\rceil)$ time. Overall, the running time is $O(nd\mu\lceil\log_\mu k\rceil)$ which is minimized for a 3-ary tree. ◀

4.2 Algorithm for the balanced case

In the case of balanced traversals, $\mathcal{G}$ is chosen to be a star graph as was done in Section 2. Let q_1 correspond to the root of the graph and $q_2, \ldots, q_k$ correspond to the leaves. In this case the objective function $\alpha_s N_i + \beta_s R_i + \langle p_s, U_i\rangle$ from the general case can be simplified:

- for $i = 2, \ldots, k$, we have that $R_i = 2\left(r'_i\|q_i\|^2 - \sum_{v_iv_j \in E} r'_j\right) = 2\left(r'_i - r'_1\right)$. Also, we have $U_i = 2\left(\sum_{p\in T_i} p\|q_i\|^2 - \sum_{p\in T_j \wedge v_iv_j\in E} p\right) = 2\left(\sum_{p\in T_i} p - \sum_{p\in T_1} p\right)$.
- for the root v_1, $R_i = 2\left(r'_i\|q_i\|^2 - \sum_{v_iv_j \in E} r'_j\right) = 2\left((k-1)r'_1 - \sum_{j=2}^{k} r'_j\right)$. Also, we can write $U_i = 2\left(\|q_i\|^2 \sum_{p\in T_i} p - \sum_{p\in T_j \wedge v_iv_j\in E} p\right) = 2\left((k-1)\sum_{p\in T_i} p - \sum_{p\in T_2\cup\cdots\cup T_k} p\right)$.

We can augment $\mathcal{G}$ with information at the nodes just as in the general case, and use the algorithm to compute the traversal. However, this would need time $O(nd\mu\lceil\log_\mu k\rceil) = O(ndk)$ since $\mu = (k-1)$ and the height of the tree is 1. Instead, we use an auxiliary balanced ternary rooted tree $\mathcal{T}$ for the algorithm. $\mathcal{T}$ contains k nodes, each associated to one of the vectors $q_1, \ldots, q_k$ in an arbitrary fashion. We augment the tree with the same information as in the general case, but with one difference: for each node v_i, the values of R_i and U_i are updated according to the adjacency in $\mathcal{G}$ and not using the edges of $\mathcal{T}$. Then we can simply use the algorithm for the general case to get a balanced partition. The modification does not affect the complexity of the algorithm.

5 Conclusion and future work

We gave efficient algorithms for a no-dimensional version of Tverberg theorem and for a colorful counterpart. To achieve this end, we presented a refinement of Sarkaria's tensor product construction by defining vectors using a graph. The choice of the graph was different for the general- and the balanced-partition cases and also influenced the time complexity of the algorithms. It would be a worthwhile exercise to look at more applications of this refined tensor product method. Another option could be to look at non-geometric generalizations based on similar ideas.

The radius bound that we obtain for the Tverberg partition is $\sqrt{k}$ off the optimal bound in [1]. This seems to be a limitation in handling Equation (4). It is not clear if this is an artifact of using tensor product constructions. It would be interesting to explore if this factor can be brought down without compromising on the algorithmic complexity. In the general partition case, setting $r_1 = \cdots = r_k$ gives a bound that is $\sqrt{\lceil\log k\rceil}$ worse than the balanced case, so there is some scope for optimization. In the colorful case, the radius bound is again $\sqrt{k}$ off the optimal [1], but with a silver lining. The bound is proportional to $\max_i \mathbb{D}(P_i)$ in contrast to $\mathbb{D}(P_1 \cup \cdots \cup P_n)$ in [1], which is better when the colors are well-separated.

The algorithm for colorful Tverberg has a worse runtime than the regular case. The challenge in improving the runtime lies a bit with selecting an optimal graph as well as the nature of the problem itself. Each iteration in the algorithm looks at each of the permutations $\pi_1, \ldots, \pi_k$ and computes the respective expectations. The two non-zero terms in the expectation are both computed using the chosen permutation. The permutation that minimizes the first term can be determined quickly if $\mathcal{G}$ is chosen as a path graph. This worsens the radius bound by $\sqrt{k-1}$. Further, computing the other (third) term of the expectation still requires $O(k)$ updates per permutation and therefore $O(k^2)$ updates per iteration, thereby eliminating the utility of using an auxiliary tree to determine the best permutation quickly. The optimal approach for this problem is unclear at the moment.

References

1 Karim Adiprasito, Imre Bárány, and Nabil Mustafa. Theorems of Carathéodory, Helly, and Tverberg without dimension. In *Proc. 30th Annu. ACM-SIAM Sympos. Discrete Algorithms (SODA)*, pages 2350–2360, 2019.

2 Noga Alon and Joel H. Spencer. *The Probalistic method*. John Wiley & Sons, 2008.

3 Jorge L. Arocha, Imre Bárány, Javier Bracho, Ruy Fabila Monroy, and Luis Montejano. Very colorful theorems. *Discrete Comput. Geom.*, 42(2):42–154, 2009.

4 Imre Bárány. A generalization of Carathéodory's theorem. *Discrete Mathematics*, 40(2-3):141–152, 1982.

5 Imre Bárány and David G. Larman. A colored version of Tverberg's theorem. *Journal of the London Mathematical Society*, s2-45(2):314–320, 1992.

6 Imre Bárány and Shmuel Onn. Colourful linear programming and its relatives. *Mathematics of Operations Research*, 22(3):550–567, 1997.

7 Pavle Blagojević, Benjamin Matschke, and Günter Ziegler. Optimal bounds for the colored Tverberg problem. *Journal of the European Mathematical Society*, 017(4):739–754, 2015.

8 Aruni Choudhary and Wolfgang Mulzer. No-dimensional tverberg theorems and algorithms. *CoRR*, abs/1907.04284, 2019. `arXiv:1907.04284`.

9 Kenneth L. Clarkson, David Eppstein, Gary L. Miller, Carl Sturtivant, and Shang-Hua Teng. Approximating center points with iterative radon points. *Internat. J. Comput. Geom. Appl.*, 6(3):357–377, 1996.

10 Aris Filos-Ratsikas and Paul W. Goldberg. The complexity of splitting necklaces and bisecting Ham sandwiches. In *Proc. 51st Annu. ACM Sympos. Theory Comput. (STOC)*, pages 638–649, 2019.

11 Sariel Har-Peled and Mitchell Jones. Journey to the center of the point set. In *Proc. 35th Int. Sympos. Comput. Geom. (SoCG)*, pages 41:1–41:14, 2019.

12 Jesús De Loera, Xavier Goaoc, Frédéric Meunier, and Nabil Mustafa. The discrete yet ubiquitous theorems of Carathéodory, Helly, Sperner, Tucker, and Tverberg. *Bulletin of the American Mathematical Society*, 56(3):415–511, 2019.

13 Jiří Matoušek, Martin Tancer, and Uli Wagner. A geometric proof of the colored Tverberg theorem. *Discrete Comput. Geom.*, 47(2):245–265, 2012.

14 Frédéric Meunier, Wolfgang Mulzer, Pauline Sarrabezolles, and Yannik Stein. The rainbow at the end of the line: A PPAD formulation of the Colorful Carathéodory theorem with applications. In *Proc. 28th Annu. ACM-SIAM Sympos. Discrete Algorithms (SODA)*, pages 1342–1351, 2017.

15 Gary L. Miller and Donald R. Sheehy. Approximate centerpoints with proofs. *Comput. Geom. Theory Appl.*, 43(8):647–654, 2010.

16 Wolfgang Mulzer and Yannik Stein. Computational aspects of the Colorful Carathéodory theorem. *Discrete Comput. Geom.*, 60(3):720–755, 2018.

17 Wolfgang Mulzer and Daniel Werner. Approximating Tverberg points in linear time for any fixed dimension. *Discrete Comput. Geom.*, 50(2):520–535, 2013.

18 Noam Nisan, Tim Roughgarden, Éva Tardos, and Vijay V. Vazirani, editors. *Algorithmic Game Theory*. Cambridge University Press, 2007.

19 Johann Radon. Mengen konvexer Körper, die einen gemeinsamen Punkt enthalten. *Mathematische Annalen*, 83:113–115, 1921.

20 Jean-Pierre Roudneff. Partitions of points into simplices with k-dimensional intersection. I. The conic Tverberg's theorem. *European Journal of Combinatorics*, 22(5):733–743, 2001.

21 Karanbir S. Sarkaria. Tverberg's theorem via number fields. *Israel Journal of Mathematics*, 79(2-3):317–320, 1992.

22 Pablo Soberón. Equal coefficients and tolerance in coloured Tverberg partitions. *Combinatorica*, 35(2):235–252, 2015.

23 Daniel Spielman. Spectral graph theory. URL: `http://www.cs.yale.edu/homes/spielman/561/`.

24 Arthur H. Stone and John W. Tukey. Generalized "Sandwich" theorems. *Duke Mathematical Journal*, 9(2):356–359, June 1942.

25 Helge Tverberg. A generalization of Radon's theorem. *Journal of the London Mathematical Society*, s1-41(1):123–128, 1966.

26 Helge Tverberg. A generalization of Radon's theorem II. *Journal of the Australian Mathematical Society*, 24(3):321–325, 1981.

27 Helge Tverberg and Siniša T. Vrećica. On generalizations of Radon's theorem and the Ham-sandwich theorem. *European Journal of Combinatorics*, 14(3):259–264, 1993.

28 Rade T. Zivaljević and Siniša T. Vrećica. An extension of the Ham sandwich theorem. *Bulletin of the London Mathematical Society*, 22(2):183–186, 1990.

29 Rade T. Zivaljević and Siniša T. Vrećica. The colored Tverberg's problem and complexes of injective functions. *Journal of Combinatorial Theory, Series A.*, 61:309–318, 1992.

Lexicographic Optimal Homologous Chains and Applications to Point Cloud Triangulations

David Cohen-Steiner
Université Côte d'Azur, Sophia Antipolis, France
Inria Sophia Antipolis - Mediterranée, France
https://www-sop.inria.fr/members/David.Cohen-Steiner/
david.cohen-steiner@inria.fr

André Lieutier
Dassault Systèmes Provence, Aix-en-Provence, France
andre.lieutier@3ds.fr

Julien Vuillamy
Université Côte d'Azur, Sophia Antipolis, France
Inria Sophia Antipolis - Mediterranée, France
Dassault Systèmes Provence, Aix-en-Provence, France
julien.vuillamy@3ds.fr

Abstract

This paper considers a particular case of the Optimal Homologous Chain Problem (OHCP) for integer modulo 2 coefficients, where optimality is meant as a minimal lexicographic order on chains induced by a total order on simplices. The matrix reduction algorithm used for persistent homology is used to derive polynomial algorithms solving this problem instance, whereas OHCP is NP-hard in the general case. The complexity is further improved to a quasilinear algorithm by leveraging a dual graph minimum cut formulation when the simplicial complex is a pseudomanifold. We then show how this particular instance of the problem is relevant, by providing an application in the context of point cloud triangulation.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases OHCP, simplicial homology, triangulation, Delaunay

Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.32

Related Version A full version of the paper is available at https://hal.archives-ouvertes.fr/hal-02391240.

Acknowledgements We would like to thank the anonymous reviewers for their helpful comments and suggestions on the submitted version of this paper.

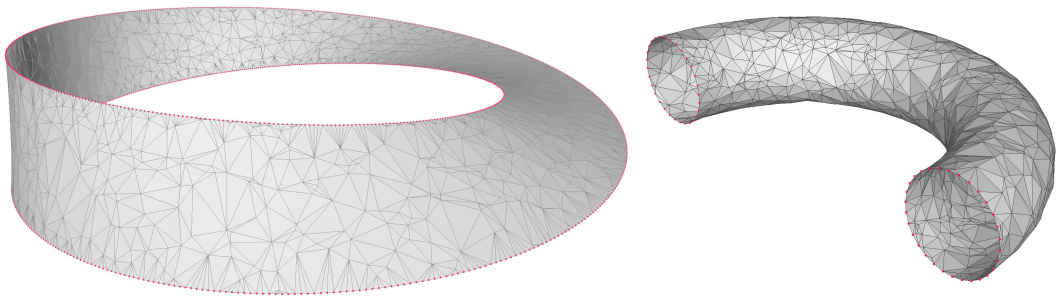


Figure 1 Open surface triangulations under imposed boundaries (red cycles).

36th International Symposium on Computational Geometry (SoCG 2020).
Editors: Sergio Cabello and Danny Z. Chen; Article No. 32; pp. 32:1–32:17
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

1.1 Problem statement

The computation of minimal simplicial homology generators has been a wide subject of interest for its numerous applications related to shape analysis, computer graphics or computer-aided design. Coined in [22], we recall the Optimal Homologous Chain Problem (OHCP):

▶ **Problem 1** (OHCP). *Given a d-chain A in a simplicial complex K and a set of weights given by a diagonal matrix W of appropriate dimension, find the L^1-norm minimal chain $\Gamma_{\min}$ homologous to A:*

$$\Gamma_{\min} = \min_{\Gamma, B} ||W \cdot \Gamma||_1 \text{ such that } \Gamma = A + \partial_{d+1} B \text{ and } \Gamma \in \boldsymbol{C_d}(K), B \in \boldsymbol{C_{d+1}}(K)$$

It has been shown that OHCP is NP-hard in the general case when using coefficients in $\mathbb{Z}_2$ [15, 9]. However, we consider a specialization of this problem: the Lexicographic Optimal Homologous Chain Problem (Lex-OHCP). Using coefficients in $\mathbb{Z}_2$, minimality is now meant according to a lexicographic order on chains induced by a total order on simplices. Formulated in the context of OHCP, this would require ordering the simplices using a total order and taking a weight matrix W with generic term $W_{ii} = 2^i$, allowing the L^1-norm minimization to be equivalent to a minimization along the lexicographic order.

1.2 Contributions

After providing some required definitions and notations (Section 2), we show how an algorithm based on the matrix reduction algorithm used for the computation of persistent homology [26] allows to solve this particular instance of OHCP in $O(n^3)$ worst case complexity (Section 3). Using a very similar process, we show that the problem of finding a minimal d-chain bounding a given $(d-1)$-cycle admits a similar algorithm with the same algorithmic complexity (Section 4). Section 5 then considers Lex-OHCP in the case where the simplicial complex K is a strongly connected $(d+1)$-pseudomanifold. By formulating it as a Lexicographic Minimum Cut (LMC) dual problem, the algorithm can be improved to a quasilinear complexity: the cost of sorting the dual edges and performing a $\mathcal{O}(E\alpha(E))$ algorithm based on disjoint-sets, where E is the number of dual edges and α is the inverse Ackermann function. Finally, Section 6 legitimizes this restriction of OHCP by characterizing the quality of lexicographic optimal homologous chains, namely in the context of point cloud triangulation. After defining a total order closely related to the Delaunay triangulation, we provide details on an open surface algorithm given a boundary as well as a watertight surface reconstruction algorithm given an interior and exterior information.

1.3 Related work

Several authors have studied algorithm complexities for the computation of L^1-norm optimal cycles in homology classes [28, 13, 9, 10, 14, 38, 24, 15, 22, 23]. However, to the best of our knowledge, considering lexicographic-minimal chains in their homology classes is a new idea. When minimal cycles are of codimension 1 in a pseudo-manifold, the idea of considering the minimal cut problem on the dual graph has been previously studied [36]. In particular, Chambers et al. [9] have considered graph duality to derive complexity results for the computation of optimal homologous cycles on 2-manifolds. Chen et al. [15] also use a reduction to a minimum cut problem on a dual graph to compute minimal non-null homologous cycles on d-complexes embedded in $\mathbb{R}^d$. Their polynomial algorithm (Theorem 5.2.3 in [15]) for

computing a homology class representative of minimal radius is reminiscent of our algorithm for computing lexicographic minimal representatives (Algorithm 4). In a recent work [23], Dey et al. consider the dual graph of pseudo-manifolds in order to obtain polynomial time algorithms for computing minimal persistent cycles. Since they consider arbitrary weights, they obtain the $\mathcal{O}(n^2)$ complexity of best known minimum cut/maximum flow algorithms [35]. The lexicographic order introduced in our work can be derived from the idea of a variational formulation of the Delaunay triangulation, first introduced in [16] and further studied in [1, 17]. Finally, many methods have been proposed to answer the problem of surface reconstruction in specific acquisition contexts [31, 32, 34]: [33] classifies a large number of these methods according to the assumptions and information used in addition to geometry. In the family of purely geometric reconstruction based on a Delaunay triangulation, one early contribution is the sculpting algorithm by Boissonnat [6]. The crust algorithm by Amenta et al. [2, 3] and an algorithm based on natural neighbors [7] were the first algorithms to guarantee a triangulation of the manifold under sampling conditions. However, these general approaches usually have difficulties far from these sampling conditions, in applications where point clouds are noisy or under-sampled. This difficulty can be circumvented by providing additional information on the nature of the surface [21, 25, 8]. Our contribution lies in this category of approaches. We provide some topological information of the surface: a boundary for the open surface reconstruction and interior and exterior regions for the closed surface reconstruction.

2 Definitions

2.1 Simplicial complexes

Consider an independent family $A = (a_0, \dots, a_d)$ of points of $\mathbb{R}^N$. We call a ***d*-simplex** σ spanned by A the set of all points: $x = \sum_{i=0}^{d} t_i a_i$, where $\forall i \in [0, d], t_i \geq 0$ and $\sum_{i=0}^{d} t_i = 1$. Any simplex spanned by a subset of A is called a **face** of σ.

A **simplicial complex** K is a collection of simplices such that every face of a simplex of K is in K and the intersection of two simplices of K is either empty either a common face.

2.2 Simplicial chains

Let K be a simplicial complex of dimension at least d. The notion of chains can be defined with coefficients in any ring but we restrict here the definition to coefficients in the field $\mathbb{Z}_2 = \mathbb{Z}/2\mathbb{Z}$. A ***d*-chain** A with coefficients in $\mathbb{Z}_2$ is a formal sum of d-simplices :

$$A = \sum_i x_i \sigma_i, \text{ with } x_i \in \mathbb{Z}_2 \text{ and } \sigma_i \in K \tag{1}$$

We denote $\boldsymbol{C_d}(K)$ the vector space over the field $\mathbb{Z}_2$ of d-chains in the complex K. Interpreting the coefficient $x_i \in \mathbb{Z}_2 = \{0, 1\}$ in front of simplex σ_i as indicating the existence of σ_i in the chain A, we can view the d-chain A as a set of simplices : for a d-simplex σ and a d-chain A, we write that $\sigma \in A$ if the coefficient for σ in A is 1. With this convention, the sum of two chains corresponds to the symmetric difference on their sets. In what follows, a d-simplex σ can also be interpreted as the d-chain containing only the d-simplex σ.

2.3 Boundary operator

For a d-simplex $\sigma = [a_0, \dots, a_d]$, the **boundary operator** is defined as the operator:

$$\partial_d : \boldsymbol{C_d}(K) \to \boldsymbol{C_{d-1}}(K)$$
$$\partial_d \sigma \underset{\text{def.}}{=} \sum_{i=0}^{d} [a_0, \dots, \widehat{a_i}, \dots, a_d]$$

where the symbol $\widehat{a_i}$ means the vertex a_i is deleted from the array. The kernel of the boundary operator $Z_d = \operatorname{Ker} \partial_d$ is called the group of d-cycles and the image of the operator $B_d = \operatorname{Im} \partial_{p+1}$ is the group of d-boundaries. We say two d-chains A and A' are **homologous** if $A - A' = \partial_{d+1} B$, for some $(d+1)$-chain B.

2.4 Lexicographic order

We assume now a total order on the d-simplices of K, $\sigma_1 < \cdots < \sigma_n$, where $n = \dim \boldsymbol{C_d}(K)$. From this order, we define a lexicographic total order on d-chains.

▶ **Definition 2** (Lexicographic total order). *For $C_1, C_2 \in \boldsymbol{C_d}(K)$:*

$$C_1 \sqsubseteq_{lex} C_2 \underset{def.}{\iff} \begin{cases} C_1 + C_2 = 0 \\ \text{or} \\ \sigma_{\max} = \max\{\sigma \in C_1 + C_2\} \in C_2 \end{cases}$$

This total order naturally extends to a strict total order $\sqsubset_{lex}$ on $\boldsymbol{C_d}(K)$.

3 Lexicographic optimal homologous chain

3.1 Problem statement

In this section, we define the Lexicographic Optimal Homologous Chain Problem (Lex-OHCP), a particular instance of OHCP (Problem 1):

▶ **Problem 3** (Lex-OHCP). *Given a simplicial complex K with a total order on the d-simplices and a d-chain $A \in \boldsymbol{C_d}(K)$, find the unique chain $\Gamma_{\min}$ defined by :*

$$\Gamma_{\min} = \min_{\sqsubseteq_{lex}} \{\Gamma \in \boldsymbol{C_d}(K) \mid \exists B \in \boldsymbol{C_{d+1}}(K), \Gamma - A = \partial_{d+1} B\}$$

▶ **Definition 4.** *A d-chain $A \in \boldsymbol{C_d}(K)$ is said **reducible** if there is a d-chain $\Gamma \in \boldsymbol{C_d}(K)$ (called **reduction**) and a $(d+1)$-chain $B \in \boldsymbol{C_{d+1}}(K)$ such that:*

$$\Gamma \sqsubset_{lex} A \quad \text{and} \quad \Gamma - A = \partial_{d+1} B$$

*If this property cannot be verified, the d-chain A is said **irreducible**. If A is reducible, we call **total reduction** of A the unique irreducible reduction of A. If A is irreducible, A is said to be its own total reduction.*

Problem 3 can be reformulated as finding the total reduction of A.

3.2 Boundary matrix reduction

With $m = \dim \boldsymbol{C_d}(K)$ and $n = \dim \boldsymbol{C_{d+1}}(K)$, we now consider the m-by-n boundary matrix ∂_{d+1} with entries in $\mathbb{Z}_2$. We enforce that rows of the matrix are ordered according to a given strict total order on d-simplices $\sigma_1 < \cdots < \sigma_m$, where the d-simplex σ_i is the basis element corresponding to the i^{th} row of the boundary matrix. The order of columns, corresponding to an order on $(d+1)$-simplices, is not relevant for this section and can be chosen arbitrarily.

For a matrix R, the index of the lowest (i.e. closest to the bottom) non-zero coefficient of a non-zero column R_j is denoted by $\mathrm{low}(j)$, or sometimes $\mathrm{low}(R_j)$ when we want to explicit the considered matrix.

Algorithm 1 is a slightly modified version of the boundary reduction algorithm presented in [26]. Indeed, for our purpose, we do not need the boundary matrix storing all the simplices of all dimensions and apply the algorithm to the sub-matrix $\partial_{d+1} : \boldsymbol{C_{d+1}}(K) \to \boldsymbol{C_d}(K)$. One checks easily that Algorithm 1 has $\mathcal{O}(mn^2)$ time complexity.

Algorithm 1 Reduction algorithm for the ∂_{d+1} matrix.

```
R = ∂_{d+1}
for j ← 1 to n do
    while R_j ≠ 0 and ∃ j_0 < j with low(j_0) = low(j) do
        R_j ← R_j + R_{j_0}
    end
end
```

We now introduce a few lemmas useful for solving Problem 3. We allow ourselves to consider each column R_j of the matrix R, formally an element of $\mathbb{Z}_2^m$, as the corresponding d-chain in the basis $(\sigma_1, \ldots, \sigma_m)$.

▶ **Lemma 5.** *A d-chain A is reducible if and only if at least one of its d-simplices is reducible.*

Proof. If there is a reducible d-simplex $\sigma \in A$, A is reducible by the d-chain $A' = A - \sigma + Red(\sigma)$, where $Red(\sigma)$ is a reduction for σ.
We suppose A to be reducible. Let $\Gamma \sqsubset_{lex} A$ be a reduction for A and B the $(d+1)$-chain such that $\Gamma - A = \partial B$. We denote $\sigma_{\max} = \max\{\sigma \in A - \Gamma\}$. Note that $\sigma_{\max}$ is homologous to $\Gamma - A + \sigma_{\max}$. The chain $\Gamma - A + \sigma_{\max}$ only contains simplices smaller than $\sigma_{\max}$, by definition of the lexicographic order (Definition 2). We have thus shown that if A is reducible, it contains at least one simplex that is reducible. ◀

▶ **Lemma 6.** *After matrix reduction (Algorithm 1), a non-zero column $R_j \neq 0$ can be described as*

$$R_j = \sigma_{\mathrm{low}(j)} + \Gamma, \text{ where } \Gamma \text{ is a reduction for } \sigma_{\mathrm{low}(j)}.$$

Proof. As all matrix operations performed on R by the reduction algorithm are linear, each non-zero column R_j of R is in the image of ∂_{d+1}. Therefore, there exist a $(d+1)$-chain B such that $R_j = \sigma_{\mathrm{low}(j)} + \Gamma = \partial_{d+1} B$, which, is equivalent in $\mathbb{Z}_2$ to $\Gamma - \sigma_{\mathrm{low}(j)} = \partial_{d+1} B$. By definition of the low of a column, we also have immediately: $\Gamma \sqsubset_{lex} \sigma_{\mathrm{low}(j)}$. For each non-zero column, the largest simplex is therefore reducible by the other d-simplices of the column. ◀

▶ **Lemma 7.** *After matrix reduction (Algorithm 1), there is a one-to-one correspondence between the reducible d-simplices and non-zero columns of R:*

$$\sigma_i \in \boldsymbol{C_d}(K) \text{ is reducible} \iff \exists j \in [1, n], R_j \neq 0 \text{ and } \mathrm{low}(j) = i$$

Proof. Lemma 6 shows immediately that the simplex corresponding to the lowest index of a non-zero column is reducible. Suppose now that a d-simplex $\tilde{\sigma}$ is reducible and let $\tilde{\Gamma}$ be a reduction of it: $\tilde{\sigma} + \tilde{\Gamma} = \partial_{d+1}B$ and $\tilde{\Gamma} \sqsubset_{lex} \tilde{\sigma}$. Algorithm 1 realizes the matrix factorization $R = \partial_{d+1} \cdot V$, where matrix V is invertible [26]. It follows that $\operatorname{Im} R = \operatorname{Im} \partial_{d+1}$. Therefore, non-zero columns of R span $\operatorname{Im} \partial_{d+1}$ and since $\tilde{\sigma} + \tilde{\Gamma} = \partial_{d+1}B \in \operatorname{Im} \partial_{d+1}$, there is a family $(R_j)_{j\in J} = (\sigma_{\text{low}(j)}, \Gamma_j)_{j\in J}$ of columns of R such that :

$$\tilde{\sigma} + \tilde{\Gamma} = \sum_{j\in J} \sigma_{\text{low}(j)} + \Gamma_j$$

Every $\sigma_{\text{low}(j)}$ represents the largest simplex of a column, and Γ_j a reduction chain for $\sigma_{\text{low}(j)}$. As observed in section VII.1 of [26], one can check that the low indexes in R are unique after the reduction algorithm. Therefore, there is a $j_{\max} \in J$ such that for all j in $J \setminus \{j_{\max}\}$, $\text{low}(j) < \text{low}(j_{\max})$, which implies:

$$\sigma_{j_{\max}} = \max\{\sigma \in \sum_{j\in J} \sigma_{\text{low}(j)} + \Gamma_j\} = \max\left\{\sigma \in \tilde{\sigma} + \tilde{\Gamma}\right\} = \tilde{\sigma}$$

We have shown that for the reducible simplex $\tilde{\sigma}$, there is a non-zero column $R_{j_{\max}}$ with $\tilde{\sigma} = \sigma_{\text{low}(j_{\max})}$ as its largest simplex. ◀

3.3 Total reduction algorithm

Combining the three previous lemmas give the intuition on how to construct the total reduction solving Problem 3: Lemma 5 allows to consider each simplex individually, Lemma 7 characterizes the reducible nature of a simplex using the reduced boundary matrix and Lemma 6 describes a column of the reduction boundary matrix as a simplex and its reduction. We now present Algorithm 2, referred to as the **total reduction algorithm**. For a d-chain Γ, $\Gamma[i] \in \mathbb{Z}_2$ denotes the coefficient of the i^{th} simplex in the chain Γ.

Algorithm 2 Total reduction algorithm.

Inputs : A d-chain Γ, the reduced boundary matrix R
for $i \leftarrow m$ **to** 1 **do**
  if $\Gamma[i] \neq 0$ ***and*** $\exists j \in [1, n]$ *with* $\text{low}(j) = i$ *in* R **then**
    $\Gamma \leftarrow \Gamma + R_j$
  end
end

▶ **Proposition 8.** *Algorithm 2 finds the total reduction of a given d-chain in $\mathcal{O}(m^2)$ time complexity.*

Proof. In Algorithm 2, let Γ_{i-1} be the value of the variable Γ after iteration i. Since the iteration counter i decreases from m to 1, the input and output of the algorithm are respectively Γ_m and Γ_0. At each iteration, Γ_{i-1} are either equal to Γ_i or $\Gamma_i + R_j$. Since $R_j \in \operatorname{Im} \partial_{d+1}$, Γ_{i-1} is in both cases homologous to Γ_i. Therefore, Γ_0 is homologous to Γ_m. We are left to show that Γ_0 is irreducible. From Lemma 5, it is enough to check that it does not contain any reducible simplex.
Let σ_i be a reducible simplex and let us show that $\sigma_i \notin \Gamma_0$. Two possibilities may occur:

- if $\sigma_i \in \Gamma_i$, then $\Gamma_{i-1} = \Gamma_i + R_j$. Since $\text{low}(j) = i$, we have $\sigma_i \in R_j$ and therefore $\sigma_i \notin \Gamma_{i-1}$.
- if $\sigma_i \notin \Gamma_i$, then $\Gamma_{i-1} = \Gamma_i$ and again $\sigma_i \notin \Gamma_{i-1}$.

Furthermore, from iterations $i-1$ to 1, the added columns R_j contain only simplices smaller than σ_i and therefore $\sigma_i \notin \Gamma_{i-1} \Rightarrow \sigma_i \notin \Gamma_0$.

Observe that using an auxiliary array allows to compute the correspondence $low(j) \to i$ in time $\mathcal{O}(1)$. The column addition nested inside the loop lead to a $\mathcal{O}(m^2)$ time complexity for Algorithm 2. ◀

It follows that Problem 3 can be solved in $\mathcal{O}(mn^2)$ time complexity, by applying successively Algorithms 1 and 2, or in $\mathcal{O}(N^3)$ complexity if N is the size of the simplicial complex.

4 Lexicographic-minimal chain under imposed boundary

4.1 Problem statement

This section will study a variant of Lex-OHCP (Problem 9) in order to solve the subsequent problem of finding a minimal d-chain bounding a given $(d-1)$-cycle (Problem 10).

▶ **Problem 9.** *Given a simplicial complex K with a total order on the d-simplices and a d-chain $\Gamma_0 \in \boldsymbol{C_d}(K)$, find :*

$$\Gamma_{\min} = \min_{\sqsubseteq_{lex}} \{\Gamma \in \boldsymbol{C_d}(K) \mid \partial_d \Gamma = \partial_d \Gamma_0\}$$

▶ **Problem 10.** *Given a simplicial complex K with a total order on the d-simplices and a $(d-1)$-cycle A, check if A is a boundary:*

$$B_A \underset{def.}{=} \{\Gamma \in \boldsymbol{C_d}(K) \mid \partial_d \Gamma = A\} \neq \varnothing$$

If it is the case, find the minimal d-chain Γ bounded by A:

$$\Gamma_{min} = \min_{\sqsubseteq_{lex}} B_A$$

In Problem 10, finding a representative Γ_0 in the set $B_A \neq \varnothing$ such that $\partial_d \Gamma_0 = A$ is sufficient: we are then taken back to Problem 9 to find the minimal d-chain $\Gamma_{\min}$ such that $\partial_d \Gamma_{\min} = \partial_d \Gamma_0 = A$.

4.2 Boundary reduction transformation matrix

As in Section 3, we will derive an algorithmic solution to Problem 9 from the result of the boundary matrix reduction algorithm. Note that, unlike Section 3 that used the ∂_{d+1} boundary operator, we are now considering ∂_d, meaning the given total order on d-simplices applies to the greater dimension of the matrix. An arbitrary order can be taken for the $(d-1)$-simplices to build the matrix ∂_d. Indeed, if we see the performed reduction in matrix notation as $R = \partial_d \cdot V$, the minimization steps in this section will be performed on the transformation matrix V, whose rows do follow the given simplicial ordering. The number of zero columns of R is the dimension of $Z_d = \operatorname{Ker} \partial_d$ [26]. Let's denote it by $n^{\text{Ker}} = \dim(Z_d)$. By selecting all columns in V corresponding to zero columns in R, we obtain the matrix V^{Ker}, whose columns $V_1^{\text{Ker}}, \ldots, V_{n^{\text{Ker}}}^{\text{Ker}}$ form a basis of Z_d. We first show a useful property on the matrix V^{Ker}. Note that the low index for any column in V^{Ker} is well defined, as V is invertible.

▶ **Lemma 11.** *Indexes* $\left\{ \operatorname{low}(V_i^{\mathrm{Ker}}) \right\}_{i\in[1,n^{\mathrm{Ker}}]}$ *are unique. If* $A \in \operatorname{Ker}\partial_d \setminus \{0\}$*, there exists a unique column* $V^{\mathrm{Ker}}_{\max}$ *of* V^{Ker} *with* $\operatorname{low}(V^{\mathrm{Ker}}_{\max}) = \operatorname{low}(A)$.

Proof. Before the boundary matrix reduction algorithm, the initial matrix V is the identity: the low indexes are therefore unique. During iterations of the algorithm, the matrix V is right-multiplied by an column-adding elementary matrix $L_{j_0,j}$, adding column j_0 to j with $j_0 < j$.

$$L_{j_0,j} = \begin{array}{c} \\ \\ \\ \\ \\ \\ \end{array} \begin{array}{c} \begin{array}{ccccccc} & & & j & & & \end{array} \\ \begin{bmatrix} 1 & & & & & & \\ & 1 & & 1 & & & \\ & & \ddots & & & & \\ & & & 1 & & & \\ & & & & \ddots & & \\ & & & & & 1 & \end{bmatrix} \end{array} \begin{array}{c} \\ \\ \\ j_0 \\ \\ \\ \end{array}$$

Therefore, the indexes $\{\operatorname{low}(V_i), V_i \in V\}$ stay on the diagonal during the reduction algorithm and are therefore unique. The restriction of V to V^{Ker} does not change this property.

If $A \in \operatorname{Ker}\partial_d \setminus \{0\}$, A can be written as a non-zero linear combination $A = \sum_{i\in I} V_i^{\mathrm{Ker}}$ of columns of V^{Ker}. By unicity of the lows of V^{Ker}, the largest low of the combination - i.e. $\max_{i\in I}\{\operatorname{low}(V_i^{\mathrm{Ker}})\}$ is in A and has to be the low of A. ◀

4.3 Total reduction with imposed boundary

We apply a similar total reduction algorithm as previously introduced in Section 3 but using the matrix V^{Ker}. In the following algorithm, $m = \dim \boldsymbol{C_d}(K)$.

Algorithm 3 Total reduction variant.

Inputs : A d-chain Γ and V^{Ker}
for $i \leftarrow m$ **to** 1 **do**
  if $\Gamma[i] \neq 0$ ***and*** $\exists j \in [1, n^{\mathrm{Ker}}]$ *with* $\operatorname{low}(j) = i$ *in* V^{Ker} **then**
    $\Gamma \leftarrow \Gamma + V_j^{\mathrm{Ker}}$
  end
end

▶ **Proposition 12.** *Algorithm 3 computes the solution for Problem 9 in* $\mathcal{O}(m^2)$ *time complexity.*

The proof is very similar to the one of Proposition 8 and available in [19].

4.4 Finding a representative of $\mathrm{B_A}$

As previously mentioned, solving Problem 10 requires deciding if the set B_A is empty and in case it is not empty, finding an element of the set B_A. Algorithm 3 can then be used to minimize this element under imposed boundary. In the following algorithm, $m = \dim \boldsymbol{C_{d-1}}(K)$ and $n = \dim \boldsymbol{C_d}(K)$.

Algorithm 4 Finding a representative.

```
Inputs : A (d − 1)-chain A, a boundary matrix R reduced by V
Γ_0 ← ∅
A_0 ← A
for i ← m to 1 do
    if A_0[i] ≠ 0 then
        if ∃j ∈ [1, n] with low(j) = i in R then
            A_0 ← A_0 + R_j
            Γ_0 ← Γ_0 + V_j
        else
            The set B_A is empty.
        end
    end
end
```

▶ **Proposition 13.** *Algorithm 4 decides if the set B_A is non-empty, and in that case, finds a representative Γ_0 such that $\partial\Gamma_0 = A$ in $\mathcal{O}(m^2)$ time complexity.*

Proof. We start by two trivial observations from the definition of a reduction. First, A is a boundary if and only if its total reduction is the null chain. Second, if a non-null chain is a boundary, then its greatest simplex is reducible.

If, at iteration i, $A_0[i] \neq 0$, then σ_i is the greatest simplex in A_0. In the case R has no column R_j such that $low(j) = i$, σ_i is not reducible by Lemma 7 and therefore A_0 is not a boundary. Since A and A_0 differ by a boundary (added columns of R), A is not a boundary either. This means the set B_A is empty.

The main difference with the previous chain reduction is we keep track of the column operations in Γ_0. If the total reduction of A is null, we have found a linear combination $(R_j)_{j\in J}$ such that $A = \sum_{j\in J} R_j$. We have also computed Γ_0 as the sum of the corresponding columns in V: $\Gamma_0 = \sum_{j\in J} V_j$. As $R = \partial_d \cdot V$, we can now verify:

$$\partial_d\Gamma_0 = \partial_d \left(\sum_{j\in J} V_j \right) = \sum_{j\in J} R_j = A$$ ◀

5 Efficient algorithm for codimension 1 (dual graph)

In this section we focus on Problem 17, a specialization of Problem 3, namely when K is a subcomplex of a $(d+1)$-pseudomanifold.

5.1 Problem statement

Recall that a d-dimensional simplicial complex is said **pure** if it is of dimension d and any simplex has at least one coface of dimension d.

▶ **Definition 14.** *A **d-pseudomanifold** is a pure d-dimensional simplicial complex for which each $(d-1)$-face has exactly two d-dimensional cofaces.*

▶ **Definition 15.** *The **dual graph** of a d-pseudomanifold $\mathcal{M}$ is the graph whose vertices are in one-to-one correspondence with the d-simplices of $\mathcal{M}$ and whose edges are in one-to-one correspondence with $(d-1)$-simplices of $\mathcal{M}$: an edge e connects two vertices v_1 and v_2 of the graph if and only if e corresponds to the $(d-1)$-face with cofaces corresponding to v_1 and v_2.*

▶ **Definition 16.** *A **strongly connected** d-pseudomanifold is a d-pseudomanifold whose dual graph is connected.*

Given a strongly connected $(d+1)$-pseudomanifold $\mathcal{M}$ and $\tau_1 \neq \tau_2$ two $(d+1)$-simplices of $\mathcal{M}$, we consider a special case of Problem 3 where $K = \mathcal{M} \setminus \{\tau_1, \tau_2\}$ and $A = \partial\tau_1$:

▶ **Problem 17.** *Given a strongly connected $(d+1)$-pseudomanifold $\mathcal{M}$ with a total order on the d-simplices and two distinct $(d+1)$-simplices (τ_1, τ_2) of $\mathcal{M}$, find:*

$$\Gamma_{\min} = \min_{\sqsubseteq_{lex}} \{\Gamma \in \boldsymbol{C_d}(\mathcal{M}) \mid \exists B \in \boldsymbol{C_{d+1}}(\mathcal{M} \setminus \{\tau_1, \tau_2\}), \Gamma - \partial\tau_1 = \partial B\}$$

▶ **Definition 18.** *Seeing a graph G as a 1-dimensional simplicial complex, we define the **coboundary operator** $\partial^0 : \boldsymbol{C_0}(G) \to \boldsymbol{C_1}(G)$ as the linear operator defined by the transpose of the matrix of the boundary operator $\partial_1 : \boldsymbol{C_1}(G) \to \boldsymbol{C_0}(G)$ in the canonical basis of simplices.*[1]

For a given graph $G = (\mathcal{V}, \mathcal{E})$, $\mathcal{V}$ and $\mathcal{E}$ respectively denote its vertex and edge sets. For a d-chain $\alpha \in \boldsymbol{C_d}(\mathcal{M})$ and a $(d+1)$-chain $\beta \in \boldsymbol{C_{d+1}}(\mathcal{M})$, $\tilde{\alpha}$ and $\tilde{\beta}$ denote their respective dual 1-chain and dual 0-chain in the dual graph $G(\mathcal{M})$ of $\mathcal{M}$. We easily see that:

▶ Remark 19. For a set of vertices $\mathcal{V}_0 \subset \mathcal{V}$, $\partial^0\mathcal{V}_0$ is exactly the set of edges in $G = (\mathcal{V}, \mathcal{E})$ that connect vertices in $\mathcal{V}_0$ with vertices in $\mathcal{V} \setminus \mathcal{V}_0$.

▶ Remark 20. Let $\mathcal{M}$ be a $(d+1)$-pseudomanifold. If $\alpha \in \boldsymbol{C_d}(\mathcal{M})$ and $\beta \in \boldsymbol{C_{d+1}}(\mathcal{M})$, then $\tilde{\alpha} = \partial^0\tilde{\beta} \iff \alpha = \partial_{d+1}\beta$.

5.2 Codimension 1 and Lexicographic Min Cut (LMC)

The order on d-simplices of a $(d+1)$-pseudomanifold $\mathcal{M}$ naturally defines a corresponding order on the edges of the dual graph: $\tau_1 < \tau_2 \iff \tilde{\tau}_1 < \tilde{\tau}_2$. This order extends similarly to a lexicographic order $\sqsubseteq_{lex}$ on sets of edges (or, equivalently, 1-chains) in the graph.

In what follows, we say a set of edges $\tilde{\Gamma}$ **disconnects** $\tilde{\tau}_1$ and $\tilde{\tau}_2$ in the graph $(\mathcal{V}, \mathcal{E})$ if $\tilde{\tau}_1$ and $\tilde{\tau}_2$ are not in the same connected component of the graph $(\mathcal{V}, \mathcal{E} \setminus \tilde{\Gamma})$.

Given a graph with weighted edges and two vertices, the min-cut/max-flow problem [27, 35] consists in finding the minimum cut (i.e. set of edges) disconnecting the two vertices, where minimum is meant as minimal sum of weights of cut edges. We consider a similar problem where the minimum is meant in term of a lexicographic order: the Lexicographic Min Cut (LMC).

▶ **Problem 21** (LMC)**.** *Given a connected graph $G = (\mathcal{V}, \mathcal{E})$ with a total order on $\mathcal{E}$ and two vertices $\tilde{\tau}_1, \tilde{\tau}_2 \in \mathcal{V}$, find the set $\tilde{\Gamma}_{\text{LMC}} \subset \mathcal{E}$ minimal for the lexicographic order $\sqsubseteq_{lex}$, that disconnects $\tilde{\tau}_1$ and $\tilde{\tau}_2$ in G.*

▶ **Proposition 22.** *$\Gamma_{\min}$ is solution of Problem 17 if and if only its dual 1-chain $\tilde{\Gamma}_{\min}$ is solution of Problem 21 on the dual graph $G(\mathcal{M})$ of $\mathcal{M}$ where $\tilde{\tau}_1$ and $\tilde{\tau}_2$ are respective dual vertices of τ_1 and τ_2.*

[1] In order to avoid to introduce non essential formal definitions, the coboundary operator is defined over chains since, for finite simplicial complexes, the canonical inner product defines a natural bijection between chains and cochains.

Proof. Problem 17 can be equivalently formulated as:

$$\Gamma_{\min} = \min_{\sqsubseteq_{lex}} \{\partial_{d+1}(\tau_1 + B) \mid B \in \boldsymbol{C_{d+1}}(\mathcal{M} \setminus \{\tau_1, \tau_2\})\} \tag{2}$$

Using Observation 20, we see that $\Gamma_{\min}$ is the minimum in Equation (2) if and only if its dual 1-chain $\tilde{\Gamma}_{\min}$ satisfies:

$$\tilde{\Gamma}_{\min} = \min_{\sqsubseteq_{lex}} \{\partial^0(\tilde{\tau}_1 + \tilde{B}) \mid \tilde{B} \subset \mathcal{V} \setminus \{\tilde{\tau}_1, \tilde{\tau}_2\}\} \tag{3}$$

Denoting $\tilde{\Gamma}_{\text{LMC}}$ the minimum of Problem 21, we need to show that $\tilde{\Gamma}_{\text{LMC}} = \tilde{\Gamma}_{\min}$.
As $\tilde{\Gamma}_{\text{LMC}}$ disconnects $\tilde{\tau}_1$ and $\tilde{\tau}_2$ in $G = (\mathcal{V}, \mathcal{E})$, $\tilde{\tau}_2$ is not in the connected component of $\tilde{\tau}_1$ in $(\mathcal{V}, \mathcal{E} \setminus \tilde{\Gamma}_{\text{LMC}})$. We define $\tilde{B}$ as the connected component of $\tilde{\tau}_1$ in $(\mathcal{V}, \mathcal{E} \setminus \tilde{\Gamma}_{\text{LMC}})$ minus $\tilde{\tau}_1$. We have that $\tilde{B} \subset \mathcal{V} \setminus \{\tilde{\tau}_1, \tilde{\tau}_2\}$. Consider an edge $e \in \partial^0(\tilde{\tau}_1 + \tilde{B})$. From Observation 19, e connects a vertex $v_a \in \{\tilde{\tau}_1\} \cup \tilde{B}$ with a vertex $v_b \notin \{\tilde{\tau}_1\} \cup \tilde{B}$. From the definition of $\tilde{B}$, $\tilde{\Gamma}_{\text{LMC}}$ disconnects v_a and v_b in G, which in turn implies $e \in \tilde{\Gamma}_{\text{LMC}}$. We have therefore shown that $\partial^0(\tilde{\tau}_1 + \tilde{B}) \subset \tilde{\Gamma}_{\text{LMC}}$. Using Equation (3), we get:

$$\tilde{\Gamma}_{\min} \sqsubseteq_{lex} \partial^0(\tilde{\tau}_1 + \tilde{B}) \sqsubseteq_{lex} \tilde{\Gamma}_{\text{LMC}} \tag{4}$$

Now we claim that if there is a $\tilde{C} \subset \mathcal{V} \setminus \{\tilde{\tau}_1, \tilde{\tau}_2\}$ with $\tilde{\Gamma} = \partial^0(\tilde{\tau}_1 + \tilde{C})$, then $\tilde{\Gamma}$ disconnects $\tilde{\tau}_1$ and $\tilde{\tau}_2$ in $(\mathcal{V}, \mathcal{E})$. Consider a path in G from $\tilde{\tau}_1$ to $\tilde{\tau}_2$. Let v_a be the last vertex of the path that belongs to $\{\tilde{\tau}_1\} \cup \tilde{C}$ and v_b the next vertex on the path (which exists since $\tilde{\tau}_2$ is not in $\{\tilde{\tau}_1\} \cup \tilde{C}$). From Observation 19, we see that the edge (v_a, v_b) must belong to $\tilde{\Gamma} = \partial^0(\tilde{\tau}_1 + \tilde{C})$. We have shown that any path in G connecting $\tilde{\tau}_1$ and $\tilde{\tau}_2$ has to contain an edge in $\tilde{\Gamma}$ and the claim is proved.
In particular, the minimum $\tilde{\Gamma}_{\min}$ disconnects $\tilde{\tau}_1$ and $\tilde{\tau}_2$ in $(\mathcal{V}, \mathcal{E})$. As $\tilde{\Gamma}_{\text{LMC}}$ denotes the minimum of Problem 21, $\tilde{\Gamma}_{\text{LMC}} \sqsubseteq_{lex} \tilde{\Gamma}_{\min}$ which, together with Equation (4), gives us $\tilde{\Gamma}_{\text{LMC}} = \tilde{\Gamma}_{\min}$. We have therefore shown the minimum defined by Equation (3) coincides with the minimum defined in Problem 21. ◀

5.3 Algorithm for Lexicographic Min Cut

In light of the new problem equivalency, we will study an algorithm solving Problem 21. As we will only consider the dual graph for this section, we leave behind the dual chain notation: vertices $\tilde{\tau}_1$ and $\tilde{\tau}_2$ are replaced by α_1 and α_2, and the solution to the problem is simply noted Γ_{LMC}. The following lemma exposes a constructive property of the solution on subgraphs.

▶ **Lemma 23.** *Consider Γ_{LMC} solution of Problem 21 for the graph $G = (\mathcal{V}, \mathcal{E})$ and $\alpha_1, \alpha_2 \in \mathcal{V}$. Let e_0 be an edge in $\mathcal{V} \times \mathcal{V}$ such that $e_0 < \min\{e \in \mathcal{E}\}$. Then:*

(a) *The solution for $(\mathcal{V}, \mathcal{E} \cup \{e_0\})$ is either Γ_{LMC} or $\Gamma_{\text{LMC}} \cup \{e_0\}$.*

(b) *$\Gamma_{\text{LMC}} \cup \{e_0\}$ is solution for $(\mathcal{V}, \mathcal{E} \cup \{e_0\})$ if and only if α_1 and α_2 are connected in $(\mathcal{V}, \mathcal{E} \cup \{e_0\} \setminus \Gamma_{\text{LMC}})$.*

Proof. Let's call Γ'_{LMC} the solution for $(\mathcal{V}, \mathcal{E} \cup \{e_0\})$. Since $\Gamma'_{\text{LMC}} \cap \mathcal{E}$ disconnects α_1 and α_2 in $(\mathcal{V}, \mathcal{E})$, one has $\Gamma_{\text{LMC}} \sqsubseteq_{lex} \Gamma'_{\text{LMC}}$. Since $\Gamma_{\text{LMC}} \cup \{e_0\}$ disconnects α_1 and α_2 in $(\mathcal{V}, \mathcal{E} \cup \{e_0\})$, we also have $\Gamma'_{\text{LMC}} \sqsubseteq_{lex} \Gamma_{\text{LMC}} \cup \{e_0\}$. Therefore, $\Gamma_{\text{LMC}} \sqsubseteq_{lex} \Gamma'_{\text{LMC}} \sqsubseteq_{lex} \Gamma_{\text{LMC}} \cup \{e_0\}$.

As $e_0 < \min\{e \in \mathcal{E}\}$, there is no set in $\mathcal{E} \cup \{e_0\}$ strictly between Γ_{LMC} and $\Gamma_{\text{LMC}} \cup \{e_0\}$ for the lexicographic order. It follows that Γ'_{LMC} is either equal to Γ_{LMC} or $\Gamma_{\text{LMC}} \cup \{e_0\}$. The choice for Γ'_{LMC} is therefore ruled by the property that it should disconnect α_1 and α_2: if α_1 and α_2 are connected in $(\mathcal{V}, \mathcal{E} \cup \{e_0\} \setminus \Gamma_{\text{LMC}})$, Γ_{LMC} does not disconnect α_1 and α_2 in $(\mathcal{V}, \mathcal{E} \cup \{e_0\})$ and $\Gamma_{\text{LMC}} \cup \{e_0\}$ has to be the solution for $(\mathcal{V}, \mathcal{E} \cup \{e_0\})$. On the other hand,

if α_1 and α_2 are not connected in $(\mathcal{V}, \mathcal{E} \cup \{e_0\} \setminus \Gamma_{\text{LMC}})$, then both Γ_{LMC} and $\Gamma_{\text{LMC}} \cup \{e_0\}$ disconnect α_1 and α_2 in $(\mathcal{V}, \mathcal{E} \cup \{e_0\})$, but as $\Gamma_{\text{LMC}} \sqsubset_{lex} \Gamma_{\text{LMC}} \cup \{e_0\}$, $\Gamma_{\text{LMC}} \cup \{e_0\}$ is not the solution for $(\mathcal{V}, \mathcal{E} \cup \{e_0\})$. ◀

Building an algorithm from Lemma 23 suggests a data structure able to check if vertices α_1 and α_2 are connected in the graph: the disjoint-set data structure, introduced for finding connected components [29], does exactly that. In this structure, each set of elements has a different root value, called representative. Calling the operation **MakeSet** on an element creates a new set containing this element. The **FindSet** operation, given an element of a set, returns the representative of the set. For all elements of the same set, **FindSet** will of course return the same representative. Finally, the structure allows merging two sets, by using the **UnionSet** operation. After this operation, elements of both sets will have the same representative.

We now describe Algorithm 5. The algorithm expects a set of edges sorted in decreasing order according to the lexicographic order.

Algorithm 5 Lexicographic Min Cut.

Inputs : $G = (\mathcal{V}, \mathcal{E})$ and $\alpha_1, \alpha_2 \in \mathcal{V}$, with $\mathcal{E} = \{e_i, i = 1, \dots, n\}$ in decreasing order
Output : Γ_{LMC}
$\Gamma_{\text{LMC}} \leftarrow \varnothing$
for $v \in \mathcal{V}$ **do**
  MakeSet(v)
end
for $e \in \mathcal{E}$ *in decreasing order* **do**
  $e = (v_1, v_2) \in \mathcal{V} \times \mathcal{V}$
  $r_1 \leftarrow$ **FindSet**(v_1), $r_2 \leftarrow$ **FindSet**(v_2)
  $c_1 \leftarrow$ **FindSet**(α_1), $c_2 \leftarrow$ **FindSet**(α_2)
  if $\{r_1, r_2\} = \{c_1, c_2\}$ **then**
    $\Gamma_{\text{LMC}} \leftarrow \Gamma_{\text{LMC}} \cup e$
  else
    UnionSet(r_1, r_2)
  end
end

▶ **Proposition 24.** *Algorithm 5 computes the solution of Problem 21 for a given graph $(\mathcal{V}, \mathcal{E})$ and two vertices $\alpha_1, \alpha_2 \in \mathcal{V}$. Assuming the input set of edges $\mathcal{E}$ are sorted, the algorithm has $\mathcal{O}(n\alpha(n))$ time complexity, where n is the cardinal of $\mathcal{E}$ and α the inverse Ackermann function.*

Proof. We denote by e_i the i^{th} edge along the decreasing order and Γ^i_{LMC} the value of the variable Γ_{LMC} of the algorithm after iteration i. The algorithm works by incrementally adding edges in decreasing order and tracking the growing connected components of the set associated with α_1 and α_2 in $(\mathcal{V}, \{e \in \mathcal{E}, e \geq e_i\} \setminus \Gamma^i_{\text{LMC}})$, for $i = 1, \dots, n$.

At the beginning, no edges are inserted, and $\Gamma^0_{\text{LMC}} = \varnothing$ is indeed solution for $(\mathcal{V}, \varnothing)$. With Lemma 23, we are guaranteed at each iteration i to find the solution for $(\mathcal{V}, \{e \in \mathcal{E}, e \geq e_i\})$ by only adding to $\Gamma^{i-1}_{\text{LMC}}$ the current edge e_i if α_1 and α_2 are connected in $\{e \in \mathcal{E}, e \geq e_i\} \setminus \Gamma^{i-1}_{\text{LMC}}$, which is done in the if-statement. If the edge is not added, we update the connectivity of the graph $(\mathcal{V}, \{e \in \mathcal{E}, e \geq e_i\} \setminus \Gamma^i_{\text{LMC}})$ by merging the two sets represented by r_1 and r_2. After each iteration, Γ^i_{LMC} is solution for $(\mathcal{V}, \{e \in \mathcal{E}, e \geq e_i\})$ and when all edges are processed, Γ^n_{LMC} is solution for $(\mathcal{V}, \mathcal{E})$.

The complexity of the **MakeSet**, **FindSet** and **UnionSet** operations have been shown to be respectively $\mathcal{O}(1)$, $\mathcal{O}(\alpha(v))$ and $\mathcal{O}(\alpha(v))$, where $\alpha(v)$ is the inverse Ackermann function on the cardinal of the vertex set [37]. Assuming sorted edges as input of the algorithm – which is performed in $\mathcal{O}(n\ln(n))$, the algorithm runs in $\mathcal{O}(n\alpha(n))$ time complexity. ◀

The similarity of Algorithm 5 with Kruskal's algorithm for minimum spanning-tree suggests an even better theoretical time complexity, by using Chazelle's algorithm [12] for minimum spanning-tree, running in $\mathcal{O}(n\alpha(n))$ complexity without requiring sorted edges.

6 Application to point cloud triangulation

In all that precedes, the order on simplices was not specified and one can wonder if choosing such an ordering makes the specialization of OHCP too restrictive for it to be useful. In this section, we give a concrete example where this restriction makes sense and provides a simple and elegant application to the problem of point cloud triangulation. Whereas all that preceded dealt with an abstract simplicial complex, we now consider a bijection between vertices and a set of points in Euclidean space, allowing to compute geometric quantities on simplices.

6.1 Simplicial ordering

Recent works have studied a characterization of the 2D Delaunay triangulation as a lexicographic minimum over 2-chains. Denote by $\mathrm{R_B}(\sigma)$ the radius of the smallest enclosing ball and $\mathrm{R_C}(\sigma)$ the radius of the circumcircle of a 2-simplex σ. Based on [20, 18], we define the total order on 2-simplices:

$$\sigma_1 \leq \sigma_2 \iff \begin{cases} \mathrm{R_B}(\sigma_1) < \mathrm{R_B}(\sigma_2) \\ \text{or} \\ \mathrm{R_B}(\sigma_1) = \mathrm{R_B}(\sigma_2) \quad \text{and} \quad \mathrm{R_C}(\sigma_1) \geq \mathrm{R_C}(\sigma_2) \end{cases} \tag{5}$$

Under generic condition on the position of points, we can show this order is total. In what follows, the lexicographic order $\sqsubseteq_{lex}$ is induced by this order on simplices. The following proposition from [20] shows a strong link between the simplex ordering and the 2D Delaunay triangulation.

▶ **Proposition 25** (Proposition 7.9 in [20]). *Let $\mathbf{P} = \{P_1, \ldots, P_N\} \subset \mathbb{R}^2$ with $N \geq 3$ be in general position and let $K_{\mathbf{P}}$ be any 2-dimensional complex containing the Delaunay triangulation of $\mathbf{P}$. Denote by $\beta_{\mathbf{P}} \in C_1(K_{\mathbf{P}})$ the 1-chain made of edges belonging to the boundary of convex hull $\mathcal{CH}(\mathbf{P})$. If $\Gamma_{\min} = \min_{\sqsubseteq_{lex}}\{\Gamma \in \boldsymbol{C_2}(K_{\mathbf{P}}), \partial\Gamma = \beta_{\mathbf{P}}\}$, the simplicial complex $|\Gamma_{\min}|$ support of $\Gamma_{\min}$ is the Delaunay triangulation of $\mathbf{P}$.*

As the 2D Delaunay triangulation has some well-known optimality properties, such as maximizing the minimal angle, we can hope that using the same order to minimize 2-chains in dimension 3 will keep some of those properties. In fact, it has been shown that for a Čech or Vietoris-Rips complex, under strict conditions linking the point set sampling, the parameter of the complex and the reach of the underlying manifold of Euclidean space, the minimal lexicographic chain using the described simplex order is a triangulation of the sampled manifold [18]. Experimental results (Figure 2) show that this property remains true relatively far from these theoretical conditions.

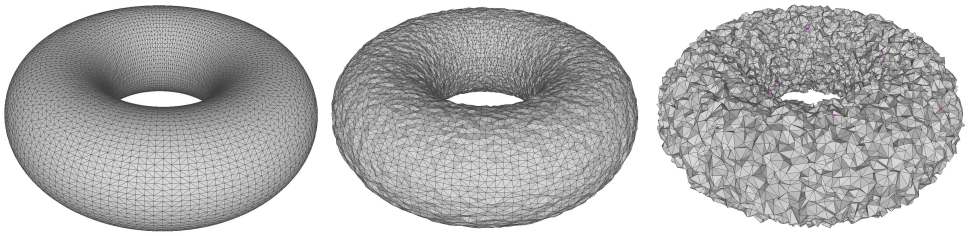


Figure 2 Watertight reconstructions under different perturbations. Under small perturbations (first two images from the left), the reconstruction is a triangulation of the sampled manifold. A few non-manifold configurations appear however under larger perturbations (Rightmost image).

6.2 Open surface triangulation

Given a point cloud sampling an open surface and a 1-cycle sampling the boundary of the surface, we generate a Čech complex of the point cloud using the Phat library [5]. The parameter of the complex should be sufficient to capture the homotopy type of the surface to reconstruct and should contain the provided cycle. After constructing the 2-boundary matrix, we apply the boundary reduction algorithm, slightly modified to store the transformation matrix V (Section 4.2). We then apply Algorithm 4 to find out if a 2-chain bounded by the cycle exists in the current Čech complex. In this case, we get a chain bounded by the provided cycle and apply Algorithm 3 to minimize the chain under imposed boundary. Otherwise, we might have to increase the Čech parameter to capture the homotopy type of the surface to reconstruct [11, 4]. Figure 1 shows results of this method.

6.3 Closed surface triangulation

Using Algorithm 5 requires a strongly connected 3-pseudomanifold: we therefore use a 3D Delaunay triangulation, for its efficiency and non-parametric nature, using the CGAL library [30], and complete it into a topological 3-sphere by connecting, for any triangle on the convex hull of the Delaunay triangulation, its dual edge to an "infinite" dual vertex.

Experimentally, sorting triangles does not require exact predicates: the $\mathrm{R_B}$ and $\mathrm{R_C}$ quantities can simply be calculated in fixed precision. The quasilinear complexity of Algorithm 5 makes it competitive in large point cloud applications. Outliers are naturally ignored and, being parameter free, the algorithm adapts to non uniform point densities, as seen in the closeup of Figure 3.

The choice of α_1 and α_2 defines the location of the closed separating surface and are chosen interactively. Although we could devise an algorithm where these inputs are not required – the algorithm would simply merge regions until only two connected components remain – this would only work for uniform and non-noisy point clouds but not make for a robust algorithm. On the contrary, adding multiple interior and exterior regions can guide the algorithm by providing better topological constraints, as depicted in Figure 4. Algorithm 5 requires to be slightly modified to take as input multiple α_1, α_2: after creating all sets with MakeSet, we need to combine all α_1 sets together, and all α_2 sets together. The algorithm remains unchanged for the rest.

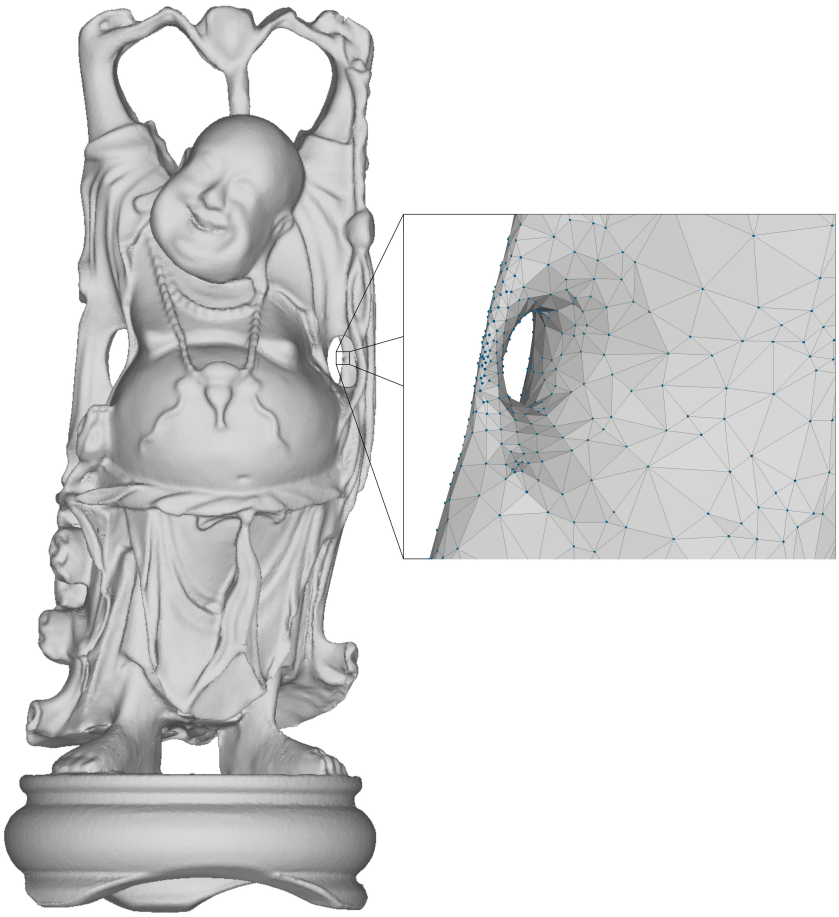


Figure 3 Closed surface triangulation of 440K points in 7.33 seconds. Beside the point cloud, the only user input is one inner tetrahedron. The closeup shows that small features are correctly recovered.

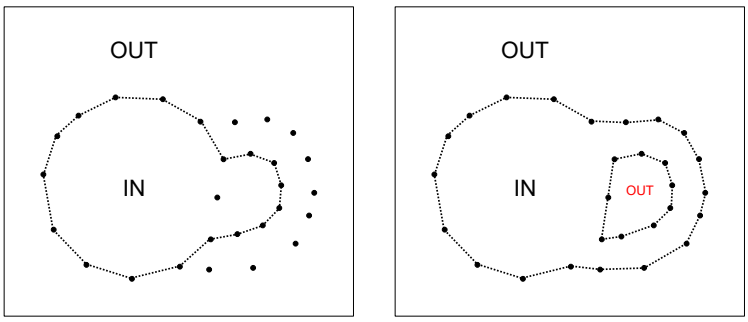


Figure 4 Providing additional topological information can improve the result of the reconstruction. Here, the lexicographic order on 1-chains is induced by edge length comparison.

References

1 Pierre Alliez, David Cohen-Steiner, Mariette Yvinec, and Mathieu Desbrun. Variational tetrahedral meshing. *ACM Trans. Graph.*, 24(3):617–625, 2005. doi:10.1145/1073204.1073238.

2 Nina Amenta, Marshall W. Bern, and Manolis Kamvysselis. A new voronoi-based surface reconstruction algorithm. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1998, Orlando, FL, USA, July 19-24, 1998*, pages 415–421, 1998. doi:10.1145/280814.280947.

3 Nina Amenta, Sunghee Choi, Tamal K. Dey, and N. Leekha. A simple algorithm for homeomorphic surface reconstruction. *Int. J. Comput. Geometry Appl.*, 12(1-2):125–141, 2002. doi:10.1142/S0218195902000773.

4 Dominique Attali, André Lieutier, and David Salinas. Vietoris-rips complexes also provide topologically correct reconstructions of sampled shapes. *Comput. Geom.*, 46(4):448–465, 2013. doi:10.1016/j.comgeo.2012.02.009.

5 Ulrich Bauer, Michael Kerber, Jan Reininghaus, and Hubert Wagner. Phat – persistent homology algorithms toolbox. *Journal of Symbolic Computation*, 78:76–90, 2017. Algorithms and Software for Computational Topology. doi:10.1016/j.jsc.2016.03.008.

6 Jean-Daniel Boissonnat. Geometric structures for three-dimensional shape representation. *ACM Trans. Graph.*, 3(4):266–286, 1984. doi:10.1145/357346.357349.

7 Jean-Daniel Boissonnat and Frédéric Cazals. Smooth surface reconstruction via natural neighbour interpolation of distance functions. In *Proceedings of the Sixteenth Annual Symposium on Computational Geometry, Clear Water Bay, Hong Kong, China, June 12-14, 2000*, pages 223–232, 2000. doi:10.1145/336154.336208.

8 J Frederico Carvalho, Mikael Vejdemo-Johansson, Danica Kragic, and Florian T Pokorny. An algorithm for calculating top-dimensional bounding chains. *PeerJ Computer Science*, 4:e153, 2018.

9 Erin W. Chambers, Jeff Erickson, and Amir Nayyeri. Minimum cuts and shortest homologous cycles. In *Proceedings of the 25th ACM Symposium on Computational Geometry, Aarhus, Denmark, June 8-10, 2009*, pages 377–385, 2009. doi:10.1145/1542362.1542426.

10 Erin Wolf Chambers and Mikael Vejdemo-Johansson. Computing minimum area homologies. In *Computer graphics forum*, volume 34, pages 13–21. Wiley Online Library, 2015.

11 Frédéric Chazal, David Cohen-Steiner, and André Lieutier. A sampling theory for compact sets in euclidean space. *Discrete & Computational Geometry*, 41(3):461–479, 2009. doi:10.1007/s00454-009-9144-8.

12 Bernard Chazelle. A minimum spanning tree algorithm with inverse-ackermann type complexity. *Journal of the ACM (JACM)*, 47(6):1028–1047, 2000.

13 Chao Chen and Daniel Freedman. Quantifying homology classes. *CoRR*, abs/0802.2865, 2008. arXiv:0802.2865.

14 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.

15 Chao Chen and Daniel Freedman. Hardness results for homology localization. *Discrete & Computational Geometry*, 45(3):425–448, 2011. doi:10.1007/s00454-010-9322-8.

16 Long Chen. Mesh smoothing schemes based on optimal delaunay triangulations. In *Proceedings of the 13th International Meshing Roundtable, IMR 2004, Williamsburg, Virginia, USA, September 19-22, 2004*, pages 109–120, 2004. URL: http://imr.sandia.gov/papers/abstracts/Ch317.html.

17 Long Chen and Michael Holst. Efficient mesh optimization schemes based on optimal delaunay triangulations. *Computer Methods in Applied Mechanics and Engineering*, 200(9):967–984, 2011. doi:10.1016/j.cma.2010.11.007.

18 David Cohen-Steiner, André Lieutier, and Julien Vuillamy. Lexicographic optimal chains and manifold triangulations. *Research Report HAL: hal-02391190*, 2019.

19 David Cohen-Steiner, André Lieutier, and Julien Vuillamy. Lexicographic optimal homologous chains and applications to point cloud triangulations. *Preprint HAL: hal-02391240*, 2019.

20 David Cohen-Steiner, André Lieutier, and Julien Vuillamy. Regular triangulations as lexicographic optimal chains. *Preprint HAL: hal-02391285*, 2019.

21 Tamal K. Dey and Samrat Goswami. Tight cocone: A water-tight surface reconstructor. *J. Comput. Inf. Sci. Eng.*, 3(4):302–307, 2003. doi:https://doi.org/10.1115/1.1633278.

22 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. doi:10.1137/100800245.

23 Tamal K. Dey, Tao Hou, and Sayan Mandal. Computing minimal persistent cycles: Polynomial and hard cases. *CoRR*, abs/1907.04889, 2019. arXiv:1907.04889.

24 Tamal K. Dey, Jian Sun, and Yusu Wang. Approximating loops in a shortest homology basis from point data. In David G. Kirkpatrick and Joseph S. B. Mitchell, editors, *Proceedings of the 26th ACM Symposium on Computational Geometry, Snowbird, Utah, USA, June 13-16, 2010*, pages 166–175. ACM, 2010. doi:10.1145/1810959.1810989.

25 Herbert Edelsbrunner. *Surface Reconstruction by Wrapping Finite Sets in Space*, pages 379–404. Springer Berlin Heidelberg, 2003. doi:10.1007/978-3-642-55566-4_17.

26 Herbert Edelsbrunner and John Harer. *Computational Topology - an Introduction*. American Mathematical Society, 2010. URL: http://www.ams.org/bookstore-getitem/item=MBK-69.

27 Jack Edmonds and Richard M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *J. ACM*, 19(2):248–264, 1972. doi:10.1145/321694.321699.

28 Jeff Erickson and Kim Whittlesey. Greedy optimal homotopy and homology generators. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2005, Vancouver, British Columbia, Canada, January 23-25, 2005*, pages 1038–1046, 2005. URL: http://dl.acm.org/citation.cfm?id=1070432.1070581.

29 Bernard A. Galler and Michael J. Fischer. An improved equivalence algorithm. *Commun. ACM*, 7(5):301–303, 1964. doi:10.1145/364099.364331.

30 Clément Jamin, Sylvain Pion, and Monique Teillaud. 3D triangulations. In *CGAL User and Reference Manual*. CGAL Editorial Board, 5.0 edition, 2019. URL: https://doc.cgal.org/5.0/Manual/packages.html#PkgTriangulation3.

31 Michael M. Kazhdan and Hugues Hoppe. Screened poisson surface reconstruction. *ACM Trans. Graph.*, 32(3):29:1–29:13, 2013. doi:10.1145/2487228.2487237.

32 Patrick Labatut, Jean-Philippe Pons, and Renaud Keriven. Robust and efficient surface reconstruction from range data. *Comput. Graph. Forum*, 28(8):2275–2290, 2009. doi:10.1111/j.1467-8659.2009.01530.x.

33 Sylvain Lefebvre and Michela Spagnuolo, editors. *Eurographics 2014 - State of the Art Reports, Strasbourg, France, April 7-11, 2014*. Eurographics Association, 2014. URL: https://diglib.eg.org/handle/10.2312/7707.

34 Yangyan Li, Xiaokun Wu, Yiorgos Chrysanthou, Andrei Sharf, Daniel Cohen-Or, and Niloy J. Mitra. Globfit: consistently fitting primitives by discovering global relations. *ACM Trans. Graph.*, 30(4):52, 2011. doi:10.1145/2010324.1964947.

35 James B. Orlin. Max flows in o(nm) time, or better. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 765–774, 2013. doi:10.1145/2488608.2488705.

36 John Matthew Sullivan. *A Crystalline Approximation Theorem for Hypersurfaces*. PhD thesis, Princeton Univ., October 1990. URL: http://torus.math.uiuc.edu/jms/Papers/thesis/.

37 Robert Endre Tarjan and Jan van Leeuwen. Worst-case analysis of set union algorithms. *J. ACM*, 31(2):245–281, 1984. doi:10.1145/62.2160.

38 Pengxiang Wu, Chao Chen, Yusu Wang, Shaoting Zhang, Changhe Yuan, Zhen Qian, Dimitris Metaxas, and Leon Axel. Optimal topological cycles and their application in cardiac trabeculae restoration. In *International Conference on Information Processing in Medical Imaging*, pages 80–92. Springer, 2017.

Finding Closed Quasigeodesics on Convex Polyhedra

Erik D. Demaine
Computer Science and Artificial Intelligence Laboratory, MIT, Cambridge, MA, USA
edemaine@mit.edu

Adam C. Hesterberg
Computer Science and Artificial Intelligence Laboratory, MIT, Cambridge, MA, USA
achester@mit.edu

Jason S. Ku
Department of Electrical Engineering and Computer Science, MIT, Cambridge, MA, USA
jasonku@mit.edu

Abstract

A ***closed quasigeodesic*** is a closed loop on the surface of a polyhedron with at most 180° of surface on both sides at all points; such loops can be locally unfolded straight. In 1949, Pogorelov proved that every convex polyhedron has at least three (non-self-intersecting) closed quasigeodesics, but the proof relies on a nonconstructive topological argument. We present the first finite algorithm to find a closed quasigeodesic on a given convex polyhedron, which is the first positive progress on a 1990 open problem by O'Rourke and Wyman. The algorithm's running time is pseudopolynomial, namely $O\left(\frac{n^2}{\varepsilon^2}\frac{L}{\ell}b\right)$ time, where ε is the minimum curvature of a vertex, L is the length of the longest edge, ℓ is the smallest distance within a face between a vertex and a nonincident edge (minimum feature size of any face), and b is the maximum number of bits of an integer in a constant-size radical expression of a real number representing the polyhedron. We take special care in the model of computation and needed precision, showing that we can achieve the stated running time on a pointer machine supporting constant-time w-bit arithmetic operations where $w = \Omega(\lg b)$.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases polyhedra, geodesic, pseudopolynomial, geometric precision

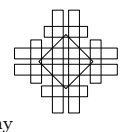
Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.33

Acknowledgements The authors thank Zachary Abel, Nadia Benbernou, Fae Charlton, Jayson Lynch, Joseph O'Rourke, Diane Souvaine, and David Stalfa for discussions related to this paper.

1 Introduction

A ***geodesic*** on a surface is a path that is locally shortest at every point, i.e., cannot be made shorter by modifying the path in a small neighborhood. A ***closed geodesic*** on a surface is a loop (closed curve) with the same property; notably, the locally shortest property must hold at all points, including the "wrap around" point where the curve meets itself. In 1905, Poincaré [20] conjectured that every convex surface has a non-self-intersecting closed geodesic.[1] In 1927, Birkhoff [5] proved this result, even in higher dimensions (for any smooth metric on the n-sphere). In 1929, Lyusternik and Schnirelmann [17] claimed that every smooth surface of genus 0 in fact has at least *three* non-self-intersecting closed geodesics. Their argument "contains some gaps" [2], filled in later by Ballmann [1].

[1] Non-self-intersecting (quasi)geodesics are often called *simple* (quasi)geodesics in the literature; we avoid this term to avoid ambiguity with other notions of "simple".



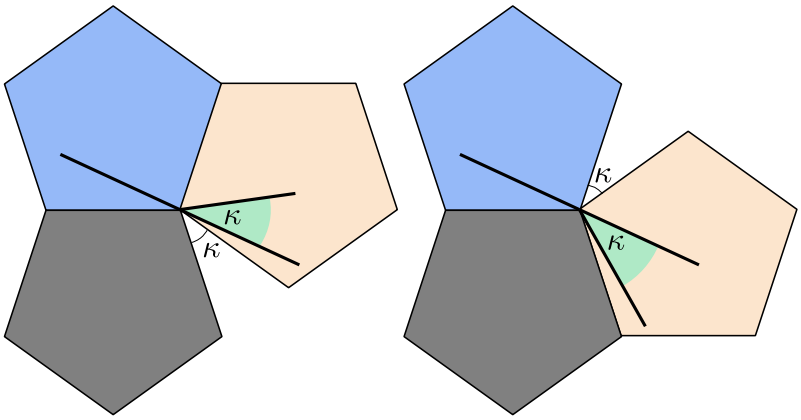


Figure 1 At a vertex of curvature κ, there is a κ-size interval of angles in which a segment of a quasigeodesic can be extended: the segment of geodesic starting on the left can continue straight in either of the pictured unfoldings, or any of the intermediate unfoldings in which the right pentagon touches only at a vertex.

For non-smooth surfaces (such as polyhedra), an analog of a geodesic is a ***quasigeodesic*** – a path with $\leq 180°$ of surface on both sides locally at every point along the path. Equivalently, a quasigeodesic can be locally unfolded to a straight line: on a face, a quasigeodesic is a straight line; at an edge, a quasigeodesic is a straight line after the faces meeting at that edge are unfolded (developed) flat at that edge; and at a vertex of curvature κ (that is, a vertex whose sum of incident face angles is $360° - \kappa$), a quasigeodesic entering the vertex at a given angle can exit it anywhere in an angular interval of length κ, as in Figure 1. Analogously, a ***closed quasigeodesic*** is a loop which is quasigeodesic. In 1949, Pogorelov [19] proved that every convex surface has at least three non-self-intersecting closed quasigeodesics, by applying the theory of quasigeodesics on smooth surfaces to smooth approximations of arbitrary convex surfaces and taking limits.

The existence proof of three closed quasigeodesics is nonconstructive, because the smooth argument uses a nonconstructive topological argument (a homotopy version of the intermediate value theorem).[2] In 1990, Joseph O'Rourke and Stacia Wyman posed the problem of finding a polynomial-time algorithm to find any closed quasigeodesic on a given convex polyhedron (aiming in particular for a non-self-intersecting closed quasigeodesic) [18]. This open problem was stated during the open problem session at SoCG 2002 (by O'Rourke) and finally appeared in print in 2007 [9, Open Problem 24.24]. Two negative results mentioned in [9] are that an n-vertex polyhedron can have $2^{\Omega(n)}$ non-self-intersecting closed quasigeodesics (an unpublished result by Aronov and O'Rourke) and that, for any k, there is a convex polyhedron whose shortest closed geodesic is not composed of k shortest paths (an unpublished result from the discussion at SoCG 2002).

Even a finite algorithm is not known or obvious. One approach is to argue that there is a closed quasigeodesic consisting of $O(n)$ (or any function $s(n)$) segments on faces. It seems plausible that the "short" closed quasigeodesics from the nonconstructive proofs satisfy

[2] A proof sketch for the existence of one closed geodesic on a smooth convex surface is as follows. By homotopy, there is a transformation of a small clockwise loop into its (counterclockwise) reversal that avoids self-intersection throughout. Consider the transformation that minimizes the maximum arclength of any loop during the transformation. By local cut-and-paste arguments, the maximum-arclength intermediate loop is in fact a closed geodesic. The same argument can be made for the nonsmooth case.

this property, but as far as we know the only proved property about them is that they are non-self-intersecting, which does not obviously suffice. (A quasigeodesic could plausibly wind many times around a curvature-bisecting loop, like the equator of a cube, somehow turn around, and symmetrically unwind, all without collisions.) If true, there are $O(n)^{s(n)}$ combinatorial types of quasigeodesics to consider, and each can be checked via the existential theory of the reals (in exponential time), resulting in an exponential-time algorithm. But we do not know how to bound $s(n)$; even the results of this paper give no upper bound on the number of segments constituting a closed quasigeodesic. In general, polyhedra such as isosceles tetrahedra have arbitrarily long non-self-intersecting closed geodesics (and even infinitely long non-self-intersecting geodesics) [14], so the only hope is to find an upper bound $s(n)$ on some (fewest-edge) closed quasigeodesic.

1.1 Our Results

We develop an algorithm that finds at least one[3] closed quasigeodesic on a given convex polyhedron in $O\left(\frac{n^2}{\varepsilon^2}\frac{L}{\ell}\right)$ real operations, where n is the number of vertices of the polyhedron, ε is the smallest curvature at a vertex, L is the length of the longest edge, and ℓ is the smallest distance within a face between a vertex and a nonincident edge (minimum feature size of any face). In the model described below in Section 1.2, these real operations take $O\left(\frac{n^2}{\varepsilon^2}\frac{L}{\ell}b\right)$ time if the input numbers are constant-size radical expressions over b-bit integers.

This running time is pseudopolynomial, so this does not yet resolve the open problem of a polynomial-time algorithm. The closed quasigeodesic output by our algorithm may be self-intersecting, even though a non-self-intersecting closed quasigeodesic is guaranteed to exist. Furthermore, the quasigeodesic path is output implicitly (in a format detailed below), as we lack a bound on the number $s(n)$ of needed segments. In Section 3, we discuss some of the difficulties involved in resolving either of these issues.

1.2 Models of Computation

Our results hold in several standard models of computation, which we pay careful attention to. While much work has been done on computational geometry in bounded-precision computational models [7, 8, 23], we are not aware of a single reference that describes all the relevant models, or with models that can handle representing polyhedra and geodesic paths. Thus we detail the possible model choices, distinguishing four aspects of the model:

1. **Combinatorial model of computation.** In combinatorial algorithms and data structures (without real numbers), there are two popular models of computation:

   a. **Pointer machine** [3, 10, 13]: Memory is decomposed into m records, each with a constant number of fields. Each field can store a w-bit integer or a pointer to another record. One record represents the machine's registers, and in constant time, the machine can read and write fields within constant pointer distance from that record, and create and/or destroy such a record.

[3] Our algorithm may in fact produce a list of closed quasigeodesics, but there are some closed quasigeodesics that it cannot find, including closed geodesics (not passing through a vertex) and possibly all non-self-intersecting closed quasigeodesics.

b. **Word RAM** [11]: Memory is an array of m words, each of which can store a w-bit integer. The first $O(1)$ words represent the machine's registers. In constant time, the machine can modify the register words or any word whose array index is given by a register word. The word RAM can simulate the pointer machine.

In both cases, the machine can also do basic w-bit integer arithmetic ($+, -, \times, \div,$ mod, AND, OR, NOT, $<, >, =$) in constant time, and we assume that $w = \Omega(\lg n)$ (the transdichotomous assumption, named for how it bridges the problem and machine [11]).

Our algorithm works in the pointer machine, and thus also in the word RAM.

2. **Real model of computation.** To deal with geometry (e.g., to represent the input), we need to handle some form of real numbers. We define three models of increasing generality that represent numbers in some binary format.

a. **Integers** (encoded in binary): Can be added, subtracted, multiplied, and compared in $O(b)$ time, where b is the total number of bits in the operands. Here we make the transdichotomous assumption that $w = \Omega(\lg b)$, in which case integer multiplication can be done in $O(b)$ time [16]. (Without this assumption, integer multiplication requires $O(b \log b)$ bit operations [12], so we would just gain a log factor in our running times.)
b. **Rationals** (encoded as two integers, numerator and denominator, in binary): Same performance as integers, plus real division in $O(b)$ time.
c. **Constant-size radical expressions over integers** (encoded as a constant-size expression tree with operators $+, -, \times, \div, \sqrt[k]{}$ and integer leaves): By known results in root separation bounds [6], if b is the total number of bits in the integer leaves, then the first $O(b)$ bits can be computed in $O(b)$ time; and if the expression is nonzero, some of these $O(b)$ bits will be nonzero, enabling exact comparison with 0. Thus, such numbers can be compared in $O(b)$ time, but when combining them arithmetically, we need to take care that the expression never grows beyond constant size. Also, we can compute the floor of such a number in $O(b)$ time.

Our algorithm works in the last model, and therefore supports inputs in any of the three models. Past work [7, 8] assumes $O(w)$-bit integer or rational inputs, which is less suitable for inputs of polyhedra, as described below.

These models contrast the real RAM model [22], where the inputs are black-box real numbers supporting radical operations $+, -, \times, \div, \sqrt[k]{}$ and comparisons in constant time. While standard in computational geometry, this model is not very realistic for a digital computer, because it does not bound the required precision, which can grow without bound. For example, the real RAM model crucially does not support converting black-box real numbers into integers (e.g., via the floor function), or else one can solve PSPACE [21] and #SAT [4] in polynomial time. Our algorithm actually needs to use the floor function, so it does not work on the "reasonable" real RAM model which lacks this operation.

3. **Polyhedron format.** The combinatorial structure of the polyhedron can be encoded as a primal or dual graph, as usual, but which real numbers should represent the geometry? Because the quasigeodesic problem is about the intrinsic geometry of the surface of a polyhedron, the input geometry can be naturally represented intrinsically as well as extrinsically, leading to three natural representations:

a. **Extrinsic coordinates:** 3D coordinates for each vertex.

b. **Intrinsic coordinates:** For each face, for some isometric embedding of the face into 2D, the 2D coordinates of each vertex of the embedded face.
c. **Intrinsic lengths:** For each face, the lengths of the edges. This representation assumes the faces have been combinatorially triangulated (so some edges may be flat).

In our real-number model of constant-size radical expressions over integers, extrinsic coordinates can be converted into intrinsic coordinates which can be converted from/to intrinsic lengths. Indeed, this feature is one of our motivations for this real-number model. (The reverse direction, from intrinsic to extrinsic, is more difficult, as it involves solving the Alexandrov problem [15].)

Our algorithm works in any of these input models.

4. **Output format.** Because we do not know any bounds on the number of segments (faces) in a closed quasigeodesic, we need to allow an implicit representation of the output. Specifically, we allow a quasigeodesic to be specified by a sequence of commands of the following form:

   follow a path R from vertex u to vertex v, traversing through some prefix of faces in the periodic sequence

   $$f_1, f_2, \ldots, f_m; f_1, f_2, \ldots, f_m; \ldots.$$

   We cannot specify the number of faces in the prefix, nor the length $\ell(R)$ of the path, but we can compute $\ell(R)$ with any desired precision: specifically, we can compute $\Omega(k)$ high-order bits of $\ell(R)$ in $O(k)$ time. We also cannot specify the direction that the path leaves u with exact precision, but we can compute $\Omega(k)$ high-order bits of the coordinates for a point p such that the direction of the path leaving u is toward p, again in $O(k)$ time. None of these quantities can be specified exactly, because the number of needed bits may be arbitrarily large, again because the path may visit arbitrarily many faces. But we can guarantee that such a path exists.

2 Algorithm

In this section, we give an algorithm to find a closed quasigeodesic on the surface of a convex polyhedron P. First, a bit of terminology: we define a ***(quasi)geodesic ray/segment*** to be a one/two-ended path that is (quasi)geodesic.

2.1 Outline

The idea of the algorithm is roughly as follows: first, we define a directed graph for which each node[4] is a pair $(V, [\varphi_1, \varphi_2])$ of a vertex V of P and a small interval of directions at it, with an edge from one such node, (U, I), to another, (V, J), if a geodesic ray starting at the polyhedron vertex U and somewhere in the interval of directions I can reach V and continue quasigeodesically everywhere in J[5]. We show how to calculate at least one out-edge from every node of that graph, so we can start anywhere and follow edges until hitting a node twice, giving a closed quasigeodesic.

[4] We use the word "node" and lower-case letters for vertices of the graph to distinguish them from vertices of a polyhedron, for which we use capital letters and the word "vertex".
[5] Since we consider only geodesic rays that can continue quasigeodesically *everywhere* in J, there are some closed quasigeodesics that we cannot find: those that leave a polyhedron vertex in a direction in an interval J for which some directions are not quasigeodesic continuations. In particular, this algorithm is unlikely to find closed quasigeodesics that turn maximally at a polyhedron vertex.

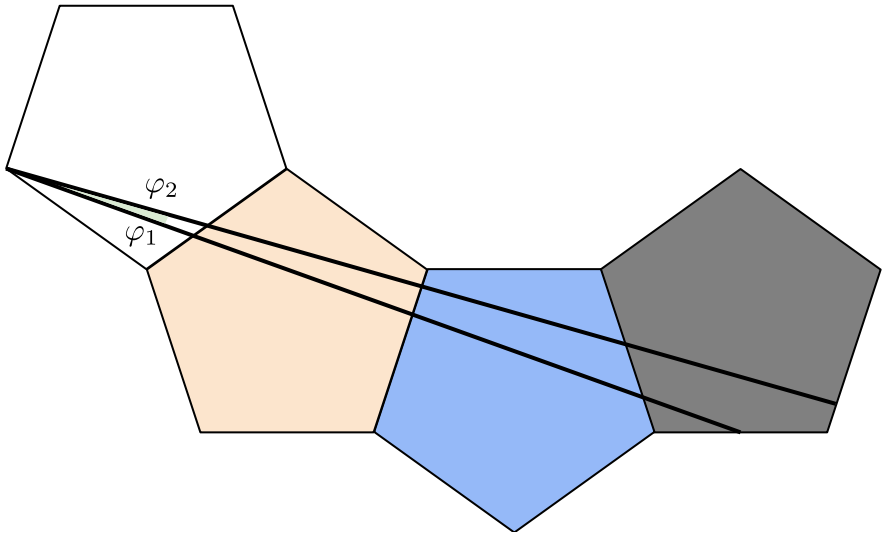


Figure 2 A segment of a geodesic is a straight line in the unfolding of the sequence of faces through which it passes, as in this unfolding of a regular dodecahedron.

The key part of this algorithm is to calculate, given a polyhedron vertex U and a range of directions as above, another vertex V that can be reached starting from that vertex and in that range of directions, even though reaching V may require crossing superpolynomially many faces. First we prove some lemmas toward that goal.

▶ **Definition 2.1.** *If X is a point on the surface of a polyhedron, φ is a direction at X, and $d > 0$, then $R(X, \varphi, d)$ is the geodesic segment starting at X in the direction φ and continuing for a distance d or until it hits a polyhedron vertex, whichever comes first.*[6] *We allow $d = \infty$; in that case, $R(X, \varphi, d)$ is a geodesic ray.*

▶ **Definition 2.2.** *If $R(X, \varphi, d)$ is a geodesic segment or ray, the* ***face sequence*** *$F(R(X, \varphi, d))$ is the (possibly infinite) sequence of faces that $R(X, \varphi, d)$ visits.*

▶ **Lemma 2.3.** *If $R_1 = R(X, \varphi_1, \infty)$ and $R_2 = R(X, \varphi_2, \infty)$ are two geodesic rays from a common starting point X with an angle between them of $\theta \in (0, \pi)$, the face sequences $F(R_1)$ and $F(R_2)$ are distinct, and the first difference between them occurs at most one face after a geodesic distance of $O(L/\theta)$.*

Proof. Given a (prefix of) $F(R_i)$, the segment of R_i on it is a straight line, so while $F(R_1) = F(R_2)$, the two geodesics R_1 and R_2 form a wedge in a common unfolding, as in Figure 2. The distance between the points on the rays at distance d from X is $2d \sin \frac{\theta}{2} > d\theta/\pi$ (since $\frac{\theta}{2} < \frac{\pi}{2}$), so at a distance of $O(L/\theta)$, that distance is at least L. So either $F(R_1)$ and $F(R_2)$ differ before then, or the next edge that R_1 and R_2 cross is a different edge, in which case $F(R_1)$ and $F(R_2)$ differ in the next face, as claimed. ◀

If we had defined L analogously to ℓ as not just the length of the longest edge but the greatest distance within a face between a polyhedron vertex and an edge not containing it, we could remove the "at most one face after" condition from Lemma 2.3.

[6] This definition is purely geometric; we reserve calculating these paths for Lemma 2.4.

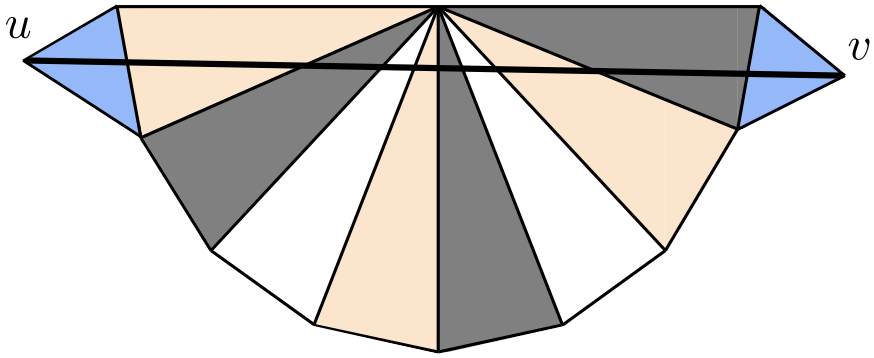


Figure 3 Even a short geodesic path between two vertices u and v may cross many edges. Equally colored faces represent copies of the same face being visited multiple times.

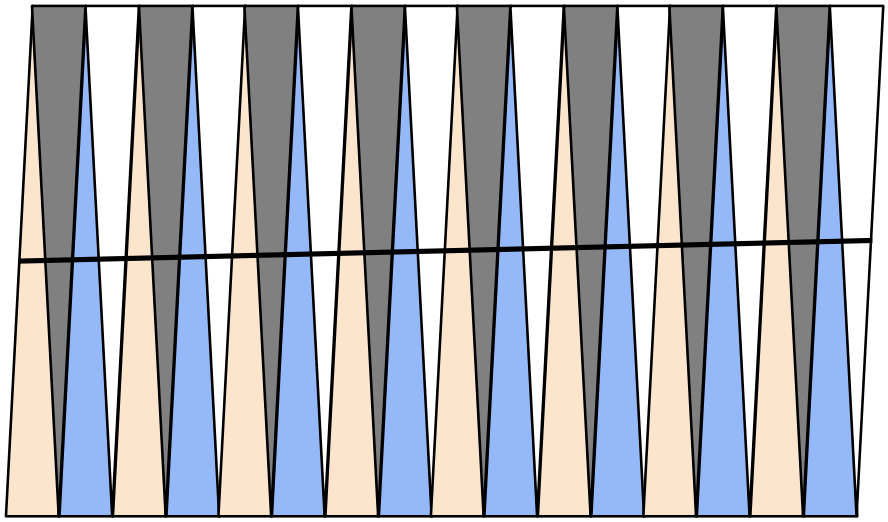


Figure 4 If a geodesic path encounters the same edge twice in nearly the same place and nearly the same direction, as is the case for the thick quasigeodesic path through the center of this figure if every fourth triangle is the same face, it may pass the same sequence of faces in the same order a superpolynomial number of times. Equally colored faces represent copies of the same face being visited multiple times.

2.2 Extending Quasigeodesic Rays

Although Lemma 2.3 gives a bound on the geodesic distance to the first difference in the face sequences (or one face before it), this gives no bound on the number of faces traversed before that difference, which might be large if the two paths come very close to a polyhedron vertex of high curvature, as in Figure 3, or repeat the same sequence of edges many times, as in Figure 4.

Nonetheless, in both of these cases, we can describe a geodesic ray's path efficiently:

▶ **Lemma 2.4.** *Let $R = (X, \varphi, d)$ be a geodesic segment with $d < \ell$. In $O(nb)$ time, we can calculate $F(R)$, expressed as a sequence S_1 of $O(n)$ faces, followed by another sequence S_2 of $O(n)$ faces and a distance over which R visits the faces of S_2 periodically*[7]*. Also, we can calculate the face, location in the face, and direction of R at its far endpoint (the one other than X).*

Proof. First, we prove the geometric fact (without calculating anything) that R is periodic from the first time it reenters any already-visited face. Second, we calculate, in $O(nb)$ time, the path of R through the non-periodic part, possibly detecting that R hits a vertex. Third, we calculate the path of R through the periodic part, in two cases: either R reenters each face at the same angle as its first entry, or not.

First, we claim that R is periodic from the first time it reenters a vertex: that is, if R enters a face f on an edge e_1 and exits[8] at a point P_2 on an edge e_2, then we claim that *every* time R enters f by e_1, it must exit f by e_2, and not any other edge e_3[9]. It must exit by a different edge from the edge e_1 by which it entered, so suppose for contradiction that in some visit to f, it enters at a point P_1 on the edge e_1 and exits at a point P_3 on another edge e_3, as shown in Figure 5. If any two of e_1, e_2, and e_3 are nonincident, then R has gone from a point on one edge to a point on a nonincident edge. By the definition of ℓ, R cannot do so without traveling a distance at least ℓ, farther than the conditions under which this lemma applies. Otherwise, e_1, e_2, and e_3 are the three edges of a triangular face, and the total geodesic distance is at least $d(P_1, P_2) + d(P_1, P_3)$. Consider the reflection e_4 of e_3 across e_2 and the reflected point P_4 on e_4. The path from P_4 to P_2 via P_1 is at least the distance from P_2 to P_4, which is at least the shortest distance from a point on e_4 to a point on e_2, which is attained at an endpoint of at least one of e_2 and e_4, say an endpoint of e_4. The path making that shortest distance (shown in gray) goes through e_1, so R travels at least the distance from e_1 to the opposite vertex, which is at least ℓ, farther than the conditions under which this lemma applies. Hence each edge crossed determines the next edge crossed, so $F(R)$ is periodic after crossing each edge at most once. Also, there are only $O(n)$ edges, so after crossing at most $O(n)$ edges, $F(R)$ repeats periodically with period $O(n)$.

Second, in total time $O(nb)$, we calculate the path of R before it repeats periodically in each face f it enters. Assume we start with an intrinsic representation of the polyhedron, with an isometric embedding of each face. We will represent the direction of a ray in a face by a pair of points, both with $O(b)$ bits, on the ray in the local coordinate system of that face; the points may or may not themselves be in the face.

Suppose that R enters f on an edge e' from a face f'. The isometry that takes the instance of e' in the embedding of f' to the instance of e' in the embedding of f is a linear transformation whose coefficients are $O(b)$ bits, which we can apply in $O(b)$ time to the pair of points representing R in f' to get a pair of points (x_0, y_0) and (x_1, y_1) representing R in f, again rounded to $O(b)$ bits.

For each edge e of f with endpoints (a, b) and (c, d), the intersection of the extension of R, which has equation $(x - x_1)(y_0 - y_1) = (x_0 - x_1)(y - y_1)$, and the extension of e, which has equation $(x - c)(b - d) = (a - c)(y - d)$, is a point (x, y) where each of x and y is a constant-depth arithmetic expressions in x_0, y_0, x_1, y_1, a, b, c, and d, so we can compute it in $O(b)$ time. Then we can check whether x is between a and c (or y is between b and d);

[7] The length of the sequence of faces may be too large to even write down the number of repetitions.

[8] If R hits a vertex of that polyhedron face f, we say that it exits on each of the two edges of f containing that vertex.

[9] In particular, R cannot exit by a vertex in any visit to f after the first.

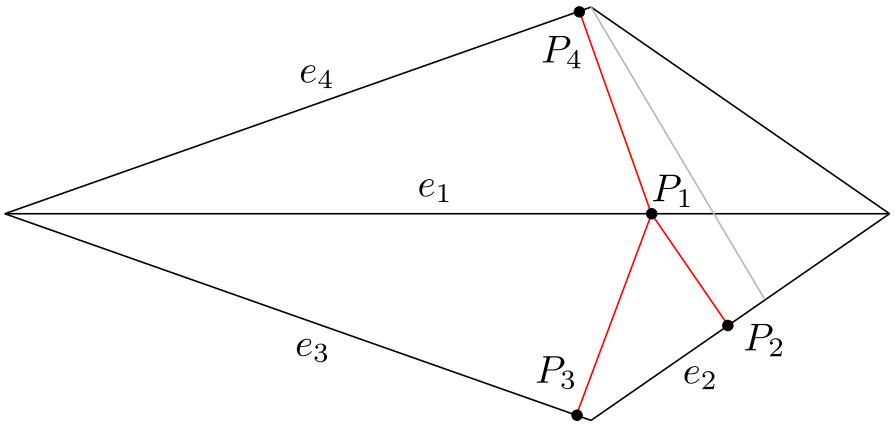


Figure 5 If a geodesic visits three edges of the same face, the total distance traveled is at least ℓ.

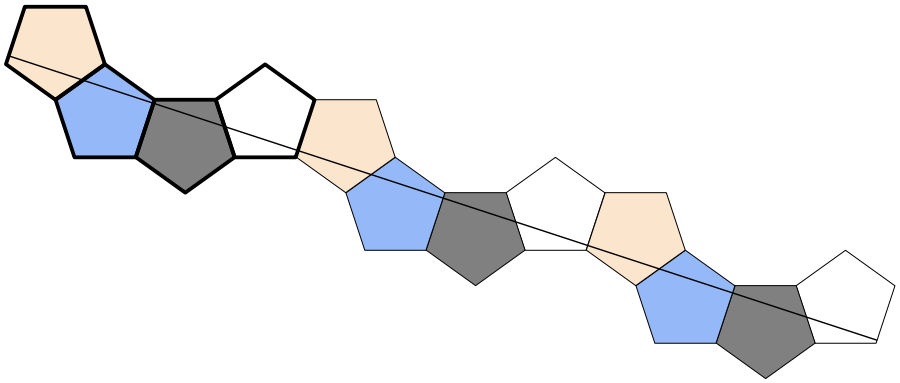


Figure 6 When a quasigeodesic path passes through the same sequence of faces several times, the unfolding of the faces it passes through repeats regularly.

having $O(b)$ bits of each is enough to do so by Section 1.2. This tells us whether R crosses e, and we have a pair of points in the embedding of f representing R, which is exactly what we need to calculate the path of R through the next face.

There are $O(n)$ pairs of a face and an edge of that face, so the total amount of computation before the face sequence repeats periodically is $O(nb)$. (If R ends at a polyhedron vertex before then, we calculate so because R exits a face by two edges at the same time, and we can compare the $O(b)$-bit leaving times in $O(b)$ time.)

Third, we calculate the periodic part of the path. Consider the shape formed by the faces $f_1, f_2, \ldots, f_k$ of $F(R)$ that repeat periodically, as in the bolded part of Figure 6. Copies of this shape attach to each other on copies of a repeated edge e; that is, the entire shape is translated and possibly rotated to identify the copies of e. The composition of the isometries that take f_1 to f_2, f_2 to f_3, and so on is an isometry that takes one copy of e to the next. By Section 1.2, we can check whether the slopes of two copies of e are equal (a case with no rotation) or not.

In the case with no rotation, as in Figure 6, all copies of each edge e are translates of each other by a constant amount, and we can describe all copies of e as line segments from $(x_0 + k\Delta x, y_0 + k\Delta y)$ to $(x_1 + k\Delta x, y_1 + k\Delta y)$ for some x_0, x_1, y_0, y_1, Δx, Δy, and all $k \in \mathbb{N}$.

Then, given the equation $(x - x_1)(y_0 - y_1) = (x_0 - x_1)(y - y_1)$ for R, we can calculate the intersection of R with the lines $(x - x_0)\Delta y = (y - y_0)\Delta x$ and $(x - x_1)\Delta y = (y - y_1)\Delta x$ in a constant number of arithmetic operations. One of those intersections is past the first copy of e and one is before it; without loss of generality, suppose that the one past the first copy of e is at $(x_0 + \kappa\Delta x, y_0 + \kappa\Delta y)$ for some $\kappa \in \mathbb{R}^+$. Then the last copy of e that R intersects is the one corresponding to $k = \lfloor \kappa \rfloor$. We can calculate that for each edge in the repeated sequence of faces (reusing the same calculated composition of isometries). The edge minimizing the resulting values of k (with ties broken by the first edge in the sequence of edges of $F(R)$) is the edge by which R leaves the periodic sequence of faces.

If there is rotation, all copies of each edge e are rotations around a consistent center point $C = (x_C, y_C)$. If the first three copies of one endpoint X of e are $X_0 = (x_0, y_0)$, $X_1 = (x_1, y_1)$, and $X_2 = (x_2, y_2)$, then we can calculate the equations of the bisectors of $\overline{X_0X_1}$ and $\overline{X_1X_2}$ in $O(1)$ arithmetic operations, so we can calculate their intersection, which is C. All copies of X are of the form $(x_C, y_C) + \sqrt{(x_0 - x_C)^2 + (y_0 - y_C)^2}(\cos(\theta_0 + k\Delta\theta), \sin(\theta_0 + k\Delta\theta))$ for some θ_0 and θ_1. (We calculate trig functions only precisely enough to take a floor: see below.) Then all copies of X are on the circle $(x - x_C)^2 + (y - y_C)^2 = (x_0 - x_C)^2 + (y_0 - y_C)^2$. We can calculate the (two) intersections of R with that circle in $O(1)$ operations. For each intersection $(x_C, y_C) + \sqrt{(x_0 - x_C)^2 + (y_0 - y_C)^2}(\cos(\theta_0 + \kappa\Delta\theta), \sin(\theta_0 + \kappa\Delta\theta))$, we can calculate $k = \lfloor \kappa \rfloor$ by calculating the first few bits of those trig functions (say, by Taylor expansions). Given such a k, the ray R intersects the kth copy of an edge e, then crosses the circle on which all copies of one endpoint of that edge are. However, if that crossing goes from outside to inside the circle, it may happen that R intersects both the kth and $(k + 1)$st copies of e, even though it left the circle in between them. So, check whether R intersects the $(k + 1)$st copy of e; if so, move on to the next-smallest value of k. There are at most $2n$ endpoints, so after $O(n)$ such operations, we find the first edge on which R leaves the periodic pattern of faces. ◀

▶ **Corollary 2.5.** *A geodesic segment $R(X, \varphi, d)$ can be implicitely representated by $O\left(\frac{d}{\ell}\right)$ subpaths, each of which visits a prefix of a periodic sequence of $O(n)$ faces, which can be computed in $O\left(n\frac{d}{\ell}b\right)$ time.*

Proof. Apply Lemma 2.4 to $R = R\left(X, \varphi, \frac{\ell}{2}\right)$ to generate a point X' and direction φ' of the endpoint of R other than X that is at least distance $\frac{\ell}{2}$ from X and traverses the prefix of some periodic sequence of $O(n)$ faces in $O(nb)$ time. Repeatedly appling Lemma 2.4 to $R\left(X', \varphi', \frac{\ell}{2}\right)$, and again at most $2d/\ell$ times proves the claim. ◀

2.3 Full Algorithm

We are now ready to state the algorithm for finding a closed quasigeodesic in quasipolynomial time:

▶ **Theorem 2.6.** *Let P be a convex polyhedron with n vertices all of curvature at least ε, let L be the length of the longest edge, let ℓ be the smallest distance within a face between a vertex and a nonincident edge, let b be the maximum number of bits of an integer in a constant-size radical expression of a real number representing P. Then, in $O\left(\frac{n^2}{\varepsilon^2}\frac{L}{\ell}b\right)$ time, we can find a closed quasigeodesic on P. The closed quasigeodesic can be implicitly represented by $O\left(\frac{n}{\varepsilon}\right)$ vertex-to-vertex paths, where each path is composed of $O\left(\frac{L}{\ell\varepsilon}\right)$ subpaths each of which visits some prefix of a periodic sequence of $O(n)$ faces.*

Proof. For each vertex V of P, divide the total angle at that vertex (that is, the angles at that vertex in the faces that meet at that vertex) into arcs of size between $\varepsilon/4$ and $\varepsilon/2 < \pi$, making $O(1/\varepsilon)$ such arcs at each vertex.

Construct a directed graph G whose nodes are pairs of a vertex V from P and one of its arcs I, giving the graph $O(n/\varepsilon)$ nodes, with an edge from a node $u = (U, I)$ to a node $v = (V, J)$ if there exists a direction in I such that a quasigeodesic ray starting in that direction from the polyhedron vertex U hits the polyhedron vertex V and can continue from every angle in J.

Let $v = (V, I)$ be a node of G, with corresponding vertex V and arc I spanning angles from φ_1 to φ_2. Compute face sequences for $R_1 = R(V, \varphi_1, L/\varepsilon)$ and $R_2 = R(V, \varphi_2, L/\varepsilon)$ and compare their face sequences $F(R_1)$ and $F(R_2)$. By Lemma 2.3, face sequences $F(R_1)$ and $F(R_2)$ differ somewhere, and their first difference determines a polyhedron vertex reachable in the wedge between R_1 and R_2 via a geodesic from V n a direction between angles φ_1 and φ_2, which can be found by scanning the sequencing. Once we reach such a vertex U, a quasigeodesic can exit the vertex anywhere in an angle equal to that vertex's curvature, which is at least ε, so for at least one of the arcs J of size at most $\varepsilon/2$ at that vertex, the quasigeodesic can exit anywhere in that arc, so we have found an outgoing edge from node v to node $u = (U, J)$.

The preceding algorithm computes an outgoing edge from any node in G, so we repeatedly traverse outgoing edges of G until a node of G is repeated. This cycle in G exactly corresponds to a closed quasigeodesic on the polyhedron, by the definition of the graph at the start of Section 2.1.

This algorithm computes $O(n/\varepsilon)$ edges of G (at most one for every graph node) before finding a cycle. To find an edge, the algorithm computes two face sequences $F(R_1)$ and $F(R_2)$, which by Corollary 2.5 can each be implicitly represented by $O\left(\frac{L}{\ell\varepsilon}\right)$ subpaths, each of which visits a prefix of a periodic sequence of $O(n)$ faces and can be computed in $O\left(\frac{n}{\varepsilon}\frac{L}{\ell}b\right)$ time. Then the geodesic corresponding to each edge of G can be computed through the longest common prefix of these face sequences in the same amount of time. Thus the whole geodesic can be described by $O(n/\varepsilon)$ such vertex-to-vertex paths, and can be constructed in $O\left(\frac{n^2}{\varepsilon^2}\frac{L}{\ell}b\right)$ time, as desired. ◀

If D is the greatest diameter of a face, then a closed quasigeodesic found by Theorem 2.6 has length $O\left(\frac{n}{\varepsilon}(\frac{L}{\varepsilon} + D)\right)$, because the quasigeodesic visits $O(n/\varepsilon)$ graph nodes and, by Lemma 2.3, goes a distance at most $L/\varepsilon + D$ between each consecutive pair.

3 Conclusion

It has been known for seven decades [19] that every convex polyhedron has a closed quasigeodesic, but our algorithm is the first finite algorithm to find one. We end with some open problems about extending our approach, though they all seem difficult.

▶ **Open Problem 1.** *Theorem 2.6 does not necessarily find a* non-self-intersecting *closed quasigeodesic, even though at least three are guaranteed to exist. Is there an algorithm to find one? In particular, can we find the* shortest *closed quasigeodesic?*

Any approach similar to Theorem 2.6 is unlikely to resolve this, for several reasons:

1. Parts of a quasigeodesic could enter a vertex at infinitely many angles. Theorem 2.6 makes this manageable by grouping similar angles of entry to a vertex, but if similar angles of entry to a vertex are combined, extensions that would be valid for some of them

but invalid for others are treated as invalid for all of them. For instance, a quasigeodesic found by Theorem 2.6 will almost never turn by the maximum allowed at any vertex, since exiting a vertex at the maximum possible turn from one entry angle to the vertex may mean exiting it with more of a turn than allowed for another very close entry angle. So there are some closed quasigeodesics not findable by Theorem 2.6, and those may include non-self-intersecting ones.

2. Given a vertex and a wedge determined by a range of directions from it, we can find *one* vertex in the wedge, but if we wish to find more than one, the problem becomes more complicated. When we seek only one vertex, we only need consider one unfolding of the faces, which the entire wedge stays in until it hits a vertex; when we pass a vertex, the unfoldings on each side of it might be different, so we multiply the size of the problem by 2 every time we pass a vertex. There may, in fact, be exponentially many non-self-intersecting geodesic paths between two vertices: for instance, Aronov and O'Rourke [9] give the example of a doubly covered regular polygon, in which a geodesic path may visit every vertex in order around the cycle but may skip vertices.

▶ **Open Problem 2.** *Theorem 2.6 is polynomial in not just n but the smallest curvature at a vertex, the length of the longest edge, and the shortest distance within a face between a vertex and an edge not containing it. Are all of those necessary? Can the last be simplified to the length of the shortest side?*

▶ **Open Problem 3.** *Can the algorithm of Theorem 2.6 be extended to nonconvex polyhedra?*

▶ **Open Problem 4.** *Is there an algorithm to find a closed quasigeodesic passing through a number of faces bounded by a polynomial function of n, ε, L, ℓ, and perhaps the minimum total angle of a polyhedron vertex? Does Theorem 2.6 already have such a bound?*

A single quasigeodesic ray may pass through a number of faces not bounded by a function of those parameters before ceasing to cycle periodically: for instance, the geodesic ray of Figure 4 does. However, we have no example for which a whole geodesic wedge passes through a number of faces not bounded by a function of those parameters before containing a vertex.

References

1 Hans Werner Ballmann. Der Satz von Lusternik und Schnirelmann. *Bonner Mathematische Schriften*, 102:1–25, 1978.

2 Werner Ballmann, Gudlaugur Thorbergsson, and Wolfgang Ziller. On the existence of short closed geodesics and their stability properties. In *Seminar on Minimal Submanifolds*, pages 53–63. Princeton University Press, 1983. URL: `https://www.researchgate.net/profile/Wolfgang_Ziller/publication/268500145_On_the_existence_of_short_closed_geodesics_and_their_stability_properties/links/58de87a7a6fdcc41bf8e987f/On-the-existence-of-short-closed-geodesics-and-their-stability-properties.pdf`.

3 Amir M. Ben-Amram. What is a "pointer machine"? *SIGACT News*, 26(2):88–95, June 1995. `doi:10.1145/202840.202846`.

4 A. Bertoni, G. Mauri, and N. Sabadini. Simulations among classes of random access machines and equivalence among numbers succinctly represented. *Annals of Discrete Mathematics*, 25:65–90, 1985.

5 George D. Birkhoff. *Dynamical Systems*, volume 9 of *Colloquium Publications*. American Mathematical Society, 1927. URL: `https://bookstore.ams.org/coll-9`.

6 Christoph Burnikel, Stefan Funke, Kurt Mehlhorn, Stefan Schirra, and Susanne Schmitt. A separation bound for real algebraic expressions. In *Proceedings of the 9th Annual European Symposium on Algorithms*, volume 2161 of *Lecture Notes in Computer Science*, pages 254–265, Aarhus, Denmark, August 2001. URL: `http://graphics.stanford.edu/~sfunke/Papers/ESA01/sepbound01.pdf`.

7 Timothy M. Chan and Mihai Pătraşcu. Transdichotomous results in computational geometry, I: Point location in sublogarithmic time. *SIAM Journal on Computing*, 39(2):703–729, 2009. doi:10.1137/07068669X.

8 Timothy M. Chan and Mihai Pătraşcu. Transdichotomous results in computational geometry, II: Offline search, 2010. Originally published at STOC 2007. arXiv:1010.1948.

9 Erik D. Demaine and Joseph O'Rourke. Geodesics: Lyusternik-Schnirelmann. In *Geometric Folding Algorithms: Linkages, Origami, Polyhedra*, section 24.4, pages 372–375. Cambridge University Press, Cambridge, 2007.

10 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.

11 Michael L. Fredman and Dan E. Willard. Surpassing the information theoretic bound with fusion trees. *Journal of Computer and System Sciences*, 47(3):424–436, 1993. doi:10.1016/0022-0000(93)90040-4.

12 David Harvey and Joris van der Hoeven. Integer multiplication in time $O(n \log n)$. HAL Preprint hal-02070778, 2019. URL: https://hal.archives-ouvertes.fr/hal-02070778.

13 Giuseppe F. Italiano and Rajeev Raman. Topics in data structures. In Mikhail J. Atallah and Marina Blanton, editors, *Algorithms and Theory of Computation Handbook*, volume 1, chapter 5, pages 5-1–5-29. CRC Press, second edition, 2010.

14 Jin-Ichi Itoh, Joël Rouyer, and Costin Vîlcu. Polyhedra with simple dense geodesics. *Differential Geometry and its Applications*, 66:242–252, 2019. doi:10.1016/j.difgeo.2019.07.001.

15 Daniel Kane, Gregory N. Price, and Erik D. Demaine. A pseudopolynomial algorithm for Alexandrov's Theorem. In *Proceedings of the 11th Algorithms and Data Structures Symposium*, volume 5664 of *Lecture Notes in Computer Science*, pages 435–446, Banff, Canada, August 2009.

16 Donald E. Knuth. *The Art of Computer Programming*, volume 2. Addison-Wesley, 1969.

17 Lazar Lyusternik and Lev Schnirelmann. Sur le probléme de trois géodésiques fermées sur les surfaces de genre 0. *Comptes Rendus de l'Académie des Sciences de Paris*, 189:269–271, 1929.

18 Joseph O'Rourke. Personal communication, 2020.

19 Aleksei Vasilevich Pogorelov. Quasi-geodesic lines on a convex surface. *Matematicheskii Sbornik*, 25(62):275–306, 1949. English translation in *American Mathematical Society Translations* 74, 1952.

20 Henri Poincaré. Sur les lignes géodésiques des surfaces convexes. *Transactions of the American Mathematical Society*, 6(3):237–274, 1905.

21 Arnold Schönhage. On the power of random access machines. In *Proceedings of the 6th International Colloquium on Automata, Languages, and Programming*, volume 71 of *Lecture Notes in Computer Science*, pages 520–529, 1979.

22 Michael Ian Shamos. *Computational Geometry*. PhD thesis, Yale University, 1978.

23 Vikram Sharma and Chee K. Yap. Robust geometric computation. In Jacob E. Goodman, Joseph O'Rourke, and Csaba D. Tóth, editors, *Handbook of Discrete and Computational Geometry*, chapter 45, pages 1189–1223. CRC Press, third edition, 2018.

The Stretch Factor of Hexagon-Delaunay Triangulations

Michael Dennis
Computer Science Division, University of California at Berkeley, Berkeley, CA, USA
michael_dennis@cs.berkeley.edu

Ljubomir Perković
School of Computing, DePaul University, Chicago, IL, USA
lperkovic@cs.depaul.edu

Duru Türkoğlu
School of Computing, DePaul University, Chicago, IL, USA
duru@cs.uchicago.edu

Abstract

The problem of computing the exact stretch factor (i.e., the tight bound on the worst case stretch factor) of a Delaunay triangulation is one of the longstanding open problems in computational geometry. Over the years, a series of upper and lower bounds on the exact stretch factor have been obtained but the gap between them is still large. An alternative approach to solving the problem is to develop techniques for computing the exact stretch factor of "easier" types of Delaunay triangulations, in particular those defined using regular-polygons instead of a circle. Tight bounds exist for Delaunay triangulations defined using an equilateral triangle and a square. In this paper, we determine the exact stretch factor of Delaunay triangulations defined using a regular hexagon: It is 2. We think that the main contribution of this paper are the two techniques we have developed to compute tight upper bounds for the stretch factor of Hexagon-Delaunay triangulations.

2012 ACM Subject Classification Theory of computation → Computational geometry; Theory of computation → Sparsification and spanners; Theory of computation → Shortest paths

Keywords and phrases Delaunay triangulation, geometric spanner, plane spanner, stretch factor, spanning ratio

Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.34

Related Version A full version of the paper is available at [7], `https://arxiv.org/abs/1711.00068`.

1 Introduction

In this paper we consider the problem of computing a tight bound on the worst case stretch factor of a Delaunay triangulation. Given a set P of points on the plane, the Delaunay triangulation T on P is a plane graph such that for every pair $u, v \in P$, (u, v) is an edge of T if and only if there is a *circle* passing through u and v with no point of P in its interior. (This definition assumes that points in P are in general position which we discuss in Section 2.) In this paper, we refer to Delaunay triangulations defined using the *circle* as ○-Delaunay triangulations. The ○-Delaunay triangulation T of P is a plane subgraph of the complete, weighted Euclidean graph $\mathcal{E}^P$ on P in which the weight of an edge is the Euclidean distance between its endpoints. Graph T is also a *spanner*, defined as a subgraph of $\mathcal{E}^P$ with the property that the distance in the subgraph between any pair of points is no more than a constant multiplicative ratio of the distance in $\mathcal{E}^P$ between the points. The constant ratio is referred to as the *stretch factor* (or *spanning ratio*) of the spanner.

The problem of computing a tight bound on the worst case stretch factor of the ○-Delaunay triangulation has been open for more than three decades. In the 1980s, when ○-Delaunay triangulations were not known to be spanners, Chew considered related, "easier"

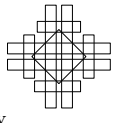



Table 1 Key stretch factor upper bounds (tight bounds are bold).

Paper	Graph	Upper Bound
[8]	○-Delaunay	$\pi(1+\sqrt{5})/2 \approx 5.08$
[9]	○-Delaunay	$4\pi/(3\sqrt{3}) \approx 2.41$
[10]	○-Delaunay	1.998
[6]	△-Delaunay	**2**
[5]	□-Delaunay	$\sqrt{10} \approx 3.16$
[2]	□-Delaunay	$\sqrt{\mathbf{4+2\sqrt{2}}} \approx \mathbf{2.61}$
[This paper]	⬡-Delaunay	**2**

structures. In 1986 [5], Chew proved that a □-Delaunay triangulation – defined using a *fixed-orientation square* instead of a circle – is a spanner with stretch factor at most $\sqrt{10}$. Following this, Chew proved that the △-Delaunay triangulation – defined using a *fixed-orientation equilateral triangle* – has a stretch factor of 2 [6]. Significantly, this bound is tight: one can construct △-Delaunay triangulations with stretch factor arbitrarily close to 2. Finally, Dobkin et al. [8] showed that the ○-Delaunay triangulation is a spanner as well. The bound on the stretch factor they obtained was subsequently improved by Keil and Gutwin [9] as shown in Table 1. The bound by Keil and Gutwin stood unchallenged for many years until Xia recently improved the bound to below 2 [10].

On the lower bound side, some progress has been made on bounding the worst case stretch factor of a ○-Delaunay triangulation. The trivial lower bound of $\pi/2 \approx 1.5707$ has been improved to 1.5846 [4] and then to 1.5932 [11].

After three decades of research, we know that the worst case stretch factor of ○-Delaunay triangulations is somewhere between 1.5932 and 1.998. Unfortunately, the techniques that have been developed so far seem inadequate for proving a tight stretch factor bound.

Rather than attempting to improve further the bounds on the stretch factor of ○-Delaunay triangulations, we follow an alternative approach. Just like Chew turned to △- and □-Delaunay triangulations to develop insights useful for showing that ○-Delaunay triangulations are spanners, we make use of Delaunay triangulations defined using regular polygons to develop techniques for computing tight stretch factor bounds. Delaunay triangulations based on regular polygons are known to be spanners (Bose et al. [3]). Tight bounds are known for △-Delaunay triangulations [6] and also for □-Delaunay triangulations (Bonichon et al. [2]) as shown in Table 1.

In this paper, we show that the worst case stretch factor of ⬡-Delaunay triangulations is 2. We present an overview of our proof in Section 3. The overview makes use of three lemmas whose detailed proofs are omitted; the proofs (briefly discussed in Sections 4, 5, and 6) are in the full version of the paper [7]. We think that our main contribution consists of two techniques that we use to compute tight upper bounds on the stretch factor of particular types of ⬡-Delaunay triangulations. In Section 7 we review the role of the techniques in the paper and explore their potential to be applied to other kinds of Delaunay triangulations.

2 Preliminaries

We consider a finite set P of points in the two-dimensional plane with an orthogonal coordinate system. The x- and y-coordinates of a point p will be denoted by $\mathtt{x}(p)$ and $\mathtt{y}(p)$, respectively. The Euclidean graph $\mathcal{E}^P$ of P is the complete weighted graph embedded in the plane whose

nodes are identified with the points of P. For every pair of nodes p and q, the edge (p, q) represents the segment $[pq]$ and the weight of (p, q) is the Euclidean distance between p and q which is $d_2(p, q) = \sqrt{(\mathtt{x}(p) - \mathtt{x}(q))^2 + (\mathtt{y}(p) - \mathtt{y}(q))^2}$. Our arguments also use the x-coordinate distance between p and q which we denote as $d_x(p, q) = |\mathtt{x}(p) - \mathtt{x}(q)|$.

Let T be a subgraph of $\mathcal{E}^P$. The length of a path in T is the sum of the weights of the edges of the path and the distance $d_T(p, q)$ in T between two points p and q is the length of the shortest path in T between them. T is a t-spanner for some constant $t > 0$ if for every pair of points p, q of P, $d_T(p, q) \leq t \cdot d_2(p, q)$. The constant t is referred to as the *stretch factor* of T.

We define a *family of spanners* to be a set of graphs T^P, one for every finite set P of points in the plane, such that for some constant $t > 0$, every T^P is a t-spanner of $\mathcal{E}^P$. We say that the stretch factor t is exact (tight) for the family (or that the worst case stretch factor is t) if for every $\epsilon > 0$ there exists a set of points P such that T^P is *not* a $(t - \epsilon)$-spanner of $\mathcal{E}^P$.

The families of spanners we consider are various types of Delaunay triangulations on a set P of points in the plane. Given a set P of points on the plane, we say that a convex, closed, simple curve in the plane is empty if it contains no point of P in its interior. The ○-Delaunay triangulation T on P is defined as follows: For every pair $u, v \in P$, (u, v) is an edge of T if and only if there is an empty *circle* passing through u and v. (This definition assumes that points are in general position which in the case of ○-Delaunay triangulations means that no four points of P are co-circular.) If, in the definition, *circle* is replaced by *fixed-orientation square* (e.g., a square whose sides are axis-parallel) or by *fixed-orientation equilateral triangle* then different triangulations are obtained: the □- and the △-Delaunay triangulations.

If, in the definition of the ○-Delaunay triangulation, we change *circle* to *fixed-orientation regular hexagon*, then a ⬡-Delaunay triangulation is obtained. In this paper we focus on such triangulations. While any fixed orientation of the hexagon is possible, we choose w.l.o.g. the orientation that has two sides of the hexagon parallel to the y-axis as shown in Fig. 1-(a). In the remainder of the paper, *hexagon* will always refer to a regular hexagon with such an orientation. We find it useful to label the vertices of the hexagon N, E_N, E_S, S, W_S, and W_N, in clockwise order and starting with the top one. We also label the sides n_e, e, s_e, s_w, w, and n_w as shown in Fig. 1-(a); we will sometimes refer to the s_e and s_w sides as the s sides and to the n_e and n_w sides as the n sides.

The definition of the ⬡-Delaunay triangulation assumes that no four points lie on the boundary of an empty hexagon. Our arguments also assume that no two points lie on a line whose slope matches the slope of a side of the hexagon (i.e. slopes ∞, $\frac{1}{\sqrt{3}}$, $-\frac{1}{\sqrt{3}}$). The *general position* assumption we therefore make in this paper consists of the above two restrictions. This assumption is made solely for the purpose of simplifying the presentation; the arguments in the paper could be extended so the results apply to all ⬡-Delaunay triangulations. Finally, we need to be aware that unlike the ○-Delaunay triangulation on P, the ⬡-Delaunay (and also the □- and △-Delaunay) triangulation on P may not contain all edges on the convex hull of P. To handle this and simplify our arguments, we add to P six additional points, very close to but not exactly (in order to satisfy the above assumptions) at coordinates $(0, \pm M)$ and $(\pm M \cos(\pi/6), \pm M \sin(\pi/6))$ where $M > 50 \max_{s,t \in P} d_2(s, t)$. The ⬡-Delaunay triangulation on this modified set of points P, consisting of the original triangulation plus additional edges between the new points and original points and also between the new points themselves, includes the edges on the convex hull of P. Also, any path in this triangulation between two points s and t from the original set P with length bounded by $2d_2(s, t)$ cannot possibly use the added points. Thus a proof of our main result for the modified triangulation will also be a proof for the original one and so we assume that the ⬡-Delaunay triangulation on P includes the edges on the convex hull of P.

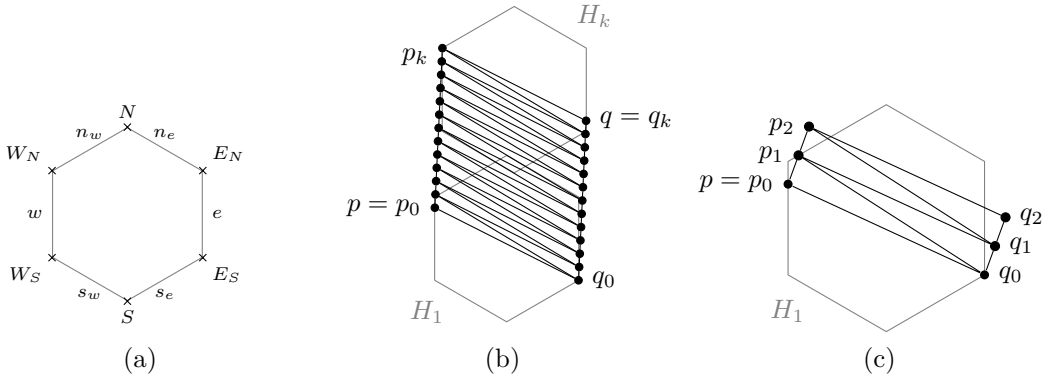


Figure 1 (a) The hexagon orientation and the side and vertex labels that we use (b) A ⬡-Delaunay triangulation with points p, q, p_k, and q_0 having coordinates $(0,0)$, $(1, \frac{1}{\sqrt{3}})$, $(\delta, \frac{2}{\sqrt{3}} - \sqrt{3}\delta)$, and $(1-\delta, -\frac{1}{\sqrt{3}} + \sqrt{3}\delta)$, respectively. For δ small enough, $d_T(p,q) \geq (2-\epsilon)d_2(p,q)$. (c) A closer look at the bottom faces of this triangulation.

We end this section with a lower bound, by Bonichon [1], on the worst case stretch factor of ⬡-Delaunay triangulations. The lower bound construction is illustrated in Fig. 1-(b) and Fig. 1-(c). The proof is omitted but appears in the full version of the paper [7].

▶ **Lemma 1.** *For every $\varepsilon > 0$, there exists a set P of points in the plane such that the ⬡-Delaunay triangulation on P has stretch factor at least $2 - \varepsilon$.*

3 Main result

In this section we state our main result and provide an overview of our proof. We start with a technical lemma that is used to prove the two key lemmas needed for the main result.

3.1 Technical lemma

Let T be the ⬡-Delaunay triangulation on a set of points P in the plane.

▶ **Definition 2.** *Let $T_1, T_2, \ldots, T_n$ be a sequence of triangles of T that a line st of finite slope intersects. This sequence of triangles is said to be* linear *w.r.t. line st if for every $i = 1, \ldots, n-1$:*

- *triangles T_i and T_{i+1} share an edge, and*
- *line st intersects the interior of that shared edge (not an endpoint).*

Our goal is to prove an upper bound on the length of the shortest path from the "leftmost" point of T_1 to the "rightmost" point of T_n, when certain conditions hold. We introduce some notation and definitions, illustrated in Fig. 2, to make this more precise.

We consider the $n-1$ shared triangle edges intersected by line st from left to right (where left and right are defined with respect to x-coordinates) and label the endpoints of the i-th edge u_i and l_i, with u_i being above line st and l_i below. We note that points typically get multiple labels and identify a point with its label(s). If line st goes through the vertex of T_1 other than u_1 and l_1, we assign that vertex both labels u_0 and l_0 (as shown in Fig. 4); otherwise, we assign labels u_0 and l_0 to the endpoints of the edge of T_1 intersected by line st other than (u_1, l_1), with u_0 being above line st (as shown in Fig. 2). Similarly, if line st goes through the vertex of T_n other than u_{n-1} and l_{n-1}, we assign it both labels u_n and l_n (as

shown in Fig. 4); otherwise, we assign labels u_n and l_n to the endpoints of the other edge of T_n intersected by line st, with u_n being above line st (as shown in Fig. 2). Note that for $1 \leq i \leq n$:

- either $T_i = \triangle(u_i, l_i, l_{i-1})$, in which case we call l_{i-1} and l_i the *left* and *right* vertices of T_i
- or $T_i = \triangle(u_{i-1}, u_i, l_i)$, in which case we call u_{i-1} and u_i the *left* and *right* vertices of T_i.

Note that only one of the above holds, except for T_1 if $u_0 = l_0$ (in which case both hold) or for T_n if $u_n = l_n$ (in which case again both hold). For every $i = 1, \ldots, n$, when $u_i = u_{i-1}$ or $l_i = l_{i-1}$ we call the corresponding vertex of T_i the *base* vertex of T_i. Note that T_1 has no base vertex if $u_0 = l_0$ and T_n has no base vertex if $u_n = l_n$ (as is the case in Fig. 4). Let U and L be the sets of all point labels u_i and l_i, respectively, and let T_{1n} be the union of $T_1, T_2, \ldots, T_n$ which we will refer to as a linear sequence of triangles as well.

Let H_i, for $1 \leq i \leq n$, be the (empty) hexagon passing through the vertices of T_i; note that this hexagon is unique due to the general position assumption. A vertex of T_i is said to be a w, e, n, or s vertex of T_i if it lies on the w side, e side, one of the n sides, or one of the s sides, respectively, of H_i (see Fig. 2). A left vertex of T_i that is a w vertex of T_i is referred to as a *left induction vertex* of T_i; similarly, a right vertex that is a e vertex is referred to as a *right induction vertex* of T_i.

Note that a base vertex cannot be an induction vertex.

▶ **Definition 3.** *We call an edge* (u_i, l_j) gentle *if its slope is between* $-\frac{1}{\sqrt{3}}$ *and* $\frac{1}{\sqrt{3}}$.

In Fig. 2 no edge (u_i, l_j) is gentle while in Fig. 4 (u_0, l_1) and (u_8, l_8) are gentle.

▶ **Definition 4.** *The linear sequence of triangles* T_{1n} *is* regular *if* T_1 *has a left induction vertex,* T_n *has a right induction vertex, and if, for every* $i = 1, \ldots, n-1$*:*

- u_i *is* not *a* s *vertex of* T_i *and* T_{i+1}*,*
- l_i *is* not *a* n *vertex of* T_i *and* T_{i+1}*, and*
- (u_i, l_i) *is not gentle.*

The linear sequence in Fig. 2 is regular while the one in Fig. 4 is not (because u_8 lies on the s_w side of H_9 – the red hexagon passing through the vertices of $T_9 = \triangle(u_8, u_9, l_9)$ – and also because edge (u_8, l_8) is gentle).

The proof of the following technical lemma is discussed in Section 5.

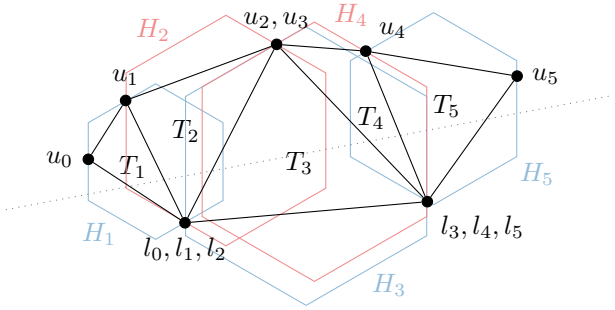


Figure 2 The dotted line (st) intersects the linear sequence of triangles $T_1, T_2, \ldots, T_5$. The vertices of each triangle T_i (u_{i-1}, u_i, l_{i-1}, l_i, two of which are equal) lie on the boundary of the hexagon H_i. Note that l_2 is the left, l_3 is the right, and $u_2 = u_3$ is the base vertex of T_3, for example. The linear sequence is regular since T_1 has a left induction vertex, T_5 has a right induction vertex, and, for $i = 1, \ldots, 4$, no u_i is a s vertex of T_i or T_{i+1}, no l_i is a n vertex of T_i or T_{i+1}, and no (u_i, l_i) is gentle.

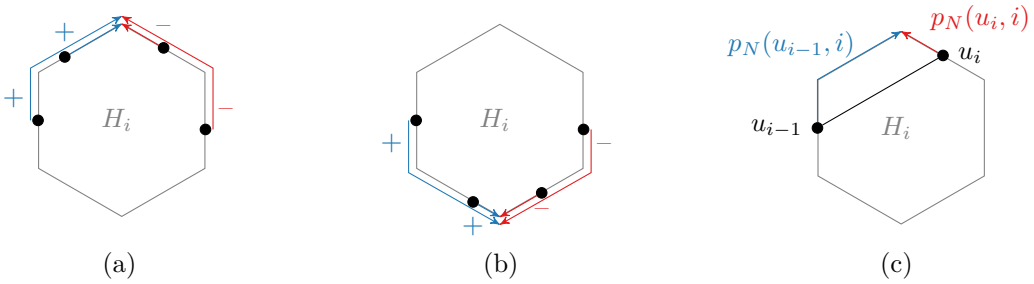


Figure 3 (a) The values of $p_N(o, i)$ are illustrated, for various points o lying on the boundary of H_i, as signed hexagon arc lengths. (b) The values of $p_S(o, i)$ are illustrated similarly.

▶ **Lemma 5** (The Technical Lemma). *If T_{1n} is a regular linear sequence of triangles then there is a path in T_{1n} from the left induction vertex p of T_1 to the right induction vertex q of T_n of length at most $\frac{4}{\sqrt{3}} d_x(p, q)$.*

Actually, what we show in Section 5 implies something stronger: If T_{1n} is regular then the lengths of the *upper* path $p, u_0, \ldots, u_n, q$ and of the *lower* path $p, l_0, \ldots, l_n, q$ add up to at most $\frac{8}{\sqrt{3}} d_x(p, q)$. It is useful to informally describe now the techniques we use to do this. For that purpose we introduce, for a point o on a side of H_i, functions $p_N(o, i)$ and $p_S(o, i)$ as the *signed* shortest distances around the perimeter of H_i from o to the N vertex and S vertex, respectively; the sign is positive if o lies on sides n_w, w, or s_w of H_i and negative otherwise (see Fig. 3-(a) and Fig. 3-(b)).

Note that the length of each edge (u_{i-1}, u_i) (assuming $u_{i-1} \neq u_i$) can be bounded by the distance from u_{i-1} to u_i when traveling clockwise along the sides of H_i. This distance is exactly $p_N(u_{i-1}, i) - p_N(u_i, i)$ as illustrated in Fig. 3-(c). This motivates the following *discrete* function, defined for $i = 0, 1, \ldots, n$ and, for convenience's sake, 1) assuming that $p = u_0$ and $q = u_n$ and 2) using an additional hexagon H_{n+1} of radius 0 centered at point q:

$$\bar{U}(i) = \sum_{j=1}^{i} (p_N(u_{j-1}, j) - p_N(u_j, j)) + p_N(u_i, i + 1).$$

Function $\bar{U}(i)$ can be used to bound the length of upper path fragments; in particular, $\bar{U}(n)$ bounds the length of the upper path from p to q. A function $\bar{L}(i)$ bounding the length of the lower path can be defined similarly. In Section 5, we will compute an upper bound for $\bar{U} + \bar{L}$ by 1) switching the analysis from a discrete one to a continuous one, with functions p_N and p_S defined not in term of index i but in terms of coordinate x for every x between $\mathtt{x}(p)$ and $\mathtt{x}(q)$ and 2) analyzing the growth rates, with respect to x, of the continuous functions p_N, p_S, and $\bar{U} + \bar{L}$. We will show that (the continuous versions of) p_N and p_S are piecewise linear functions with growth rates $\frac{2}{\sqrt{3}}$, $\frac{4}{\sqrt{3}}$, or $\frac{6}{\sqrt{3}}$, and that $\bar{U} + \bar{L}$ is also piecewise linear with growth rate equal to the growth rate of $p_N + p_S$ which can be $\frac{4}{\sqrt{3}}$, $\frac{6}{\sqrt{3}}$, or $\frac{8}{\sqrt{3}}$. Lemma 5 will follow from the last (largest) growth rate.

With the technical lemma in hand, we can now state the first of the two key lemmas that we need to prove our main result.

3.2 The amortization lemma

The first of our two key lemmas is a strengthening of the (Technical) Lemma 5 under two restrictions. The first restriction is that T_{1n} is defined with respect to a line st whose slope m_{st} is restricted to $0 < m_{st} < \frac{1}{\sqrt{3}}$. With that restriction we get the following properties:

▶ **Lemma 6.** *Let T_{1n} be a linear sequence with respect to line st with slope m_{st} such that $0 < m_{st} < \frac{1}{\sqrt{3}}$. For every i s.t. $1 \leq i \leq n$:*

- *If u_{i-1} lies on side s_w of H_i or l_i lies on side n_e of H_i then (u_{i-1}, l_i) is gentle.*
- *If l_{i-1} lies on side n_w of H_i or u_i lies on the s_e side of H_i then (l_{i-1}, u_i) is gentle.*
- *None of the following can occur: u_{i-1} lies on side s_e of H_i, l_i lies on side n_w of H_i, l_{i-1} lies on side n_e of H_i, and u_i lies on the s_w side of H_i.*

Note, for example, that u_8 lies on side s_w of hexagon H_9 in Fig. 4 and that edge (u_8, l_9) is gentle.

Proof. If u_{i-1} lies on side s_w of some hexagon H_i then, since $0 < m_{st} < \frac{1}{\sqrt{3}}$ and by general position assumptions, either $u_{i-1} = u_i$ and l_{i-1} and l_i must lie on sides s_e and e of H_i, respectively, or $l_{i-1} = l_i$ must lie on side s_e or e of H_i. Either way, the slope of the line going through u_{i-1} and l_i must be between $-\frac{1}{\sqrt{3}}$ and $\frac{1}{\sqrt{3}}$. Similar arguments can be used to handle the remaining three cases in the first two bullet points.

Let the left and right intersection points of line st with hexagon H_i be h_{i-1} and h_i. Note that when traveling clockwise along the sides of H_i the points will be visited in this order: $h_{i-1}, u_{i-1}, u_i, h_i, l_i, l_{i-1}$. If u_{i-1} lies on side s_e of H_i then $i > 1$ and, because $0 < m_{st} < \frac{1}{\sqrt{3}}$, either u_i (if $u_{i-1} \neq u_i$) or l_i (if $l_{i-1} \neq l_i$) would have to lie on side s_e of H_i as well, which violates our general position assumption for the set of points P. The remaining three cases are handled similarly. ◀

By the above lemma, under the restriction $0 < m_{st} < \frac{1}{\sqrt{3}}$, if T_{1n} has no gentle edge then it is regular and (Technical) Lemma 5 applies. A narrower but much stronger version of (Technical) Lemma 5 applies as well if another restriction is made. To state the second restriction we need some additional terminology.

Let $l_i \in L$ and $u_j \in U$. If $i \leq j$ and $\mathtt{x}(l_i) < \mathtt{x}(u_j)$ then we say that l_i *occurs before* u_j, and if $j \leq i$ and $\mathtt{x}(u_j) < \mathtt{x}(l_i)$ then we say that u_j *occurs before* l_i.

▶ **Definition 7.** *Given points $l_i \in L$ and $u_j \in U$ such that one occurs before the other, a path between them is* gentle *if the length of the path is not greater than $\sqrt{3}d_x(u_j, l_i) - (\boldsymbol{y}(u_j) - \boldsymbol{y}(l_i))$.*

See Fig. 4 for an illustration of a gentle path. Note that a gentle edge is a gentle path (e.g., (u_0, l_1) and $(u_8, l_8 = l_9)$ in Fig. 4).

The following is the key to our proof of the main result of this paper:

▶ **Lemma 8** (The Amortization Lemma)**.** *Let T_{1n} be a regular linear sequence with respect to line st with slope m_{st}. If $0 < m_{st} < \frac{1}{\sqrt{3}}$ and if T_{1n} contains no gentle path then there is a path in T_{1n} from the left induction vertex p of T_1 to the right induction vertex q of T_n of length at most $(\frac{5}{\sqrt{3}} - 1)d_x(p, q)$.*

We will discuss the proof of the Amortization Lemma in Section 6; the proof builds on the analysis done in Section 5 to prove (Technical) Lemma 5. Instead of using function $\bar{U}$, however, we consider the discrete function

$$U(i) = d_{T_{1i}}(p, u_i) + p_N(u_i, i + 1)$$

defined for $i = 0, 1, \ldots, n$ and, for convenience's sake, 1) assuming that $p = u_0$ and $q = u_n$ and 2) using additional hexagon H_{n+1} of radius 0 centered at point q. An equivalent discrete function $L(i)$ using points l_i instead of u_i can be defined. Note that $U(n) + L(n)$ is exactly twice the distance in T_{1n} from p to q. To bound $U + L$, we will switch the analysis to a continuous one just as we did for $\bar{U} + \bar{L}$. We will see that, except for a finite number of discontinuities, the continuous version of $U + L$ has the same growth rate as $\bar{U} + \bar{L}$, which

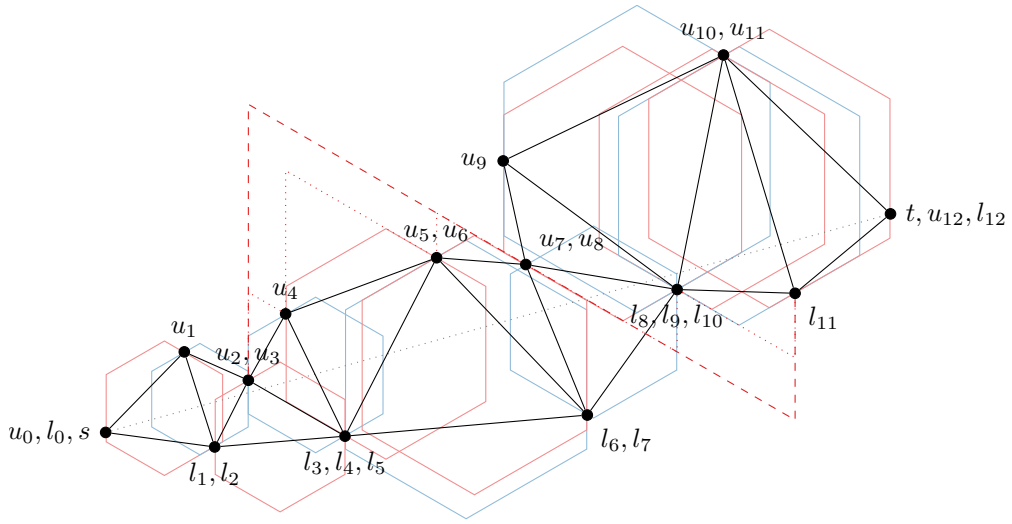


Figure 4 A gentle path from u_2 to l_{11} is one whose length is at most $\sqrt{3}d_x(u_2, l_{11}) - (\mathtt{y}(u_2) - \mathtt{y}(l_{11}))$, i.e. the length of the red dashed piecewise linear curve from u_2 to l_{11} (consisting of two vertical segments and a third with slope $-\frac{1}{\sqrt{3}}$). The path $u_2 = u_3, u_4, u_5 = u_6, u_7 = u_8, l_8 = l_9 = l_{10}, l_{11}$, easily seen to be bounded–in length–by the red dotted piecewise linear curve, is gentle. This path can be extended with edge (l_{11}, t) to a canonical gentle path from u_2 to t; the proof of (Main) Lemma 13, in this particular case, combines the bound on the length of this path together with the bound on the length of a path from s to u_2 obtained via induction.

is the growth rate of $p_N + p_S$. We will consider the intervals when the growth rate of (the continuous version of) $U + L$ is higher than $2(\frac{5}{\sqrt{3}} - 1)$ (i.e., when its growth rate is $\frac{8}{\sqrt{3}}$) and we will amortize the extra $2 - \frac{2}{\sqrt{3}}$ growth over intervals when its growth rate is smaller than $2(\frac{5}{\sqrt{3}} - 1)$ (i.e., when its growth rate is $\frac{4}{\sqrt{3}}$ or $\frac{6}{\sqrt{3}}$). The amortization can usually be done because when the growth rate of $U + L$ is large, the intervals must be relatively short compared to intervals when its growth is smaller, otherwise a gentle path can be shown to exist. To get our tight bound however, we will need to do more and show that at certain points (which are points of discontinuity) we need to use "cross-edges" (l_i, u_i). This is because when the amortization is not possible there is a long enough interval, say from hexagon H_i to hexagon H_j, when the growth rate of $U + L$ is $\frac{8}{\sqrt{3}}$ most of the time. It turns out that in that case one of U or L has growth rate bounded by $\frac{2}{\sqrt{3}}$ (say, U) and the other (L) by $\frac{6}{\sqrt{3}}$. This means that path $l_i, l_{i+1}, \ldots, l_j$ has relatively large length with respect to $\Delta(x)$ and that $u_i, u_{i+1}, \ldots, u_j$ is a relatively short path that can be used to replace the long subpath $l_i, l_{i+1}, \ldots, l_j$ with the shorter subpath $l_i, u_i, u_{i+1}, \ldots, u_j, l_j$ in a path from p to q. The $\frac{5}{\sqrt{3}} - 1$ stretch factor bound is the result of a min-max optimization between the two subpaths from l_i to l_j, and it is tight as we show in Section 7.

Next we turn to the case when the sequence of triangles T_{1n} contains a gentle path.

3.3 The gentle path lemma

Just as in the previous subsection, we consider a linear sequence of triangles T_{1n} defined with respect to a line st with slope m_{st} satisfying $0 < m_{st} < \frac{1}{\sqrt{3}}$. We now consider the case when T_{1n} contains a gentle path and state the other of our two key lemmas. We start with two definitions:

▶ **Definition 9.** *We say that linear sequence T_{1n} is* standard *if T_1 has a left induction vertex or $u_0 = l_0$, T_n has a right induction vertex or $u_n = l_n$, and neither the base vertex of T_1 (if any) nor the base vertex of T_n (if any) is the endpoint of a gentle path in T_{1n}.*

Note that if $u_0 = l_0$ and $u_n = l_n$ both hold (i.e., line st goes through those points) then T_{1n} is trivially standard because T_1 and T_n cannot have base vertices.

▶ **Definition 10.** *Let T_{1n} be a standard linear sequence. A gentle path in T_{1n} from p to q, where p occurs before q, is* canonical in T_{1n} *(or simply* canonical *if T_{1n} is clear from the context) if p is a right induction vertex of T_i for some $i \geq 1$ or p is the left vertex of T_1 and if q is a left induction vertex of T_j for some $j \leq n$ or q is the right vertex of T_n.*

For example, the gentle path $u_2 = u_3, u_4, u_5 = u_6, u_7 = u_8, l_8 = l_9 = l_{10}, l_{11}, l_{12}$ in Fig. 4 is canonical.

The second key lemma, which we will use alongside (Amortization) Lemma 8 to prove our main result, is stated next; its proof is discussed in Section 4.

▶ **Lemma 11** (The Gentle Path Lemma)**.** *Let T_{1n} be a linear sequence of triangles with respect to a line st with slope m_{st} such that $0 < m_{st} < \frac{1}{\sqrt{3}}$. If T_{1n} is standard and contains a gentle path then the path can be extended to a canonical gentle path in T_{1n}.*

The main idea behind the proof of this lemma is that a gentle path between $u_r \in U$ and $l_s \in L$ (where, say, $r \leq s$ and $\mathtt{x}(u_r) < \mathtt{x}(u_s)$) in T_{1n} can be extended using edge (u_{r-1}, u_r), unless $r = 0$ or u_r is a right induction vertex of T_r, or using edge (l_s, l_{s+1}), unless $s = n$ or l_s is a left induction vertex of T_{s+1}. In other words, a gentle path can be extended unless it is canonical.

We are now ready to state our main result and provide a proof that uses the two key lemmas.

3.4 The main result and the main lemma

▶ **Theorem 12.** *The stretch factor of a $\hexagon$-Delaunay triangulation is at most 2.*

To prove Theorem 12 we need to show that between any two points s and t of a set of points P there is, in the $\hexagon$-Delaunay triangulation T on P, a path from s to t of length at most $2d_2(s,t)$. Let m_{st} be the slope of the line st passing through s and t. Thanks to the hexagon's rotational and reflective symmetries as well as our general position assumptions, we can rotate the plane around s and possibly reflect the plane with respect to the x-axis to ensure that $0 < m_{st} < \frac{1}{\sqrt{3}}$. Given this assumption, our main theorem will follow from:

▶ **Lemma 13** (The Main Lemma)**.** *For every pair of points $s, t \in P$ with $0 < m_{st} < \frac{1}{\sqrt{3}}$:*

$$d_T(s,t) \leq \max\left\{\frac{5}{\sqrt{3}} - 1, \sqrt{3} + m_{st}\right\} d_x(s,t). \tag{1}$$

Before we prove this lemma, we show that it implies the main theorem.

Proof of Theorem 12. W.l.o.g., we assume that s has coordinates $(0,0)$, t lies in the positive quadrant, $m_{st} < \frac{1}{\sqrt{3}}$, and $d_2(s,t) = 1$. With these assumptions it follows that $\frac{\sqrt{3}}{2} < \mathtt{x}(t) = d_x(s,t) < 1$ and we need to show that $d_T(s,t) \leq 2$.

By Lemma 13, either $d_T(s,t) \leq (\frac{5}{\sqrt{3}} - 1) d_x(s,t) \leq (\frac{5}{\sqrt{3}} - 1) < 2$ or

$$d_T(s,t) \leq (\sqrt{3} + m_{st}) d_x(s,t) = \sqrt{3} d_x(s,t) + d_y(s,t) = \sqrt{3} d_x(s,t) + \sqrt{1 - d_x(s,t)^2}$$

which attains its maximum, over the interval $[\frac{\sqrt{3}}{2}, 1]$, at $d_x(s,t) = \frac{\sqrt{3}}{2}$ giving $d_T(s,t) \leq 2$. ◀

We now turn to the proof of (Main) Lemma 13. We start by noting that if there is a point p of P on the segment $[st]$ then (1) would follow if (1) holds for the pairs of points s, p and p, t; we can therefore assume that no point of P other than s and t lies on the segment $[st]$. We can also assume, as argued in Section 2, that segment $[st]$ does not intersect the outer face of the triangulation T. We assume w.l.o.g. that s has coordinates $(0, 0)$ and thus t lies in the positive quadrant.

Let $T_1, T_2, T_3, \ldots, T_n$ be the sequence of triangles of the triangulation T that line segment $[st]$ intersects when moving from s to t (refer to Fig. 4). (Recall that we assume that segment $[st]$ does not intersect the outer face of T.) Clearly, T_{1n} is a linear sequence of triangles and we assign labels u_i and l_i to the points and define sets U and L as described in Subsection 3.1. We note that all arguments in the rest of this paper use only points and edges of T_{1n}.

Note that, since st must intersect the interior of H_1, s can only lie on the n_w, w, or s_w sides of H_1; by Lemma 6, if s lies on the n_w side of H_1 then $(s, u_1) = (l_0, u_1)$ is gentle, and if s lies on the s_w side of H_1 then $(s, l_1) = (u_0, l_1)$ is gentle. Similarly, t can only lie on the n_e, e, or s_e sides of H_n; if t lies on the s_w side of H_n then $(t, l_{n-1}) = (u_n, l_{n-1})$ is gentle, and if t lies on the n_w side of H_n then $(t, u_{n-1}) = (l_n, u_{n-1})$ is gentle. Note that this means that if T_{1n} has no gentle edge then it is regular.

We now informally describe the approach we use to prove (Main) Lemma 13. We first note that (Amortization) Lemma 8 and (Gentle Path) Lemma 11 rely on (Technical) Lemma 5. We will prove (Main) Lemma 13 that bounds the length of the shortest path in T_{1n} from s to t as follows. If T_{1n} does not contain a gentle path then it is regular and the proof follows from (Amortization) Lemma 8. If T_{1n} contains a gentle path then by (Gentle Path) Lemma 11 it must contain a canonical gentle path $\mathcal{G}$ from, in general, a right induction vertex of T_i to a left induction vertex of T_j, where $0 \leq i < j \leq n$. We can assume, using (Gentle Path) Lemma 11, that $\mathcal{G}$ is maximal in the sense that it is not a subpath of any other gentle path in T_{1n}. The maximality of $\mathcal{G}$ will guarantee that neither T_{1i} nor T_{jn} contains a gentle path whose endpoint is the base vertex of T_i or the base vertex of T_j, respectively. Therefore T_{1i} and T_{jn} are standard and we then proceed by induction to prove a "more general" version of (Main) Lemma 13 for T_{1i} and T_{jn}. The obtained bounds on the lengths of shortest paths from s to the right induction vertex of T_i and from the left induction vertex of T_j to t are combined with the bound on the length of gentle path $\mathcal{G}$ to complete the proof of (Main) Lemma 13. Our reliance on induction means that we need to restate the Main Lemma so it is amenable to an inductive proof:

▶ **Lemma 14** (The Generalized Main Lemma). *Let $s, t \in P$ such that $0 < m_{st} < \frac{1}{\sqrt{3}}$ and let T_{1n} be the linear sequence of triangles that segment $[st]$ intersects. If T_{ij}, for some i, j such that $1 \leq i \leq j \leq n$, is standard, p is the left vertex of T_i, and q is the right vertex of T_j then*

$$d_{T_{ij}}(p, q) \leq \max\{\frac{5}{\sqrt{3}} - 1, \sqrt{3} + m_{st}\} d_x(p, q).$$

Note that (Main) Lemma 13 is a special case of this statement when $i = 1$ and $j = n$ since T_{1n} is (trivially) standard, s is the left vertex of T_1, and t is the right vertex of T_n.

Proof. We proceed by induction on $j - i$. If T_{ij} is standard and there is no gentle path in T_{ij} (the base case) then, by Lemma 6, the linear sequence of triangles in T_{ij} is regular and thus, by (Amortization) Lemma 8, we have $d_T(p, q) \leq (\frac{5}{\sqrt{3}} - 1) d_x(p, q)$.

If T_{ij} is standard and there is a gentle path in T_{ij}, then, by Lemma 11, there exist points $u_{i'}$ and $l_{j'}$ in T_{ij} such that there is a canonical gentle path between $u_{i'}$ and $l_{j'}$ in T_{ij}. We also assume that the canonical path between $u_{i'}$ and $l_{j'}$ is maximal in the sense that it is not a

proper subpath of a gentle path in T_{ij}. W.l.o.g., we assume that $u_{i'}$ occurs before $l_{j'}$, and so $i-1 \leq i' \leq j' \leq j$, $\mathtt{x}(u_{i'}) < \mathtt{x}(l_{j'})$, and $d_T(u_{i'}, l_{j'}) \leq \sqrt{3}d_x(u_{i'}, l_{j'}) - (\mathtt{y}(u_{i'}) - \mathtt{y}(l_{j'}))$. Since $u_{i'}$ is either s or above st and $l_{j'}$ is either t or below st, it follows that $-(\mathtt{y}(u_{i'}) - \mathtt{y}(l_{j'})) \leq m_{st}d_x(u_{i'}, l_{j'})$. Therefore, $d_T(u_{i'}, l_{j'}) \leq (\sqrt{3} + m_{st})d_x(u_{i'}, l_{j'})$.

Since the gentle path from $u_{i'}$ to $l_{j'}$ is canonical, either $u_{i'}$ is a right induction vertex of $T_{i'}$ and $i' \geq i$ or $u_{i'} = u_{i-1}$. In the first case, because $u_{i'}$ is on side e of $H_{i'}$ the base vertex $l_{i'-1} = l_{i'}$ of $T_{i'}$ must satisfy $\mathtt{x}(l_{i'}) < \mathtt{x}(u_{i'})$. Suppose that $l_{i'}$ is the endpoint of a gentle path in $T_{ii'}$ from, say, point $u_{i''}$ then we would have

$$\begin{aligned} d_{T_{ij}}(u_{i''}, l_{j'}) &\leq d_{T_{ij}}(u_{i''}, l_{i'}) + d_2(l_{i'}, u_{i'}) + d_{T_{ij}}(u_{i'}, l_{j'}) \\ &\leq \sqrt{3}d_x(u_{i''}, l_{i'}) - (\mathtt{y}(u_{i''}) - \mathtt{y}(l_{i'})) + \sqrt{3}d_x(l_{i'}, u_{i'}) - (\mathtt{y}(l_{i'}) - \mathtt{y}(u_{i'})) \\ &\quad + \sqrt{3}d_x(u_{i'}, l_{j'})) - (\mathtt{y}(u_{i'}) - \mathtt{y}(l_{j'})) \\ &\leq \sqrt{3}d_x(u_{i''}, l_{j'}) - (\mathtt{y}(u_{i''}) - \mathtt{y}(l_{j'}). \end{aligned}$$

This contradicts the maximality of the canonical gentle path from $u_{i'}$ to $l_{j'}$. This means that $l_{i'}$ is not the endpoint of a gentle path in $T_{ii'}$. Since $u_{i'}$ is a right induction vertex of $T_{i'}$, it follows that $T_{ii'}$ is standard, the inductive hypothesis applies, and $d_T(p, u_{i'}) \leq \max\{\frac{5}{\sqrt{3}} - 1, \sqrt{3} + m_{st}\}d_x(p, u_{i'})$. In the second case, because T_{ij} is standard, $u_{i'}$ cannot be the base vertex of T_i and so $u_{i'} = p$ and the same inequality holds trivially.

Similarly, we can show that $d_T(l_{j'}, q) \leq \max\{\frac{5}{\sqrt{3}} - 1, \sqrt{3} + m_{st}\}d_x(l_{j'}, q)$. Thus:

$$\begin{aligned} d_T(p, q) &\leq d_T(p, u_{i'}) + d_T(u_{i'}, l_{j'}) + d_T(l_{j'}, q) \\ &\leq \max\{\frac{5}{\sqrt{3}} - 1, \sqrt{3} + m_{st}\}(d_x(p, u_{i'}) + d_x(l_{j'}, q)) + (\sqrt{3} + m_{st})d_x(u_{i'}, l_{j'}) \\ &\leq \max\{\frac{5}{\sqrt{3}} - 1, \sqrt{3} + m_{st}\}d_x(p, q) \end{aligned}$$

◀

4 Proof of (gentle path) lemma 11

The main idea behind the proof of the Gentle Path lemma is that a gentle path between $u_r \in U$ and $l_s \in L$ (where, say, $r \leq s$ and $\mathtt{x}(u_r) < \mathtt{x}(u_s)$) in T_{1n} can be extended using edge (u_{r-1}, u_r), unless $r = 0$ or u_r is a right induction vertex of T_r, or using edge (l_s, l_{s+1}), unless $s = n$ or l_s is a left induction vertex of T_{s+1}. In other words, a gentle path from r to s is either canonical or can be extended to a canonical path from $u_{r'}$ to $l_{s'}$ as illustrated in Fig. 5.

5 Proof of (technical) lemma 5

We prove this lemma via a framework that uses continuous versions of the discrete functions (p_N, p_S, etc.) informally introduced in Subsection 3.1. We start by defining functions $H(x)$, $T(x)$, $u(x)$, $\ell(x)$, $\mathtt{r}(x)$, $w(x)$, and $e(x)$ for $\mathtt{x}(p) \leq x \leq \mathtt{x}(q)$ as illustrated in Fig. 6. Let point c_i be the center of hexagon H_i, for $i = 1, \ldots, n$. For x such that $\mathtt{x}(c_i) \leq x < \mathtt{x}(c_{i+1})$, $H(x)$ is the hexagon whose center has abscissa x and that has points $u_i = u(x)$ and $l_i = \ell(x)$ on its boundary. Intuitively, function $H(x)$ from $x = \mathtt{x}(c_i)$ to $x = \mathtt{x}(c_{i+1})$ models the "pushing" of hexagon H_i through u_i and l_i up until it becomes H_{i+1}. Function $\mathtt{r}(x)$ is the minimum radius of $H(x)$ and $w(x) = x - \mathtt{r}(x)$ and $e(x) = x + \mathtt{r}(x)$ are the abscissas of the w and e sides, respectively, of $H(x)$. Finally, we define $T(x) = T_{1i}$ when $\mathtt{x}(c_i) \leq x < \mathtt{x}(c_{i+1})$.

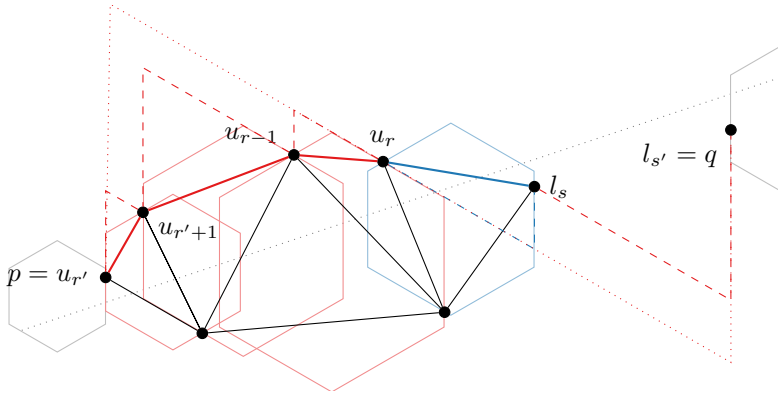


Figure 5 Illustration of the proof of Lemma 11 in the case when the gentle path from u_r to l_s is just a gentle edge. For every i such that $r' < i \leq r$ and u_i is the right vertex of H_i, hexagon H_i and the edge (u_{i-1}, u_i) are shown in red. Each edge (u_{i-1}, u_i) has slope greater than $-\frac{1}{\sqrt{3}}$ and therefore has length bounded by $\sqrt{3}d_x(u_{i-1}, u_i) - (\mathtt{y}(u_{i-1}) - \mathtt{y}(u_i))$, a value equal to the total length of the two intersecting, red, dashed segments going north from u_{i-1} and north-west from u_i. The total length of the two dashed blue segments is an upper bound on the length of the edge (u_r, l_s) and the total length of the dotted red line segments represent the upper bound $\sqrt{3}d_x(p, q) - (\mathtt{y}(p) - \mathtt{y}(q))$ on the length of the path $p = u_{r'}, \ldots, u_r, l_s, l_{s+1}, \ldots, l_{s'} = q$.

For a point o on a side of $H(x)$, we define functions $p_N(o, x)$ and $p_S(o, x)$ as the *signed* shortest distances around the perimeter of $H(x)$ to the N vertex and S vertex, respectively, with sign $\mathrm{sgn}(x - x(o))$. As Fig. 7-(a) and Fig. 7-(b) illustrate, these signs are positive for o on the n_w, w, or s_w sides of $H(x)$ and negative for o on n_e, e, or s_e sides. We omit o and use the shorthand notation $p_N(x)$ if $o = u(x)$ and $p_S(x)$ if $o = \ell(x)$.

Functions $U(x)$ and $L(x)$, used to bound the length of the shortest path from p to q and illustrated in Fig. 8-(a), are defined as follows for $\mathtt{x}(p) \leq x \leq \mathtt{x}(q)$:

$$U(x) = d_{T(x)}(p, u(x)) + p_N(x) \qquad L(x) = d_{T(x)}(p, \ell(x)) + p_S(x)$$

We note that $U(\mathtt{x}(q)) + L(\mathtt{x}(q))$ is exactly twice the distance in T_{1n} from p to q. We will compute an upper bound for function $U + L$ by bounding its growth rate.

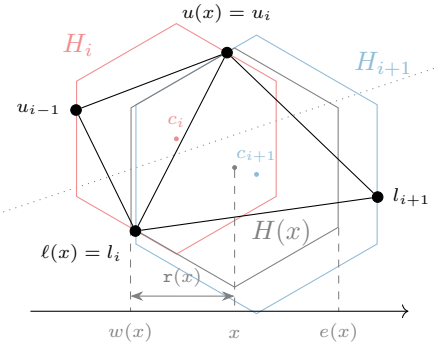


Figure 6 Intuitively, function $H(x)$ from $x = \mathtt{x}(c_i)$ to $x = \mathtt{x}(c_{i+1})$ models the "pushing" of hexagon H_i through u_i and l_i up until it becomes H_{i+1}.

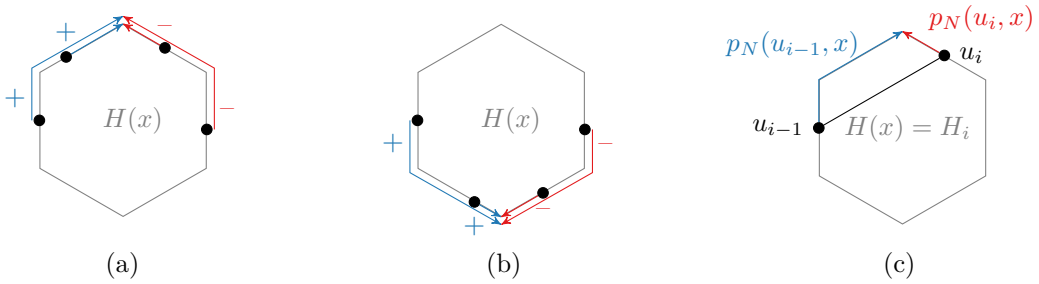


Figure 7 (a) The values of $p_N(o, x)$ are shown, for various points o lying on the boundary of $H(x)$, as signed hexagon arc lengths. (b) The values of $p_S(o, x)$ are shown similarly.

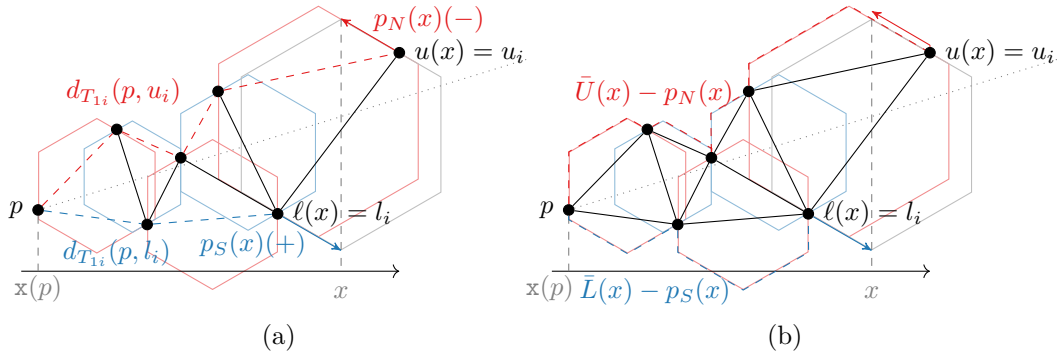


Figure 8 (a) Definition of $U(x)$ and $L(x)$. For example, $U(x)$ for $\mathtt{x}(c_i) \leq x < \mathtt{x}(c_{i+1})$ is the sum of the length of the shortest path from p to u_i in T_{1i} (illustrated as the red dashed path) and $p_N(x)$ (of negative value and represented as a red arrow). (b) Definition of $\bar{U}(x)$ and $\bar{L}(x)$. When $\mathtt{x}(c_i) \leq x < \mathtt{x}(c_{i+1})$ for example, $\bar{U}(x) - p_N(x)$ is an upper bound (equal to the length of the sequence of red dashed hexagon arcs going from p to u_i) on the length of the upper path $p, u_0, u_1, \ldots, u_{i-1}, u_i$.

As Fig. 7-(c) illustrates, the length of each edge (u_{i-1}, u_i), with u_{i-1}, u_i lying on the boundary of $H(x) = H_i$, is bounded by $p_N(u_{i-1}, x) - p_N(u_i, x)$. This and a similar insight about each (l_{i-1}, l_i) motivate functions $\bar{U}(x)$ and $\bar{L}(x)$ that bound the lengths of the upper and lower paths in T_{1n} and that are defined as follows for $\mathtt{x}(c_i) \leq x \leq \mathtt{x}(c_{i+1})$ (see Fig. 8-(b)):

$$\bar{U}(x) = \sum_{j=1}^{i} (p_N(u_{j-1}, \mathtt{x}(c_j)) - p_N(u_j, \mathtt{x}(c_j))) + p_N(x)$$

$$\bar{L}(x) = \sum_{j=1}^{i} (p_S(l_{j-1}), \mathtt{x}(c_j)) - p_S(l_j, \mathtt{x}(c_j))) + p_S(x)$$

When $\mathtt{x}(c_i) < x < \mathtt{x}(c_{i+1})$, functions $\bar{U}(x)$ and $\bar{L}(x)$ as well as $U(x)$ and $L(x)$ have rates of growth that depend solely on the last term ($p_N(x)$ or $p_S(x)$). We show that functions p_N and p_S are monotonically increasing piecewise linear and bound the rate of growth of p_N and p_S using elementary geometric arguments illustrated in Fig. 9. Figure 9-(c) illustrates a case when the growth rate of $p_N + p_S$, and therefore also of $\bar{U} + \bar{L}$ and of $U + L$, is $\frac{8}{\sqrt{3}}$.

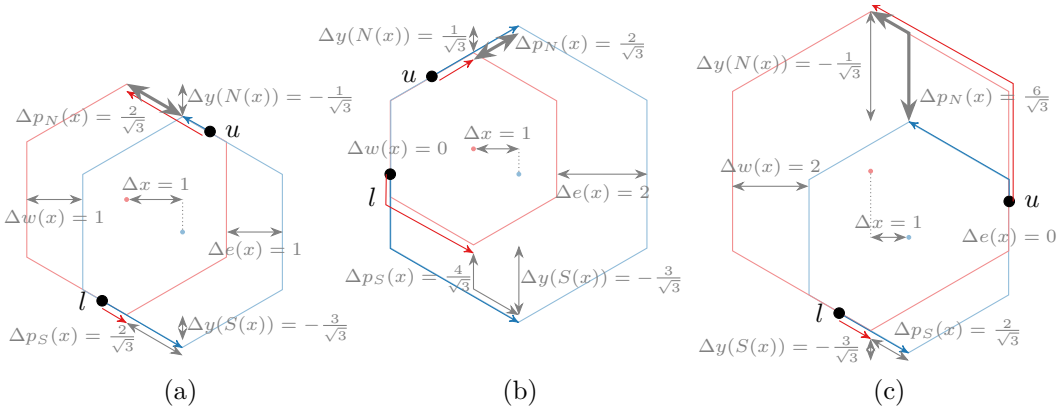


Figure 9 Constructions demonstrating growth rates, with respect to $\Delta x = 1$, of p_N, p_S and other functions for three different placements of $u(x) = u$ and $\ell(x) = l$ on the boundary of $H(x)$.

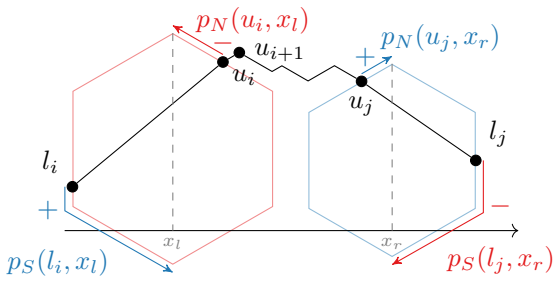


Figure 10 Illustrated is a situation in which the growth rate of $p_S(x)$ is $\frac{6}{\sqrt{3}}$ between $x = x_l$ and $x = x_r$. In that case the growth rate of $p_N(x)$ is $\frac{2}{\sqrt{3}}$. For large enough such intervals $[x_l, x_r]$, the path $l_i, u_i, u_{i+1}, \ldots, u_j, l_j$ is a shortcut for $l_i, l_{i+1}, \ldots, l_j$ and therefore $L(x_r)$ is smaller than what the growth rate of $p_S(x)$ would indicate. The stretch factor bound we obtain is the result of a min-max optimization between the two subpaths from l_i to l_j, and it is tight as we show in Fig. 11.

6 Proof of (amortization) lemma 8

The proof of the lemma builds on the framework discussed in the previous section and on a careful analysis of the growth rates of p_N and p_S when T_{1n} contains no gentle path. We show that in that case the average growth rate of $U + L$ is at most $2\left(\frac{5}{\sqrt{3}} - 1\right)$.

Our main approach is to spread (i.e., amortize) the "extra" $\frac{2}{\sqrt{3}}$ of the $\frac{8}{\sqrt{3}}$ growth rate over wider intervals of time that, as we show, include time intervals during which the growth rate is smaller. To achieve our tight bound of $2\left(\frac{5}{\sqrt{3}} - 1\right)$, however, we need to do more and also include "cross-edges" (l_i, u_i) as illustrated in Fig. 10.

7 Conclusion

The approach we use to bound the length of the shortest path in a Delaunay triangulation T between points s and t is to consider the linear sequence T_{1n} of triangles of T that segment $[st]$ intersects. We show that, in general, T_{1n} can be split into 1) disjoint linear sequences of triangles $T_{i_1 j_1}, T_{i_2 j_2}, \ldots, T_{i_k, j_k}$ that contain no gentle path and 2) $k - 1$ gentle paths with a gentle path connecting the right vertex of T_{j_l} with the left vertex of $T_{i_{l+1}}$ for $l = 1, \ldots, k - 1$.

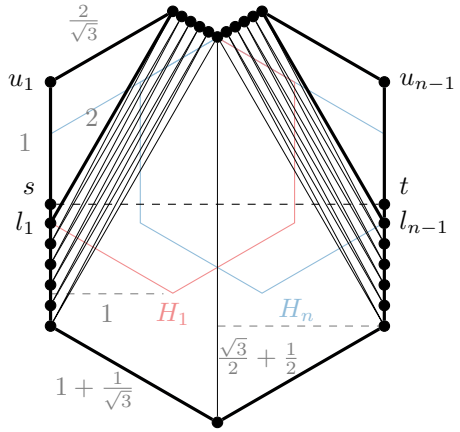


Figure 11 The Mickey Mouse ⬡-Delaunay triangulation. The inradii of H_1 and H_n are both set to 1. Edges that belong to a shortest path from s to t are in bold.

The worst case stretch factor for the Delaunay triangulation is then the maximum between the worst case stretch factors for 1) a path connecting the leftmost and rightmost points in a linear sequence T_{ij} that contains no gentle path and 2) a gentle path.

(Main) Lemma 13 and Lemma 1 show that the worst case stretch factor for ⬡-Delaunay triangulations comes from gentle path constructions. It turns out that similar conclusions can also be made regarding △- and □-Delaunay triangulations.

For ○-Delaunay triangulations, the situation seems to be different. The lower bound construction by Bose et al. [4] corresponds to a gentle path construction and has stretch factor 1.5846. The lower bound construction by Xia and Zhang [11] corresponds to a linear sequence that contains no gentle path and has stretch factor 1.5932. We think that the worst case stretch factor for ○-Delaunay triangulations will come from a construction similar to the one by Xia and Zhang [11]. Therefore, to get a tight bound on the stretch factor of a ○-Delaunay triangulation one needs to develop techniques that give tight bounds on the stretch factor of a linear sequence that contains no gentle path.

We have done so for ⬡-Delaunay triangulations. Our (Amortization) Lemma 8 implies that for ⬡-Delaunay triangulations the worst case stretch factor for a linear sequence T_{ij} with no gentle paths is $(\frac{5}{\sqrt{3}} - 1)$. It turns out that our analysis is tight: Figure 11 shows a construction–which we name the Mickey Mouse ⬡-Delaunay triangulation–that, for any $\epsilon > 0$, can be extended to a ⬡-Delaunay triangulation whose shortest path between s and t is at least $(\frac{5}{\sqrt{3}} - 1)d_x(s,t) - \epsilon$. Unsurprisingly, the construction corresponds to the lower bound construction by Xia and Xhang [11] for ○-Delaunay triangulations.

Based on this we think that the techniques we developed for obtaining the tight bound in Lemma 8 will be useful in obtaining better upper bounds for the stretch factor of other kinds of Delaunay triangulations.

References

1 Nicolas Bonichon, 2011. Personal communication.

2 Nicolas Bonichon, Cyril Gavoille, Nicolas Hanusse, and Ljubomir Perković. Tight stretch factors for L_1- and L_∞-Delaunay triangulations. *Computational Geometry*, 48(3):237–250, 2015. doi:10.1016/j.comgeo.2014.10.005.

3 Prosenjit Bose, Paz Carmi, Sebastien Collette, and Michiel Smid. On the stretch factor of convex Delaunay graphs. *Journal of Computational Geometry*, 1(1):41–56, 2010. doi:10.20382/jocg.v1i1a4.

4 Prosenjit Bose, Luc Devroye, Maarten Löffler, Jack Snoeyink, and Vishal Verma. Almost all Delaunay triangulations have stretch factor greater than $\pi/2$. *Computational Geometry*, 44(2):121–127, 2011. doi:10.1016/j.comgeo.2010.09.009.

5 L. Paul Chew. There is a planar graph almost as good as the complete graph. In *Proceedings of the 2nd Annual ACM Symposium on Computational Geometry (SoCG)*, pages 169–177, 1986. doi:10.1145/10515.10534.

6 L. Paul Chew. There are planar graphs almost as good as the complete graph. *Journal of Computer and System Sciences*, 39(2):205–219, 1989. doi:10.1016/0022-0000(89)90044-5.

7 Michael Dennis, Ljubomir Perković, and Duru Türkoğlu. The stretch factor of hexagon-delaunay triangulations, 2017. arXiv:1711.00068.

8 David P. Dobkin, Steven J. Friedman, and Kenneth J. Supowit. Delaunay graphs are almost as good as complete graphs. *Discrete & Computational Geometry*, 5(4):399–407, 1990. doi:10.1007/BF02187801.

9 J. Mark Keil and Carl A. Gutwin. Classes of graphs which approximate the complete Euclidean graph. *Discrete & Computational Geometry*, 7(1):13–28, 1992. doi:10.1007/BF02187821.

10 Ge Xia. The stretch factor of the Delaunay triangulation is less than 1.998. *SIAM Journal on Computing*, 42(4):1620–1659, 2013. doi:10.1137/110832458.

11 Ge Xia and Liang Zhang. Toward the tight bound of the stretch factor of Delaunay triangulations. In *Proceedings of the 23rd Annual Canadian Conference on Computational Geometry (CCCG)*, 2011. URL: http://www.cccg.ca/proceedings/2011/papers/paper57.pdf.

Flipping Geometric Triangulations on Hyperbolic Surfaces

Vincent Despré
Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France
vincent.despre@loria.fr

Jean-Marc Schlenker
Department of Mathematics, University of Luxembourg, Luxembourg
http://math.uni.lu/schlenker/
jean-marc.schlenker@uni.lu

Monique Teillaud
Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France
https://members.loria.fr/Monique.Teillaud/
monique.teillaud@inria.fr

Abstract

We consider geometric triangulations of surfaces, i.e., triangulations whose edges can be realized by disjoint geodesic segments. We prove that the flip graph of geometric triangulations with fixed vertices of a flat torus or a closed hyperbolic surface is connected. We give upper bounds on the number of edge flips that are necessary to transform any geometric triangulation on such a surface into a Delaunay triangulation.

2012 ACM Subject Classification Theory of computation → Computational geometry; Mathematics of computing → Discrete mathematics; Mathematics of computing → Geometric topology

Keywords and phrases Hyperbolic surface, Topology, Delaunay triangulation, Algorithm, Flip graph

Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.35

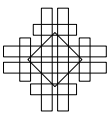
Funding The authors were partially supported by the grant(s) ANR-17-CE40-0033 of the French National Research Agency ANR (project SoS) and INTER/ANR/16/11554412/SoS of the Luxembourg National Research fund FNR (https://members.loria.fr/Monique.Teillaud/collab/SoS/).

1 Introduction

We investigate triangulations of two categories of surfaces: flat tori, i.e., surfaces of genus 1 with a locally Euclidean metric, and hyperbolic surfaces, i.e., surfaces of genus at least 2 with a locally hyperbolic metric (these surfaces will be introduced more formally in Section 2.1).

Triangulations of surfaces can be considered in a purely topological manner: a triangulation of a surface is a graph whose vertices, edges and faces partition the surface and whose faces have three (non-necessarily distinct) vertices. However, when the surface is equipped with a Euclidean or hyperbolic structure, it is possible to consider *geometric* triangulations, i.e., triangulations whose edges can be realized as geodesic segments that can only intersect at common endpoints (Definition 2). Note that a geometric triangulation can still have loops and multiple edges, but no contractible loop and no contractible cycle formed of two edges. We will prove that any Delaunay triangulation (Definition 4) of the considered surfaces is geometric (Proposition 8).

The flip graph of triangulations of the Euclidean plane is known to be connected; moreover the number of edge flips that are needed to transform any given triangulation with n vertices in the plane into the Delaunay triangulation has complexity $\Theta(n^2)$ [14]. We are interested in generalizations on this result to surfaces. Flips in triangulations of surfaces will be defined precisely later (Definition 5), for now we can just think of them as similar to edge flips in



triangulations of the Euclidean plane. Geodesics only locally minimize the length, so the edges of a geometric triangulation are generally not shortest paths. We will prove that the number of geometric triangulations on a set of points can be infinite, whereas the flip graph of "shortest path" triangulations is small but not connected in most situations [8].

▶ **Definition 1.** *Let $(\mathcal{M}_2, h)$ be either a torus $(\mathbb{T}^2, h)$ equipped with a Euclidean structure h or a closed oriented surface (S, h) equipped with a hyperbolic structure h. Let $V \subset \mathcal{M}_2$ be a set of n points. The* geometric flip graph $\mathcal{F}_{\mathcal{M}_2,h,V}$ *of $(\mathcal{M}_2, h, V)$ is the graph whose vertices are the geometric triangulations of $(\mathcal{M}_2, h)$ with vertex set V and where two vertices are connected by an edge if and only if the corresponding triangulations are related by a flip.*

Our results are mainly interesting in the hyperbolic setting, which is richer than the flat setting. However, to help the readers' intuition, we also present them for flat tori, where they are slightly simpler to prove and might even be considered as folklore. The geometric flip graph is known to be connected for the special case of flat surfaces with conical singularities and triangulations whose vertices are these singularities [19].

The main results of this paper are:

- The geometric flip graph of $(\mathcal{M}_2, h, V)$ is connected (Theorems 12 and 14).
- The Delaunay triangulation can be reached from any geometric triangulation by a path in the geometric flip graph $\mathcal{F}_{\mathcal{M}_2,h,V}$ whose length is bounded by n^2 times a quantity measuring the *quality* of the input triangulation (Theorems 16 and 19).

If an initial triangulation of the surface only having one vertex is given, then the Delaunay triangulation can thus be computed incrementally by inserting points one by one in a very standard way: for each new point, the triangle containing it is split into three, then the Delaunay property is restored by propagating flips. This approach, based on flips, can handle triangulations of a surface with loops and multiarcs, which is not the case for the approach based on Bowyer's incremental algorithm [6, 5].

2 Background and notation

2.1 Surfaces

In this section, we first recall a few notions, then we illustrate them for the two classes of surfaces (flat tori and hyperbolic surfaces) that we are interested in.

Let $\mathcal{M}_2$ be a closed oriented surface, i.e., a compact connected oriented 2-manifold without boundary. There is a unique simply connected surface $\widetilde{\mathcal{M}_2}$, called the *universal cover* of $\mathcal{M}_2$, equipped with a projection $\rho : \widetilde{\mathcal{M}_2} \to \mathcal{M}_2$ that is a local diffeomorphism. There is a natural action on $\widetilde{\mathcal{M}_2}$ of the fundamental group $\pi_1(\mathcal{M}_2)$ of $\mathcal{M}_2$ so that for all $p \in \mathcal{M}_2$, $\rho^{-1}(p)$ is an orbit under the action of $\pi_1(\mathcal{M}_2)$. We will denote as $\widetilde{p}$ a *lift* of p, i.e., one of the elements of the orbit $\rho^{-1}(p)$. A *fundamental domain* in $\widetilde{\mathcal{M}_2}$ for the action of $\pi_1(\mathcal{M}_2)$ on $\widetilde{\mathcal{M}_2}$ is a connected subset Ω of $\widetilde{\mathcal{M}_2}$ that intersects each orbit in exactly one point, or, equivalently, such that the restriction of ρ to Ω is a bijection from Ω to $\mathcal{M}_2$ [17]. The genus g of $\mathcal{M}_2$ is its number of handles. In this paper, we consider surfaces with constant curvature (0 or -1). The value of the curvature is given by the Gauss-Bonnet Theorem and thus only depends on the genus: a surface of genus 0 only admits spherical structures (not considered here); a flat torus is a surface of genus 1 and admits Euclidean structures; a surface of genus 2 and above admits only hyperbolic structures (see below).

From now on, $\mathcal{M}_2$ will denote either a flat torus or a closed hyperbolic surface.

Flat tori

We denote by $\mathbb{T}^2$ the topological torus, that is, the product $\mathbb{T}^2 = \mathbb{S}^1 \times \mathbb{S}^1$ of two copies of the circle. Flat tori are obtained by taking the quotient of the Euclidean plane by an Abelian group generated by two independent translations. There are in fact many different Euclidean structures on $\mathbb{T}^2$; if one considers Euclidean structures up to homothety – which is sufficient for our purposes here – a Euclidean structure is uniquely determined by a vector u in the upper half-plane $\mathbb{R} \times \mathbb{R}_{>0}$: to such a vector u is associated the Euclidean structure $(\mathbb{T}^2, h_u) \sim \mathbb{R}^2/(\mathbb{Z}e_1 + \mathbb{Z}u)$, where $e_1 = (1, 0)$ and $u = (u_x, u_y) \in \mathbb{R}^2$ is linearly independent from e_1. The orbit of a point of the plane is a lattice. The area A_h of the surface is $|u_y|$. The plane $\mathbb{R}^2$, equipped with the Euclidean metric, is then isometric to the universal cover of the corresponding quotient surface.

Hyperbolic surfaces

We now consider a closed oriented surface S (a compact oriented surface without boundary) of genus $g \geq 2$. Such a surface does not admit any Euclidean structure, but it admits many *hyperbolic* structures, corresponding to metrics of constant curvature -1, locally modeled on the hyperbolic plane $\mathbb{H}^2$. Given a hyperbolic structure h on S, the surface (S, h) is isometric to the quotient $\mathbb{H}^2/G$, where G is a (non-Abelian) discrete subgroup of the isometry group $PSL(2, \mathbb{R})$ of $\mathbb{H}^2$ isomorphic to the fundamental group $\pi_1(S)$. The universal cover $\widetilde{S}$ is isometric to the hyperbolic plane $\mathbb{H}^2$.

2.2 The Poincaré disk model of the hyperbolic plane

In the Poincaré disk model [1], the hyperbolic plane is represented as the open unit disk $\mathbb{D}^2$ of $\mathbb{R}^2$. The geodesic lines consist of circular arcs contained in the disk $\mathbb{D}^2$ and that are orthogonal to its boundary (Figure 1 (left)). The model is conformal, i.e., the Euclidean angles measured in the plane are equal to the hyperbolic angles.

We won't need the exact expression of the hyperbolic metric here. However, the notion of a hyperbolic circle is relevant to us. Three non-collinear points in the hyperbolic plane $\mathbb{H}^2$ determine a *circle*, which is the restriction to the Poincaré disk of a Euclidean circle or line. If C is a Euclidean circle or line and $\phi : \mathbb{D}^2 \to \mathbb{D}^2$ is an isometry of the hyperbolic plane, then $\phi(C \cap \mathbb{D}^2)$ is still the intersection with $\mathbb{D}^2$ of a Euclidean circle or a line.

A key difference with the Euclidean case is that the "circle" defined by 3 non-collinear points in $\mathbb{H}^2$ is generally not compact (i.e., it is not included in the Poincaré disk). The compact circles are sets of points at constant (hyperbolic) distance from a point. Non-compact circles are either hypercycles, i.e., connected components of the set of points at constant (hyperbolic) distance from a hyperbolic line, or horocycles (Figure 1 (right)) [13].[1] Therefore, the relatively elementary tools that can be used for flat tori must be refined for hyperbolic surfaces. Still, some basic properties of circles still hold for non-compact circles. A non-compact circle splits the hyperbolic plane into two connected regions; we will call a *disk* the region of the Poincaré disk that is convex, in the hyperbolic sense (Figure 2).

Triangulations of hyperbolic spaces have been studied [3] and implemented in CGAL in 2D [4]. Note that that previous work was not considering non-compact circles as circles.

[1] A synthetic presentation can be found at `http://en.wikipedia.org/wiki/Hypercycle_(geometry)`

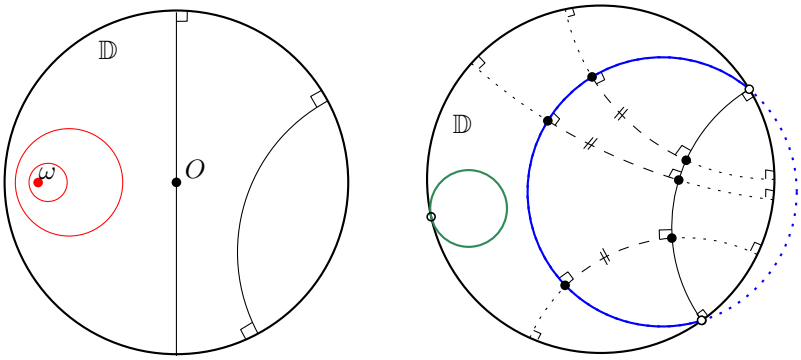


Figure 1 The Poincaré disk. Left: Geodesic lines (black) and compact circles (red) centered at point ω. Right: A horocycle (green). A hypercycle (blue), whose points have constant distance from the black geodesic line.

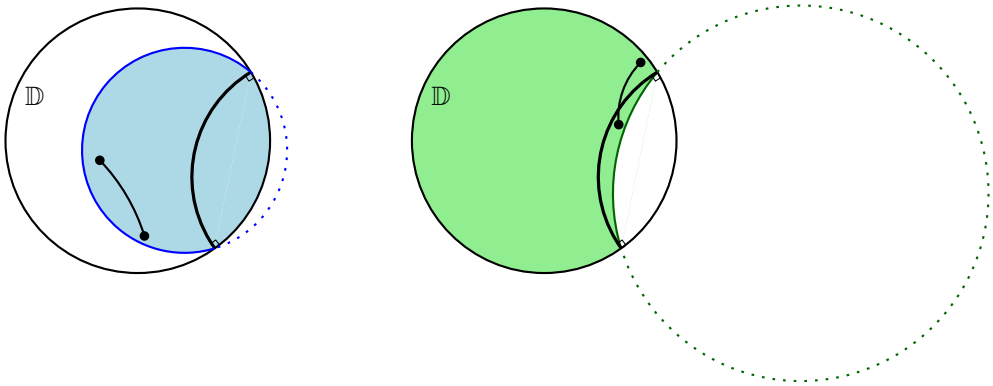


Figure 2 Shaded: Two (convex) non-compact disks.

2.3 Triangulations on surfaces

Let $(\mathcal{M}_2, h)$ be either a torus $(\mathbb{T}^2, h)$ equipped with a Euclidean structure h or a closed surface (S, h) equipped with a hyperbolic structure h.

For a given finite set of points $V \subset \mathcal{M}_2$, we will consider any two topological triangulations T and T' of $\mathcal{M}_2$ with vertex set V as *equivalent* if for any two vertices u and v in V, the edges of T with vertices u and v are in one-to-one correspondence with the edges of T' with the same vertices u and v through homotopies with fixed points.

Recall that given two distinct points $v, w \in \mathcal{M}_2$, any homotopy class of paths on $\mathcal{M}_2$ with endpoints v and w contains a unique geodesic segment [10, Chapter 1]. So, any triangulation is equivalent to a unique *geodesic* triangulation, i.e., a triangulation whose edges are geodesic segments. Note that the edges of a geodesic triangulation can intersect in their interiors.

▶ **Definition 2.** *A triangulation T on $\mathcal{M}_2$ is said to be* geometric *for h if the edges of its equivalent geodesic triangulation do not intersect except at common endpoints.*

If T is a triangulation of $\mathcal{M}_2$, its inverse image $\rho^{-1}(T)$ is the (infinite) triangulation of $\widetilde{\mathcal{M}_2}$ whose vertices, edges and faces are the lifted images by ρ^{-1} of those of T.

▶ **Definition 3.** *The* diameter $\Delta(T)$ *of a geodesic T is the smallest diameter of a fundamental domain that is the union of lifts of the triangles of T in $\widetilde{\mathcal{M}_2}$.*

The diameter $\Delta(T)$ is not smaller than the diameter of (S, h). It is unclear how to compute $\Delta(T)$ algorithmically and the problem looks difficult. However bounds are easy to obtain: $\Delta(T)$ is at least equal to the maximum of the diameters of the triangles of $\rho^{-1}(T)$ in $\widetilde{\mathcal{M}_2}$ and is at most the sum of the diameters of these triangles.

▶ **Definition 4.** *We say that a triangulation T of $\mathcal{M}_2$ is a* Delaunay *triangulation if for each face f of T and any face $\widetilde{f}$ of $\rho^{-1}(T)$, the open disk in $\widetilde{\mathcal{M}_2}$ that is bounded by the circle passing through the three vertices of $\widetilde{f}$ is* empty*, i.e., it contains no vertex of $\rho^{-1}(T)$.*

It follows that if T is a geodesic Delaunay triangulation of $\mathcal{M}_2$ with vertex set V, then $\rho^{-1}(T)$ is the Delaunay triangulation in $\widetilde{\mathcal{M}_2}$ of $\rho^{-1}(V)$. So, for a non-degenerate set of points on $\mathcal{M}_2$, since the Delaunay triangulation of their lifts in $\widetilde{\mathcal{M}_2}$ is unique, there is also a unique geodesic Delaunay triangulation on $\mathcal{M}_2$. However its edges may *a priori* intersect.

For a degenerate set V of points, at least two adjacent triangles in the possible Delaunay triangulations of $\rho^{-1}(V)$ in $\widetilde{\mathcal{M}_2}$ have cocircular vertices. Any triangulation of the subset $\mathcal{C}$ of $\rho^{-1}(V)$ consisting of c cocircular points is a Delaunay triangulation. Any of these triangulations can be transformed in any other by $O(c)$ flips [14]. From now on, we can thus assume that the set of points V on the surfaces that we consider is always non-degenerate.

We will see in Section 3 that any Delaunay triangulation of $\mathcal{M}_2$ is in fact geometric.

Remark that, even for a hyperbolic surface, the closure of every empty disk in the universal cover $\mathbb{H}^2$ is compact. Indeed, any non-compact disk contains at least one disk of any diameter, so, at least one disk of diameter $\Delta(T)$, thus it contains a fundamental domain (actually, infinitely many fundamental domains) and its interior cannot be empty.

Let us now give a natural definition for flips in triangulations of surfaces.

▶ **Definition 5.** *Let T be a geometric triangulation of $\mathcal{M}_2$. Let (v_1, v_2, v_3) and (v_2, v_1, v_4) be two adjacent triangles in T, sharing the edge $e = (v_1, v_2)$. Let us lift the quadrilateral (v_1, v_2, v_3, v_4) to a quadrilateral $(\widetilde{v_1}, \widetilde{v_2}, \widetilde{v_3}, \widetilde{v_4})$ in $\widetilde{\mathcal{M}_2}$ so that $(\widetilde{v_1}, \widetilde{v_2}, \widetilde{v_3})$ and $(\widetilde{v_2}, \widetilde{v_1}, \widetilde{v_4})$ form two adjacent triangles of $\rho^{-1}(T)$ sharing the edge $\widetilde{e} = (\widetilde{v_1}, \widetilde{v_2})$.*

Flipping e in T consists of replacing the diagonal $\widetilde{e}$ in the quadrilateral $(\widetilde{v_1}, \widetilde{v_2}, \widetilde{v_3}, \widetilde{v_4})$ (which lies in $\widetilde{\mathcal{M}_2}$, i.e., $\mathbb{R}^2$ or $\mathbb{H}^2$) by the geodesic segment $(\widetilde{v_3}, \widetilde{v_4})$, then projecting the two new triangles $(\widetilde{v_3}, \widetilde{v_4}, \widetilde{v_2})$ and $(\widetilde{v_4}, \widetilde{v_3}, \widetilde{v_1})$ to $\mathcal{M}_2$ by ρ.

We say that the flip of T along e is Delaunay *if the triangulation is* locally Delaunay *in the quadrilateral after the flip, i.e., the disk inscribing $(\widetilde{v_3}, \widetilde{v_4}, \widetilde{v_2})$ does not contain $\widetilde{v_1}$ (and the disk inscribing $(\widetilde{v_4}, \widetilde{v_3}, \widetilde{v_1})$ does not contain $\widetilde{v_2}$).*

An edge e is said to be Delaunay flippable *if the flip along e is Delaunay.*

Note that even though T is geometric in this definition, the triangulation after a flip is not necessarily geometric. We will prove later (Lemma 9) that a Delaunay flip transforms a geometric triangulation into a geometric triangulation.

Triangulations and polyhedral surfaces

The Euclidean plane can be identified with the plane $(z = 1)$ in $\mathbb{R}^3$, while the Poincaré model of the hyperbolic plane can be identified with the unit disk in that plane. We can now use the stereographic projection $\sigma : \mathbb{S}^2 \setminus \{s_0\} \to \mathbb{R}^2$ to send the unit sphere $\mathbb{S}^2$ to this plane $(z = 1)$, where $s_0 = (0, 0, -1)$ is the pole. In this projection, each point $p \neq s_0$ on the sphere is sent to the unique intersection with the plane $(z = 1)$ of the line going through s_0 and p. The inverse image of the plane $(z = 1)$ is $\mathbb{S}^2 \setminus \{s_0\}$, while the inverse image of the disk containing the Poincaré model of the hyperbolic plane is a disk, which is the set of points of $\mathbb{S}^2$ above a horizontal plane.

Let $T^\star$ be a triangulation of the Euclidean or the hyperbolic plane – for instance, $T^\star$ could be the inverse image $\rho^{-1}(T)$ of a triangulation T of a surface $(\mathcal{M}_2, h)$, in which case $T^\star$ has infinitely many vertices. We associate to $T^\star$ a polyhedral surface Σ in $\mathbb{R}^3$, constructed as follows. The construction is similar to the classic duality originally presented with a paraboloid in the case of (finite) triangulations in a Euclidean space [9]. It can also be seen as a simpler version, sufficient for our purpose, of the construction presented for triangulations in hyperbolic spaces using the space of spheres [3].

- The vertices of Σ are the inverse images on $\mathbb{S}^2$ by σ of the vertices of $T^\star$.
- The edges of Σ are line segments in $\mathbb{R}^3$ corresponding to the edges of $T^\star$ and the faces of Σ are triangles in $\mathbb{R}^3$ corresponding to the faces of $T^\star$.

Note that Σ is not necessarily convex. We can make the following well-known remarks. Let t_1 and t_2 be two triangles of $T^\star$ sharing an edge e, and let t_1^Σ and t_2^Σ be the corresponding faces of the polyhedral surface Σ, sharing the edge e^Σ. Then Σ is concave at e^Σ if and only if e is Delaunay flippable. Flipping e in the triangulation $T^\star$ in the plane corresponds to replacing the two faces t_1^Σ and t_2^Σ of Σ by the two other faces of the tetrahedron formed by their vertices. That tetrahedron lies between Σ and $\mathbb{S}^2$. We obtain a new edge $e^{\Sigma'}$ at which the new polyhedral surface Σ' is convex, and which is strictly closer to $\mathbb{S}^2$ than Σ. By an abuse of language, we will say that Σ' *contains* Σ, which we will denote as $\Sigma \subset \Sigma'$.

As a consequence, Σ is convex if and only if $T^\star$ is Delaunay.

There is a direct corollary of this statement: Given a (non-degenerate, see above) discrete set V of points in $\mathbb{R}^2$ or $\mathbb{H}^2$, there is a unique Delaunay triangulation with this set of vertices.

However we are going to see in the next two sections that there can be infinitely many geometric (non-Delaunay) triangulations on a surface, with the same given finite vertex set.

3 Geometric triangulations of surfaces

We consider now Dehn twists, which are usually considered as acting on the space of metrics on a surface [7], but are defined here equivalently, for simplicity, as acting on triangulations of a closed oriented surface $(\mathcal{M}_2, h)$ equipped with a fixed Euclidean or hyperbolic structure (figures in this section illustrate the flat case, but the results are proved for both flat and hyperbolic cases). Let T be a triangulation of $(\mathcal{M}_2, h)$, with vertex set V, and let c be an oriented homotopically non-trivial simple closed curve on $\mathcal{M}_2 \setminus V$. We define a new triangulation $\tau_c(T)$ of $\mathcal{M}_2$ by performing a *Dehn twist* along c: whenever an edge e of T intersects c at a point p, we orient e so that the unit vectors of the tangent plane along e and c form a positively oriented basis (see Figure 3 (left)), and then replace e by the oriented path following e until p, then following c until it comes back to p, then following e until its endpoint (see Figure 3 (right)). This defines a map τ_c from the space of triangulations of $\mathbb{T}^2$ with vertex set V to itself. Note that, even if T is a geometric triangulation, $\tau_c(T)$ is not necessarily geometric. If we denote by $-c$ the curve c with the opposite orientation, then one easily checks that $\tau_{-c} = \tau_c^{-1}$.

▶ **Lemma 6.** *There exists a geometric triangulation T of $(\mathcal{M}_2, h)$ and a simple closed curve $c \subset \mathcal{M}_2$ such that for all $k \in \mathbb{Z}$, $\tau_c^k(T)$ is geometric.*

Proof. Let us focus on the hyperbolic case (the construction is easier in the flat case). Consider a pants decomposition of $\mathcal{M}_2$ and denote as $\mathcal{C}$ the set of its boundary curves, which are simple closed geodesics. Let us choose c in $\mathcal{C}$ and $\varepsilon > 0$. We denote by c_-, c_+ the two hypercycles at distance ε from c on both sides of c. The value of ε must be sufficiently small so that the region between c_- and c_+ is an annulus drawn on $\mathcal{M}_2$ that does not intersect

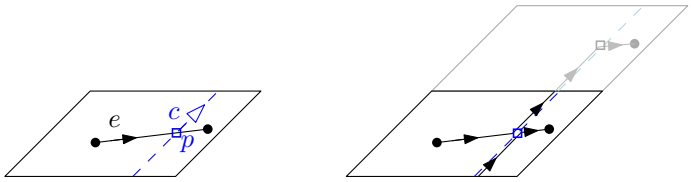


Figure 3 Transformation of an edge e by the Dehn twist along c on a flat torus $\mathbb{T}^2$. Here the black parallelepiped is a fundamental domain, and the gray one, used for the construction of the image of e by τ_c, is another fundamental domain, image through an element of the the group Γ of isometries.

any curve in $\mathcal{C} \setminus \{c\}$. Each curve in $\mathcal{C} \setminus \{c\}$ is split into two geodesic segments by putting two points on it; let us add the two segments as edges of T. Let us put two points on c_- (resp. c_+) and add as edges of T the two geodesic segments between them, whose union forms a curve homotopic to c. Each pair of pants not bounded by c, as well as the two "shortened pants" bounded by c_- and c_+, can be decomposed into two hexagons, which can easily been triangulated with geodesic edges. All these edges are left unchanged by τ_c (or τ_{-c}) as they do not intersect c. The annulus between c_- and c_+ can be triangulated with four edges intersecting c exactly once. We realize the image by τ_c of each of these four edges as a geodesic segment – there is a unique choice in the homotopy class of the path described above (Figure 4). The annulus is convex, as the projection onto $\mathcal{M}_2$ of the intersection of two (convex) disks (Figure 2), so, the geodesic segment is completely contained in it. Let

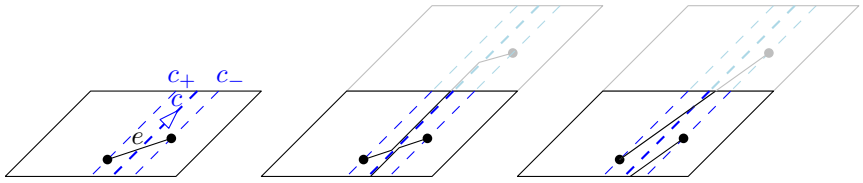


Figure 4 Image of e by a Dehn twist (middle), realized as a geodesic edge (right).

e, e' be two edges of T. If either e or e' does not intersect c, then their images by τ_c (or τ_{-c}) remain disjoint, as they lie in different regions separated by c_- and c_+. If e and e' intersect c, then again their images by τ_c (or τ_{-c}) remain disjoint, as their endpoints appear in the same order on c_- and c_+ and two geodesic lines cannot intersect more than once (Figure 5). As a consequence, $\tau_c(T)$ and $\tau_{-c}(T)$ are geometric. They are not equivalent as each edge e

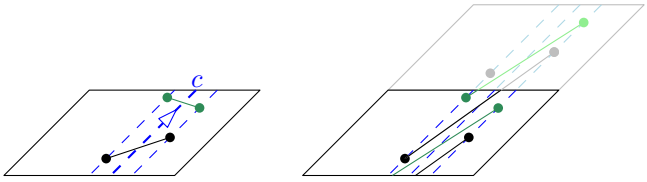


Figure 5 The Dehn twist of two edges along c for two edges intersecting c.

crossing c is replaced by an edge that does not lie in the same homotopy class as e. The same result follows by induction for $\tau_c^k(T)$ for any $k \in \mathbb{Z}$. ◀

▶ **Corollary 7.** *For any closed oriented surface $(\mathcal{M}_2, h)$, there exists a finite set of points $V \subset \mathcal{M}_2$ such that the graph of geometric triangulations with vertex set V is infinite.*

▶ **Proposition 8.** *Any Delaunay triangulation of a closed oriented surface $(\mathcal{M}_2, h)$ is geometric.*

Proof. Let V be a finite set of points on $\mathcal{M}_2$, and let T be a Delaunay triangulation of $(\mathcal{M}_2, h)$ with vertex set V. Realize every edge of T as a the unique geodesic segment in its homotopy class, so that T is geodesic. We argue by contradiction and suppose that T is not geometric, so that there are two edges e_1 and e_2 that intersect in their interiors. We then lift e_1 and e_2 to edges $\widetilde{e_1}$ and $\widetilde{e_2}$ of $\rho^{-1}(T)$ whose interiors still intersect.

We can find two distincts faces $\widetilde{f_1}$ and $\widetilde{f_2}$ of $\rho^{-1}(T)$ such that $\widetilde{e_1}$ is an edge of $\widetilde{f_1}$ and $\widetilde{e_2}$ is an edge of $\widetilde{f_2}$. Let $\widetilde{C_1}$ and $\widetilde{C_2}$ be the circles inscribing $\widetilde{f_1}$ and $\widetilde{f_2}$, respectively. Since $\rho^{-1}(T)$ is Delaunay, $\widetilde{C_1}$ and $\widetilde{C_2}$ bound empty disks $\widetilde{D_1}$ and $\widetilde{D_2}$, i.e., open disks not containing any point of $\rho^{-1}(V)$. Recall that, as mentioned in Section 2.3, the closures of empty disks are compact even in the hyperbolic case, and that $\widetilde{e_1} \subset \widetilde{D_1}$ and $\widetilde{e_2} \subset \widetilde{D_2}$ (edges are considered as open). The two circles $\widetilde{C_1}$ and $\widetilde{C_2}$ intersect twice as the intersection point of $\widetilde{e_1}$ and $\widetilde{e_2}$ lies in $\widetilde{D_1} \cap \widetilde{D_2}$. Let $\widetilde{L}$ be the geodesic line through the intersection points. The endpoints of $\widetilde{e_1}$ are on $\widetilde{C_1} \setminus \widetilde{D_2}$ and those of $\widetilde{e_2}$ are on $\widetilde{C_2} \setminus \widetilde{D_1}$, so the two pairs of endpoints are on opposite sides of $\widetilde{L}$. As a consequence, $\widetilde{e_1}$ and $\widetilde{e_2}$ are on opposite sides of $\widetilde{L}$, so they cannot intersect. This leads to a contradiction. ◀

4 The flip algorithm

Let us consider a closed oriented surface $(\mathcal{M}_2, h)$. The flip algorithm consists of performing Delaunay flips in any order, starting from a given input geometric triangulation of $\mathcal{M}_2$, until there is no more Delaunay flippable edge.

In this section, we first define a data structure that supports this algorithm, then we prove the correctness of the algorithm.

4.1 Data structure

In both cases of a flat or hyperbolic surface, the group of isometries defining the surface is denoted as G. We assume that a fundamental domain Ω^0 is given. By definition (Section 2.1), $\widetilde{\mathcal{M}_2}$ is the union $G(\Omega^0)$ of the images of Ω^0 under the action of G.

To represent a triangulation on the surface, we propose a data structure generalizing the data structure previously introduced for triangulations of flat orbifolds [6] and triangulations of the Bolza surface [15]. The combinatorics of the triangulation is given by the set of its vertices V on the surface and the set of its triangles, where each triangle gives access to its three vertices in V and its three adjacent triangles, and each vertex gives access to one of its incident triangles. The geometry of the triangulation is given by the set $\widetilde{V}^0$ of the lifts of its vertices that lie in the fundamental domain Ω^0 and one lift $\widetilde{t}^0$ in $\widetilde{\mathcal{M}_2}$ of each triangle $t = (v_{0,t}; v_{1,t}; v_{2,t})$ of the triangulation, chosen among the (one, two, or three) lifts of t in $\widetilde{\mathcal{M}_2}$ having at least one vertex in Ω^0: $\widetilde{t}^0$ has at least one of its vertices $\widetilde{v_{i,t}}^0$ in Ω^0 ($i = 0, 1$, or 2); then the other vertices of $\widetilde{t}^0$ are images $g_{i+1,t} \cdot \widetilde{v_{i+1,t}}^0$ and $g_{i+2,t} \cdot \widetilde{v_{i+2,t}}^0$ of two vertices in $\widetilde{V}^0$, where $g_{i+1,t}$ and $g_{i+2,t}$ are elements of G (indices are taken modulo 3). In the data structure, each vertex v on the surface has access to its representative $\widetilde{v}^0$, and each triangle t on the surface has access to the isometries $g_{0,t}, g_{1,t}$, and $g_{2,t}$ allowing to construct $\widetilde{t}^0$, at least one of the isometries being the identity $\mathbb{1}_G$. Note that two triangles t and t' of T that are adjacent on the surface are represented by two triangles $\widetilde{t}^0$ and $\widetilde{t'}^0$, which are not necessarily adjacent in $\widetilde{\mathcal{M}_2}$ (Figure 6 (left)). However, there is an isometry g in G such that $\widetilde{t}^0$ and $g \cdot \widetilde{t'}^0$ are adjacent.

Let T be an input triangulation given as such a data structure. Figure 6 illustrates a Delaunay flip performed on two adjacent triangles t and t' on the surface. The triangle $\widetilde{t'}^0$ is first moved so that the vertices of the edge to be flipped coincide. Then the edge is flipped. The isometries in the two triangles created by the flip are easy to compute from the isometries stored in t and t'. Note that the order in which isometries are composed is crucial in the hyperbolic case, as they do not commute. We have shown that the data structure can be maintained through flips.

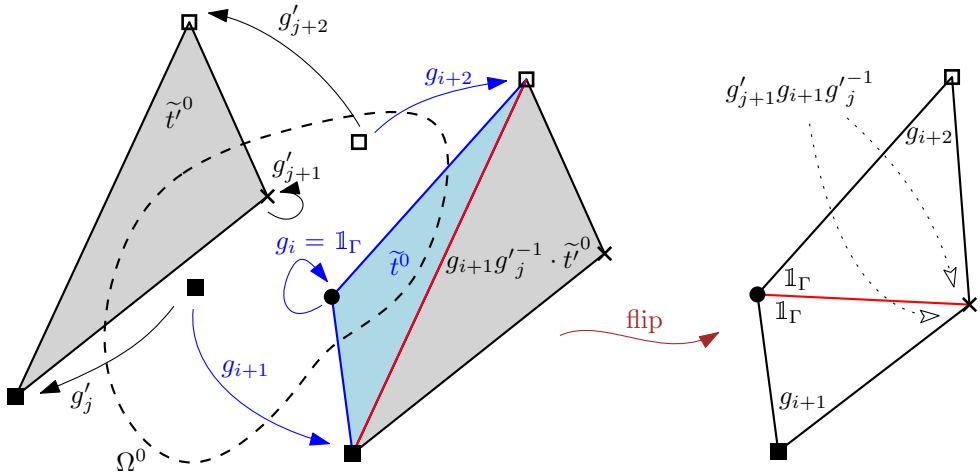


Figure 6 A flip. Here (hyperbolic) triangles are represented schematically with straight edges. Left: the two triangles $\widetilde{t}^0$ and $\widetilde{t'}^0$ before the flip. Here $g_i = \mathbb{1}_G$. Right: the isometries in the two triangles created by the flip.

4.2 Correctness of the algorithm

The following statement is a key starting point.

▶ **Lemma 9.** *Let T be a geometric triangulation of $(\mathcal{M}_2, h)$, and let T' be obtained from T by a Delaunay flip. Then T' is still geometric.*

Proof. Let e be a Delaunay flippable edge and $\widetilde{e}$ a lift in $\widetilde{\mathcal{M}_2}$. Denote the vertices of $\widetilde{e}$ by $\widetilde{v}$ and $\widetilde{v'}$. Let $\widetilde{t_1}$ and $\widetilde{t_2}$ be the triangles of $\rho^{-1}(T)$ incident to $\widetilde{e}$. To prove that T' is geometric, it is sufficient to prove that $\widetilde{t_1} \cup \widetilde{t_2}$ is a strictly convex quadrilateral.

Let $\widetilde{C_1}$ (resp. $\widetilde{C_2}$) be the circle through the three vertices of $\widetilde{t_1}$ (resp. $\widetilde{t_2}$). Note that $\widetilde{C_1}$ and $\widetilde{C_2}$ may be non-compact. Let $\widetilde{D_1}$ and $\widetilde{D_2}$ be the corresponding disks (as defined in Section 2.2 on case of non-compact circles). The disk $\widetilde{D_1}$ (resp. $\widetilde{D_2}$) is convex (in the Euclidean plane if $\mathcal{M}_2$ is a flat torus, or in the sense of hyperbolic geometry if $\mathcal{M}_2$ is a hyperbolic surface) and contains $\widetilde{t_1}$ (resp. $\widetilde{t_2}$). The fact that e is Delaunay flippable then implies that $\widetilde{t_1}$ and $\widetilde{t_2}$ are contained in $\widetilde{D_1} \cap \widetilde{D_2}$ (see Figure 7). As a consequence, the sum of angles of $\widetilde{t_1}$ and $\widetilde{t_2}$ at $\widetilde{v}$ is smaller than the interior angle at $\widetilde{v}$ of $\widetilde{D_1} \cap \widetilde{D_2}$, which is at most π, and similarly at $\widetilde{v'}$. As a consequence, the quadrilateral $\widetilde{t_1} \cup \widetilde{t_2}$ is strictly convex at $\widetilde{v}$ and $\widetilde{v'}$. Since it is strictly convex at its other two vertices (as each of these vertices is a vertex of a triangle), it is strictly convex, and the statement follows. ◀

The following lemma, using the diameter of the triangulation (Definition 3), is central in the proof of the termination of the algorithm (Theorem 14) for hyperbolic surfaces and in its analysis for both flat tori and hyperbolic surfaces (Section 5).

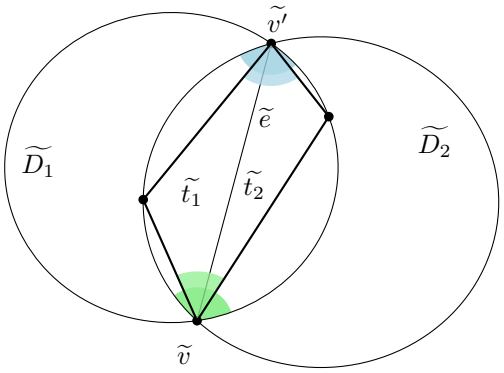


Figure 7 The quadrilateral is convex (edges are represented schematicaly as straight line segments).

▶ **Lemma 10.** *Let T be a geometric triangulation of $(\mathcal{M}_2, h)$. Then, the flip algorithm starting from T will never insert an edge longer than $2\Delta(T)$.*

Note that the length of an edge can be measured on any of its lifts in the universal covering space $\widetilde{\mathcal{M}_2}$.

Proof. Let T_k be the triangulation obtained from $T = T_0$ after k flips and let Σ_k be the corresponding polyhedral surface of $\mathbb{R}^3$ as defined in Section 2.3. Since we perform only Delaunay flips, $\Sigma_0 \subset \ldots \subset \Sigma_k \subset \Sigma_{k+1}$ (with the abuse of language mentioned in Section 2.3).

We will prove the result by contradiction. Let us assume that T_k has an edge e of length larger than $2\Delta(T)$. Let Ω be a fundamental domain of $\mathcal{M}_2$ having diameter $\Delta(T)$, given as the union of lifts of triangles of $T = T_0$ (it is not clear how to compute such a fundamental domain efficiently but its existence is clear). Let v be the midpoint of e and $\widetilde{v}$ its lift in Ω. Let $\widetilde{e} = (\widetilde{v_1}, \widetilde{v_2})$ be the unique lift of e whose midpoint is $\widetilde{v}$. The domain Ω is strictly included in the disk $\widetilde{D}$ of radius $\Delta(T)$ and centered at $\widetilde{v}$, by definition of $\Delta(T)$ (see Figure 8 (left)).

Let P_D denote the plane in $\mathbb{R}^3$ containing the circle on $\mathbb{S}^2$ that is the boundary of $\sigma^{-1}(D)$ (recall that σ denotes the stereographic projection, see Section 2.3), and let p denote the point $\sigma^{-1}(\widetilde{v})$ on $\mathbb{S}^2$. As $p \in \sigma^{-1}(\Omega) \subset \sigma^{-1}(\widetilde{D})$, the projection p^{Σ_0} of p onto Σ_0 lies above P_D (Figure 8 (right)).

Now, denote the edge $\sigma^{-1}(\widetilde{e})$ on $\mathbb{S}^2$ as (p_1, p_2). The points p_1 and p_2 lie outside $\sigma^{-1}(D)$. So, the corresponding edge $e^{\Sigma} = [p_1, p_2]$ of Σ_k lies below the plane P_C, thus the projection $p^{\Sigma_k} \in [p_1, p_2]$ of p onto Σ_k lies below P_C.

From what we have shown, p^{Σ_k} is a point of Σ_k that lies strictly between the pole s_0 and the point p^{Σ_0} of Σ_0, which contradicts the inclusion $\Sigma_0 \subset \Sigma_k$. ◀

We will now show that, for any order, the flip algorithm terminates and returns the Delaunay triangulation of the surface. The proof given for the hyperbolic case would also work for the flat case. However we propose a more elementary proof for the flat case.

Flat tori

The case of flat tori is easy, and might be considered as folklore. However, as we have not found a reference, we give the details here for completeness.

We define the weight of a triangle t of a geometric triangulation T of $\mathbb{T}^2$ as the number of vertices of $\rho^{-1}(T)$ that lie in the open circumdisk of a lift of t. The weight $w(T)$ of T is defined as the sum of the weights of its triangles.

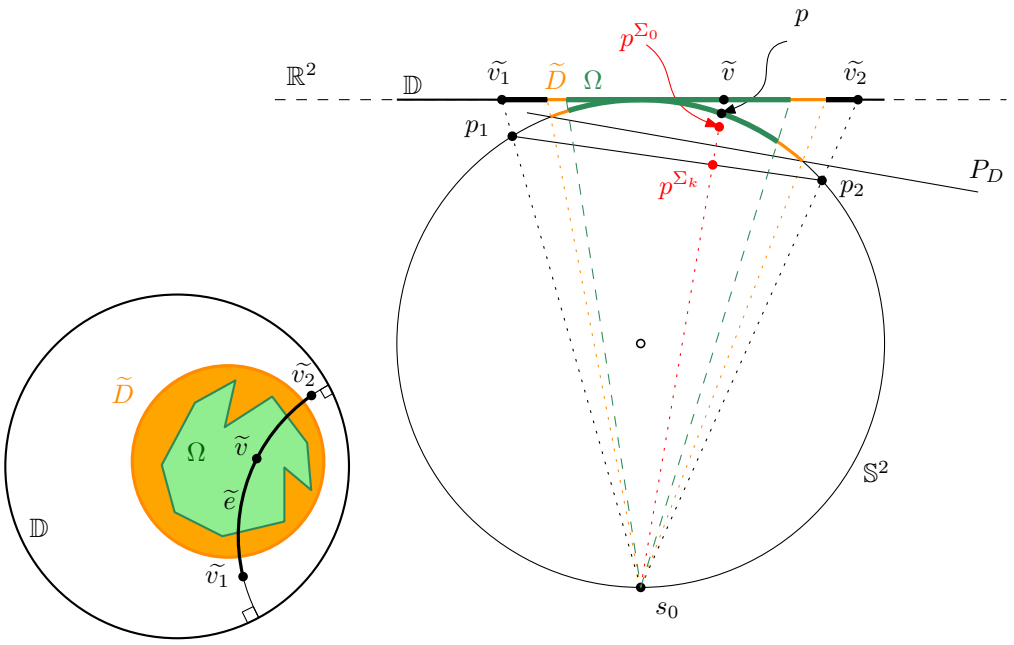


Figure 8 Illustration for the proof of Lemma 10 (for a hyperbolic surface). Left: notation in $\mathbb{H}^2$. Right: contradiction seen in a cutting plane in $\mathbb{R}^3$.

▶ **Lemma 11.** *The weight $w(T)$ of a triangulation T of a flat torus $(\mathbb{T}^2, h)$ is finite. Let T' be the triangulation obtained from a geometric triangulation T after performing a Delaunay flip. Then $w(T') \leq w(T) - 2$.*

Proof. The closed circumdisk of any triangle in $\mathbb{R}^2$ is compact, so, it can only contain a finite number of vertices of $\rho^{-1}(T)$. The sum $w(T)$ of these numbers over triangles of T is clearly finite as the number of triangles of T is finite. Let us now focus on a quadrilateral in $\mathbb{R}^2$ that is a lift of the quadrilateral on $\mathbb{T}^2$ whose diagonal e is flipped. Let $\widetilde{D_1}$ and $\widetilde{D_2}$ denote the two open circumdisks in $\mathbb{R}^2$ before the flip and $\widetilde{D'_1}$ and $\widetilde{D'_2}$ denote the two open circumdisks after the flip, then $\widetilde{D'_1} \cup \widetilde{D'_2} \subset \widetilde{D_1} \cup \widetilde{D_2}$ and $\widetilde{D'_1} \cap \widetilde{D'_2} \subset \widetilde{D_1} \cap \widetilde{D_2}$ (see Figure 9). Moreover, by definition of a Delaunay flip, the union $\widetilde{D'_1} \cup \widetilde{D'_2}$ contains at least two fewer

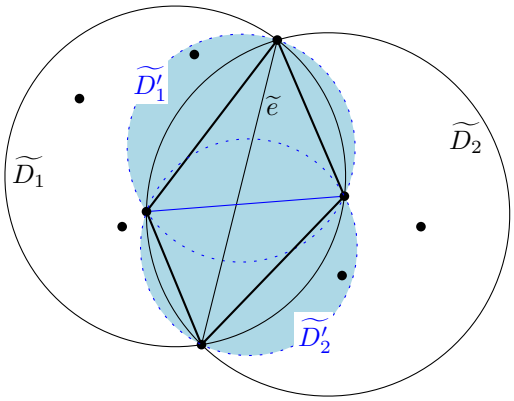


Figure 9 Circumdisks $\widetilde{D_1}$ and $\widetilde{D_2}$ before flipping $\widetilde{e}$ and $\widetilde{D_1}$ and $\widetilde{D'_2}$ after the Delaunay flip.

vertices of $\rho^{-1}(T)$ than $\widetilde{D_1} \cup \widetilde{D_2}$, which are the two vertices of the quadrilateral that are not vertices of $\widetilde{e}$. This concludes the proof. ◀

The result follows trivially:

▶ **Theorem 12.** *Let T be a geometric triangulation of a flat torus with finite vertex set V. The flip algorithm terminates and outputs the Delaunay triangulation of V.*

▶ **Corollary 13.** *The geometric flip graph $\mathcal{F}_{\mathbb{T}^2,h,V}$ is connected.*

Hyperbolic surfaces

To show that the flip algorithm terminates in the hyperbolic case, we cannot mimic the proof presented for the flat tori since the circumcircle of a hyperbolic triangle can be non-compact (see Section 2.2) and thus can have an infinite weight. Note also that the proof cannot use a property on the angles of the Delaunay triangulation similar to what holds in the Euclidean case: in $\mathbb{H}^2$, the locus of points seeing a segment with a given angle is not a circle arc, and thus the Delaunay triangulation of a set of points in $\mathbb{H}^2$ does not maximize the smallest angle of triangles. The proof relies on Lemma 10.

▶ **Theorem 14.** *Let T be a geometric triangulation of a closed hyperbolic surface with finite vertex set V. The flip algorithm terminates and outputs the Delaunay triangulation of V.*

Proof. We use the same notation as in the proof Lemma 10. Once an edge of T_k is flipped, it can never reappear in the triangulation, as the corresponding segment in $\mathbb{R}^3$ becomes interior to the polyhedral surface Σ_{k+1} (see Section 2.3) and further surfaces $\Sigma_{k'}, k' \geq k+1$. In addition, all the introduced edges have length smaller than $2\Delta(T)$ by Lemma 10. Moreover, there is only a finite number of edges with vertices in V that are shorter than $2\Delta(T)$ on S, as a circle given by a center and a bounded radius is compact. So, the flip algorithm terminates. The output does not have any Delaunay flippable edge, so, it is the Delaunay triangulation. ◀

▶ **Corollary 15.** *The geometric flip graph $\mathcal{F}_{S,h,V}$ is connected.*

5 Algorithm analysis

For a triangulation on n vertices in the Euclidean plane, counting the weights of triangulations leads to the optimal $O(n^2)$ bound. However the same argument does not yield a bound even for the flat torus, since points must be counted in the universal cover.

▶ **Theorem 16.** *For any triangulation T with n vertices of a torus $(\mathbb{T}^2, h)$, there is a sequence of flips of length $C_h \cdot \Delta(T)^2 \cdot n^2$ connecting T to a Delaunay triangulation of $(\mathbb{T}^2, h)$, where C_h only depends on h.*

Proof. Let $e = (v_1, v_2)$ be an edge appearing during the flip algorithm, and $\widetilde{v_1}$ (resp. $\widetilde{v_2}$) be a lift of v_1 (resp. v_2), such that $(\widetilde{v_1}, \widetilde{v_2})$ is a lift $\widetilde{e}$ of e. The point $\widetilde{v_2}$ lies in a circle C of diameter $4\Delta(T)$ centered at $\widetilde{v_1}$ by Lemma 10. Let M be the affine transformation that maps the lattice of the lifts of v_2 to the square lattice $\mathbb{Z}^2$. $M(C)$ is a convex set and from Pick's theorem [20],[2] the number of points of $\mathbb{Z}^2$ in $M(C)$ is smaller than $\text{area}(M(C)) + 1/2 \cdot \text{perimeter}(M(C)) + 1$, which is also a bound on the number of possible points $\widetilde{v_2}$ in C and thus the number of possible edges e. The area of $M(C)$ is $1/A_h \cdot \text{area(C)}$ since $det(M) = 1/A_h$, but there is no simple formula for its perimeter. As already mentioned in the proof of Theorem 14, an edge can never reappear after it was flipped. Moreover, there are $n^2/2$ sets of points $\{v_1, v_2\}$ (v_1 and v_2 may be the same point), which yields the result. ◀

[2] See also `https://en.wikipedia.org/wiki/Pick's_theorem#Inequality_for_convex_sets`

The rest of this section is devoted to computing the number of edges not longer than $2\Delta(T)$ between two fixed points v_1 and v_2 on a hyperbolic surface (S, h). Counting the number of points in a disk of fixed radius would give an exponential bound because the area of a circle in $\mathbb{H}^2$ is exponential in its radius [16]. Note that we only consider geodesic edges, so we only need to count homotopy classes of simple paths. The behavior of the number N_l of simple closed curves smaller than a fixed length l is well understood: N_l/l^{6g-6} converges to a positive constant depending "continuously" on the structure h [18]. However, we need a result for geodesic segments instead of geodesic closed curves, and Mirzakhani's proof is too deep and relies on too sophisticated structures to easily be generalized. So, we will only prove an upper bound on the number of segments. Such an upper bound could be derived from the theory of measured laminations of Thurston, which is also quite intricate. Fortunately, a more comprehensible proof, specific to simple closed geodesic curves on hyperbolic structures, can be found in a book published by the French Mathematical Society [12, 4.III, p.61-67] [11]. While recalling the main steps of the proof, we show how to extend it to geodesic segments.

Let $\Gamma = \{\gamma_i, i = 1, \dots, 3g-3\}$ be a set of $3g-3$ simple disjoint closed geodesics on (S, h) not containing v_1 and v_2 that forms a pants decomposition on S, where each γ_i belongs to two different pairs of pants. A set $\{\overline{\gamma_i}, i = 1, \dots, 3g-3\}$ of disjoint closed annuli is defined on S, where each $\overline{\gamma_i}$ is a tubular neighborhood of γ_i containing none of v_1, v_2. This yields a decomposition of S into $3g-3$ annuli $\overline{\gamma_i}$ $(i = 1, \dots, 3g-3)$ and $2g-2$ pairs of "short pants" P_j $(j = 1, \dots, 2g-2)$. For $i = 1, \dots, 3g-3$, let us denote as $\partial\overline{\gamma_i}$ any one of the two curves bounding the annulus $\overline{\gamma_i}$ (this is an abuse of notation but should not introduce any confusion). In each pair of pants $P_j, j = 1, \dots, 2g-2$, for each boundary $\partial\overline{\gamma}$, an arc J_i^{γ} is drawn in P_i, going from the boundary of $\overline{\gamma}$ to itself that separates the other two boundaries of P_i and that has minimal length.

Two curves γ' and γ'' are associated to each $\gamma \in \Gamma$ in the following way (Figure 10). The annulus $\overline{\gamma}$ is glued with the two pairs of pants P_i and P_j between which it is lying, which yields a sphere with four boundaries: $\partial\overline{\gamma_{i,1}}$ and $\partial\overline{\gamma_{i,2}}$ bounding P_i and $\partial\overline{\gamma_{j,1}}$ and $\partial\overline{\gamma_{j,2}}$ bounding P_j. A curve γ' is then defined: it coincides with J_i^{γ} in P_i and J_j^{γ} in P_j, it separates $\partial\overline{\gamma_{i,1}}$ and $\partial\overline{\gamma_{j,1}}$ from $\partial\overline{\gamma_{i,2}}$ and $\partial\overline{\gamma_{j,2}}$, and it has exactly 2 crossings with γ. The curve γ'' is defined in the same way, separating $\partial\overline{\gamma_{i,1}}$ and $\partial\overline{\gamma_{j,2}}$ from $\partial\overline{\gamma_{i,2}}$ and $\partial\overline{\gamma_{j,1}}$.

For each P_i and $m_{i,1}, m_{i,2}, m_{i,3} \in \mathbb{N}$, a model of a multiarc is fixed in P_i, having $m_{i,1}$, $m_{i,2}$ and $m_{i,3}$ intersections with the three boundaries $\partial\overline{\gamma_{i,1}}$, $\partial\overline{\gamma_{i,2}}$ and $\partial\overline{\gamma_{i,3}}$ of P_i (if one exists). The model is chosen among all the possible multiarcs as the one that has a minimal number of intersections with the three arcs $J_i^{\gamma_{i,j}} (j = 1, 2, 3)$ of P_i. The model is unique, up to homeomorphisms of the pair of pants, and those homeomorphisms are rather simple to understand since they can be decomposed into three Dehn twists around curves homotopic to the three boundaries of the pair of pants.

Let now f be a path between v_1 and v_2 on S. We decompose f into three parts: (v_1, w_1), $f^w = (w_1, w_2)$ and (w_2, v_2) where w_1 and w_2 are the first and the last points of f on an annulus boundary. We "push" all the twists of f^w into the annuli $\overline{\gamma}, \gamma \in \Gamma$, and obtain a *normal form* homotopic to f, whose definition adapts the definition given in the book [12] for closed curves:

1. It is simple.
2. It has a minimal number m_i of intersections with each $\gamma_i, i = 1, \dots, 3g-3$.
3. In each $P_j, j = 1, \dots, 2g-2$, it is homotopic with fixed endpoints to the model that corresponds to the number of intersections with its boundaries. For P_{j_1} (resp. P_{j_2}) containing v_1 (resp. v_2), only the intersections different from w_1 (resp. w_2) are counted.

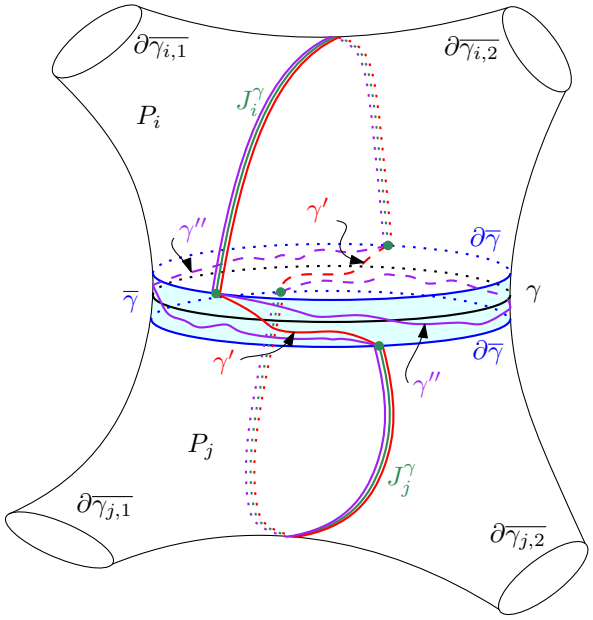


Figure 10 Two adjacent pairs of pants P_i and P_j.

4. Between v_1 and w_1 (resp. w_2 and v_2), it has a minimal number of intersections with the three arcs $J_{j_1}^{\gamma_{j_1},k}(k=1,2,3)$ in P_{j_1} containing v_1 (resp. $J_{j_2}^{\gamma_{j_2},k}$ in P_{j_2} containing v_2).
5. It has a minimal number t_i of intersections with γ_i' inside $\overline{\gamma_i}$, for any $i=1,\dots,3g-3$.
6. It has a minimal number s_i of intersections with γ_i'' inside $\overline{\gamma_i}$, for any $i=1,\dots,3g-3$.

The existence of a normal form is clear. The two forms of the path f are used to define two notions of complexity: its geodesic form is used to define its length, which can be seen as a geometric complexity, whereas its *normal coordinates* m_i, s_i and t_i can be seen as a combinatorial complexity. Lemma 18 shows some equivalence between the two notions of complexity. We first show that a fixed set of coordinates corresponds to a finite number of possible non-homotopic paths.

▶ **Lemma 17.** *For any set of coordinates $m_i, t_i, s_i, i=1,\dots,3g-3$, there are at most $9(\max_{\{i=1,\dots,3g-3\}}(m_i))^2$ non-homotopic normal forms.*

Proof. Let f be a path, decomposed as above into (v_1,w_1), $f^w=(w_1,w_2)$ and (w_2,v_2). In each pair of pants not containing any endpoint v_1 or v_2, fixing the m_i, s_i and t_i leads to a unique homotopy class of models [12, Lemma 5, p.63]. For the two (not necessarily different) pairs of pants P_{j_1} and P_{j_2} containing v_1 and v_2, w_1 and w_2 are in fact fixing unique models (see Figure 11). There are three possible annulus boundaries $\partial\overline{\gamma_{j,i}}, i=1,2,3$ for w_1 in the pair of pants P_j that contains v_1 (resp. $\overline{\gamma_{j,i}}$ for w_2), so, at most $3\max_{\{i\}}(m_i)$ possibilities for each of them. The choices for w_1 and w_2 are independent and the result follows. ◀

▶ **Lemma 18.** *Let f be a geodesic segment of length l, then there exists a constant c_h such that the coordinates m_i, t_i and $s_i, i=1,\dots,3g-3$ of the normal form of f are smaller than $c_h \cdot l$.*

Proof. For any simple closed geodesic δ on S, the geodesic form of f intersects δ in a minimal number k_δ of points, since they are both geodesics. If ε_δ is the width of a tubular neighborhood of δ, then $l \geq \varepsilon_\delta(k_\delta-1)$ [2, Lemma 3.1]. Each coordinate m_i, t_i and s_i of f

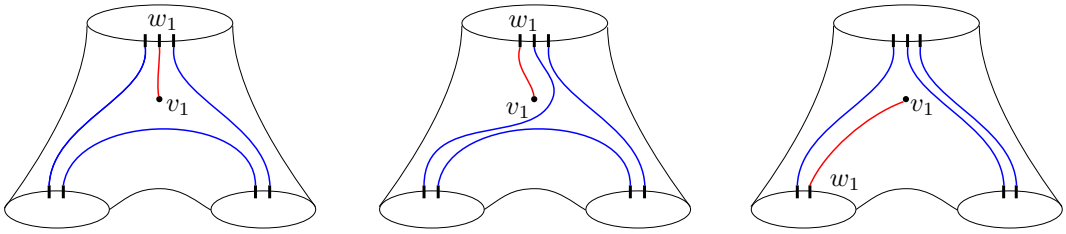


Figure 11 Three possible choices for w_1. The two left choices correspond to the same model, but the orderings on the upper boundary lead to non-homotopic paths. The right choice leads to different models.

corresponds to the minimal number of intersections with a curve. The number m_i corresponds to γ_i. The number t_i is actually not larger than the number of intersections of f with the geodesic curve that is homotopic to γ_i' (γ_i' is generally not geodesic), and similarly s_i is not larger than the number of intersections of f with the geodesic homotopic to γ_i''. These curves $\gamma_i, \gamma_i', \gamma_i''$ only depend on (S, h), so, we can take ε_h to be the largest of all the $9g-9$ widths $\varepsilon_{\gamma_i}, \varepsilon_{\gamma_i'}, \varepsilon_{\gamma_i''}$ and we obtain $l \geq \varepsilon_h \cdot \max(m_i, t_i, s_i)$ and thus $\max(m_i, t_i, s_i) \leq 1/\varepsilon_h \cdot l$. ◀

▶ **Theorem 19.** *For any hyperbolic structure h on S and any triangulation T of (S, h), there is a sequence of flips of length at most $C_h \cdot \Delta(T)^{6g-4} \cdot n^2$ in the geometric flip graph connecting T to a Delaunay triangulation of (S, h).*

Proof. Let N_{v_1,v_2} be the number of segments from v_1 to v_2 shorter than $l = 2 \cdot \Delta(T)$. From the previous lemma, we obtain that the $9g-9$ coordinates m_i, t_i, and s_i of any such segment f are smaller than $c_h \cdot 2\Delta(T)$. It appears that, $\forall i, m_1 = t_i + s_i$, $t_i = m_i + s_i$ or $s_i = m_i + t_i$ [12, Lemma 6, p.64 & Fig.5, p.65]. So, if we fix m_i and t_i there are at most 3 possible s_i. Lemma 17 and 18 proves that there are $9(c_h \cdot 2\Delta(T))^2$ potential segments for each coordinate set. We obtain a bound for N_{v_1,v_2}: $N_{v_1,v_2} \leq 9(c_h \cdot 2\Delta(T))^2 \cdot 3(c_h \cdot 2\Delta(T))^{6g-6}$ and thus, there is a constant C_h' such that $N_{v_1,v_2} \leq C_h' \cdot \Delta(T)^{6g-4}$. Since there are $1/2 \cdot n^2$ possible sets $\{v_1, v_2\}$, we obtain the bound on the number of edges. ◀

References

1 Marcel Berger. *Geometry*. Springer, 1996.

2 Joan S Birman and Caroline Series. Geodesics with bounded intersection number on surfaces are sparsely distributed. *Topology*, 24(2):217–225, 1985.

3 Mikhail Bogdanov, Olivier Devillers, and Monique Teillaud. Hyperbolic Delaunay complexes and Voronoi diagrams made practical. *Journal of Computational Geometry*, 5:56–85, 2014. doi:10.20382/jocg.v5i1a4.

4 Mikhail Bogdanov, Iordan Iordanov, and Monique Teillaud. 2D hyperbolic Delaunay triangulations. In *CGAL User and Reference Manual*. CGAL Editorial Board, 4.14 edition, 2019. URL: https://doc.cgal.org/latest/Manual/packages.html#PkgHyperbolicTriangulation2.

5 Mikhail Bogdanov, Monique Teillaud, and Gert Vegter. Delaunay triangulations on orientable surfaces of low genus. In *Proceedings of the Thirty-second International Symposium on Computational Geometry*, pages 20:1–20:17, 2016. doi:10.4230/LIPIcs.SoCG.2016.20.

6 Manuel Caroli and Monique Teillaud. Delaunay triangulations of closed Euclidean d-orbifolds. *Discrete & Computational Geometry*, 55(4):827–853, 2016. URL: http://hal.inria.fr/hal-01294409, doi:10.1007/s00454-016-9782-6.

7 Andrew J. Casson and Steven A. Bleiler. *Automorphisms of surfaces after Nielsen and Thurston*, volume 9 of *London Mathematical Society Student Texts*. Cambridge University Press, Cambridge, 1988. doi:10.1017/CBO9780511623912.

8 Carmen Cortés, Clara I Grima, Ferran Hurtado, Alberto Márquez, Francisco Santos, and Jesus Valenzuela. Transforming triangulations on nonplanar surfaces. *SIAM Journal on Discrete Mathematics*, 24(3):821–840, 2010. doi:10.1137/070697987.

9 H. Edelsbrunner and R. Seidel. Voronoi diagrams and arrangements. *Discrete & Computational Geometry*, 1(1):25–44, 1986. doi:10.1007/BF02187681.

10 Benson Farb and Dan Margalit. *A Primer on Mapping Class Groups (PMS-49)*. Princeton University Press, 2012. URL: http://www.jstor.org/stable/j.ctt7rkjw.

11 Albert Fathi, François Laudenbach, and Valentin Poénaru. *Thurston's Work on Surfaces (MN-48)*. Princeton University Press, 2012.

12 Albert Fathi, François Laudenbach, Valentin Poénaru, et al. *Travaux de Thurston sur les surfaces*, volume 66–67 of *Astérisque*. Société Mathématique de France, Paris, 1979.

13 Martin Gardner. Chapter 19: Non-Euclidean geometry. In *The Last Recreations*. Springer, 1997.

14 F. Hurtado, M. Noy, and J. Urrutia. Flipping edges in triangulations. *Discrete & Computational Geometry*, 3(22):333–346, 1999. doi:10.1007/PL00009464.

15 Iordan Iordanov and Monique Teillaud. Implementing Delaunay triangulations of the Bolza surface. In *Proceedings of the Thirty-third International Symposium on Computational Geometry*, pages 44:1–44:15, 2017. doi:10.4230/LIPIcs.SoCG.2017.44.

16 Gregorii A Margulis. Applications of ergodic theory to the investigation of manifolds of negative curvature. *Functional analysis and its applications*, 3(4):335–336, 1969.

17 William S. Massey. *A basic course in algebraic topology*, volume 127 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 1991.

18 Maryam Mirzakhani. Growth of the number of simple closed geodesics on hyperbolic surfaces. *Annals of Mathematics*, 168(1):97–125, 2008.

19 Guillaume Tahar. Geometric triangulations and flips. *C. R. Acad. Sci. Paris, Ser. I*, 357:620–623, 2019.

20 J. Trainin. An elementary proof of Pick's theorem. *Mathematical Gazette*, 91(522):536–540, 2007.

An Efficient Algorithm for 1-Dimensional (Persistent) Path Homology

Tamal K. Dey
Department of Computer Science and Engineering,
The Ohio State University, Columbus, OH, 43210, USA
tamaldey@cse.ohio-state.edu

Tianqi Li
Department of Computer Science and Engineering,
The Ohio State University, Columbus, OH, 43210, USA
li.6108@osu.edu

Yusu Wang
Department of Computer Science and Engineering,
The Ohio State University, Columbus, OH, 43210, USA
yusu@cse.ohio-state.edu

Abstract

This paper focuses on developing an efficient algorithm for analyzing a directed network (graph) from a topological viewpoint. A prevalent technique for such topological analysis involves computation of homology groups and their persistence. These concepts are well suited for spaces that are not directed. As a result, one needs a concept of homology that accommodates orientations in input space. Path-homology developed for directed graphs by Grigoryan, Lin, Muranov and Yau has been effectively adapted for this purpose recently by Chowdhury and Mémoli. They also give an algorithm to compute this path-homology. Our main contribution in this paper is an algorithm that computes this path-homology and its persistence more efficiently for the 1-dimensional (H_1) case. In developing such an algorithm, we discover various structures and their efficient computations that aid computing the 1-dimensional path-homology. We implement our algorithm and present some preliminary experimental results.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases computational topology, directed graph, path homology, persistent path homology

Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.36

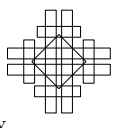
Related Version A full version of this paper is available at `https://arxiv.org/abs/2001.09549`.

Funding This work is in part supported by National Science Foundation under grants CCF1740761, DMS-1547357, and RI-1815697.

1 Introduction

When it comes to graphs, traditional topological data analysis has focused mostly on undirected ones. However, applications in social networks [1, 15], brain networks [16], and others require processing directed graphs. Consequently, topological data analysis for these applications needs to be adapted accordingly to account for directedness. Recently, some work [4, 14] have initiated to address this important but so far neglected issue.

Since topological data analysis uses persistent homology as a main tool, one needs a notion of homology for directed graphs. Of course, one can forget the directedness and consider the underlying undirected graph as a simplicial 1-complex and use a standard persistent homology pipeline for the analysis. However, this is less than desirable because the important information involving directions is lost. Currently, there are two main approaches that have




been proposed for dealing with directed graphs. One uses directed clique complexes [8, 14] and the other uses the concept of path homology [10]. In the first approach, a k-clique in the input directed graph is turned into a $(k-1)$-simplex if the clique has a single source and a single sink. The resulting simplicial complex is subsequently analyzed with the usual persistent homology pipeline. One issue with this approach is that there could be very few cliques with the required condition and thus accommodating only a very few higher dimensional simplices. In the worst case, only the undirected graph can be returned as the directed clique complex if each 3-clique is a directed cycle. The second approach based on path homology alleviates this deficiency. Furthermore, certain natural functorial properties, such as Künneth formula, do not hold for the clique complex [10].

The path homology, originally proposed by Grigoryan, Lin, Muranov and Yau in 2012 [10] and later studied by [5, 11, 12], has several properties that make it a richer mathematical structure. For example, there is a concept of homotopy under which the path homology is preserved; it accommodates Künneth formula; and the path homology theory is dual to the cohomology theory of digraphs introduced in [12]. Furthermore, persistent path homology developed in [5] is shown to respect a stability property for its persistent diagrams.

To use path homologies effectively in practice, one needs efficient algorithms to compute them. In particular, we are interested in developing efficient algorithms for computing 1-dimensional path homology and its persistent version because even for this case the current state of the art is far from satisfactory: Given a directed graph G with n vertices, the most efficient algorithm proposed in [5] has a time complexity $O(n^9)$ (more precisely, their algorithm takes $O(n^{3+3d})$ to compute the $(d-1)$-dimensional persistent path-homology).

The main contribution of this paper is stated in Theorem 1. The reduced time complexity of our algorithm can be attributed to the fact that we compute the boundary groups more efficiently. In particular, it turns out that for 1-dimensional path homology, the boundary group is determined by bigons, certain triangles, and certain quadrangles in the input directed graph. The bigons and triangles can be determined relatively easily. It is the boundary quadrangles whose computation and size determine the time complexity. The authors in [5] compute a basis of these boundary quadrangles by constructing a certain generating set for the 2-dimensional chain group by a nice column reduction algorithm (being different from the standard simplicial homology, it is non-trivial to do reduction for path homology). We take advantage of the concept of *arboricity* and related results in graph theory, together with other efficient strategies, to enumerate a much smaller set generating the boundary quadrangles. Computing the cycle and boundary groups efficiently both for non-persistent and persistent homology groups is the key to our improved time complexity.

▶ **Theorem 1.** *Given a directed graph G with n vertices and m edges, set $r = \min\{\mathsf{a}(G)m, \sum_{(u,v)\in E}(d_{in}(u) + d_{out}(v))\}$, where $\mathsf{a}(G)$ is the so-called arboricity of G (with $\mathsf{a}(G) = O(n)$), and $d_{in}(u)$ and $d_{out}(u)$ are the in-degree and out-degree of u, respectively. There is an $O(rm^{\omega-1} + m\alpha(n))$ time algorithm for computing the* 1-dimensional persistent path homology *for G where $\omega < 2.373$ is the exponent for matrix multiplication[1], and $\alpha(\cdot)$ is the inverse Ackermann function.*

This also gives an $O(rm^{\omega-1} + m\alpha(n))$ time algorithm for computing the 1*-dimensional path homology H_1 of G.*

In particular, for a planar graph G, $\mathsf{a}(G) = O(1)$ and the time complexity becomes $O(n^{\omega})$.

[1] That is, the fastest algorithm to multiply two $r \times r$ matrices takes time $O(r^{\omega})$.

The *arboricity* $\mathsf{a}(G)$ of a graph G mentioned in Theorem 1 denotes the minimum number of edge-disjoint spanning forests into which G can be decomposed [13]. It is known that in general, $\mathsf{a}(G) = O(n)$, but it can be much smaller. For example, $\mathsf{a}(G) = O(1)$ for planar graphs and $\mathsf{a}(G) = O(g)$ for a graph embedded on a genus-g surface [3]. Hence, for planar graphs, we can compute 1-dimensional persistent path homology in $O(n^\omega)$ time whereas the algorithm in [5] takes $O(n^5)$ time[2].

In the full version, we develop an algorithm to compute 1-dimensional *minimal path homology basis* [7, 9], and also show experiments demonstrating the efficiency of our new algorithms.

Organization of the paper. After characterizing the 1-dimensional path homology group H_1 in Section 3, we first propose a simple algorithm to compute it. In Section 4, we consider its persistent version and present an improved and more efficient algorithm.

2 Background

We briefly introduce some necessary background for path homology. Interested readers can refer to [10] for more details. The original definition can be applied to structures beyond directed graphs; but for simplicity, we use directed graphs to introduce the notations.

Given a directed graph $G = (V, E)$, we denote (u, v) as the directed edge from u to v. A *self-loop* is defined to be the edge (u, u) from u to itself. Throughout this paper, we assume that G does not have self-loops. We also assume that G does not have multi-edges, i.e. for every ordered pair u, v, there is at most one directed edge from u to v. For notational simplicity, we sometimes use index i to refer to vertex $v_i \in V = \{v_1, \ldots, v_n\}$.

Let $\mathbb{F}$ be a field with 0 and 1 being the additive and multiplicative identities respectively. We use $-a$ to denote the additive inverse of a in $\mathbb{F}$. An *elementary d-path* on V is simply a sequence $i_0, i_1, \cdots, i_d$ of $d+1$ vertices in V. We denote this path by $e_{i_0,i_1,\cdots,i_d}$. Let $\Lambda_d = \Lambda_d(G, \mathbb{F})$ denote the $\mathbb{F}$-linear space of all linear combinations of elementary d-paths with coefficients from $\mathbb{F}$. It is easy to check that the set $\{e_{i_0,\cdots,i_d} \mid i_0, \cdots, i_d \in V\}$ is a basis for Λ_d. Each element p of Λ_d is called a *d-path*, and it can be written as

$$p = \sum\nolimits_{i_0,\cdots,i_d \in V} a_{i_0\cdots i_d} e_{i_0\cdots i_d}, \text{ where } a_{i_0\cdots i_d} \in \mathbb{F}.$$

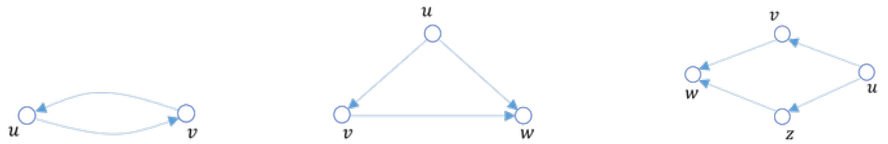
Similar to simplicial complexes, there is a well-defined *boundary operator* $\partial : \Lambda_d \to \Lambda_{d-1}$:

$$\partial e_{i_0\cdots i_d} = \sum_{i_0,\cdots,i_d \in V} (-1)^j e_{i_0\cdots\hat{i}_j\cdots i_d},$$

where $\hat{i}_k$ means the omission of index i_k. The boundary of a path $p = \sum_{i_0,\cdots,i_d\in V} a_{i_0\cdots i_d} \cdot e_{i_0\cdots i_d}$, is thus $\partial p = \sum_{i_0,\cdots,i_d\in V} a_{i_0\cdots i_d} \cdot \partial e_{i_0\cdots i_d}$. We set $\Lambda_{-1} = 0$ and note that Λ_0 is the set of $\mathbb{F}$-linear combinations of vertices in V. Lemma 2.4 in [10] shows that $\partial^2 = 0$.

Next, we restrict to real paths in directed graphs. Specifically, given a directed graph $G = (V, E)$, call an elementary d-path $e_{i_0,\cdots,i_d}$ *allowed* if there is an edge from i_k to i_{k+1} for all k. Define $\mathcal{A}_d$ as the space of all allowed d-paths, that is, $\mathcal{A}_d := \text{span}\{e_{i_0\cdots i_d} : e_{i_0\cdots i_d} \text{ is allowed}\}$. An elementary d-path $i_0 \cdots i_d$ is called *regular* if $i_k \neq i_{k+1}$ for all k, and is *irregular* otherwise. Clearly, every allowed path is regular since there is no self-loop. However,

[2] The original time complexity stated in the paper is $O(n^9)$ for 1-dimensional case. However, one can improve it to $O(n^5)$ by a more refined analysis for planar graphs.



(a) Bigon. **(b)** Boundary triangle. **(c)** Boundary quadrangle.

Figure 1 Examples of 1-boundaries.

the boundary map ∂ on Λ_d may create a term resulting into an irregular path. For example, $\partial e_{uvu} = e_{vu} - e_{uu} + e_{uv}$ is irregular because of the term e_{uu}. To deal with this case, the term containing consecutive repeated vertices is identified with 0 [10]. Thus, for the previous example, we get $\partial e_{uvu} = e_{vu} - 0 + e_{uv} = e_{vu} + e_{uv}$. The boundary map ∂ on $\mathcal{A}_d$ is taken to be the boundary map for Λ_d restricted on $\mathcal{A}_d$ with this modification: where all terms with consecutive repeated vertices created by the boundary map ∂ are replaced with 0's.

Unfortunately, after restricting to the space of allowed paths $\mathcal{A}_*$, the inclusion that $\partial\mathcal{A}_d \subset \mathcal{A}_{d-1}$ may not hold any more; that is, the boundary of an allowed d-path is not necessarily an allowed $(d-1)$-path. To this end, we adopt a stronger notion of allowed paths: an allowed path p is *∂-invariant* if ∂p is also allowed. Let $\Omega_d := \{p \in \mathcal{A}_d \mid \partial p \in \mathcal{A}_{d-1}\}$ be the space generated by all ∂-invariant paths. We then have $\partial\Omega_d \subset \Omega_{d-1}$ (as $\partial^2 = 0$). This gives rise to the following *chain complex of ∂-invariant allowed paths*:

$$\cdots \Omega_d \xrightarrow{\partial} \Omega_{d-1} \xrightarrow{\partial} \cdots \Omega_d \xrightarrow{\partial} \Omega_0 \xrightarrow{\partial} 0.$$

We can now define the homology groups of this chain complex. The d-th cycle group is defined as $\mathsf{Z}_d = \operatorname{Ker} \partial|_{\Omega_d}$, and elements in Z_d are called d-cycles. The d-th boundary group is defined as $\mathsf{B}_d = \operatorname{Im} \partial|_{\Omega_{d+1}}$, with elements of B_d being called d-boundary cycles (or simply d-boundaries). The resulting *d-dimensional path homology group* is $\mathsf{H}_d(G, \mathbb{F}) = \mathsf{Z}_d/\mathsf{B}_d$.

2.1 Examples of 1-boundaries

Below we give three examples of 1-boundaries; see Figure 1.

Bi-gon. A *bi-gon* is a 1-cycle $e_{uv} + e_{vu}$ consisting of two edges (u, v) and (v, u) from E; see Figure 1(a). Consider the 2-path e_{uvu}. We have that its boundary is $\partial(e_{uvu}) = e_{vu} - e_{uu} + e_{uv} = e_{vu} + e_{uv}$. Since both e_{vu} and e_{uv} are allowed 1-paths, it follows that any bi-gon $e_{vu} + e_{uv}$ of G is necessarily a 1-boundary.

Boundary triangle. Consider the 1-cycle $C = e_{vw} - e_{uw} + e_{uv}$ of G (it is easy to check that $\partial C = 0$). Now consider the 2-path e_{uvw}: its boundary is then $\partial(e_{uvw}) = e_{vw} - e_{uw} + e_{uv} = C$. Note that every summand in the boundary is allowed. Thus C is a 1-boundary. We call any triangle in G isomorphic to C a *boundary triangle*. Note that a boundary triangle always has one sink and one source; see the source u and sink w in Figure 1(b). In what follows, we use $(u, w \mid v)$ to denote a boundary triangle where u is the source and w is the sink.

Boundary quadrangle. Consider the 1-cycle $C = e_{uv} + e_{vw} - e_{uz} - e_{zw}$ from G. It is easy to check that C is the boundary of the 2-path $e_{uvw} - e_{uzw}$, as $\partial(e_{uvw} - e_{uzw}) = e_{vw} - e_{uw} + e_{uv} - (e_{zw} - e_{uw} + e_{uz}) = e_{vw} + e_{uv} - e_{zw} - e_{uz} = C$. We call any quadrangle isomorphic to C a *boundary quadrangle*.

In the remainder of the paper, we use $R(u, v, w, z)$ to represent a quadrangle; i.e, a 1-cycle consisting of 4 edges whose *undirected version* has the form $(u, v) + (v, w) + (w, z) + (z, u)$. (Note that a quadrangle may not be a boundary quadrangle). We denote a boundary quadrangle $e_{uv} + e_{vw} - e_{uz} - e_{zw}$ by $\{u, w \mid v, z\}$, where u and w are the source and sink of this boundary quadrangle respectively.

3 Computing 1-dimensional path homology H_1

Note that the 1-dimensional ∂-invariant path space $\Omega_1 = \Omega_1(G)$ is the space generated by all edges [10] because the boundary of every edge is allowed by definition.

Now consider the 1-cycle group $\mathsf{Z}_1 \subseteq \Omega_1$; that is, Z_1 is the kernel of ∂ applied to Ω_1. We show below that a basis of Z_1 can be computed by considering a spanning tree of the undirected version of G, which denoted by G_u. This is well known when $\mathbb{F}$ is $\mathbb{Z}_2$. It is easy to see that this spanning tree based construction also works for arbitrary field $\mathbb{F}$.

Specifically, let T be a rooted spanning tree of G_u with root r, and $\bar{T} := G_u \setminus T$. For every edge $e = (v_1, v_2) \in \bar{T}$, let c_e be the 1-cycle (under $\mathbb{Z}_2$) obtained by summing e and all edges on the paths π_1 and π_2 between v_1 and r, and v_2 and r respectively. The cycles $\{c_e, e \in \bar{T}\}$ form a basis of 1-cycle group of G_u under $\mathbb{Z}_2$ coefficient. Now for every such cycle c_e in G_u, we also have a cycle in $\Omega_1(G)$ containing same edges with c_e which are assigned a coefficient 1 or -1 depending on their orientations in G. We call this 1-cycle in $\Omega_1(G)$ also c_e. Then we have the following proposition, whose proof is in the full version.

▶ **Proposition 2.** *The cycles $\{c_e | e \in \bar{T}\}$ in $\Omega_1(G)$ form a basis for Z_1 under any coefficient field $\mathbb{F}$.*

Now, we show a relation between 1-dimensional homology, cycles, bigons, triangles and quadrangles. Recall that bi-gons, boundary triangles and boundary quadrangles are specific types of 1-dimensional boundaries with two, three or four vertices, respectively; see Section 2.1. The following theorem is similar to Proposition 2.9 from [11], where the statement there is under coefficient ring $\mathbb{Z}$. For completeness, we include the (rather similar) proof for our case in the full version.

▶ **Theorem 3.** *Let $G = (V, E)$ be a directed graph. Let Q denote the space generated by all boundary triangles, boundary quadrangles and bi-gons in G. Then we have $\mathsf{B}_1 = \mathsf{Q}$.*

▶ **Corollary 4.** *The 1-dimensional path homology group satisfies that $\mathsf{H}_1 = \mathsf{Z}_1/\mathsf{Q}$.*

3.1 A simple algorithm

Theorem 3 and Corollary 4 provide us a simple framework to compute H_1. Below we only focus on the computation of the rank of H_1; but the algorithm can easily be modified to output a basis for H_1 as well. Later in Section 4, we will develop a more efficient and sophisticated algorithm for the 1-dimensional *persistent* path homology H_1, which as a by-product, also gives a more efficient algorithm to compute H_1.

In the remaining of this paper, we represent each cycle in Z_1 with a vector. Assume all edges are indexed from 1 to m as $e_1, \cdots, e_m$ where m is the number of edges. Then, each 1-cycle C is an m-dimensional vector, where $C[i] \in \mathbb{F}$ records the coefficient for edge e_i in C.

(Step 1): cycle group Z_1. By Proposition 2, $rank(\mathsf{Z}_1) = |E| - |V| + 1$ for directed graph $G = (V, E)$. The computation of the rank takes $O(1)$ time (or $O(|V|^2)$ time if we need to output a basis of it explicitly).

Algorithm 1 A simple first algorithm to compute rank of H_1.

```
procedure CompH1-Simple(G, t)
    (Step 1): Compute rank of 1-cycle group Z_1
    (Step 2): Compute rank of 1-boundary group B_1
        (Step 2.a) Compute a generating set C of 1-boundary cycles that generates B_1
        (Step 2.b) From C compute a basis for B_1
    Return rank(H_1) = rank(Z_1) − rank(B_1).
end procedure
```

(Step 2): boundary group B_1. Note that by Theorem 3, we can compute the set of all bigons, boundary triangles and boundary quadrangles as a *generating set* C of 1-boundary cycles (meaning that it generates the boundary group B_1) for (Step 2.a). However, such a set C could have size $\Omega(n^2)$ even for a planar graph, where $n = |V|$; see Figure 2. (For a general graph, the number of boundary quadrangles could be $\Theta(n^4)$.)

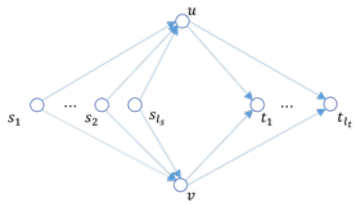


Figure 2 There are n vertices but $l_s \cdot l_t = \Theta(n^2)$ quadrangles, $l_s = \lfloor (n-2)/2 \rfloor$ and $l_t = \lceil (n-2)/2 \rceil$.

To make (Step 2.b) efficient, we wish to have a generating set C of 1-boundary cycles with *small cardinality*. To this end, we leverage a classical result of [3] to reduce the size of C.

Given an undirected graph G, its *arboricity* $\mathsf{a}(G)$ is the minimum number of edge-disjoint spanning forests which G can be decomposed into [13]. An alternative definition is

$$\mathsf{a}(G) = \max_{H \text{ is a subgraph of } G} \frac{|E(H)|}{|V(H)| - 1}.$$

From this definition, it is easy to see (and well-known, see e.g, [3]) that:

▶ **Observation 3.1.**
(1) *If G is a planar graph, or a graph with bounded vertex degrees, then* $\mathsf{a}(G) = O(1)$.
(2) *If G is a graph embedded on a genus g surface, then* $\mathsf{a}(G) = O(g)$.
(3) *In general, if G does not contain self-loops, then* $\mathsf{a}(G) = O(n)$.

We will leverage some classical results from [3]. First, to represent quadrangles, we use the following *triple-list* representation [13] : a triple-list $(u, v, \{w_1, w_2, \cdots, w_l\})$ means that for each i, w_i is adjacent to both u and v, where we say u' and v' are adjacent if either (u', v') or (v', u') are in E (i.e, u' and v' are adjacent when disregarding directions). Given such a triple-list $\xi = (u, v, \{w_1, w_2, \cdots, w_l\})$, it is easy to see that u, w_i, v, w_j form the *consecutive vertices* of a quadrangle in the undirected version of graph G; and we also say that the undirected quadrangle $R(u, w_i, v, w_j)$ is *covered* by this triple-list. We say that a vertex z is in a triple-list $(u, v, \{w_1, w_2, \cdots, w_l\})$ if it is in the set $\{w_1, w_2, \cdots, w_l\}$.

The size of a triple-list is the total number of vertices contained in it. This triple-list ξ thus represents $\Theta(l^2)$ number of undirected quadrangles in G succinctly with $\Theta(l)$ size.

▶ **Proposition 5** ([3]).
(1) *Let G be a connected undirected graph with n vertices and m edges. There is an algorithm listing all the triangles in G in $O(\mathsf{a}(G)m)$ time.*
(2) *There is an algorithm to compute a set of triple-lists which covers all quadrangles in a connected graph G in $O(\mathsf{a}(G)m)$ time. The total size complexity of all triple-lists is $O(\mathsf{a}(G)m)$.*
Using the above result, we can have the following theorem, with proof in the full version.

▶ **Theorem 6.** *Let $G = (V, E)$ be a directed graph with n vertices and m edges. We can compute a generating set C of 1-boundary cycles for B_1 with cardinality $O(\mathsf{a}(G)m)$ in time $O(\mathsf{a}(G)m)$.*

It then follows from Theorem 3 that (Step 2.a) can be implemented in $O(\mathsf{a}(G)m)$ time, producing a generating set of cardinality $O(\mathsf{a}(G)m)$. Finally, representing each boundary cycle in C as a vector of dimension $m = |E|$, we can then compute the rank of cycles in C in $O(|\mathsf{C}|m^{\omega-1}) = O(\mathsf{a}(G)m^{\omega})$,where $\omega < 2.373$ is the exponent for matrix multiplication [2].

Putting everything together, we have that

▶ **Theorem 7.** *Given a directed graph $G = (V, E)$ with $n = |V|$ and $m = |E|$, Algorithm 1 computes the rank of the 1-dimensional path homology group H_1 in $O(\mathsf{a}(G)m^{\omega})$ time.*

The algorithm can be extended to compute a basis for H_1 with the same time complexity.

For example, by Observation 3.1, if G is a planar graph, then we can compute H_1 in $O(n^{\omega})$. For a graph G embedded on a genus g surface, H_1 can be computed in $O(gn^{\omega})$ time. In contrast, we note that the algorithm of [5] takes $O(n^5)$ time for planar graphs.

4 Computing persistent path homology H_1

The concept of arboricity used in the previous section does not consider edge directions. Indeed, our algorithm to compute a generating set C as given in the proof of Theorem 6 first computes a (succinct) representation of all quadrangles, whether they contribute to boundary quadrangles or not. On the other hand, as Figure 3 illustrates, a graph G can have no boundary quadrangle despite the fact that the graph is dense (with $\Theta(n^2)$ edges and thus $\mathsf{a}(G) = \Theta(n)$ arboricity). Another way to view this is that the example has no allowed 2-path, and thus no ∂-invariant 2-paths and consequently no 1-boundary cycles. Our algorithm will be more efficient if it can also respect the number of allowed elementary 2-paths.

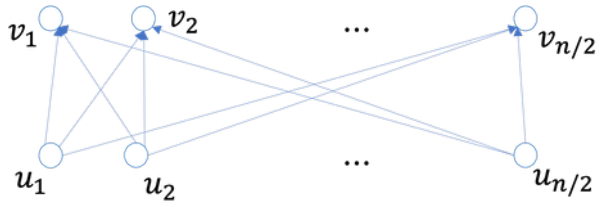


Figure 3 A dense graph with no boundary quadrangle.

In fact, a more standard and natural way to compute a basis for the 1-boundary group proceeds by taking the boundary of ∂-invariant 2-paths. The complication is that unlike in the simplicial homology case, it is not immediately evident how to compute a basis for Ω_2 (the space of ∂-invariant 2-paths). Nevertheless, Chowdhury and Mémoli presented

an elegant algorithm to show that a basis for B_1 (and H_1) can still be computed using careful column-based matrix reductions [5]. The time complexity of their algorithm is $O((\sum_{(u,v)\in E}(d_{in}(u)+d_{out}(v)))mn^2)$ which depends on the number of elementary 2-paths[3].

In this section, we present an algorithm that can take advantage of both of the previous approaches (the algorithm of [5] and Algorithm 1). Similar to [5], we will now consider the persistent path homology setting, where we will add directed edges in G one by one incrementally. Hence our algorithm can compute the *persistent* H_1 w.r.t. a filtration. However different from [5], instead of reducing a matrix with columns corresponding to all elementary allowed 2-paths, we will follow a similar idea as in Algorithm 1 and add a generating set of boundary cycles each time we consider a new directed edge.

4.1 Persistent path homology

We now introduce the definition of the persistent path homology [5]. The *persistent vector space* is a family of vector spaces together with linear maps $\{U^\delta \xrightarrow{\mu_{\delta,\delta'}} U^{\delta'}_{\delta\le\delta'\in\mathbb{R}}\}$ so that: (1) $\mu_{\delta,\delta}$ is the identity for every $\delta\in\mathbb{R}$; and (2) $\mu_{\delta,\delta''}=\mu_{\delta,\delta'}\circ\mu_{\delta',\delta''}$ for each $\delta\le\delta'\le\delta''\in\mathbb{R}$.

Let $G=(V,E,w)$ be a weighted directed graph where V is the vertex set, E is the edge set, and w is the weight function $w: E\to\mathbb{R}^+$. For every $\delta\in\mathbb{R}^+$, a directed graph G^δ can be constructed as $G^\delta=(V^\delta=V, E^\delta=\{e\in E: w(e)\le\delta\})$. This gives rise to a filtration of graphs $\{G^\delta\hookrightarrow G^{\delta'}\}_{\delta\le\delta'\in\mathbb{R}}$ using the natural inclusion map $i_{\delta,\delta'}: G^\delta\hookrightarrow G^{\delta'}$.

▶ **Definition 8** ([5]). *The 1-dimensional persistent path homology of a weighted directed graph $G=(V,E,w)$ is defined as the persistent vector space $\mathbb{H}_1:=\{\mathsf{H}_1(G^\delta)\xrightarrow{i_{\delta,\delta'}}\mathsf{H}_1(G^{\delta'})\}_{\delta\le\delta'\in\mathbb{R}}$. The 1-dimensional path persistence diagram $Dg(G)$ of G is the persistence diagram of $\mathbb{H}_1$.*

To compute the path homology $\mathsf{H}_1(G)$ of an unweighted directed graph $G=(V,E)$, we can order edges in E arbitrarily with the index of an edge in this order being its weight. The rank of $\mathsf{H}_1(G)$ can then be retrieved from the 1-dimensional persistent homology group induced by this filtration by considering only those homology classes that "never die".

4.2 A more efficient algorithm

In what follows, to simplify presentation, we assume that we are given a directed graph $G=(V,E)$, where edges are already sorted $e_1,\dots,e_m$ in increasing order of their weights. Let $G^{(i)}=(V,E^{(i)}=\{e_1,\dots,e_i\})$ denote the subgraph of G spanned by the edges $e_1,\dots,e_i$; and set $G^{(0)}=(V,\varnothing)$. We now present an algorithm to compute the 1-dimensional persistent path homology induced by the nesting sequence $G^{(0)}\subseteq G^{(1)}\subseteq\cdots G^{(m)}$. In particular, in Algorithm 2, as we insert each new edge e_s, moving from $G^{(s-1)}$ to $G^{(s)}$, we maintain a basis for $\mathsf{Z}_1(s):=\mathsf{Z}_1(G^{(s)})$ and for $\mathsf{B}_1(s):=\mathsf{B}_1(G^{(s)})$, updated from $\mathsf{Z}_1(s-1)$ and $\mathsf{B}_1(s-1)$ and output new persistent pairs. On a high level, this algorithm follows the standard procedure in [6]; details are in the full version.

4.2.1 Procedure GenSet(s)

Note that $G^{(s)}$ is obtained from $G^{(s-1)}$ by inserting a new edge $e_s=(u,v)$ to $G^{(s-1)}$. At this point, we have already maintained a basis B for $\mathsf{B}_1(G^{(s-1)})$. Our goal is to compute a set of generating boundary cycles C_s such that $B\cup\mathsf{C}_s$ contains a basis for $\mathsf{B}_1(G^{(s)})$.

[3] The time complexity given in the paper [5] assumes that the input directed graph is complete, and takes $O(n^9)$ to compute H_1. However, a more refined analysis of their time complexity shows that it can be improved to $O((\sum_{(u,v)\in E} d_{in}(u)+d_{out}(v))mn^2)$.

Algorithm 2 Compute 1-D persistent path homology for a directed graph $G = (V, E)$.

```
1: procedure PERSISTENCE(G)
2:     Order the edges in non-decreasing order of their weights: e_1, ..., e_m.
3:     Set G^(0) = (V, ∅), current basis for 1-boundary group is B = ∅.
4:     for s = 1 to m do
5:         Call GENSET(s) to compute a generating set C_s containing a basis for newly
       generated 1-boundary cycles moving from G^(s−1) to G^(s).
6:         Call FINDPAIRS(s) to output new persistent pairs, and update the boundary basis
       B for G^(s).
7:     end for
8: end procedure
```

We first inspect the effect of adding edge $e_s = (u, v)$ to $G^{(s)}$. Two cases can happen:

Case-A: The endpoints u and v are in different connected components in (the undirected version of) $G^{(s-1)}$, and after adding e_s, those two components are merged into a single one in $G^{(s)}$. In this case, no cycle is created, nor does the boundary group change. Thus $\mathsf{Z}_1(G^{(s-1)}) = \mathsf{Z}_1(G^{(s)})$ and $\mathsf{B}_1(G^{(s-1)}) = \mathsf{B}_1(G^{(s)})$. We say that edge e_s is *negative* in this case (as it kills in H_0).
The algorithm maintains the set of negative edges seen so far, which is known to form a spanning forest T_s of V. (Here, we abuse the notation slightly and say that a set of directed edges span a tree for a set of vertices if they do so when directions are ignored.) The algorithm maintains T_s via a union-find data structure.

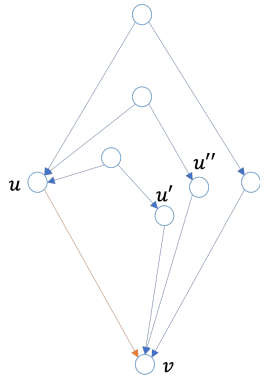


Figure 4 The insertion of edge (u, v) increases the rank of the boundary group by 3.

Case-B: The endpoints u and v are already in the same connected component in $G^{(s-1)}$. After adding this edge e_s, new cycles are created in $G^{(s)}$. Hence e_s is *positive* in this case (as it creates an element in Z_1; although different from the standard simplicial homology, it may not necessarily create an element in H_1 as we will see later).
Whether e_s is positive or negative can be easily determined by performing two Find operations in the union-find data structure representing T_{s-1}. A $\mathsf{Union}(u, v)$ operation is performed to update T_{s-1} to T_s if e_s is negative.

We now describe how to handle (Case-B). After adding edge e_s, multiple cycles containing e_s can be created in $G^{(s)}$. Nevertheless, by Proposition 2, the dimension of Z_1 increases only by 1. On the other hand, the addition of e_s may create new boundary cycles. Interestingly, it could increase the rank of B_1 by more than 1. See Figure 4 for an example where $rank(\mathsf{B}_1)$ increases by 3; and note that this number can be made arbitrarily large.

As mentioned earlier, in this case, we wish to compute a set of generating boundary cycles C_s such that $B \cup \mathsf{C}_s$ contains a basis for $\mathsf{B}_1(G^{(s)})$.

Similar to Algorithm 1, using Theorem 3, we choose some bigons, boundary triangles and boundary quadrangles and add them to C_s. In particular, since C_s only accounts for the newly created boundary cycles, we only need to consider bigons, boundary triangles and boundary quadrangles that contain e_s. We now describe the construction of C_s, which is initialized to be $\varnothing$.

(i) *Bigons.* At most one bigon can be created after adding e_s (namely, the one that contains e_s). We add it to C_s if this bigon exists.

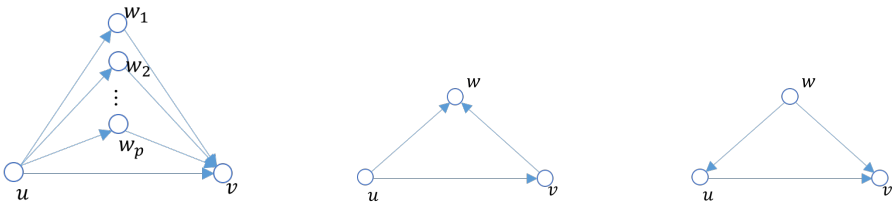


Figure 5 Three types of boundary triangles incident to $e_s = (u, v)$.

(ii) *Boundary triangles.* There could be three types of newly created boundary triangles containing $e_s = (u, v)$. The first case is when u is the source and v is the sink; see Figure 5(a). In this case multiple 2-paths may exist from u to v, $e_{uw_1v}, e_{uw_2v}, \cdots e_{uw_pv}$, forming multiple boundary triangles of this type containing e_s. However, we only need to add *one* triangle of them into C_s, say $(u, v \mid w_1)$ since every other triangle $(u, v \mid w_j)$ can be written as a linear combination of $(u, v \mid w_1)$ and an existing boundary quadrangle $(u, v \mid w_1, w_j)$ in $G^{(s-1)}$.

For the second case (see Figure 5(b)) where u is the source but v is not the sink, we include all such boundary triangles to C_s. We also add all boundary triangles of the last type in which v is the sink but u is not the source to C_s; see Figure 5(c). It is easy to see that $\mathsf{C}_s \cup B$ can generate all new boundary triangles containing $e_s = (u, v)$.

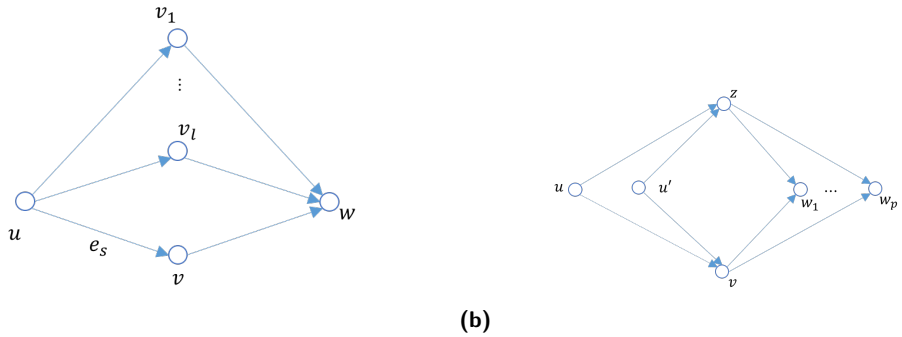


Figure 6 (a) Examples of new boundary quadrangles with u being the source. (b) Not all boundary quadrangles in M will be added to the generating set C_s.

(iii) *Boundary quadrangles.* Given an edge $e_s = (u, v)$, there are two types of the boundary quadrangles incident to it: one has u as the source; the other has v as the sink. We focus on the first case; see Figure 6(a). The second case can be handled symmetrically.

In particular, we will first compute a set M and then select a subset of quadrangles from M for adding to C_s.

Specifically, take any successor w of v, that is, there is an edge $(v, w) \in G^{(s-1)}$ forming an allowed 2-path e_{uvw} in $G^{(s)}$. Before introducing the edge e_s, there may be multiple allowed 2-paths $e_{uv_1w}, e_{uv_2w}, \cdots, e_{uv_lw}$ in $G^{(s-1)}$; see Figure 6 (a). For each such 2-path $e_{uv_kw}, 1 \leq k \leq l$, a new boundary quadrangle $(u, w \mid v, v_k)$ containing $e_s = (u, v)$ will be created. However, among all such 2-paths $e_{uv_1w}, \cdots, e_{uv_lw}$, we will pick just one 2-path, say e_{uv_1w} and only add the quadrangle $(u, w \mid v, v_1)$ formed by e_{uvw} and e_{uv_1w} to M. Observe that any other boundary quadrangle containing 2-path e_{uvw}, say $(u, w \mid v, v_k)$, can be written as a linear combination of the quadrangle $(u, w \mid v, v_1)$ and boundary quadrangle $(u, w \mid v_k, v_1)$ which is already in $G^{(s-1)}$ (and in the span of B which is a basis for $\mathsf{B}_1(G^{(s-1)})$). In other words, $(u, w \mid v, v_1) \cup B$ generates any other boundary quadrangle containing 2-path e_{uvw}.

We perform this for each successor w of v. Hence this step adds at most $d_{out}^{G^{(s-1)}}(v)$ number of boundary quadrangles to the set M.

Not all quadrangles in M will be added to C_s. In particular, suppose we have p quadrangles $A = \{(u, w_j \mid v, z) : 1 \leq j \leq p\} \subseteq M$ incident to the newly inserted edge $e_s = (u, v)$ as well as another vertex z, i.e. there are edges $(u, z), (w_j, z)$ and (v, w_j), $1 \leq j \leq p$; see Figure 6 (b). If there does not exist any other vertex u' such that edges $(u', z), (u', v) \in G^{(s-1)}$, then we add *all* quadrangles in A to C_s. If this is not the case, let u' be another vertex such that (u', z) and (u', v) are already in $G^{(s-1)}$; see Figure 6 (b). In this case, we only add *one* quadrangle from set A, say, $(u, w_1|v, z)$ to the generating set C_s.

It is easy to check that any other quadrangle $(u, w_j \mid v, z)$, $1 < j \leq p$, can be written as the combination of $(u, w_1 \mid v, z)$, $(u', w_1 \mid v, z)$ and $(u', w_j \mid v, z)$. As the latter two quadrangles are boundary quadrangles from $G^{(s-1)}$, they can already be generated by B. The entire process takes time $O(|M|) = O(d_{out}^{G^{(s-1)}}(v))$. It is also easy to see that $B \cup \mathsf{C}_s$ can generate any boundary quadrangle containing $e_s = (u, v)$ and with u being its source.

The case when v is the sink of a boundary quadrangle is handled symmetrically in time $O(d_{in}^{G^{(s-1)}}(u))$. Hence the total time to compute a generating set C_s is $O(d_{in}^{G^{(s-1)}}(u) + d_{out}^{G^{(s)}}(v))$ when inserting a single edge $e_s = (u, v)$.

4.3 Analysis of Algorithm 2

Correctness of the algorithm. Notice that the invariant that B is a basis for $G^{(s)}$ at the end of the for-loop (line-7 of Algorithm 2) is maintained. Furthermore, B is always in reduced form which is maintained via left-to-right column additions only. Hence the algorithm computes the 1-dimensional persistent path homology correctly [6].

Time complexity analysis. The remainder of this section is devoted to determining the time complexity of Algorithm 2. Specifically, we first show the following theorem.

▶ **Theorem 9.** *Across all stages $s \in [1, m]$, the total cardinality of the generating set $\mathsf{C} = \cup_s \mathsf{C}_s$ is $O(\min\{\mathsf{a}(G)m, \sum_{(u,v)\in E}(d_{in}(u) + d_{out}(v))\})$. The total time taken by procedure NewBasis(s) for all $s \in [1, m]$ is $O(m + \sum_{(u,v)\in E}(d_{in}(u) + d_{out}(v))\})$.*

Proof. We will count separately the number of bigons, boundary triangles, and boundary quadrangles added to any C_s. Set $r = \min\{\mathsf{a}(G)m, \sum_{(u,v)\in E}(d_{in}(u) + d_{out}(v))\}$.

(i) *Bigons:* First, it is easy to see that for each edge $e_s = (u, v)$ with $s \in [1, m]$, at most one bigon (incident to e_s) is added. Besides, if $d_{out}(v) = 0$, there is no bigon incident to e_s. Hence the total number ever added to C is $O(\min\{m, \sum_{(u,v)\in E}(d_{out}(v))\}) = O(r)$ and it takes $O(m)$ time to compute them.

(ii) *Boundary triangles:* For boundary triangles, we know from Proposition 5 that there are altogether $O(\mathsf{a}(G)m)$ triangles (thus at most $O(\mathsf{a}(G)m)$ boundary triangles) in a graph G and they can all be enumerated in $O(\mathsf{a}(G)m)$ time. Obviously, the number of boundary triangles ever added to C is at most $O(\mathsf{a}(G)m)$.
We now argue that the number of boundary triangles added to C is also bounded by $O(\sum_{(u,v)\in E}(d_{in}(u) + d_{out}(v)))$. Note that, for every 2-path, at most one boundary triangle is added to the set. Since the number of 2-paths is indeed $\Theta(\sum_{(u,v)\in E}(d_{in}(u) + d_{out}(v)))$, the number of triangles we add is $O(\sum_{(u,v)\in E}(d_{in}(u) + d_{out}(v)))$. Recall that there are three cases for boundary triangles added; see Figure 5. The time spent for the first case for every s is $O(1)$by recording any 2-path e_{uwv}, and $O(d_{in}(u) + d_{out}(v))$ for the last two cases. Thus the total time spent at adding boundary triangles incident to e_s and identifying triangles to be added to C_s for all $s \in [1, m]$ takes $O(m + \sum_{(u,v)\in E}(d_{in}(u) + d_{out}(v)))$ time.

(iii) *Boundary quadrangles:* The situation here is somewhat opposite to that of the boundary triangles: Specifically, it is easy to see that this step accesses at most $O(d_{in}(u)+d_{out}(v))$ boundary quadrangles when handling edge $e_s = (u, v)$. Hence the number of boundary quadrangles it can add to C_s is at most $O(d_{in}(u) + d_{out}(v))$. The total number of boundary quadrangles ever added to C is thus bounded by $O(\sum_{(u,v)\in E}(d_{in}(u) + d_{out}(v)))$.
We now prove that the number of boundary quadrangles ever added to C is also bounded by $O(\mathsf{a}(G)m)$. We use the existence of a succinct representation of all quadrangles as specified in Proposition 5 to help us argue this upper bound. Notice that our algorithm **does not** compute this representation. It is only used to provide this complexity analysis.

Specifically, by Proposition 5, we can compute a list L of triple-lists with $O(\mathsf{a}(G)m)$ total size complexity, which generates all undirected quadrangles. Following the proof of Theorem 6(see the full version), we can further refine this list, where each triple-list $\xi \in L$ further gives rise to three lists that are of type-1, 2, or 3. Let $\widehat{L}$ denote this refinement of L, consisting of lists of type-1, 2 or 3. From the proof of Theorem 6, we know that the total size complexity for all lists in $\widehat{L}$ is still $O(\mathsf{a}(G)m)$. This also implies that the cardinality of $\widehat{L}$ is bounded by $|\widehat{L}| = O(\mathsf{a}(G)m)$.

We now denote by $\mathcal{R}$ the set of all boundary quadrangles ever added to $\mathsf{C} = \cup_s \mathsf{C}_s$ by Algorithm 2. Furthermore, let

$$\mathsf{P} := \{(\xi, w) \mid \xi \in \widehat{L}, w \in \xi\}.$$

Below we show that we can find an injective map $\pi : \mathcal{R} \to \mathsf{P}$. But first, note that $|\mathsf{P}|$ is proportional to the total size complexity of $\widehat{L}$ and thus is bounded by $O(\mathsf{a}(G)m)$.

We now establish the injective map $\pi : \mathcal{R} \to \mathsf{P}$. Specifically, we process each boundary quadrangle *in the order* that they are added to C. Consider a boundary quadrangle $R = R(u, v, w, z)$ added to C_s while processing edge $e_s = (u, v)$. There are two cases: The first is that R is of the form $(u, w \mid v, z)$ in which u is the source of this quadrangle. The second is that it has the form $(w, v \mid u, z)$ in which v is the sink. We describe the map $\pi(R)$ for the first case, and the second one can be analyzed symmetrically.

By construction of L, there is at least one triple-list $\xi \in L$ covering $R = (u, w \mid v, z)$. There are three possibilities:

Case-a: The triple-list ξ is of the form $\xi = \xi_{uw} = (u, w, \{\cdots\})$. In this case, the boundary quadrangle $R = (u, w \mid v, z)$ is in a type-1 list $\xi^{(1)}_{uw} = (u, w, S) \in \widehat{L}$, and both $v, z \in S$. We now claim that the pair $(\xi^{(1)}_{uw}, v) \in \mathsf{P}$ has not yet been mapped (i.e, there is no $R' \in C$ with

$\pi(R') = (\xi^{(1)}_{uw}, v)$ yet), and we can thus set $\pi(R) = (\xi^{(1)}_{uw}, v) \in \mathsf{P}$. Suppose on the contrary there already exists $R' \in C$ that we processed earlier than R with $\pi(R') = (\xi^{(1)}_{uw}, v)$. In that case, $R' = (u, w \mid v, z')$ must contain the 2-path e_{uvw} as well. Since R' is processed earlier than R, and edge $e_s = (u, v)$ is the most recent edge added, R' must be added when we process e_s as well (as R' contains e_s). However, Algorithm 2 in this case only adds *one* quadrangle containing the 2-path e_{uvw}, meaning that R' cannot exist (as otherwise, we would not have added R to C_s; recall Figure 6 (a)). Hence, the map π so far remains injective.

Case-b: The triple-list ξ is of the form $\xi = \xi_{wu} = (w, u, \{\cdots\})$. In this case, this quadrangle is covered by the type-2 list $\xi^{(2)}_{wu} \in \widehat{L}$. We handle this in a manner symmetric to (Case-a) and map $\pi(R) = (\xi^{(2)}_{wu}, v)$.

Case-c: The last case is that R is generated by triple-list ξ of the form $\xi_{vz} = (v, z, \{\cdots\})$. In this case, the quadrangle $R = (u, w \mid v, z)$ will be covered by the type-3 list $\xi^{(3)}_{vz} = (v, z, S_1, S_2)$ with $u \in S_1$ and $w \in S_2$; see Figure 7. We now argue that at least one of $(\xi^{(3)}_{vz}, u)$ and $(\xi^{(3)}_{vz}, w)$ has *not* been mapped under π yet.

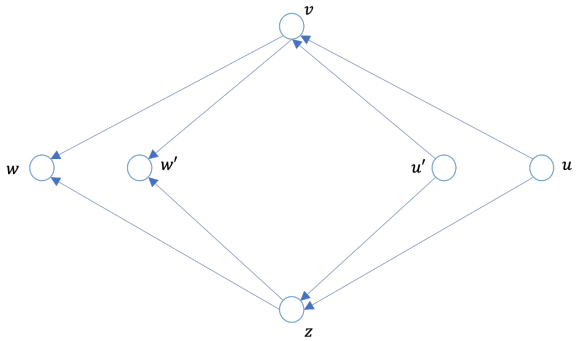


Figure 7 At least one of $(\xi^{(3)}_{vz}, u)$ and $(\xi^{(3)}_{vz}, w)$ has not been mapped yet.

Suppose this is not the case and we already have both $\pi(Q_1) = (\xi^{(3)}_{vz}, u)$ and $\pi(Q_2) = (\xi^{(3)}_{vz}, w)$. Then Q_1 is necessarily of the form $(u, w' \mid v, z)$ and Q_2 is of the form $(u', w \mid v, z)$; and both Q_1 and Q_2 are processed before R. See Figure 7. Furthermore, Q_1 is only added when we process edge e_s. However, in this case, once Q_1 is added, Algorithm 2 will not add further quadrangle containing edges (u, v) and (u, z) (recall the handling of Figure 6 (b)). Hence R cannot be added to C_s in this case.

In other words, it cannot be that both Q_1 and Q_2 already exist, and hence we can set $\pi(R)$ to be one of $(\xi^{(3)}_{vz}, u)$ and $(\xi^{(3)}_{vz}, w)$ that is not yet mapped. Consequently, the map π we construct remains injective.

We process all quadrangles in C in order. The final $\pi : \mathcal{R} \to \mathsf{P}$ is injective, meaning that $|\mathcal{R}| \leq |\mathsf{P}|$ and thus $|\mathcal{R}| = O(\mathsf{a}(G)m)$.

Putting everything together, we have that the total number of boundary quadrangles added to C is bounded by $O(\min\{\mathsf{a}(G)m, \sum_{(u,v)\in E}(d_{in}(u) + d_{out}(v))\})$.

Finally, Algorithm 2 spends $O(m + \sum_{(u,v)\in E}(d_{in}(u) + d_{out}(v)))$ time to handle both cases in Figure 6. The theorem then follows. ◀

Combined with some standard matrix operations, the above theorem gives Theorem 1. The details of the proof are given in the full version of the paper.

▶ Remark 10. We note that neither term in $r = \min\{\mathsf{a}(G)m, \sum_{(u,v)\in E}(d_{in}(u) + d_{out}(v))\}$ always dominates. In particular, it is easy to find examples where one term is significantly smaller (asymptotically) than the other. For example, for any planar graph G, $\mathsf{a}(G)m = O(n)$. However, it is easy to have a planar graph where the second term $\sum_{(u,v)\in E}(d_{in}(u)+d_{out}(v)) = \Omega(n^2)$; see e.g, Figure 2.

On the other hand, it is also easy to have a graph G where $\sum_{(u,v)\in E}(d_{in}(u) + d_{out}(v)) = O(1)$ yet $\mathsf{a}(G)m = \Theta(n^3)$. Indeed, consider the bipartite graph in Figure 3, where for each edge $(u, v) \in E$, $d_{in}(u) + d_{out}(v) = 0$. However, this graph has $\mathsf{a}(G) = \Theta(n)$, $m = \Theta(n^2)$ and thus $\mathsf{a}(G)m = \Theta(n^3)$.

▶ Remark 11. We note that the time complexity of the algorithm proposed by Chowdhury and Mémoli in [5] to compute the $(d-1)$-dimensional persistence path homology takes $O(n^{3+3d})$ time. However, for the case $d = 2$, a more refined analysis shows that in fact, their algorithm takes only $O((\sum_{(u,v)\in E}(d_{in}(u) + d_{out}(v)))mn^2)$ time.

Compared with our algorithm, which takes time $O(rm^{\omega-1})$ with $r = \min\{\mathsf{a}(G)m, \sum_{(u,v)\in E}(d_{in}(u) + d_{out}(v))\}$ and $\omega < 2.373$, observe that our algorithm can be significantly faster (when $\mathsf{a}(G)m$ is much smaller than $\sum_{(u,v)\in E}(d_{in}(u)+d_{out}(v))$. For example, for planar graphs, our algorithm takes $O(n^\omega)$ time, whereas the algorithm of [5] takes $O(n^5)$ time.

Finally, in the full version of the paper, we extend our algorithm to compute the so-called *minimal path homology basis* efficiently, and provide some preliminary experimental results, including showing the efficiency of our algorithm compared to the previous best algorithm over several datasets.

5 Concluding remarks

A natural question is whether it is possible to have a more efficient algorithm for computing (persistent) path homology of higher dimensions improving the work of [5]. Another question is whether we can compute a minimal path homology basis faster improving our current time bound $O(m^\omega n)$ (see the full version of the paper).

References

1 Wei Chen, Yajun Wang, and Siyu Yang. Efficient influence maximization in social networks. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 199–208. ACM, 2009.

2 Ho Yee Cheung, Tsz Chiu Kwok, and Lap Chi Lau. Fast matrix rank algorithms and applications. *Journal of the ACM (JACM)*, 60(5):31, 2013.

3 Norishige Chiba and Takao Nishizeki. Arboricity and subgraph listing algorithms. *SIAM Journal on Computing*, 14(1):210–223, 1985.

4 Samir Chowdhury and Facundo Mémoli. A functorial dowker theorem and persistent homology of asymmetric networks. *Journal of Applied and Computational Topology*, 2(1-2):115–175, 2018.

5 Samir Chowdhury and Facundo Mémoli. Persistent path homology of directed networks. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1152–1169. SIAM, 2018.

6 David Cohen-Steiner, Herbert Edelsbrunner, and Dmitriy Morozov. Vines and vineyards by updating persistence in linear time. In *Proceedings of the twenty-second annual symposium on Computational geometry*, pages 119–126. ACM, 2006.

7 Tamal K Dey, Tianqi Li, and Yusu Wang. Efficient algorithms for computing a minimal homology basis. In *Latin American Symposium on Theoretical Informatics*, pages 376–398, 2018.

8 Pawel Dlotko, Kathryn Hess, Ran Levi, Max Nolte, Michael Reimann, Martina Scolamiero, Katharine Turner, Eilif Muller, and Henry Markram. Topological analysis of the connectome of digital reconstructions of neural microcircuits. *arXiv preprint arXiv:1601.01580*, 2016.

9 Jeff Erickson and Kim Whittlesey. Greedy optimal homotopy and homology generators. In *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1038–1046. Society for Industrial and Applied Mathematics, 2005.

10 Alexander Grigor'yan, Yong Lin, Yuri Muranov, and Shing-Tung Yau. Homologies of path complexes and digraphs. *arXiv preprint arXiv:1207.2834*, 2012.

11 Alexander Grigor'yan, Yong Lin, Yuri Muranov, and Shing-Tung Yau. Homotopy theory for digraphs. *arXiv preprint arXiv:1407.0234*, 2014.

12 Alexander Grigor'yan, Yong Lin, Yuri Muranov, and Shing-Tung Yau. Cohomology of digraphs and (undirected) graphs. *Asian J. Math*, 19(5):887–931, 2015.

13 F. Harary. *Graph Theory*. Addison Wesley series in mathematics. Addison-Wesley, 1971. URL: `https://books.google.com/books?id=q8OWtwEACAAJ`.

14 Paolo Masulli and Alessandro EP Villa. The topology of the directed clique complex as a network invariant. *SpringerPlus*, 5(1):388, 2016.

15 Ron Milo, Shai Shen-Orr, Shalev Itzkovitz, Nadav Kashtan, Dmitri Chklovskii, and Uri Alon. Network motifs: simple building blocks of complex networks. *Science*, 298(5594):824–827, 2002.

16 Lav R Varshney, Beth L Chen, Eric Paniagua, David H Hall, and Dmitri B Chklovskii. Structural properties of the caenorhabditis elegans neuronal network. *PLoS computational biology*, 7(2):e1001066, 2011.

Persistence of the Conley Index in Combinatorial Dynamical Systems

Tamal K. Dey
Department of Computer Science and Engineering, The Ohio State University, Columbus, OH, USA
http://www.cse.ohio-state.edu/~dey.8
dey.8@osu.edu

Marian Mrozek
Division of Computational Mathematics, Faculty of Mathematics and Computer Science, Jagiellonian University, Kraków, Poland
http://ww2.ii.uj.edu.pl/~mrozek
marian.mrozek@uj.edu.pl

Ryan Slechta
Department of Computer Science and Engineering, The Ohio State University, Columbus, OH, USA
http://www.cse.ohio-state.edu/~slechta.3
slechta.3@osu.edu

Abstract

A combinatorial framework for dynamical systems provides an avenue for connecting classical dynamics with data-oriented, algorithmic methods. Combinatorial vector fields introduced by Forman [6, 7] and their recent generalization to multivector fields [15] have provided a starting point for building such a connection. In this work, we strengthen this relationship by placing the Conley index in the persistent homology setting. Conley indices are homological features associated with so-called isolated invariant sets, so a change in the Conley index is a response to perturbation in an underlying multivector field. We show how one can use zigzag persistence to summarize changes to the Conley index, and we develop techniques to capture such changes in the presence of noise. We conclude by developing an algorithm to "track" features in a changing multivector field.

2012 ACM Subject Classification Mathematics of computing → Algebraic topology; Theory of computation → Computational geometry

Keywords and phrases Dynamical systems, combinatorial vector field, multivector, Conley index, persistence

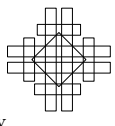
Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.37

Related Version https://arxiv.org/abs/2003.05579

Funding *Tamal K. Dey*: Partially supported by NSF grants CCF-1740761 and DMS-1547357.
Marian Mrozek: Partially supported by the Polish National Science Center under Maestro Grant No. 2014/14/A/ST1/00453.
Ryan Slechta: Supported by NSF grant DMS-1547357.

1 Introduction

At the end of the 19th century, scientists became aware that the very fruitful theory of differential equations cannot provide a description of the asymptotic behavior of solutions in situations when no analytic formulas for solutions are available. This observation affected Poincaré's study on the stability of our celestial system [17] and prompted him to use the methods of dynamical systems theory. The fundamental observation of the theory is that solutions limit in invariant sets. Examples of invariant sets include stationary solutions, periodic orbits, connecting orbits, and many more complicated sets such as chaotic invariant sets discovered in the second half of the 20th century [11]. Today, the Conley index [4, 12] is




among the most fundamental topological descriptors that are used for analyzing invariant sets. The Conley index is defined for *isolated invariant sets* which are maximal invariant sets in some neighborhood. It characterizes whether isolated invariant sets are attracting, repelling or saddle-like. It is used to detect stationary points, periodic solutions and connections between them. Moreover, it provides methods to detect and characterize different chaotic invariant sets. In particular, it was used to prove that the system discovered by Lorenz [11] actually contains a chaotic invariant set [14]. The technique of multivalued maps used in this proof may be adapted to dynamical systems known only from finite samples [13]. Unfortunately, unlike the case when an analytic description of the dynamical system is available, the approach proposed in [13] lacks a validation method. This restricts possible applications in today's data-driven world. In order to use topological persistence as a validation tool, we need an analog of dynamical systems for discrete data. In the case of a dynamical system with continuous time, the idea comes from the fundamental work of R. Forman on discrete Morse theory [7] and combinatorial vector fields [6]. This notion of a combinatorial vector field was recently generalized to that of a combinatorial multivector field [15]. Since then, the Conley index has been constructed in the setting of combinatorial multivector fields [10, 15]. The aim of this research is to incorporate the ideas of topological persistence into the study of the Conley index.

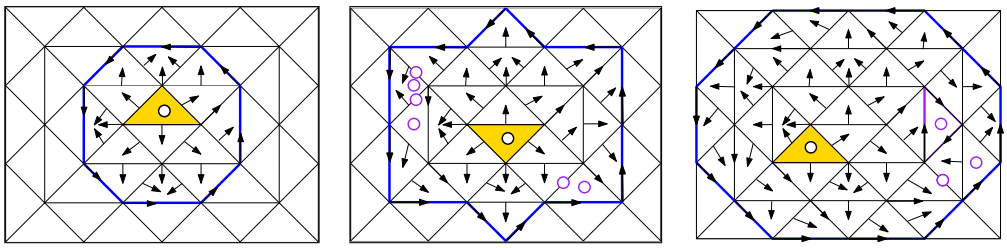


Figure 1 Three multivector fields. In each field, there is a periodic attractor in blue. Such an attractor is an example of an invariant set. The reader will notice that all flows which enter the periodic attractor can ultimately be traced back to simplices marked with circles. These simplices are individually invariant sets, and they correspond to the notion of fixed points. In each multivector field, the gold triangle corresponds to the notion of a repelling fixed point, while the triangles and edges with magenta circles are spurious. Notice that in the third multivector field there is a spurious periodic attractor. However, despite spurious invariant sets, in all three multivector fields the predominant feature is a repelling triangle, from which most emanating flow terminates in the periodic attractor. We aim to develop a quantitative summary of this behavior.

Given a simplicial complex K, Forman defined a combinatorial vector field $\mathcal{V}$ as a partition of K into three sets $L \sqcup U \sqcup C$ where a bijective map $\mu : L \to U$ pairs a p-simplex $\sigma \in L$ with a $(p+1)$-simplex $\tau = \mu(\sigma)$. This pair can be thought of as a vector originating in σ and terminating in τ. Using these vectors, Forman defined a notion of flow for discrete vector fields called a V-path. These paths correspond to the classical notion of integral lines in smooth vector fields. Multivector fields proposed in [15] generalize this concept by allowing a vector to have multiple simplices (dubbed *multivectors*) and more complicated dynamics.

An extension to the idea of the V-path from Forman's theory is called a *solution* for a multivector field $\mathcal{V}$. A solution in $\mathcal{V}$ is a possibly infinite sequence of simplices $\{\sigma_i\}$ such that σ_{i+1} is either a face of σ_i or in the same multivector as σ_i. Solutions may be doubly infinite (or bi-infinite), right infinite, left infinite, or finite. Solutions that are not doubly infinite are *partial*. Bi-infinite solutions correspond to invariant sets in the combinatorial setting.

In Figure 1, one can see a sequence of multivector fields, each of which contains multiple isolated invariant sets. In principle, one would like to choose an isolated invariant set from each of the multivector fields and obtain a description of how the Conley index of these sets changes. Obtaining such a description is highly nontrivial, and it is the main contribution of this paper. Given a sequence of isolated invariant sets, we use the theory of zigzag persistence [3] to extract such a description. In [5], the authors studied the persistence of the Morse decomposition of multivector fields, but this is the first time that the Conley index has been placed in a persistence framework. We also provide schemes to automatically select isolated invariant sets and to limit the effects of noise on the persistence of the Conley index.

2 Preliminaries

Throughout this paper, we will assume that the reader has a basic understanding of both point set and algebraic topology. In particular, we assume that the reader is well-versed in homology. For more information on these topics, we encourage the reader to consult [9, 16].

2.1 Multivectors and Combinatorial Dynamics

In this subsection, we briefly recall the fundamentals of multivector fields as established in [5, 10, 15]. Let K be a finite simplicial complex with face relation $\leq$, that is, $\sigma \leq \sigma'$ if and only if σ is a face of σ'. Equivalently, $\sigma \leq \sigma'$ if $V(\sigma) \subseteq V(\sigma')$, where $V(\sigma)$ denotes the vertex set of σ. For a simplex $\sigma \in K$, we let $\mathsf{cl}(\sigma) := \{\tau \in K \,|\, \tau \leq \sigma\}$ and for a set $A \subseteq K$, we let $\mathsf{cl}(A) := \{\tau \leq \sigma \,|\, \sigma \in A\}$. We say that $A \subseteq K$ is *closed* if $\mathsf{cl}(A) = A$. The reader familiar with the Alexandrov topology [1, Section 1.1] will immediately notice that this notation and terminology is aligned with the topology induced on K by the relation $\leq$.

▶ **Definition 1** (Multivector, Multivector Field). *A subset $A \subseteq K$ is called a* multivector *if for all $\sigma, \sigma' \in A$, $\tau \in K$ satisfying $\sigma \leq \tau \leq \sigma'$, we have that $\tau \in A$. A* multivector field *over K is a partition of K into multivectors.*

Every multivector is said to be either regular or critical. To define critical multivectors, we define the *mouth* of a set as $\mathsf{m}(A) := \mathsf{cl}(A) \setminus A$. The multivector V is critical if the relative homology $H_p(\mathsf{cl}(V), \mathsf{m}(V)) \neq 0$ in some dimension p. Otherwise, V is regular. Simplices in critical multivectors are marked with circles in Figures 1, 2, and 3. Throughout this paper, all references to homology are references to simplicial homology. Note that $H_p(\mathsf{cl}(V), \mathsf{m}(V))$ is thus well defined because $\mathsf{m}(S) \subseteq \mathsf{cl}(S) \subseteq K$. Intuitively, a multivector V is regular if $\mathsf{cl}(V)$ can be collapsed onto $\mathsf{m}(V)$. In Figure 2, the red triangle with its two edges is a critical multivector V because $H_1(\mathsf{cl}(V), \mathsf{m}(V))$ is nontrivial. Similarly, the gold colored triangles (denoted τ) and the green edge (denoted σ) are critical because $H_2(\mathsf{cl}(\tau), \partial\tau)$ and $H_1(\mathsf{cl}(\sigma), \partial\sigma)$ are nontrivial, where we use $\partial\sigma$ to denote the boundary of a simplex σ.

A multivector field over K induces a notion of dynamics. For $\sigma \in K$, we denote the multivector containing σ as $[\sigma]$. If the multivector field $\mathcal{V}$ is not clear from context, we will use the notation $[\sigma]_{\mathcal{V}}$. We now use a multivector field $\mathcal{V}$ on K to define a multivalued map $F_{\mathcal{V}} : K \multimap K$. In particular, we let $F_{\mathcal{V}}(\sigma) := \mathsf{cl}(\sigma) \cup [\sigma]$. Such a multivalued map induces a notion of flow on K. In the interest of brevity, for $a, b \in \mathbb{Z}$, we set $\mathbb{Z}_{[a,b]} = [a, b] \cap \mathbb{Z}$ and define $\mathbb{Z}_{(a,b]}$, $\mathbb{Z}_{[a,b)}$, $\mathbb{Z}_{(a,b)}$ as expected. A *path* from σ to σ' is a map $\rho : \mathbb{Z}_{[a,b]} \to K$, where $\rho(a) = \sigma$, $\rho(b) = \sigma'$, and for all $i \in \mathbb{Z}_{(a,b]}$, we have that $\rho(i) \in F_{\mathcal{V}}(\rho(i-1))$. Similarly, a *solution* to a multivector field over K is a map $\rho : \mathbb{Z} \to K$ where $\rho(i) \in F_{\mathcal{V}}(\rho(i-1))$.

▶ **Definition 2** (Essential Solution). *A solution $\rho : \mathbb{Z} \to K$ is an* essential solution *of multivector field $\mathcal{V}$ on K if for each $i \in \mathbb{Z}$ where $[\rho(i)]$ is regular, there exists an $i^-, i^+ \in \mathbb{Z}$ where $i^- < i < i^+$ and $[\rho(i^-)] \neq [\rho(i)] \neq [\rho(i^+)]$.*

For a set $A \subseteq K$, let $\mathsf{eSol}(A)$ denote the set of essential solutions ρ such that $\rho(\mathbb{Z}) \subseteq A$. If the relevant multivector field is not clear from context, we use the notation $\mathsf{eSol}_{\mathcal{V}}(A)$. We define the *invariant part* of A as $\mathsf{Inv}(A) = \{\sigma \in A \mid \exists \rho \in \mathsf{eSol}(A), \rho(0) = \sigma\}$. We say that A is *invariant* or an *invariant set* if $\mathsf{Inv}(A) = A$. If the multivector field is not clear from context, we use the notation $\mathsf{Inv}_{\mathcal{V}}(A)$.

Solutions and invariant sets are defined in accordance to their counterparts in the classical setting. For more information on the classical counterparts of these concepts, see [2]. As in the classical setting, we have a notion of isolation for combinatorial invariant sets.

▶ **Definition 3** (Isolated Invariant Set, Isolating Neighborhood). *An invariant set $A \subseteq N$, N closed, is* isolated by *N if all paths $\rho : \mathbb{Z}_{[a,b]} \to N$ for which $\rho(a), \rho(b) \in A$ satisfy $\rho(\mathbb{Z}_{[a,b]}) \subseteq A$. The closed set N is said to be an* isolating neighborhood *for S.*

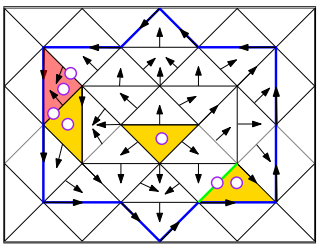


Figure 2 A multivector field with several invariant sets, isolated by the entire rectangle, N. Note that for each colored triangle σ, since $[\sigma]$ is critical, there is an essential solution $\rho : \mathbb{Z} \to N$ where $\rho(i) = \sigma$ for all i. Likewise for the green edge. Since the periodic attractor is composed of regular vectors, there is no such essential solution for any given simplex in the periodic attractor. However, by following the arrows in the periodic attractor we still get an essential solution.

2.2 Conley Indices

The Conley index of an isolated invariant set is a topological invariant used to characterize features of dynamical systems [4, 12]. In both the classical and the combinatorial settings, the Conley index is determined by index pairs.

▶ **Definition 4.** *Let S be an isolated invariant set. The pair of closed sets (P, E) subject to $E \subseteq P \subseteq K$ is an* index pair *for S if all of the following hold:*

1. $F_{\mathcal{V}}(E) \cap P \subseteq E$
2. $F_{\mathcal{V}}(P \setminus E) \subseteq P$
3. $S = \mathsf{Inv}(P \setminus E)$

In addition, an index pair is said to be a *saturated index pair* if $S = P \setminus E$. In Figure 3, the gold, critical triangle σ is an isolated invariant set. The reader can easily verify that $(\mathsf{cl}(\sigma), \mathsf{cl}(\sigma) \setminus \{\sigma\})$ is an index pair for σ. In fact, this technique is a canonical way of picking an index pair for an isolated invariant set. This is formalized in the following proposition.

▶ **Proposition 5** ([10, Proposition 4.3]). *Let S be an isolated invariant set. The pair $(\mathsf{cl}(S), \mathsf{m}(S))$ is a saturated index pair for S.*

However, there are several other natural ways to find index pairs. Figure 3 shows another index pair for the same gold triangle σ. By letting $P := \mathsf{cl}(\sigma) \cup S_P$ and $E := \mathsf{m}(\sigma) \cup S_E$, where S_P and S_E are the set of simplices reachable from paths originating in $\mathsf{cl}(S)$, $\mathsf{m}(S)$ respectively, we obtain a much larger index pair. In Figure 3, P is the set of all colored simplices, while E is the set of all colored simplices which are not gold.

In principle, it is important that the Conley index be independent of the choice of index pair. Fortunately, it is also known that the relative homology given by an index pair for an isolated invariant set S is independent of the choice of index pair.

▶ **Theorem 6** ([10, Theorem 4.15]). *Let (P_1, E_1) and (P_2, E_2) be index pairs for the isolated invariant set S. Then $H_p(P_1, E_1) \cong H_p(P_2, E_2)$ for all p.*

The *Conley Index* of an isolated invariant set S in dimension p is then given by the relative homology group $H_p(P, E)$ for any index pair of S denoted (P, E).

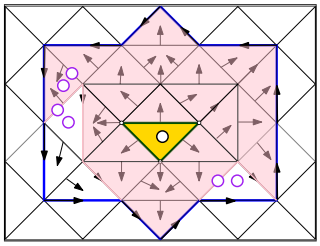


Figure 3 Two index pairs for the gold triangle, denoted σ. The first is given by $(\mathsf{cl}(\sigma), \mathsf{m}(\sigma))$ where $\mathsf{m}(\sigma)$ is in green and $\mathsf{cl}(\sigma) \setminus \mathsf{m}(\sigma)$ is exactly the gold triangle. The second index pair is $(\mathsf{pf}(\mathsf{cl}(\sigma)), \mathsf{pf}(\mathsf{m}(\sigma)))$, where $\mathsf{pf}(\mathsf{m}(\sigma))$ consists of those simplices which are colored pink, green, and blue, while $\mathsf{pf}(\mathsf{cl}(\sigma))$ consists of all colored simplices. Note that the second index pair is also an index pair in N, where N is taken to be the entire rectangle.

3 Conley Index Persistence

We move to establishing the foundations for persistence of the Conley Index. Given a sequence of multivector fields $\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_n$ on a simplicial complex K, one may want to quantify the changing behavior of the vector fields. One such approach is to compute a sequence of isolated invariant sets $S_1, S_2, \ldots, S_n$ under each multivector field, and then to compute an index pair for each isolated invariant set. By Proposition 5, a canonical way to do this is to take the closure and mouth of each isolated invariant set to obtain a sequence of index pairs $(\mathsf{cl}(S_1), \mathsf{m}(S_1)), (\mathsf{cl}(S_2), \mathsf{m}(S_2)), \ldots, (\mathsf{cl}(S_n), \mathsf{m}(S_n))$. A first idea is to take the element-wise intersection of consecutive index pairs, which results in the zigzag filtration:

$$(\mathsf{cl}(S_1), \mathsf{m}(S_1)) \supseteq (\mathsf{cl}(S_1) \cap \mathsf{cl}(S_2), \mathsf{m}(S_1) \cap \mathsf{m}(S_2)) \subseteq (\mathsf{cl}(S_2), \mathsf{m}(S_2)) \cdots (\mathsf{cl}(S_n), \mathsf{m}(S_n))$$

Taking the relative homology groups of the pairs in the zigzag sequence, we obtain a zigzag persistence module. We can extract a barcode corresponding to a decomposition of this module:

$$H_p(\mathsf{cl}(S_1), \mathsf{m}(S_1)) \leftarrow H_p(\mathsf{cl}(S_1) \cap \mathsf{cl}(S_2), \mathsf{m}(S_1) \cap \mathsf{m}(S_2)) \rightarrow H_p(\mathsf{cl}(S_2), \mathsf{m}(S_2)) \leftarrow \cdots \rightarrow H_p(\mathsf{cl}(S_n), \mathsf{m}(S_n)).$$

However, the chance that this approach works in practice is low. In general, two isolated invariant sets S_1, S_2 need not overlap, and hence their corresponding index pairs need not intersect. For example, if one were to take the blue periodic solutions in the multivector

fields in Figure 1 to be S_1, S_2, S_3, by using Proposition 5 one gets the index pairs $(S_1, \emptyset)$, $(S_2, \emptyset)$, and $(S_3, \emptyset)$ (since $\mathsf{cl}(S_i) = S_i$). Note that in such a case, the intermediate pairs are $(S_1 \cap S_2, \emptyset)$ and $(S_2 \cap S_3, \emptyset)$. But $S_1 \cap S_2$ and $S_2 \cap S_3$ intersect only at vertices, so none of the 1-cycles persist beyond their multivector field. This is problematic in computing the persistence, because intuitively there should be an H_1 generator that persists through all three multivector fields. To increase the likelihood that two index pairs intersect, we consider a special type of index pair called an *index pair in N*.

▶ **Definition 7.** *Let S be an invariant set isolated by N under $\mathcal{V}$. The pair of closed sets (P, E) satisfying $E \subseteq P \subseteq N$ is an* index pair for S in N *if all of the following conditions are met:*

1. $F_{\mathcal{V}}(P) \cap N \subseteq P$
2. $F_{\mathcal{V}}(E) \cap N \subseteq E$
3. $F_{\mathcal{V}}(P \setminus E) \subseteq N$, *and*
4. $S = \mathsf{Inv}(P \setminus E)$.

As is expected, such index pairs in N are index pairs.

▶ **Theorem 8.** *Let (P, E) be an index pair in N for S. The pair (P, E) is an index pair for S in the sense of Definition 4.*

Proof. Note that by condition three of Definition 7, if $\sigma \in P \setminus E$, then $F_{\mathcal{V}}(\sigma) \subseteq N$. Condition one of Definition 7 implies that $F_{\mathcal{V}}(\sigma) \cap N = F_{\mathcal{V}}(\sigma) \subseteq P$, which is condition two of Definition 4. Likewise, by condition two of Definition 7, if $\sigma \in E$, then $F_{\mathcal{V}}(\sigma) \cap N \subseteq E$. Note that $P \subseteq N$, so it follows that $F_{\mathcal{V}}(\sigma) \cap P \subseteq F_{\mathcal{V}}(\sigma) \cap N \subseteq E$, which is condition one of Definition 4. Finally, condition four of Definition 7 directly implies condition three of Definition 4. ◀

An additional advantage to considering index pairs in N is that the intersection of index pairs in N is an index pair in N. In general, $(\mathsf{cl}(S_1) \cap \mathsf{cl}(S_2), \mathsf{m}(S_1) \cap \mathsf{m}(S_2))$ is not an index pair. However, for index pairs in N, we get the next two results which involve the notion of a new multivector field obtained by intersection. Given two multivector fields $\mathcal{V}_1$, $\mathcal{V}_2$, we define $\mathcal{V}_1 \overline{\cap} \mathcal{V}_2 := \{V_1 \cap V_2 \mid V_1 \in \mathcal{V}_1,\ V_2 \in \mathcal{V}_2\}$.

▶ **Theorem 9.** *Let $(P_1, E_1), (P_2, E_2)$ be index pairs in N for S_1, S_2 under $\mathcal{V}_1, \mathcal{V}_2$. The set $\mathsf{Inv}((P_1 \cap P_2) \setminus (E_1 \cap E_2))$ is isolated by N under $\mathcal{V}_1 \overline{\cap} \mathcal{V}_2$.*

Proof. To contradict, we assume that there exists a path $\rho : \mathbb{Z}_{[a,b]} \to N$ under $\mathcal{V}_1 \overline{\cap} \mathcal{V}_2$ where $\rho(a), \rho(b) \in \mathsf{Inv}((P_1 \cap P_2) \setminus (E_1 \cap E_2))$ and there exists some $i \in (a, b) \cap \mathbb{Z}$ where $\rho(i) \notin \mathsf{Inv}((P_1 \cap P_2) \setminus (E_1 \cap E_2))$. Note that by the the definition of an index pair, $F_{\mathcal{V}}(P) \cap N \subseteq P$. Hence, it follows by an easy induction argument that since $F_{\mathcal{V}_1 \overline{\cap} \mathcal{V}_2}(\sigma) \subseteq F_{\mathcal{V}_1}(\sigma), F_{\mathcal{V}_2}(\sigma)$, we have that $\rho(\mathbb{Z}_{[a,b]}) \subseteq P_1, P_2$. This directly implies that $\rho(\mathbb{Z}_{[a,b]}) \subseteq P_1 \cap P_2$. In addition, it is easy to see that ρ can be extended to an essential solution in $P_1 \cap P_2$, which we denote $\rho' : \mathbb{Z} \to N$, by some simple surgery on essential solutions. This is because there must be essential solutions $\rho_1, \rho_2 : \mathbb{Z} \to (P_1 \cap P_2) \setminus (E_1 \cap E_2)$ where $\rho_1(a) = \rho(a)$ and $\rho_2(b) = \rho(b)$, as $\rho(a)$ and $\rho(b)$ are both in essential solutions. Hence, $\rho'(x) = \rho_1(x)$ if $x \leq a$, $\rho'(x) = \rho(x)$ if $a \leq x \leq b$, and $\rho'(x) = \rho_2(x)$ if $b \leq x$. Since ρ' is an essential solution, we have that $\rho(\mathbb{Z}_{[a,b]}) \subseteq \mathsf{Inv}(P_1 \cap P_2)$, but also that $\rho(\mathbb{Z}_{[a,b]}) \not\subseteq \mathsf{Inv}((P_1 \cap P_2) \setminus (E_1 \cap E_2))$. Therefore, we must have that $\rho(i) \in E_1 \cap E_2$. But by the same reasoning as before, it follows that $\rho(\mathbb{Z}_{[i,b]}) \subseteq E_1 \cap E_2$. Hence, $b \notin (P_1 \cap P_2) \setminus (E_1 \cap E_2)$, a contradiction. ◀

▶ **Theorem 10.** *Let (P_1, E_1) and (P_2, E_2) be index pairs in N under $\mathcal{V}_1, \mathcal{V}_2$. The tuple $(P_1 \cap P_2, E_1 \cap E_2)$ is an index pair for $\mathsf{Inv}((P_1 \cap P_2) \setminus (E_1 \cap E_2))$ in N under $\mathcal{V}_1 \overline{\cap} \mathcal{V}_2$.*

Proof. We proceed by using the conditions in Definition 7 to show that $(P_1 \cap P_2, E_1 \cap E_2)$ is an index pair in N. Note that $F_{\mathcal{V}_1 \overline{\cap} \mathcal{V}_2}(P_1 \cap P_2) \cap N \subseteq F_{\mathcal{V}_1}(P_1) \cap F_{\mathcal{V}_2}(P_2) \cap N$, which is immediate by the definition of F and considering $\mathcal{V}_1 \overline{\cap} \mathcal{V}_2$. Note that since (P_1, E_1) and (P_2, E_2) are index pairs in N, we know from Definition 7 that $F_{\mathcal{V}_1}(P_1) \cap N \subseteq P_1$ and $F_{\mathcal{V}_2}(P_2) \cap N \subseteq P_2$. Therefore $F_{\mathcal{V}_1 \overline{\cap} \mathcal{V}_2}(P_1 \cap P_2) \cap N \subseteq P_1 \cap P_2$. This implies the first condition in Definition 7. This argument also implies the second condition by replacing P with E.

Now, we aim to show that $(P_1 \cap P_2, E_1 \cap E_2)$ satisfies condition three in Definition 7. Consider $\sigma \in (P_1 \cap P_2) \setminus (E_1 \cap E_2)$. Without loss of generality, we assume $\sigma \notin E_1$. Therefore, $\sigma \in P_1 \setminus E_1$, so $F_{\mathcal{V}_1}(\sigma) \subseteq N$ by the definition of an index pair in N. Hence, since $F_{\mathcal{V}_1 \overline{\cap} \mathcal{V}_2}(\sigma) \subseteq F_{\mathcal{V}_1}(\sigma)$, condition three is satisfied.

Finally, note that $\mathsf{Inv}((P_1 \cap P_2) \setminus (E_1 \cap E_2))$ is obviously equal to $\mathsf{Inv}((P_1 \cap P_2) \setminus (E_1 \cap E_2))$, so condition four holds as well. ◀

Hence, if (P_i, E_i) are index pairs in N, these theorems gives a meaningful notion of persistence of Conley index through the decomposition of the following zigzag persistence module:

$$H_p(P_1, E_1) \leftarrow H_p(P_1 \cap P_2, E_1 \cap E_2) \rightarrow H_p(P_2, E_2) \leftarrow \cdots \rightarrow H_p(P_n, E_n). \tag{1}$$

Because of the previous two theorems, when one decomposes the above zigzag module, one is actually capturing a changing Conley index. This contrasts the case where one only considers index pairs of the form $(\mathsf{cl}(S_i), \mathsf{m}(S_i))$, because $(\mathsf{cl}(S_i) \cap \mathsf{cl}(S_{i+1}), \mathsf{m}(S_i) \cap \mathsf{m}(S_{i+1}))$ need not be an index pair for any invariant set.

As has been established, the pair $(\mathsf{cl}(S), \mathsf{m}(S))$ is an index pair, but it need not be an index pair in N. We introduce a canonical approach to transform $(\mathsf{cl}(S), \mathsf{m}(S))$ to an index pair in N by using the *push forward*.

▶ **Definition 11.** *The* push forward $\mathit{pf}(S)$ *of a set S in N, N closed, is the set of all simplices in S together with those $\sigma \in N$ such that there exists a path $\rho : \mathbb{Z}_{[a,b]} \to N$ where $\rho(a) \in S$ and $\rho(b) = \sigma$.*

If N is not clear from context, we use the notation $\mathsf{pf}_N(S)$. The next series of results imply that an index pair in N can be obtained by taking the push forward of $(\mathsf{cl}(S), \mathsf{m}(S))$.

▶ **Proposition 12.** *If $S \subseteq K$ is an isolated invariant set with isolating neighborhood N under $\mathcal{V}$, then $\mathit{pf}(\mathit{m}(S)) \cap \mathit{cl}(S) = \mathit{m}(S)$.*

▶ **Proposition 13.** *If $S \subseteq K$ is an isolated invariant set with isolating neighborhood N under $\mathcal{V}$, then $\mathit{pf}(\mathit{m}(S)) \cup \mathit{cl}(S) = \mathit{pf}(\mathit{cl}(S))$.*

▶ **Proposition 14.** *If $S \subseteq K$ is an isolated invariant set with isolating neighborhood N, then $\mathit{pf}(\mathit{cl}(S)) \setminus \mathit{pf}(\mathit{m}(S)) = \mathit{cl}(S) \setminus \mathit{m}(S) = S$.*

Proofs for Propositions 12, 13, and 14 are included in the full version. Crucially, from these propositions we get the following.

▶ **Theorem 15.** *If S is an isolated invariant set then $(\mathit{pf}(\mathit{cl}(S)), \mathit{pf}(\mathit{m}(S)))$ is an index pair in N for S.*

Proof. First, we note that since the index pair $(\mathsf{cl}(S), \mathsf{m}(S))$ is saturated, it follows that $S = \mathsf{Inv}(\mathsf{cl}(S) \setminus \mathsf{m}(S)) = \mathsf{cl}(S) \setminus \mathsf{m}(S)$. But since by Proposition 14 $\mathsf{cl}(S) \setminus \mathsf{m}(S) = \mathsf{pf}(\mathsf{cl}(S)) \setminus \mathsf{pf}(\mathsf{m}(S))$, it follows that $S = \mathsf{pf}(\mathsf{cl}(S)) \setminus \mathsf{pf}(\mathsf{m}(S)) = \mathsf{Inv}(\mathsf{pf}(\mathsf{cl}(S)) \setminus \mathsf{pf}(\mathsf{m}(S)))$, which satisfies condition four of being an index pair in N.

We show that $F_{\mathcal{V}}(\mathsf{pf}(\mathsf{cl}(S))) \cap N \subseteq \mathsf{pf}(\mathsf{cl}(S))$. Let $x \in \mathsf{pf}(\mathsf{cl}(S))$, and assume that $y \in F_{\mathcal{V}}(x) \cap N$. There must be a path $\rho : \mathbb{Z}_{[a,b]} \to N$ where $\rho(a) \in \mathsf{cl}(S)$ and $\rho(b) = x$, by the definition of the push forward. Thus, we can construct an analogous path $\rho' : \mathbb{Z}_{[a,b+1]} \to N$ where $\rho'(i) = \rho(i)$ for $i \in \mathbb{Z}_{[a,b]}$ and $\rho'(b+1) = y$. Hence, $y \in \mathsf{pf}(\mathsf{cl}(S))$ by definition. Identical reasoning can be used to show that $F_{\mathcal{V}}(\mathsf{pf}(\mathsf{m}(S))) \cap N \subseteq \mathsf{pf}(\mathsf{m}(S))$, so $(\mathsf{pf}(\mathsf{cl}(S)), \mathsf{pf}(\mathsf{m}(S)))$ also meets the first two conditions required to be an index pair.

Finally, we show that $F_{\mathcal{V}}(\mathsf{pf}(\mathsf{cl}(S)) \setminus \mathsf{pf}(\mathsf{m}(S))) \subseteq N$. By Proposition 14, this is equivalent to showing that $F_{\mathcal{V}}(\mathsf{cl}(S) \setminus \mathsf{m}(S)) \subseteq N$. Since $(\mathsf{cl}(S), \mathsf{m}(S))$ is an index pair for S, it follows that $F_{\mathcal{V}}(\mathsf{cl}(S) \setminus \mathsf{m}(S)) \subseteq \mathsf{cl}(S)$. Note that since $N \supseteq S$ is closed, it follows that $\mathsf{cl}(S) \subseteq N$. Hence, $F_{\mathcal{V}}(\mathsf{pf}(\mathsf{cl}(S)) \setminus \mathsf{pf}(\mathsf{m}(S))) \subseteq N$, and all conditions for an index pair in N are met. ◀

An example of an index pair induced by the push forward can be seen in Figure 3. Hence, instead of considering a zigzag filtration given by a sequence of index pairs $(\mathsf{cl}(S_1), \mathsf{m}(S_1))$, $(\mathsf{cl}(S_2), \mathsf{m}(S_2))$, …, $(\mathsf{cl}(S_n), \mathsf{m}(S_n))$, a canonical choice is to instead consider the zigzag filtration given by the the sequence of index pairs $(\mathsf{pf}(\mathsf{cl}(S_1)), \mathsf{pf}(\mathsf{m}(S_1)))$, $(\mathsf{pf}(\mathsf{cl}(S_2)), \mathsf{pf}(\mathsf{m}(S_2)))$, …, $(\mathsf{pf}(\mathsf{cl}(S_n)), \mathsf{pf}(\mathsf{m}(S_n)))$.

Choosing S_i is highly application specific, so in our implementation we choose $S_i := \mathsf{Inv}_{\mathcal{V}_i}(N)$. This decision together with the previous theorems gives Algorithm 1 for computing the persistence of the Conley Index. Index pairs and barcodes computed by Algorithm 1 can be seen in Figure 4.

Algorithm 1 Scheme for computing the persistence of the Conley Index, fixed N.

Input: Sequence of multivector fields $\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_n$, closed set $N \subseteq K$.
Output: Barcodes corresponding to persistence of Conley Index
$i \leftarrow 1$
while $i <= n$ **do**
  $S_i \leftarrow \mathsf{Inv}_{\mathcal{V}_i}(N)$
  $(P_i, E_i) \leftarrow (\mathsf{pf}(\mathsf{cl}(S_i)), \mathsf{pf}(\mathsf{m}(S_i)))$
  $i \leftarrow i + 1$
end
return *zigzagPers*$((P_1, E_1) \supseteq (P_1 \cap P_2, E_1 \cap E_2) \subseteq (P_2, E_2) \supseteq \ldots \subseteq (P_n, E_n))$

3.1 Noise-Resilient Index Pairs

The strategy given for producing index pairs in N produces saturated index pairs. Equivalently, the cardinality of $P \setminus E$ is minimized. This is problematic in the presence of noise, where if $\mathcal{V}_2$ is a slight perturbation of $\mathcal{V}_1$ we frequently have that $\mathsf{Inv}_{\mathcal{V}_1}(N) \neq \mathsf{Inv}_{\mathcal{V}_2}(N)$. This gives a perturbation in our generated index pairs and in particular a perturbation in $P \setminus E$. As the Conley Index is obtained by taking relative homology, taking the intersection of index pairs (P_1, E_2) and (P_2, E_2) where $P_i \setminus E_i = \mathsf{Inv}_{\mathcal{V}_i}(N)$ can result in a "breaking" of bars in the barcode. An example can be seen in Figure 5, where because of noise, the two $P \setminus E$ do not overlap, and hence a 2-dimensional homology class which intuitively should persist throughout the interval does not. In Figure 5, the Conley indices of the invariant sets consisting of the singleton critical triangles in $\mathcal{V}_1$ and $\mathcal{V}_2$ (the left and right multivector fields) have rank 1 in dimension 2 because the homology group H_2 of P (which is the entire complex in both cases) relative to E (which is all pink simplices) has rank 1. However, the generators for $H_2(P_1, E_1)$ and $H_2(P_2, E_2)$ are both in the intersection field $\mathcal{V}_1 \overline{\cap} \mathcal{V}_2$. Hence, rather than

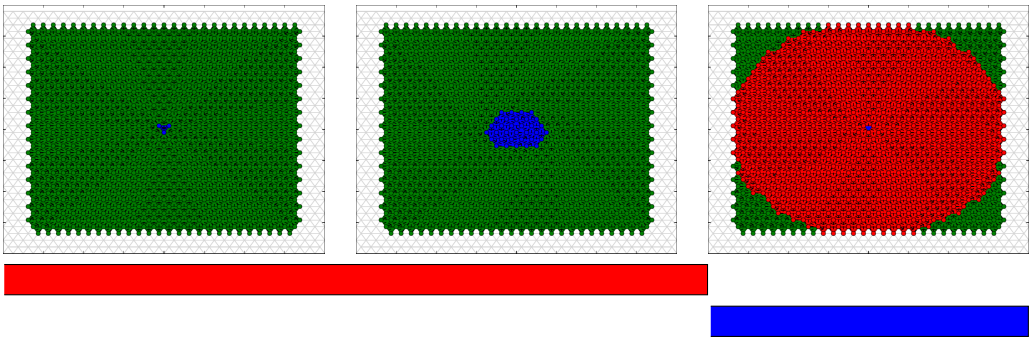


Figure 4 Examples of index pairs computed by using the push forward on multivector fields induced by a differential equation. A sequence of multivector fields was generated from a λ-parametrized differential equation undergoing supercritical Hopf bifurcation [8, Section 11.2]. The consecutive images (from left to right) present a selection from this sequence: the case when $\lambda < 0$ and there is only an attracting fixed point inside N; the case when $\lambda > 0$ is small and N contains a repelling fixed point, a small attracting periodic trajectory and all connecting trajectories; the case when $\lambda > 0$ is large and the periodic trajectory is no longer contained in N. In all three images, we depict N in green, E in red, and $P \setminus E$ in blue. Note that in the leftmost image, the only invariant set is a triangle which represents an attracting fixed point. For this invariant set in this N, the only relative homology group which is nontrivial is $H_0(P, E)$, which has a single homology generator. In the middle image, the invariant sets represent a repelling fixed point, a periodic attractor, and heteroclinic orbits which connect the repelling fixed point with the periodic attractor. Note that the relative homology has not changed from the leftmost case, so the only nontrivial homology group is $H_0(P, E)$. In the rightmost image, the periodic attractor is no longer entirely contained within N, so the only invariant set corresponds to a repelling fixed point. Here, the only nontrivial homology group is $H_2(P, E)$, which has one generator, so the Conley index has changed. Algorithm 1 captures this change. The persistence barcode output by Algorithm 1 is below index pairs, where a H_0 generator (red bar) lasts until the periodic trajectory leaves N, at which point it is replaced by an H_2 generator (blue bar).

one generator persisting through all three multivector fields, we get two bars that overlap at the intersection field. The difficulty is rooted in the fact that the sets $W_1 = P_1 \setminus E_1$, $W_2 = P_2 \setminus E_2$, and $W_{12} = (P_1 \cap P_2) \setminus (E_1 \cap E_2)$ do not have a common intersection.

To address this problem, we propose an algorithm to expand the size of $P \setminus E$. It is important to note that a balance is needed to ensure a large E as well as a large $P \setminus E$. If E_1 and E_2 are too small, then it is easy to see that E_1 and E_2 may not intersect as expected even though consecutive vector fields are very similar. The following proposition is very useful for computing a balanced index pair.

▶ **Proposition 16.** *Let (P, E) be an index pair for S in N under $\mathcal{V}$. If $V \subseteq E$ is a regular multivector where $E' := E \setminus V$ is closed, then (P, E') is an index pair for S in N.*

We include the proof for Proposition 16 in the full version. Figure 6 illustrates how enlarging $P \setminus E$ by removing regular vectors as Proposition 16 suggests can help mitigate the effects of noise on computing Conley index persistence. Contrast this example with the example in Figure 5. Denoting $W_i = P_i \setminus E_i$ for $i = 1, 2$ and $W_{12} = (P_1 \cap P_2) \setminus (E_1 \cap E_2)$ in both figures, we see that $W_1 \cap W_{12} \cap W_2$ is empty in Figure 5 while in Figure 6 it consists of three critical simplices each marked with a circle. Hence, in Figure 6 a single generator persists throughout the interval, unlike in Figure 5.

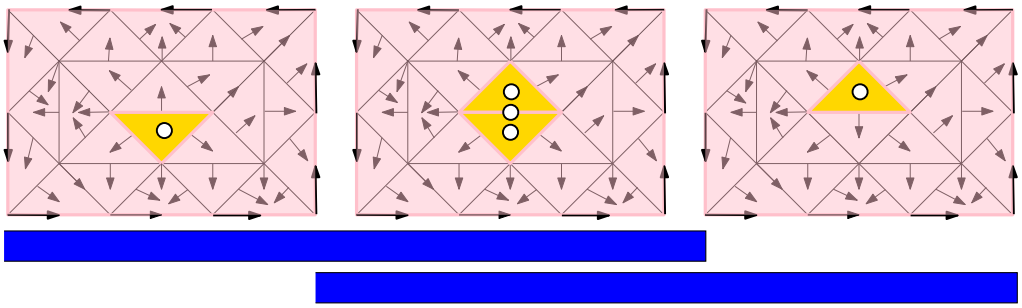


Figure 5 Infeasibilty of the index pair $(\mathsf{pf}(\mathsf{cl}(S)), \mathsf{pf}(\mathsf{m}(S)))$: The sets $E = \mathsf{pf}(\mathsf{m}(S))$ are colored pink in all three images, while the invariant sets which equal $P \setminus E$ are golden in all three images. (left) $\mathcal{V}_1 : P_1 \setminus E_1$ consists of a single golden triangle; (right) $\mathcal{V}_2$: $P_2 \setminus E_2$ consists of the single golden triangle; (middle) $(P_1 \cap P_2) \setminus (E_1 \cap E_2)$ consists of two golden triangles (excluding the edge between them) in the intersection field $\mathcal{V}_1 \overline{\cap} \mathcal{V}_2$. The barcode for index pairs is depicted by two blue bars, each of which represents a 2-dimensional homology generator. Ideally, these would be a single bar.

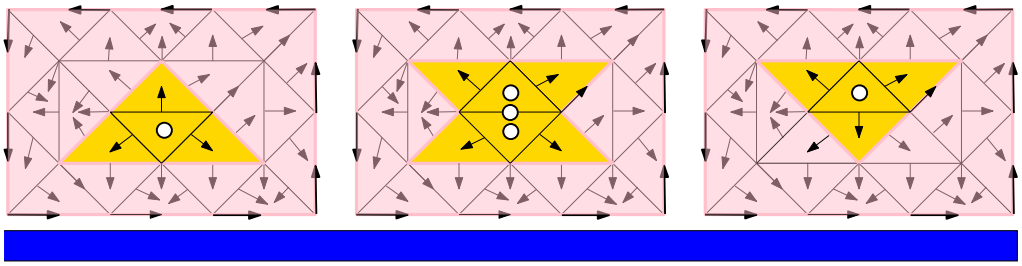


Figure 6 Enlarging $P \setminus E$ which is gold in all three pictures while E is colored pink. (left) $\mathcal{V}_1$; (right) $\mathcal{V}_2$; (middle) $\mathcal{V}_1 \overline{\cap} \mathcal{V}_2$. Note that there is one bar in the barcode, in contrast with Figure 5.

3.2 Computing a Noise-Resilient Index Pair

We give a method for computing a noise-resilient index pair by using techniques demonstrated in the previous subsection. Note that by Theorem 15, we have that $(\mathsf{pf}(\mathsf{cl}(S)), \mathsf{pf}(\mathsf{m}(S)))$ is an index pair for invariant set S in N. Hence, we adopt the strategy of taking $P = \mathsf{pf}(\mathsf{cl}(S))$ and $E = \mathsf{pf}(\mathsf{m}(S))$, and we aim to find some collection $R \subseteq E$ so that $(P, E \setminus R)$ remains an index pair in N. Finding an appropriate R is a difficult balancing act: one wants to find an R so that $P \setminus (E \setminus R)$ is sufficiently large, so as to capture perturbations in the isolated invariant set as described in the previous section, but not so large that E is small and perturbations in E are not captured. If R is chosen to be as large as possible, then a small shift in E may results in $(E \setminus R) \cap (E' \setminus R')$ having a different topology than E or E' leading to a "breaking" of barcodes analogous to the case described in the previous section.

Before we give an algorithm for outputting such an R, we first define a δ-collar.

▶ **Definition 17.** *We define the* δ-collar *of an invariant set* $S \subseteq K$ *recursively:*

1. *The* 0*-collar of* S *is* $\mathsf{cl}(S)$.
2. *For* $\delta > 0$*, the* δ*-collar of* S *is the set of simplices* σ *in the* $(\delta - 1)$*-collar of* S *together with those simplices* τ *where* τ *is a face of some* σ *with a face* τ' *in the* $(\delta - 1)$*-collar of* S.

For an isolated invariant set S, we will let $C_\delta(S)$ denote the δ-collar of S. Together with Proposition 16, δ-collars give a natural algorithm for finding an R to enlarge $P \setminus E$.

In particular, we use Algorithm 2 for this purpose.

Algorithm 2 `findR`$(S, P, E, \mathcal{V}, \delta)$.

Input: Isolated invariant set S with respect to $\mathcal{V}$ contained in some closed set N, Index pair (P, E) in N with respect to $\mathcal{V}$, $\delta \in \mathbb{Z}$
Output: List of simplices R such that $(P, E \setminus R)$ is an index pair for S in N.

$R \leftarrow$ `new ; set()`
$vecSet \leftarrow \{[\sigma] \in \mathcal{V} \mid [\sigma] \subseteq E \cap C_\delta(S) \wedge [\sigma] \cap \partial(E) \neq \emptyset \wedge [\sigma] \cap \partial(P) = \emptyset\}$
$vec \leftarrow$ `new ; queue()`
`appendAll`$(vec, vecSet)$
while *size*$(vec) > 0$ **do**
  $[\sigma] \leftarrow$ `pop`(vec)
  if *isClosed*$((E \setminus R) \setminus [\sigma])$ *and* $[\sigma] \subseteq E \setminus R$ *and* *isRegular*$([\sigma])$ **then**
    $R \leftarrow R \cup [\sigma]$
    $mouthVecs \leftarrow \{[\tau] \mid \tau \in \mathsf{m}([\sigma]) \wedge [\tau] \subseteq C_\delta(S)\}$
    `appendAll`$(vec, mouthVecs)$
  end
end
return R

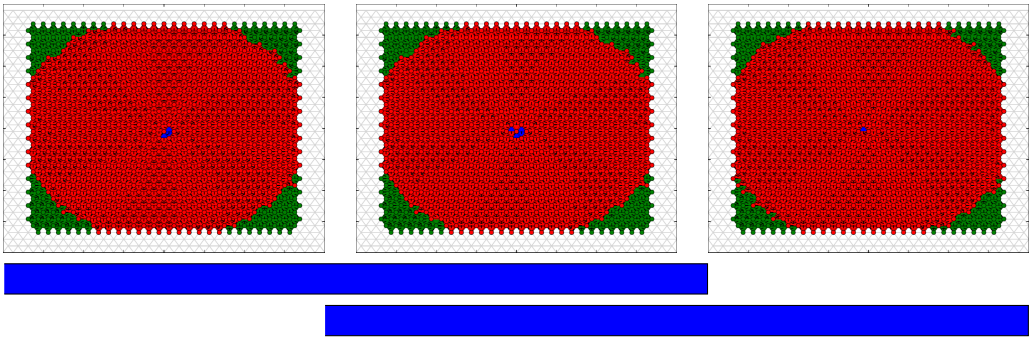


Figure 7 Index pairs on two slightly perturbed multivector fields (left, right) and their intersection (middle). As before, the isolating neighborhood N is in green, E is in red, and $P \setminus E$ is in blue. Note that we have the same difficulty as in Figure 5, where there are two homology generators in the intersection multivector field, so we get a broken bar code.

▶ **Theorem 18.** *Let R be the output of Algorithm 2 applied to index pair (P, E) in N for isolated invariant set S. The pair $(P, E \setminus R)$ is an index pair for S in N.*

The proof for Theorem 18 can be found in the full version. Hence, Algorithm 2 provides a means by which the user may enlarge $P \setminus E$. As this algorithm is parameterized, a robust choice of δ may be application specific. We also include some demonstrations on the effectiveness of using this technique. A real instance of the difficulty can be seen in Figure 7, while the application of Algorithm 2 with $\delta = 5$ to solve the problem is found in Figure 8.

4 Tracking Invariant Sets

In the previous section, we established the persistence of the Conley index of invariant sets in consecutive multivector fields which are isolated by a single isolating neighborhood. In this section, we develop an algorithm to "track" an invariant set over a sequence of isolating neighborhoods. A classic example is a hurricane, where if one were to sample wind velocity at times $t_0, t_1, \ldots, t_n$, there may be no fixed N which captures the eye of the hurricane at all t_i without also capturing additional, undesired invariant sets at some t_j.

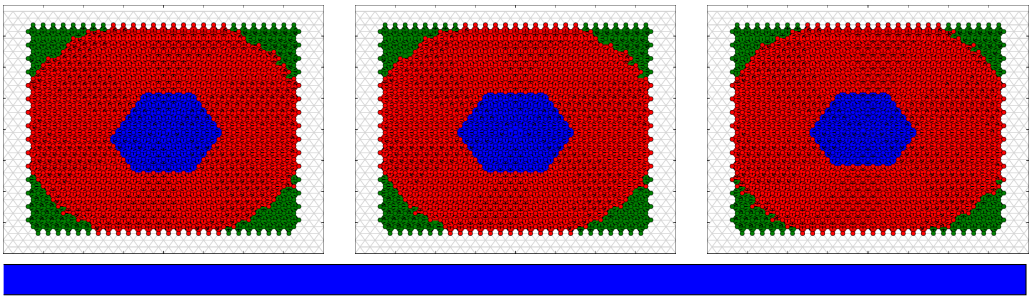


Figure 8 The same index pairs as in Figure 7 with the same color scheme, but after applying Algorithm 2 to reduce the size of E. This forces a 2-dimensional homology generator to persist across both multivector fields (left, right) and their intersection (middle).

4.1 Changing the Isolating Neighborhood

Thus far, we have defined a notion of persistence of the Conley Index for some fixed isolating neighborhood N and simplicial complex K. This setting is very inflexible – one may want to incorporate domain knowledge to change N so as to capture changing features of a sequence of sampled dynamics. We now extend the results in Section 3 to a setting where N need not be fixed. Throughout this section, we consider multivector fields $\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_n$ with corresponding isolated invariant sets $S_1, S_2, \ldots, S_n$. In addition, we assume that there exist isolating neighborhoods $N_1, N_2, \ldots, N_{n-1}$ where N_i isolates both S_i and S_{i+1}. We will also require that if $1 < i < n$, the invariant set S_i is isolated by $N_i \cup N_{i+1}$. Note that for each i where $1 < i < n$, there exist two index pairs for S_i: one index pair $(P_i^{(i-1)}, E_i^{(i-1)})$ in N_{i-1} and another index pair $(P_i^{(i)}, E_i^{(i)})$ in N_i. In the case of $i = 1$, there is only one index pair $(P_1^{(1)}, E_1^{(1)})$ for S_i. Likewise, in the case of $i = n$, there is a single index pair $(P_n^{(n-1)}, E_n^{(n-1)})$.

By applying the techniques of Section 3, we obtain a sequence of persistence modules:

$$
\begin{aligned}
&H_p\left(P_1^{(1)}, E_1^{(1)}\right) \longleftarrow H_p\left(P_1^{(1)} \cap P_2^{(1)}, E_1^{(1)} \cap E_2^{(1)}\right) \longrightarrow H_p\left(P_2^{(1)}, E_2^{(1)}\right) \\
&H_p\left(P_2^{(2)}, E_2^{(2)}\right) \longleftarrow H_p\left(P_2^{(2)} \cap P_3^{(2)}, E_2^{(2)} \cap E_3^{(2)}\right) \longrightarrow H_p\left(P_3^{(2)}, E_3^{(2)}\right) \\
&H_p\left(P_3^{(3)}, E_3^{(3)}\right) \longleftarrow H_p\left(P_3^{(3)} \cap P_4^{(3)}, E_3^{(3)} \cap E_4^{(3)}\right) \longrightarrow H_p\left(P_4^{(3)}, E_4^{(3)}\right) \qquad (2) \\
&\qquad\qquad \vdots \\
&H_p\left(P_{n-1}^{(n-1)}, E_{n-1}^{(n-1)}\right) \longleftarrow H_p\left(P_{n-1}^{(n-1)} \cap P_n^{(n-1)}, E_{n-1}^{(n-1)} \cap E_n^{(n-1)}\right) \longrightarrow H_p\left(P_n^{(n-1)}, E_n^{(n-1)}\right).
\end{aligned}
$$

In the remainder of this subsection, we develop the theory necessary to combine these modules into a single module. Without any loss of generality, we will combine the first modules into a single module, which will imply a method to combine all of the modules into one.

First, we note that by Theorem 6, we have that $H_p(P_2^{(1)}, E_2^{(1)}) \cong H_p(P_2^{(2)}, E_2^{(2)})$. To combine the persistence modules

$$\begin{aligned} H_p(P_1^{(1)}, E_1^{(1)}) &\longleftarrow H_p(P_1^{(1)} \cap P_2^{(1)}, E_1^{(1)} \cap E_2^{(1)}) \longrightarrow H_p(P_2^{(1)}, E_2^{(1)}) \\ H_p(P_2^{(2)}, E_2^{(2)}) &\longleftarrow H_p(P_2^{(2)} \cap P_3^{(2)}, E_2^{(2)} \cap E_3^{(2)}) \longrightarrow H_p(P_3^{(2)}, E_3^{(2)}). \end{aligned} \tag{3}$$

into a single module, it is either necessary to explicitly find a simplicial map which induces an isomorphism $\phi : H_p(P_2^{(1)}, E_2^{(1)}) \to H_p(P_2^{(2)}, E_2^{(2)})$, or to construct some other index pair for S_2 denoted (P, E) such that $P_2^{(1)}, P_2^{(2)} \subset P$ and $E_2^{(1)}, E_2^{(2)} \subset E$. This would allow substituting both occurrences of $(P_2^{(1)}, E_2^{(1)})$ or $(P_2^{(2)}, E_2^{(2)})$ for (P, E), and allow the combining of all the modules in Equation 2 into a single module. Since constructing the isomorphism given by Theorem 6 is fairly complicated, we opt for the second approach. First, we define a special type of index pair that is sufficient for our approach.

▶ **Definition 19** (Strong Index Pair). *Let (P, E) be an index pair for S under $\mathcal{V}$. The index pair (P, E) is a* strong index pair *for S if for each $\tau \in E$, there exists a $\sigma \in S$ such that there is a path $\rho : \mathbb{Z}_{[a,b]} \to P$ where $\rho(a) = \sigma$ and $\rho(b) = \tau$.*

Intuitively, a strong index pair (P, E) for S is an index pair for S where each simplex $\tau \in E$ is reachable from a path originating in S. Strong index pairs have the following useful property.

▶ **Theorem 20.** *Let S denote an invariant set isolated by N, N', and $N \cup N'$ under $\mathcal{V}$. If (P, E) and (P', E') are strong index pairs for S in N, N' under $\mathcal{V}$, then the pair $(\mathsf{pf}_{N \cup N'}(P \cup P'), \mathsf{pf}_{N \cup N'}(E \cup E'))$ is a strong index pair for S in $N \cup N'$ under $\mathcal{V}$.*

The proof for Theorem 20 can be found in the full version. Crucially, this theorem gives a persistence module

$$H_p(P_2^{(1)}, E_2^{(1)}) \longrightarrow H_p\left(\mathsf{pf}\left(P_2^{(1)} \cup P_2^{(2)}\right), \mathsf{pf}\left(E_2^{(1)} \cup E_2^{(2)}\right)\right) \longleftarrow H_p(P_2^{(2)}, E_2^{(2)}) \tag{4}$$

where the arrows are given by the inclusion. Note that since these are all index pairs for the same S, it follows that we have $H_p\left(P_2^{(1)}, E_2^{(1)}\right) \cong H_p\left(\mathsf{pf}\left(P_2^{(1)} \cup P_2^{(2)}\right), \mathsf{pf}\left(E_2^{(1)} \cup E_2^{(2)}\right)\right) \cong H_p\left(P_2^{(2)}, E_2^{(2)}\right)$. Hence, we will substitute $H_p\left(\mathsf{pf}\left(P_2^{(1)} \cup P_2^{(2)}\right), \mathsf{pf}\left(E_2^{(1)} \cup E_2^{(2)}\right)\right)$ into the persistence module. By using the modules in Equation 2, we get a new sequnece of persistence modules

$$\begin{aligned}
&H_p\left(P_1^{(1)}, E_1^{(1)}\right) \longleftarrow H_p\left(P_1^{(1)} \cap P_2^{(1)}, E_1^{(1)} \cap E_2^{(1)}\right) \longrightarrow H_p\left(\mathsf{pf}\left(P_2^{(1)} \cup P_2^{(2)}\right), \mathsf{pf}\left(E_2^{(1)} \cup E_2^{(2)}\right)\right) \\
&H_p\left(\mathsf{pf}\left(P_2^{(1)} \cup P_2^{(2)}\right), \mathsf{pf}\left(E_2^{(1)} \cup E_2^{(2)}\right)\right) \longleftarrow H_p\left(P_2^{(2)} \cap P_3^{(2)}, E_2^{(2)} \cap E_3^{(2)}\right) \longrightarrow H_p\left(\mathsf{pf}\left(P_3^{(2)} \cup P_3^{(3)}\right), \mathsf{pf}\left(E_3^{(2)} \cup E_3^{(3)}\right)\right) \\
&H_p\left(\mathsf{pf}\left(P_3^{(2)} \cup P_3^{(3)}\right), \mathsf{pf}\left(E_3^{(2)} \cup E_3^{(3)}\right)\right) \longleftarrow H_p\left(P_3^{(3)} \cap P_4^{(3)}, E_3^{(3)} \cap E_4^{(3)}\right) \longrightarrow H_p\left(\mathsf{pf}\left(P_4^{(3)} \cup P_4^{(4)}\right), \mathsf{pf}\left(E_4^{(3)} \cup E_4^{(4)}\right)\right) \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \vdots \\
&H_p\left(\mathsf{pf}\left(P_{n-1}^{(n-2)} \cup P_{n-1}^{(n-1)}\right), \mathsf{pf}\left(E_{n-1}^{(n-2)} \cup E_{n-1}^{(n-1)}\right)\right) \longleftarrow H_p\left(P_{n-1}^{(n-1)} \cap P_n^{(n-1)}, E_{n-1}^{(n-1)} \cap E_n^{(n-1)}\right) \longrightarrow H_p\left(P_n^{(n-1)}, E_n^{(n-1)}\right).
\end{aligned} \tag{5}$$

which can immediately be combined into a single persistence module.

This approach is not without it's disadvantages, however. Namely, if (P, E) and (P', E') are index pairs for S in N and N', it requires that (P, E) and (P', E') are strong index pairs and that S is isolated by $N \cup N'$. Fortunately, the push forward approach to computing an index pair in N gives a strong index pair.

▶ **Theorem 21.** *Let S be an isolated invariant set where N is an isolating neighborhood for S. The pair $(\mathit{pf}(\mathit{cl}(S)), \mathit{pf}(\mathit{m}(S)))$ is a strong index pair in N for S.*

Proof. We note that by Theorem 20, the pair $(\mathsf{pf}(\mathsf{cl}(S)), \mathsf{pf}(\mathsf{m}(S)))$ is an index pair for S in N. Hence, it is sufficient to show that the index pair is strong. Note that by definition, for all $\sigma \in \mathsf{m}(S)$, there exists a $\tau \in S$ such that σ is a face of τ. Hence, $\sigma \in F_{\mathcal{V}}(\tau)$. Note that $\mathsf{pf}(\mathsf{m}(S))$ is precisely the set of simplices σ' for which there exists a path originating in $\mathsf{m}(S)$ and terminating at σ', so it is immediate that there is a path originating in S and terminating at σ. Hence, the pair $(\mathsf{pf}(\mathsf{cl}(S)), \mathsf{pf}(\mathsf{m}(S)))$ is a strong index pair. ◀

Our enlarging scheme given in Algorithm 2 does not affect the strongness of an index pair.

▶ **Theorem 22.** *Let R be the output of applying Algorithm 2 to the strong index pair (P, E) in N for S with some parameter δ. The pair $(P, E \setminus R)$ is a strong index pair for S in N.*

Proof. Theorem 18 gives that $(P, E \setminus R)$ is an index pair for S in N, so it is sufficient to show that such an index pair is strong. Note that P does not change, but the strongness of index pairs only requires paths to be in P. Since all paths in (P, E) are also paths in $(P, E \setminus R)$, it follows that $(P, E \setminus R)$ is a strong index pair in N. ◀

These theorems give us a canonical scheme for choosing invariant sets from a sequence of multivector fields and then computing the barcode of persistence module given in Equation (5). We give our exact scheme in Algorithm 3.

The astute reader will notice an important detail about Algorithm 3. Namely, the `find` function is parameterized by a nonnegative integer δ, and the function has not yet been defined. In particular, said function must output a closed $N_i \supseteq S_i$ such that S_i is isolated by $N_{i-1} \cup N_i$. An obvious choice is to let $N_i := N_{i-1}$, but such an approach does not allow one to capture essential solutions that "move" outisde of $N_{i-1} = N_i$ as the multivector fields change. We give a nontrivial `find` function in the next subsection that can be used to capture such changes in an essential solution.

4.2 Finding Isolating Neighborhoods

Given an invariant set S isolated by N with respect to $\mathcal{V}$, we now propose a method to find a closed, nontrivial $N' \subseteq K$ such that $N \cup N'$ isolates S. Our method relies heavily on the concept of δ-collar introduced in Section 3. In fact, we will let $N' = C_\delta(S) \setminus R$ such that $N \cup N'$ isolates S. Hence, it is sufficient to devise an algorithm to find $C_\delta(S) \setminus R$. Before we give and prove the correctness of the algorithm, we briefly introduce the notion of the *push backward* of some set S in N, denoted $\mathsf{pb}_N(S)$. We let $\mathsf{pb}_N(S) = \{x \in N \mid \exists\ \rho\ :\ \mathbb{Z}_{[a,b]} \to N,\ \rho(a) = x, \rho(b) \in S\}$. Essentially, the push backward of S in N is the set of simplices $\sigma \in N$ for which there exists a path in N from σ to S.

We now prove that $N \cup (C_\delta(S)) \setminus R$ isolates S. Note that since $S \subseteq C_\delta(S) \setminus R$, this also implies that $C_\delta(S) \setminus R$ isolates S.

▶ **Theorem 23.** *Let S denote an invariant set isolated by $N \subseteq K$ under $\mathcal{V}$. If $C_\delta(S) \setminus R$ is the output of Algorithm 4 on inputs $S, N, \mathcal{V}, \delta$, then the closed set $N \cup (C_\delta(S) \setminus R)$ isolates S.*

Algorithm 3 Scheme for computing the persistence of the Conley Index, variable N.

Input: Sequence of multivector fields $\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_n$, closed set $N_0 \subset K$, $\delta \in \mathbb{Z}$.
Output: Barcodes corresponding to persistence of Conley Index
$i \leftarrow 1$
while $i <= n$ **do**
  $S_i \leftarrow \mathsf{Inv}_{\mathcal{V}_i}(N_{i-1})$
  $\left(P'_{i,1}, E'_{i,1}\right) \leftarrow \left(\mathsf{pf}_{N_{i-1}}\left(\mathsf{cl}\left(S_i\right)\right), \mathsf{pf}_{N_{i-1}}\left(\mathsf{m}\left(S_i\right)\right)\right)$
  $R_{i,1} \leftarrow \texttt{findR}(S_i, P'_{i,1}, E'_{i,1}, \mathcal{V}, \delta)$
  $\left(P_i^{(1)}, E_i^{(1)}\right) \leftarrow \left(P'_{i,1}, E'_{i,1} \setminus R_{i,1}\right)$
  $N_i \leftarrow \texttt{find}(S_i, N_{i-1}, \mathcal{V}, \delta)$
  $\left(P'_{i,2}, E'_{i,2}\right) \leftarrow \left(\mathsf{pf}_{N_i}\left(\mathsf{cl}\left(S_i\right)\right), \mathsf{pf}_{N_i}\left(\mathsf{m}\left(S_i\right)\right)\right)$
  $R_{i,2} \leftarrow \texttt{findR}(S_i, P'_{i,2}, E'_{i,2}, \mathcal{V}, \delta)$
  $\left(P_i^{(2)}, E_i^{(2)}\right) \leftarrow \left(P'_{i,2}, E'_{i,2} \setminus R_{i,2}\right)$
  if $i = 1$ **then**
    $\left(P_i, E_i\right) \leftarrow \left(P_i^{(2)}, E_i^{(2)}\right)$
  else if $i = n$ **then**
    $\left(P_i, E_i\right) \leftarrow \left(P_i^{(1)}, E_i^{(1)}\right)$
  else
    $\left(P_i, E_i\right) \leftarrow \left(\mathsf{pf}_{N_{i-1} \cup N_i}\left(P_i^{(1)} \cup P_i^{(2)}\right), \mathsf{pf}_{N_{i-1} \cup N_i}\left(E_i^{(1)} \cup E_i^{(2)}\right)\right)$
  end
  $i \leftarrow i + 1$
end
return *zzPers* $\left(\left(P_1, E_1\right) \supseteq \left(P_1^{(2)} \cap P_2^{(1)}, E_1^{(2)} \cap E_2^{(1)}\right) \subseteq \left(P_2, E_2\right) \supseteq \ldots \subseteq \left(P_n, E_n\right)\right)$

Proof. For a contradiction, assume that there exists a path $\rho : \mathbb{Z}_{[a,b]} \to N \cup (C_\delta(S) \setminus R)$ so that $\rho(a), \rho(b) \in S$ where there is an i satisfying $a < i < b$ with $\rho(i) \notin S$. Note that since N isolates S, if $N \cup C_\delta(S) \setminus R$ does not isolate S, then there must exist a first $k \in \mathbb{Z}_{[a,b]}$ such that $\rho(k) \in C_\delta(S) \setminus N$ and $F_{\mathcal{V}}(\rho(k)) \cap \mathsf{pb}_N(S) \neq \emptyset$. If this were not the case, then N would not isolate S. Without loss of generality, we assume that for all $a < j < k$, we have that $\rho(j) \notin S$. Note that for all $j \in \mathbb{Z}_{[a+1,k-1]}$, when $\rho(j)$ is removed from the stack V, if $\rho(j+1)$ has not been visited, then $\rho(j+1)$ is added to the stack. Hence, this implies that if any $\rho(j)$ is visited, then $\rho(k)$ will be added to R. If this were not the case, there would exist some $\rho(j)$ such that when $\rho(j)$ was removed from the stack, $\rho(j+1)$ was not visited and was not added to the stack. This implies that $F_{\mathcal{V}}(\rho(j)) \cap \mathsf{pb}_N(S) \neq \emptyset$, which contradicts $\rho(k)$ being the first such simplex in the path.

Hence, since $\rho(a+1)$ is added to the stack, it follows that $\rho(k)$ is added to R, which implies that $\rho\left(\mathbb{Z}_{[a,b]}\right) \not\subset N \cup (C_\delta(S) \setminus R)$. Note too that $N \cup C_\delta(S) \setminus R$ must be closed, as if there is a $\sigma \in N$ such that $\rho(k) \leq \sigma$, then $\rho(k) \in N$ because N is closed, a contradiction. But when $\rho(k)$ is removed from $C_\delta(S)$, any of its cofaces which are in $C_\delta(S)$ are also removed. Hence, N is closed, $C_\delta(S) \setminus R$ is closed, so their union must be closed. ◀

Hence, we use Algorithm 4 as the `find` function in our scheme given in Algorithm 3. We give an example of our implementation of Algorithm 3 using the `find` function in Figure 9.

Algorithm 4 `find`$(S, N, \mathcal{V}, \delta)$.

Input: Invariant set S isolated by N under $\mathcal{V}$, $\delta \in \mathbb{Z}$
Output: Closed set $N' \supseteq S$ such that $N \cup N'$ isolates S under $\mathcal{V}$

```
V ← new ; stack()
R ← new ; set()
pb ← pb_N(S)
foreach σ ∈ C_δ(S) ∪ N do
  | setUnvisited(σ)
end
foreach σ ∈ S do
  | adj ← cl(σ) ∪ [v]_V
  | foreach τ ∈ adj do
  |   | if τ ∉ S and τ ∈ C_δ(S) ∪ N then
  |   |   | push(V, τ)
  |   | end
  | end
end
while size(V) > 0 do
  | v ← pop(V)
  | if !hasBeenVisited(v) then
  |   | setVisited(v)
  |   | if (cl(v) ∪ [v]_V) ∩ pb ≠ ∅ then
  |   |   | add(R, v)
  |   |   | cf ← cofaces(v)
  |   |   | addAll(R, cf)
  |   | else
  |   |   | foreach σ ∈ (cl(v) ∪ [σ]_V) ∩ (C_δ(S) ∪ N) do
  |   |   |   | push(V, σ)
  |   |   | end
  |   | end
  | end
end
return C_δ(S) \ R
```

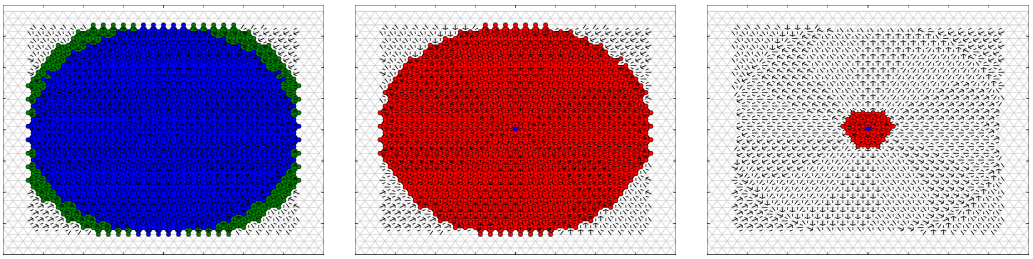


Figure 9 Three different index pairs generated from our scheme in Algorithm 3. The isolating neighborhood is in green, E is in red, and $P \setminus E$ is in blue. Note how the isolating neighborhood changes by defining a collar around the invariant sets (which are exactly equal to $P \setminus E$). Between the left and middle multivector fields, the periodic attractor partially leaves K, so the maximal invariant set in N is reduced to just a triangle. Hence, the size of N drastically shrinks between the middle and right multivector fields.

5 Conclusion

In this paper, we focused on computing the persistence of Conley indices of isolated invariant sets. Our preliminary experiments show that the algorithm can effectively compute this persistence in the presence of noise. It will be interesting to derive a stability theory for this persistence. Toward that direction, we have included a promising result on the stability of isolated invariant sets in the full version. In designing the tracking algorithm, we have made certain choices about the isolated neighborhoods and the invariant sets. Are there better choices? Which ones work better in practice? A thorough investigation with data sets in practice is perhaps necessary to settle this issue.

References

1 J. A. Barmak. *Algebraic Topology of Finite Topological Spaces and Applications*. Lecture Notes in Mathematics **2032**. Springer Verlag, Berlin - Heidelberg - New York, 2011.

2 N. P. Bhatia and G. P. Szegö. *Dynamical Systems: Stability Theory and Applications*. Lecture Notes in Mathematics **35**. Springer Verlag, Berlin - Heidelberg - New York, 1967.

3 Gunnar Carlsson and Vin de Silva. Zigzag persistence. *Foundations of Computational Mathematics*, 10(4):367–405, August 2010. doi:10.1007/s10208-010-9066-0.

4 C. Conley. Isolated invariant sets and the Morse index. In *CBMS Regional Conference Series 38, American Mathematical Society*, 1978.

5 Tamal K. Dey, Mateusz Juda, Tomasz Kapela, Jacek Kubica, Michal Lipinski, and Marian Mrozek. Persistent homology of Morse decompositions in combinatorial dynamics. *SIAM Journal on Applied Dynamical Systems*, 18(1):510–530, 2019. doi:10.1137/18M1198946.

6 R. Forman. Combinatorial vector fields and dynamical systems. *Mathematische Zeitschrift*, 228:629–681, 1998. doi:10.1007/PL00004638.

7 R. Forman. Morse theory for cell complexes. *Advances in Mathematics*, 134:90–145, 1998. doi:10.1006/aima.1997.1650.

8 J. Hale and H. Koçak. *Dynamics and Bifurcations*. Texts in Applied Mathematics **3**. Springer-Verlag, 1991.

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

10 Michał Lipiński, Jacek Kubica, Marian Mrozek, and Thomas Wanner. Conley-Morse-Forman theory for generalized combinatorial multivector fields on finite topological spaces. arXiv:1911.12698 [math.DS], 2019. arXiv:1911.12698.

11 Edward N. Lorenz. *Deterministic Nonperiodic Flow*, pages 25–36. Springer New York, New York, NY, 2004. doi:10.1007/978-0-387-21830-4_2.

12 K. Mischaikow and M. Mrozek. *The Conley Index*. Handbook of Dynamical Systems II: Towards Applications. (B. Fiedler, ed.) North-Holland, 2002.

13 K. Mischaikow, M. Mrozek, J. Reiss, and A. Szymczak. Construction of symbolic dynamics from experimental time series. *Physical Review Letters*, 82:1144–1147, February 1999. doi:10.1103/PhysRevLett.82.1144.

14 Konstantin Mischaikow and Marian Mrozek. Chaos in the Lorenz equations: a computer-assisted proof. *Bulletin of the American Mathematical Society*, 33:66–72, 1995. doi:10.1090/S0273-0979-1995-00558-6.

15 Marian Mrozek. Conley–Morse–Forman theory for combinatorial multivector fields on Lefschetz complexes. *Foundations of Computational Mathematics*, 17(6):1585–1633, December 2017. doi:10.1007/s10208-016-9330-z.

16 J.R. Munkres. *Topology*. Featured Titles for Topology Series. Prentice Hall, Incorporated, 2000.

17 H.J. Poincaré. Sur le probleme des trois corps et les équations de la dynamique. *Acta Mathematica*, 13:1–270, 1890.

On Implementing Straight Skeletons: Challenges and Experiences

Günther Eder
Universität Salzburg, FB Computerwissenschaften, Austria
geder@cs.sbg.ac.at

Martin Held
Universität Salzburg, FB Computerwissenschaften, Austria
held@cs.sbg.ac.at

Peter Palfrader
Universität Salzburg, FB Computerwissenschaften, Austria
palfrader@cs.sbg.ac.at

Abstract

We present CGAL implementations of two algorithms for computing straight skeletons in the plane, based on exact arithmetic. One code, named SURFER2, can handle multiplicatively weighted planar straight-line graphs (PSLGs) while our second code, MONOS, is specifically targeted at monotone polygons. Both codes are available on GitHub. We discuss algorithmic as well as implementational and engineering details of both codes. Furthermore, we present the results of an extensive performance evaluation in which we compared SURFER2 and MONOS to the straight-skeleton package included in CGAL. It is not surprising that our special-purpose code MONOS outperforms CGAL's straight-skeleton implementation. But our tests provide ample evidence that also SURFER2 can be expected to be faster and to consume significantly less memory than the CGAL code. And, of course, SURFER2 is more versatile because it can handle multiplicative weights and general PSLGs as input. Thus, SURFER2 currently is the fastest and most general straight-skeleton code available.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases weighted straight skeleton, implementation, algorithm engineering, experiments, CGAL, CORE

Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.38

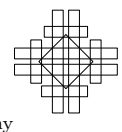
Supplementary Material Our source codes are provided on GitHub and can be used freely under the GPL(v3) license: `https://github.com/cgalab/monos` and `https://github.com/cgalab/surfer2`.

Funding Work supported by Austrian Science Fund (FWF): Grants ORD 53-VO and P31013-N31.

1 Introduction

Straight skeletons were introduced to computational geometry by Aichholzer et al. [2]. Suppose that the edges of a simple polygon P move inwards with unit speed in a self-parallel manner, thus generating mitered offsets inside of P. Then the *(unweighted) straight skeleton* of P is the geometric graph whose edges are given by the traces of the vertices of the shrinking mitered offset curves of P; see Figure 1, left. The process of simulating the shrinking offsets is called *wavefront propagation*. Straight skeletons are known to have applications in diverse fields, with the modeling of roof-like structures being one of the more prominent ones [19, 15, 16] We refer to Huber [17] for a detailed discussion of typical applications.

As a first generalization of straight skeletons, the *multiplicatively weighted straight skeleton* was introduced early on by Aichholzer and Aurenhammer [1] and then by Eppstein and Erickson [13]. In the presence of multiplicative weights, wavefront edges no longer move



at unit speed. Rather, they move at different speeds: Every edge of P is assigned its own constant speed; see Figure 1. Although weighted straight skeletons have been used in applications for years, their characteristics were studied only recently by Biedl et al. [5, 6].

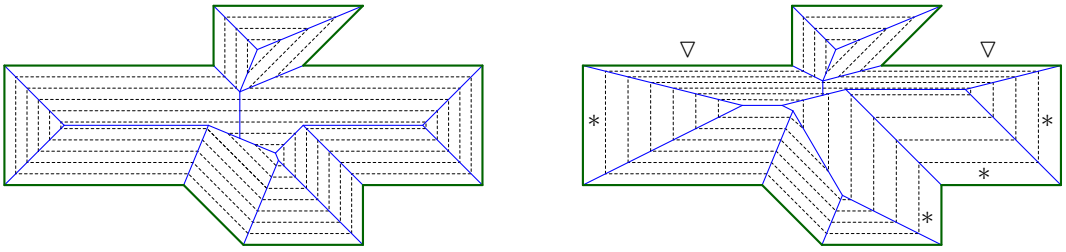


Figure 1 Left: The (unweighted) straight skeleton (in blue) plus a family of wavefronts (dashed) for the green polygon. Right: The weighted straight skeleton for the case that edges marked with $*$ have twice the weight and edges marked with $\triangledown$ have half the weight of the unmarked edges.

Held and Palfrader [15] define an *additively weighted straight skeleton* as the geometric graph whose edges are the traces of vertices of the wavefronts over the propagation period, with additive weights being assigned to the edges of P. The presence of additive weights means that the edges of P start to move at different points in time. In [15, 16] they study straight skeletons of planar straight-line graphs (PSLGs) with a combination of both additive and multiplicative weights. (A PSLG is an embedding of a planar graph such that all edges are straight-line segments that do not intersect pairwise except at common end-points.)

The straight-skeleton algorithms with the best worst-case bounds are due to Eppstein and Erickson [13] and Vigneron et al. [10, 23]. These algorithms seem difficult to implement. Indeed, progress on implementations has been rather limited so far. Felkel and Obdržálek [14] describe a simple straight-skeleton algorithm but it turned out to be flawed [24]. The first comprehensive code for computing straight skeletons was implemented by Cacciola [9] and is shipped with CGAL [22]. It is based on the algorithm by Felkel and Obdržálek [14], but was modified significantly to make it work correctly [8]. It handles polygons with holes as input.

The straight-skeleton code Bone by Huber and Held [18] handles PSLGs as input and runs in $O(n \log n)$ time and $O(n)$ space in practice. However, it is not capable of handling weighted skeletons. As a first step to extend Bone to weighted skeletons, Eder and Held [12] generalize motorcycle graphs to weighted motorcycle graphs. Still, while this is a fascinating research area of its own, this does not yet provide an avenue for an actual implementation.

2 Contribution

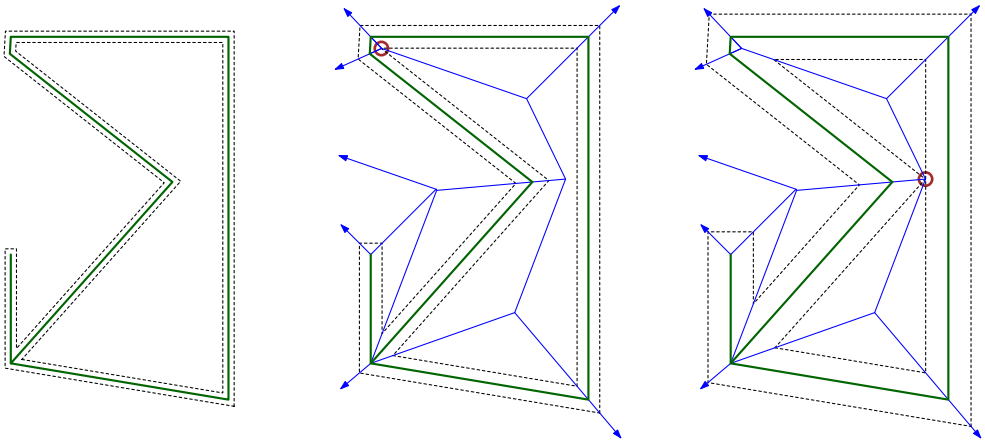
Palfrader et al. [21] describe a prototype implementation of a straight-skeleton code named Surfer. Since Surfer is the fastest implementation of unweighted straight skeletons known to us, it was the natural candidate for a re-implementation and extension to multiplicative weights. As the handling of degenerate cases had been fairly tricky for unweighted input, we decided to base the new implementation, Surfer2, on exact arithmetic. (However, Surfer2 has an option to run it on conventional floating-point arithmetic.) In order to be able to compare Surfer2 to a special-purpose algorithm for computing straight skeletons of monotone polygons, which is known to have an $\mathcal{O}(n \log n)$ worst-case complexity, we also implemented the algorithm by Biedl et al. [4]. This implementation was named Monos.

We subjected our implementations to extensive practical tests. Summarizing, Surfer2 is slower than Cgal's straight-skeleton code for small data sets, of about the same speed for polygons that contain 1000 vertices, and 100 times faster for polygons with 10 000 vertices. The memory footprint of Surfer2 is linear while Cgal's code consumes a quadratic amount

of memory. For monotone polygons, our code MONOS achieves a speed-up of about 1.5 compared to SURFER2. Our tests showed clearly that engineering considerations do yield practical speed-ups and that there is a significant price to pay for the use of exact arithmetic.

3 Straight skeleton and wavefront propagation

While every input edge of a polygon emanates a wavefront edge only to one side, namely to the interior of the polygon, PSLG edges emanate wavefront edges on both sides. As in the case of polygons, all wavefront edges are joined-up along angular bisectors of the input edges. In order to handle input vertices of degree one, at each such vertex v one additional wavefront edge is created that moves away from v and its corresponding PSLG edge e in a direction orthogonal to e; see the left-most vertical segment in Figure 2a. This way, the wavefronts form closed polygons even in the presence of degree-one input vertices.



(a) initial wavefront. **(b)** edge event. **(c)** split event.

Figure 2 Wavefronts (dashed) at specific times for a PSLG (green), straight skeleton (blue).

Initially, the wavefront coincides with the input edges. If sufficiently little time has elapsed after the start of the wavefront propagation such that no event has occurred yet, then the wavefront forms a mitered offset of the PSLG. The polygons of this mitered offset partition the plane into two (possibly disconnected or multiply-connected) areas: one area that has been swept by the wavefront and another area that has not yet been reached. As time increases and the propagation continues, the area swept by the wavefront gets larger and larger, and wavefront edges will shrink to zero length, thus collapsing in *edge events*; see Figure 2b. Furthermore, vertices may move into the interior of non-incident wavefront edges in *split events*; see Figure 2c. Every split event splits one wavefront edge. Depending on the topology of the not-yet swept region affected, such an event will also split the region into two polygons or reduce the number of holes of the region.

This wavefront-propagation process continues until no more events happen. Then the straight skeleton is the planar graph whose edges are the traces of wavefront vertices during the propagation. Note that some wavefront vertices on the outer region will continue to escape to infinity; these vertices trace out rays that form unbounded arcs of the skeleton.

Simulating the wavefront propagation is a common building block of several straight-skeleton algorithms. These algorithms differ primarily in how they find the next event. All future events already known are usually maintained in a priority queue of event times. It is standard to use a min-heap to store the times of those future events.

4 Straight skeleton of a monotone polygon

4.1 Monotone algorithm

Biedl et al. [4] describe an $\mathcal{O}(n \log n)$ time algorithm to compute the straight skeleton of a simple n-vertex monotone polygon $\mathcal{P}$. Without loss of generality we assume $\mathcal{P}$ to be x-monotone. Their algorithm consists of two steps: (i) The polygon $\mathcal{P}$ is split into an upper and lower monotone chain, and the straight skeleton of each chain is computed individually. (ii) The final straight skeleton $\mathcal{S}(\mathcal{P})$ is obtained by merging these two straight skeletons.

We employ a classical wavefront propagation to obtain the straight skeleton of a monotone chain in $\mathcal{O}(n \log n)$ time [4]. The monotonicity of the chains guarantees that every change of the wavefront topology is witnessed by an edge event. That is, split events cannot occur.

In the second step, the skeletons of the upper and lower chains are merged to form $\mathcal{S}(\mathcal{P})$. This merge is based on a left-to-right traversal of the two chains and their respective straight-skeleton faces. Starting at the leftmost vertex of both chains, the angular bisector between the incident edges is constructed. It intersects arcs of both the top and bottom straight skeleton. We stop at the first intersection reached and modify the respective straight skeleton locally by creating a node. Then the next bisector is constructed between the two edges that induce the faces of the upper and lower skeleton we are currently in. This process is repeated until the rightmost vertex of $\mathcal{P}$ is reached, thus obtaining the final skeleton $\mathcal{S}(\mathcal{P})$.

4.2 Implementational details

Monos uses Cgal's `Exact_predicates_exact_constructions_kernel_with_sqrt` algebraic kernel, which is backed by Core's `Core::Expr` exact number type.

Intersection computations. A major part of the merge step are intersection tests and intersection computations between bisectors and skeleton arcs: Careful engineering resulted in substantial performance improvements for these intersection tests. Initially, we applied Cgal's `do_intersect` and `intersection` rather naively to an arc segment (seen as a straight-line segment) and a bisector. However, explicitly deciding whether the end-points of an arc segment lie on different sides of the supporting line of a merge bisector is sufficient to decide whether an intersection occurs. Once we know that an intersection occurs then we apply Cgal's `intersection` routine to the supporting lines of the arc and the bisector. A test on more than one thousand input polygons with at least 10 000 vertices each shows average runtime savings of about 9 % when using the latter method.

Collinear edges. Input that contains collinear edges needs special care. The crux is that no "direction" need be implied by a wavefront vertex that traces out a bisector. Assume that two collinear wavefront edges become adjacent during the computation of the straight skeleton of a monotone chain. At the time of the event we have a straight-skeleton node p at which the event occurs. Hence we define the bisector of the two collinear edges to be perpendicular to them and to start at p, thereby moving into the same direction as the wavefront edges.

During the merge step, it is less obvious how to proceed in such a case. In Figure 3, left, we see that the left-to-right merge has arrived at the point p. The next bisector, b, is defined by the input edges e_i and e_j. If the merge had progressed from right to left, then b would be horizontal and it would start at the point p'', in analogy to how this situation would be handled if it occurred during the construction of the straight skeleton of a monotone chain. See the horizontal red dotted segment, b, in Figure 3, left: This bisector need not intersect p

but can arrive anywhere along the boundary of the face of e_i or the face of e_j. Hence, we need to look ahead and find the point p' where it intersects one of the two boundaries. Note that this "look ahead" does not break the algorithm's complexity as we only have to walk along the boundaries of the two faces we stepped into. At some point, the boundaries intersect, which is the end-point of b. If b is not incident to p, then we need to add the start node p' of b as well; see Figure 3, right. Afterwards we can proceed with the standard merge.

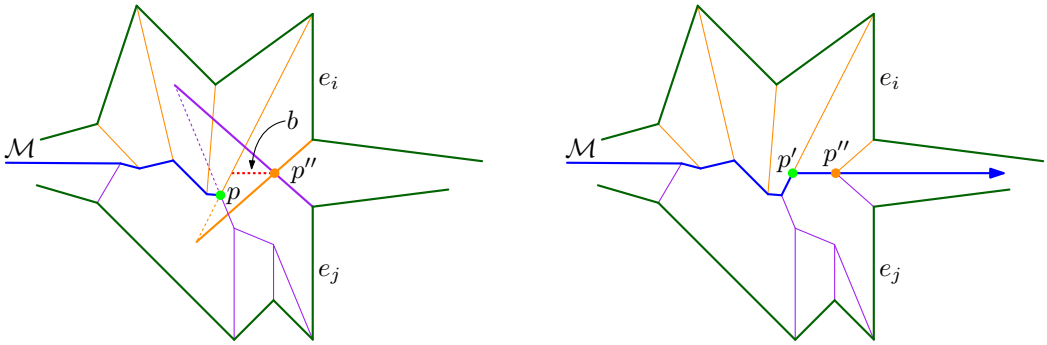


Figure 3 Left: Constructing the merge chain $\mathcal{M}$ at a specific point p where collinearities have to be handled. Right: Looking ahead lets us identify p' as start of the next horizontal bisector.

Min-heap. Initially, MONOS employed C++'s `std::set` as a priority queue due to the requirement to update elements. This structure tends to be implemented as a red-black tree and provides all operations required at appropriate asymptotic costs. However, profiling MONOS showed that a significant amount of time was spent in queuing operations. Therefore, we switched to a self-developed binary min-heap already in use in SURFER2. Performance tests showed an average performance gain of 5 %.

5 Weighted straight skeleton of a PSLG

5.1 Triangulation-based algorithm

Aichholzer and Aurenhammer [1] describe an algorithm for computing the straight skeleton of general PSLGs in the plane. It carries over to multiplicatively weighted input in a natural way, provided that all weights are positive. (Biedl et al. [5] show that most of the theory falls apart if negative weights are allowed.) Their approach constructs the straight skeleton by simulating the wavefront propagation. As the wavefront sweeps the plane, they maintain a kinetic triangulation of that part of the plane which has not yet been swept. This triangulation is obtained by triangulating the area inside the convex hull of all wavefronts. Furthermore, all edges of this convex hull are linked with a dummy vertex at infinity.

The area of each triangle of this kinetic triangulation changes over time as its vertices, which are vertices of the wavefront, move along angular (straight-line) bisectors of the input edges. Since each vertex moves at constant velocity, the area of each triangle is a quadratic function in time. As long as no triangle collapses to zero area these triangles serve as certificates for the validity of the kinetic triangulation: The wavefront does not change combinatorially while no triangle collapses. Every change in the topology of the wavefront, i.e., every edge event and every split event, is witnessed by a triangle collapse. Of course, the roots of the quadratic functions give the collapse times of the triangles.

Note that not all triangle collapses witness a corresponding event of the wavefront. Some triangle collapses correspond to internal events only, where the triangulation changes as a wavefront vertex moves over a triangulation diagonal. Aichholzer and Aurenhammer [1] call this type of event a *flip event* because it merely requires the flip of a triangulation diagonal.

The number of edge and split events is linear because the straight skeleton has a linear combinatorial complexity. However, no better bound than $O(n^3)$ is known for the maximum number of flip events for an n-vertex PSLG, with a theoretical worst-case lower bound of $\Omega(n^2)$ [17]. Consequently, the algorithm's worst-case runtime is bounded by $\Omega(n^2 \log n)$ and by $\mathcal{O}(n^3 \log n)$. Our previous work [21] showed that this algorithm runs in $\mathcal{O}(n \log n)$ time in practice when using IEEE 754 arithmetic. In the sequel we present our implementation, Surfer2, of this algorithm based on exact arithmetic, and with support for weighted input.

5.2 Implementational details

Surfer2 uses some of Cgal's geometric primitives, such as Points and Segments. It defaults to using Cgal's `Exact_predicates_exact_constructions_kernel_with_sqrt` algebraic kernel, which is backed by Core's `Core::Expr` exact number type. Surfer2 uses none of Cgal's advanced data structures, with the exception of Cgal's `DCEL` arrangement [20], which is used for the purpose of presenting the straight skeleton to the user. In addition to a library API, Surfer2, like Monos, provides both a command line interface and a GUI. GraphML [7] is our input and output format, as it allows weighted input.

Data structures. To represent the wavefront and the area not yet swept by the wavefront, our implementation explicitly has objects of kinetic triangles, wavefront edges, and wavefront vertices. For each kinetic triangle, we store pointers to the three incident wavefront vertices, and on each side we either have a pointer to the neighboring triangle or a pointer to the incident wavefront edge. Every wavefront edge has references to its two incident wavefront vertices as well as a pointer to the incident kinetic triangle. Wavefront vertices trace out straight skeleton arcs during the propagation period, moving along the bisector of input edges. Every vertex stores pointers to its incident wavefront edges, i.e., the edges that emanate from the input edges defining the bisector along which the vertex moves. Furthermore, it stores the time when it came into existence, and, if it has already been stopped by an event, the time when it was stopped. Additionally, each vertex caches its velocity (as a function of its incident edges), its start location, and also its stop location (if applicable).

We start out by constructing a constrained Delaunay triangulation of the input graph using Cgal's 2D-Triangulation package [25]. From this initially static triangulation we construct the kinetic triangulation just outlined: Neighborhood relations between triangles from the initial triangulation are copied to the kinetic triangulation for triangles that are not split by a constraint. At each constraint, we initialize two wavefront edges, one moving in each direction, that are incident to one kinetic triangle each. Lastly, we initialize the wavefront vertices and compute their velocities based on their incident wavefront edges.

Event types and event queue. The event queue is a binary min-heap whose elements are kinetic triangles, with an order imposed by the next event each triangle witnesses. An unbounded triangle, which is defined by the vertex at infinity and one edge of the convex hull, serves not only as a witness for its edge collapsing but also for one of its vertices leaving the convex hull. For each bounded triangle, we store the time of its next collapse.

Finding the collapse time of a triangle is straightforward in theory. (Recall that it is a root of a second-degree polynomial given by the signed determinant of its moving vertices.) Checking when a vertex leaves the convex hull can be achieved by considering triangles of

three subsequent convex-hull vertices. In practice we would like to know not only the time but also the type of event. This helps to ensure we make progress with the wavefront propagation. Progress is guaranteed if we process an event that is a split event or an edge event: There is only a linear number of such "real" events because each event effectively causes a collapsed triangle to be removed from the triangulation. The crux comes to light when investigating flip events. A flip event happens when a wavefront vertex becomes incident on a triangulation diagonal. We consider the quadrilateral formed by the two triangles along this diagonal, and flip the diagonal within this quadrilateral. If the other triangle also collapsed at precisely the same time, then after this flip we still have two triangles that collapse right now, and we might do another flip that re-establishes the previous configuration; cf. Figure 4.

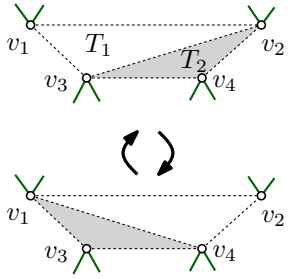


Figure 4 Potential flip-event loop when, at time t, the vertices v_1 through v_4 all become collinear.

To handle multiple events that occur at the same time we apply a twofold strategy: First, we always prioritize "real" events over flip events. (This can be done with no additional cost by using a secondary key for the priority queue.) Second, we need to impose an order on flip events. Consider a maximal set $\mathcal{T}$ of neighboring triangles that all collapse at the same time to line segments on the same supporting line, without any triangulation diagonal shrinking to zero length. Note that a flip in a triangle will always cause its longest edge to be flipped. Let T be the triangle that has the longest edge e among the triangles of $\mathcal{T}$. If the other triangle, T', that is incident at e does not belong to $\mathcal{T}$, then it has positive area, and performing the flip will reduce the number of elements in $\mathcal{T}$ and we have made progress. If, however, $T' \in \mathcal{T}$ then performing the flip will cause the longest edge within $\mathcal{T}$ to become strictly smaller. Hence, a monotonicity argument implies that we make progress, too.

Another type of event that shows up in practice if the input need not be in general position is given by non-incident vertices that coincide at some point in time. Topologically, this is equivalent to a split event as previously non-incident wavefront portions become incident. From the viewpoint of a kinetic triangulation this event looks more like an edge event, with the exception that it is not a wavefront edge that collapsed but a triangulation diagonal. Handling this event is thus similar to handling two edge events, with one event per triangle incident at the collapsed diagonal. Figure 5, left, shows such an event.

When parallel wavefront elements move into each other, events happen as triangles collapse. The wavefront edges that comprise these parallel wavefront elements are incident to different kinetic triangles. As we process these collapsed triangles, one after the other, we want to leave the kinetic triangulation in a consistent state between events. Generally, this means creating new kinetic vertices to replace old ones as edges collapse and splits happen, which involves computing the velocity of the new kinetic vertex. The velocity is such that the wavefront vertex remains on the intersection of the wavefront edges as they propagate, i.e., it points along the angular bisector of the incident edges and has the appropriate speed. However, what is the velocity of a wavefront vertex between opposing wavefront edges that

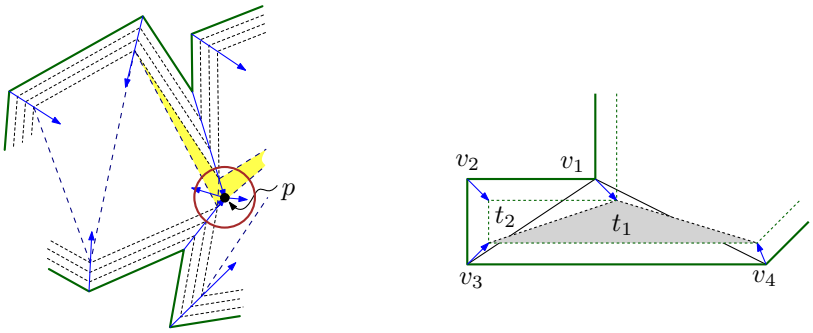


Figure 5 Left: The two shaded triangles are witnesses for an event at p, where two reflex vertices move into each other. Right: Infinitely fast vertices get created when the triangles collapse. If t_1 is processed first, an infinitely fast vertex is created at the locus of v_1 moving to the left, and one at the locus of v_2 and v_3 moving to the right, both incident to t_2. If t_2 is processed first, only one infinitely fast vertex is created at the locus of v_2 and v_3 moving to the right.

overlap and span an angle of zero? The limit, as the angle approaches zero, is infinite speed with a direction along the wavefront edges. Therefore, in our implementation we have the concept of infinitely fast vertices. These get created when opposing wavefront edges become incident and share a vertex; cf. Figure 5, right. Whenever we create an infinitely fast vertex, the triangle that is incident to this vertex is processed next as we consider it to have an event immediately. If more than one triangle is incident to an infinitely fast wavefront vertex v, then we pick the one that has the shorter constraint incident to v.

During the wavefront propagation, instances may occur where three or more wavefront vertices become collinear and remain so in the course of the propagation. If these vertices are adjacent on the convex hull, then the determinant check for whether a vertex leaves the convex hull would show a zero at all times. When encountering an always-zero determinant, depending on the specific configuration, scheduling an event for "right now" may resolve the issue, e.g., by flipping a triangulation diagonal such that the three vertices involved no longer are incident to the same kinetic triangles. If three vertices on the convex hull are collinear, then such a flip would not be advisable because removing the middle vertex from the convex hull (and keeping it on the books as a non-convex-hull vertex) would cause a bounded triangle with an always-zero determinant.

Collapse time and order of events. While the actual collapse time is one of the roots of a quadratic polynomial, solving quadratics is not always necessary. Avoiding root finding for quadratic polynomials will increase accuracy when working with limited-precision data types, and it will result in less complex expression trees when working with exact numbers as provided by CORE's `CORE::Expr`. In particular, it will avoid the computation of another square root. Hence, we try to employ geometric knowledge derived from local combinatoric properties as much as this is possible. For instance, a triangle with two incident wavefront edges will never see a flip event: If it sees an event at all, it will be the collapse of either one or both of its wavefront edges. (In the latter case the entire triangle collapses.)

Triangles with exactly one incident wavefront edge can encounter all three types of events – edge, split, and flip events – but we can determine each such event without computing the roots of a determinant: The times of split and flip events can be found by considering the distance between the vertex opposite the wavefront edge to the supporting line of the

wavefront edge. This distance is linear in time and when it passes through zero, we either have a flip or split event as the vertex comes to lie on the supporting line of the wavefront edge. Likewise, an unbounded triangle witnessing a wavefront edge collapse needs to consider only the linear function that models the wavefront edge's length. Unbounded triangles will not need to witness collapses of a triangulation diagonal because this event will be witnessed by the neighboring (bounded) triangle. Finding the time when a vertex leaves the convex hull may require determining the roots of a quadratic polynomial if no incident wavefront edge is on the convex hull. Otherwise, we employ the distance-to-supporting-line method again. The only bounded triangles for which we need to consider quadratic polynomials are triangles not incident to a wavefront edge: These triangles can cause flip events.

We process the events in order of their event times, breaking ties based on the event type: Events with infinitely fast vertices come first, then the other real events, and finally we process flip events and convex-hull-update events. Ties among flip events are broken by handling the event with the longest edge first. Since our implementation uses exact arithmetic provided by CORE, all these comparisons can be done exactly. Event handling generally is straightforward and consists of updating the kinetic triangulation to remain valid after the event, for real events dropping collapsed triangles, and stopping kinetic vertices and creating new ones. (Recall that we know the type of event.)

Once no event is left in the queue with a finite event time, the list of kinetic vertices with their start and stop times and positions yield the arcs of the final straight skeleton. As post processing, we construct a CGAL DCEL (doubly-connected edge list) structure from the kinetic vertices. Each kinetic vertex and each input segment corresponds to two DCEL half-edges, and each DCEL face is incident to one input segment or terminal input vertex.

Pre-emptive flips. Flip events are internal events of the kinetic triangulation, and as such they are not directly needed for the wavefront propagation process. They are always the result of a reflex vertex moving over the opposite diagonal in a triangle. As already outlined, flip events can lead to loops and thus require special handling. We tested whether we can reduce the number of flip events by restructuring triangles as soon as they are created, either initially or as the result of another event, rather than wait for a triangle to collapse before we do a flip. The very simple approach that we chose was to immediately flip a triangulation diagonal that is incident to two convex vertices if the flip would make it incident to a reflex vertex, in the assumption that this reflex vertex is then less likely to cause a flip event later.

We benchmarked this approach and found that it results in fewer flip events for about 98 % of our test runs. In the worst example we got 4 % more flip events, and in the best case flip events were reduced by slightly more than 10 %, with a median change of 3 % fewer flips. The benefit is less clear-cut when looking at runtime improvements, though. We noticed an improvement of about 1 % in the median and a surprising 50 % in the best case, but also a 30 % slowdown in the worst case. Still, pre-emptive flips are the default set-up for SURFER2.

Component-based priority queues. Closed input loops partition the plane into connected components. (And in practice many PSLGs contain such loops.) The basic algorithm deals with events, witnessed by triangle collapses, without any consideration of the component the triangle is in. But the straight skeleton within one component is entirely independent of the straight skeleton within any other component: The different wavefronts never interact during their respective propagation processes. Hence, it would suffice to ensure that events are ordered correctly within each component but we could handle events from different components independently. Of course, in the perfect world of the Real RAM model it does not matter whether k items are to be stored in one or a few comparison-based priority queues, since handling k items takes $\Theta(k \log k)$ time one way or the other.

Once we talk about an implementation the Real RAM model is no longer applicable, though. Our implementation uses CORE's `CORE::Expr` number type as shipped with CGAL. This gives us the luxury of actually having exact arithmetic, thus making it easy to know which events happen simultaneously. The significant price to be paid is that comparisons are no longer unit-cost: Each CORE expression variable carries with it its expression tree as the entire history of how it was derived, as well as a numerical approximation with error bounds. Whenever we need to compare two numbers, CORE can calculate more digits should this be needed to ascertain the exact relationship. This takes extra time and memory. In general, such comparisons are near-instantaneous when the values are (sufficiently) different. However, they can take a very long time if the numbers are actually equal. (Profiling showed that one such comparison may even take many seconds.)

To avoid comparing event times from different components we assign an integer identification number to each component and use this number as a secondary key for the event ordering in the priority queue. We added instrumentation to our implementation to count how often we actually compare equal event times during the wavefront propagation process. For random input in general position, the likelihood of events in different parts of the plane to happen at the same time is zero. Our tests confirmed that for random input there is nothing to gain by per-component event ordering. However, avoiding comparing event times from different components may yield substantial savings for some input classes: A reduction of the number of simultaneous events by a quarter may result in a reduction of the runtime by a factor of five in extreme cases! Savings can be expected to occur for inputs with vertices on integer grids, or, e.g., if two components both contain tight polygonal approximations of circular arcs with the same radius.

6 Results

Setup. All runtime tests were carried out on a 2015 Intel Core i7-6700 CPU. For most of our tests, memory consumption was limited to 10 GiB. The codes were compiled with `clang++`, version 7.0.1, against CGAL 5.0 except where stated otherwise.

The default setting for CGAL's straight-skeleton package [9] is to use the exact predicates but inexact constructions kernel that ships with CGAL. The use of this kernel ensures that the straight skeleton computed is combinatorially correct, even if the locations of the nodes need not be correct. CGAL's straight skeleton package can also be run with the exact predicates and exact constructions kernel. However this causes the runtimes to increase by a factor of roughly 100. Our codes, MONOS and SURFER2, use the CGAL exact predicates and exact constructions with square-root kernel by default, as we construct wavefront vertices with velocities, and these computations involve square roots.

We tested both CGAL and SURFER2 on many different classes of polygons. Our test data consists of real-world (multiply-connected) polygons as well as of synthetic data generated by RPG [3] and similar tools provided by the Salzburg Database [11]. For a polygon that has holes we used CGAL's straight-skeleton code that supports holes. Otherwise we used the implementation which only supports simple polygons.

Additionally, we tested both of our codes, MONOS and SURFER2, with large monotone polygons, for up to 10^6 vertices. (We did not run CGAL on these inputs due to memory constraints.) Given the lack of a sufficiently large number of monotone real-world inputs, we used RPG [3] to automatically generate thousands of monotone input polygons. We also ran SURFER2 on hundreds of real-world PSLGs. Most of our data came from GIS sources and

represents road and river networks, contour lines and the like. The runtimes on those inputs are quite comparable to the runtimes for polygons. That is, the test results presented here are also representative for SURFER2's performance on real-world data.

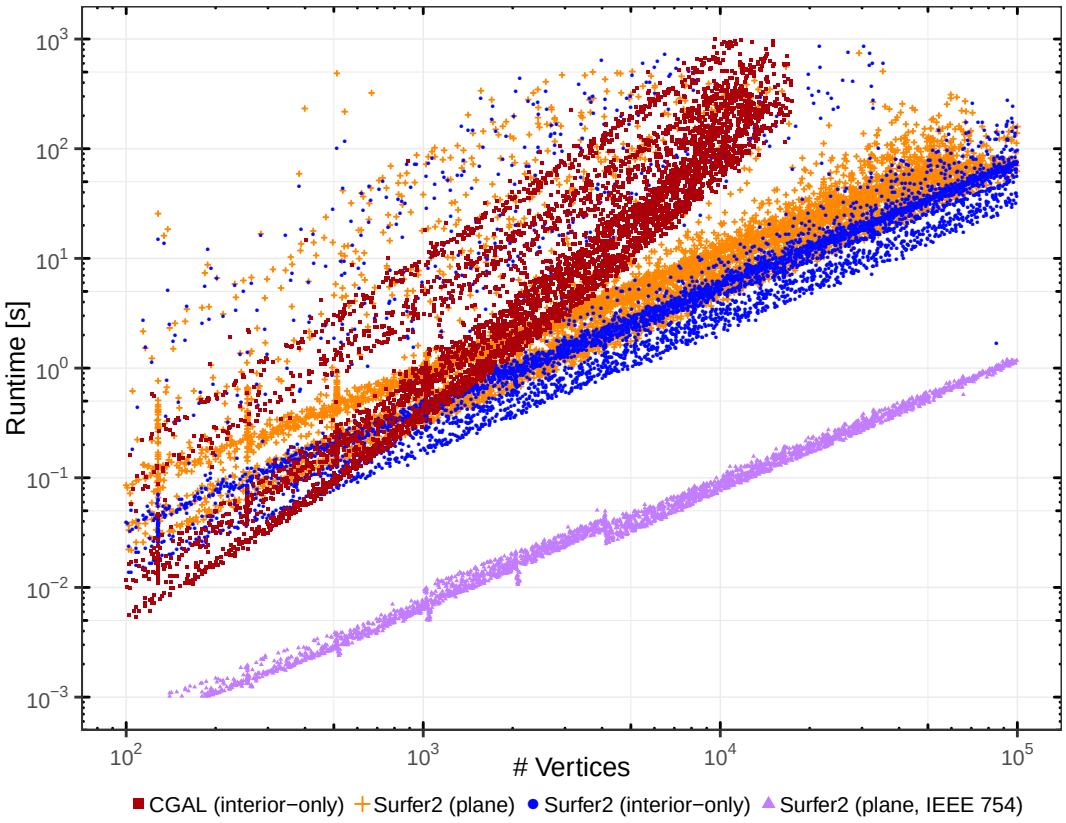


Figure 6 Runtimes of CGAL's code and different variants of SURFER2.

Runtimes and memory consumption. While CGAL's code computes the straight skeleton either in the interior or the exterior of a polygon, SURFER2 can do both in one run because it treats a polygon as a PSLG. But SURFER2 can also be restricted to just the interior or the exterior of a polygon. Hence, for the plot in Figure 6 we ran SURFER2 twice, once applied to the entire plane and once for just the interiors of the polygons that were also handled by CGAL. Our tests make it evident that SURFER2 is significantly faster than CGAL for a large fraction of the inputs. In particular, its runtime seems to exhibit an $n \log n$ growth, compared to the clearly quadratic increase of the runtime of CGAL's code. Figure 7 shows that SURFER2's memory consumption grows (mostly) linearly while CGAL's code requires a clearly quadratic amount of memory.

The results for CGAL's straight skeleton package were to be expected because (at least back in 2010) it computed potential split events for each pair of reflex vertex and wavefront edge [17, Section 2.5.4]. Indeed, tests carried out in 2010 indicated that it requires $\mathcal{O}(n^2 \log n)$ time and $\Theta(n^2)$ space for n-vertex polygons, as discussed in [18]. Our test results suggest that the same algorithm and implementation are applied in the current CGAL 5.0, which is the version used in our tests. The theoretical upper bound on the runtime complexity of SURFER2 is given by $O(n^3 \log n)$. Thus, its very decent practical performance is noteworthy.

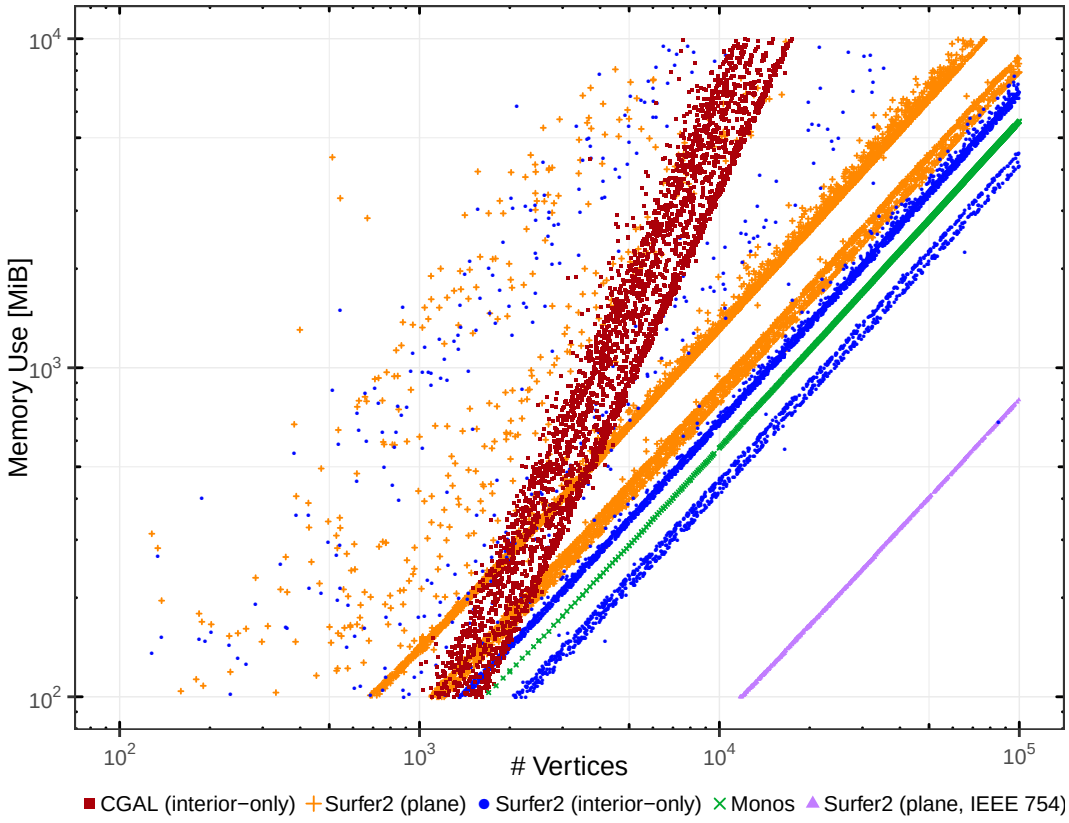


Figure 7 Memory use of CGAL, SURFER2 variants, and MONOS.

We also ran SURFER2 with IEEE 754 double as a number type instead of CORE's `CORE::Expr`. We admit, though, that SURFER2 is not (yet) fully prepared to deal with finite-precision arithmetic as it assumes that the order of the events is given reliably. With sufficient engineering effort, however, such requirements can be relaxed and the code could be made to run reliably even with finite-precision arithmetic. Hence, the current version of SURFER2 with the IEEE 754 backend works well on input in general position, but fails for some inputs when several events happen at exactly the same time and position. Nevertheless, we wanted to see the runtime and memory characteristics in order to get a better impression of the practical costs of using `CORE::Expr`: Therefore the plots in Figures 6 and 7 also show results for SURFER2 with IEEE 754 arithmetic.

The $\mathcal{O}(n \log n)$ bound for the complexity of MONOS is apparent in Figure 8, left. Given that MONOS is a special-purpose code designed specifically for handling monotone polygons, it had to be expected that it outperforms SURFER2 consistently for such input.

Dependence on input characteristics. There are outliers both in the runtime as well as in the memory consumption of SURFER2 which are clearly visible in Figures 6 and 7. (To a lesser extent this noise is visible for CGAL, too.) Given the fact that such outliers do not show up for the IEEE 754-based version of SURFER2, there is reason to assume that this behavior is not intrinsic to SURFER2's algorithm but that it has its roots in the use of exact arithmetic. To probe this issue further we investigated which input classes trigger these outliers. Figure 9 shows the runtimes of both codes for three different inputs classes.

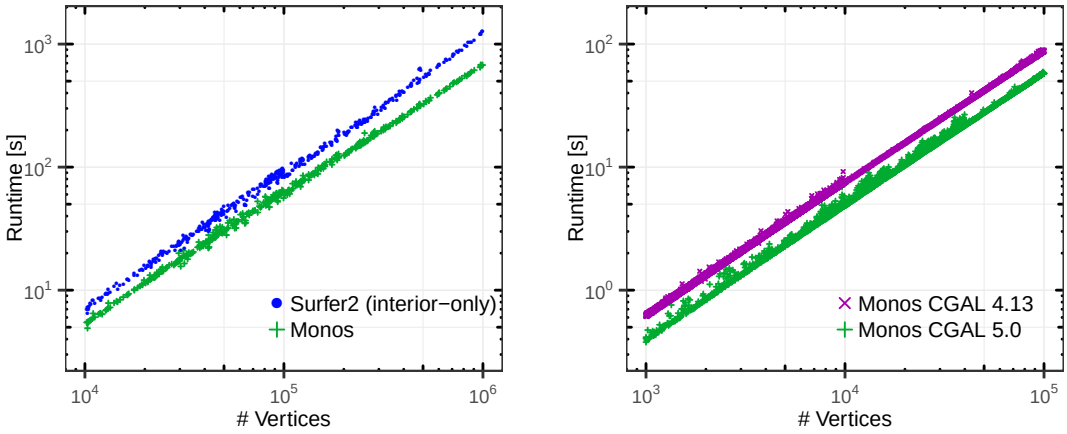


Figure 8 Monos vs. Surfer2 on monotone inputs, and Monos with Cgal 4 vs. Cgal 5.

The class of input that was most time-consuming to handle for all implementations were our random octagonal polygons. All polygons out of this group have interior angles that are multiples of 45°. We were surprised to see that axis-parallel input is not troublesome per se. The key difference between both groups of polygons is given by the fact that all octagonal polygons have their vertices on an integer grid while the vertices of the orthogonal polygons are (random) real numbers. Hence, for the octagonal polygons many events tend to happen at exactly the same time when opposite edges become incident in different parts of the polygon. It is apparent that the proper time-wise ordering of these events incurs a significant cost if `CORE::Expr` is used. The presence of parallel edges does not automatically increase the runtime of Surfer2, though. The likely explanation is that many events cause infinitely-fast vertices, which are then, in turn, easy to sort as they are handled immediately. For comparison purposes we ran the code on random simple polygons as a third class of input, with random vertex coordinates. Having any kind of co-temporal event is highly unlikely for those inputs, as are infinitely-fast vertices.

The excessive amounts of time consumed by ordering simultaneous events became even more apparent when we studied the impact of multiplicative weights on Surfer2's runtime. As expected, a difference in the timings for weighted and unweighted random polygons was hardly noticeable. For our randomly weighted octagonal polygons we expected reduced runtimes and fewer outliers even when using exact arithmetic, due to very few truly simultaneous events. And, indeed, this was confirmed impressively by our tests; see Figure 9.

Experiences with CGAL. While running our tests, we also compared Cgal versions 4.13 and 5.0, as the latter was released only recently. We witnessed an improvement in the performance of our codes for the newer version; see Figure 8, right.

We also spent a considerable amount of time on debugging Surfer2, just to find out in the end that some of our inputs trigger a bug in the `CORE::Expr` number type. This bug occurs for both Cgal 4.13 and Cgal 5.0. In specific reproducible cases the floating-point filter fails. As a result predicates are evaluated incorrectly, such as Cgal/Core claiming that $0.49770 < 0.01047$. We have reported this bug to the Cgal project (Bug#4296).

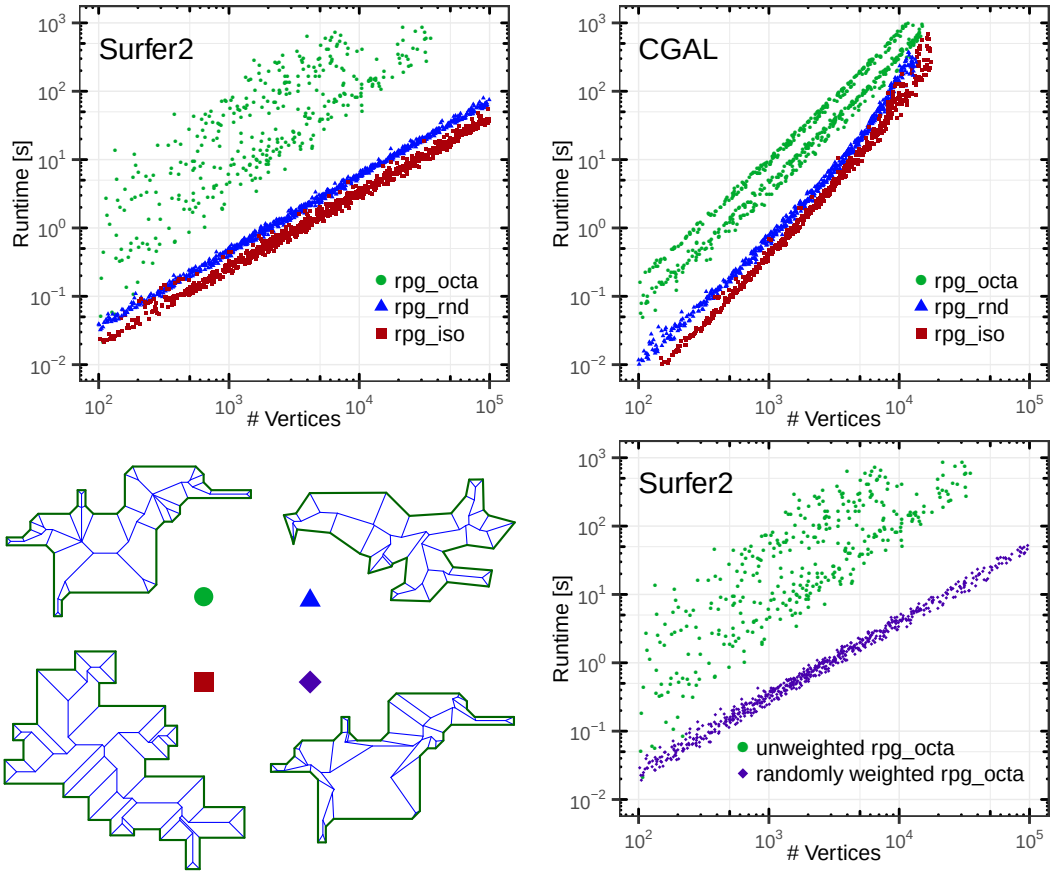


Figure 9 Top: The effect of different input classes on the runtime of SURFER2 and on CGAL. Bottom-left: Samples for different input classes (left to right, top to bottom): Octagonal input (on integer grid), random polygon, orthogonal polygon not on integer grid, and weighted octagonal input. Bottom-right: SURFER2 runtimes for unweighted and randomly weighted octagonal input.

7 Discussion and conclusion

Our implementations MONOS and SURFER2 show that an $\mathcal{O}(n \log n)$ runtime can be achieved for the computation of straight skeletons of n-vertex monotone polygons and PSLGs, even if the edges of the PSLG are weighted by positive multiplicative weights. Engineering considerations do not improve the asymptotic behavior but help to improve the practical efficiency. Hence, our implementational efforts can be regarded as successful.

But our tests also make it evident that the use of the `CORE::Expr` number type forces one to abandon the concept of unit-cost comparisons: Multiple events that occur simultaneously have a significant impact on the practical runtime of SURFER2 if the `CORE::Expr` number type is used, while no impact is visible for the standard IEEE 754 arithmetic. It is obvious that it would pay off to detect groups of simultaneous events in a way that does not involve comparing the event times, as it is done by our current implementation. As discussed, a per-component ordering of the events in different priority queues reduces the burden that simultaneous events put on the efficiency. In our current version, we apply per-component computations only for different components induced by the input PSLG. If we could cheaply detect the splitting of a wavefront into two components and afterwards use new component

numbers in future updates of the priority queue, then we might be able to exploit this idea even further. How to carry out a dynamic maintenance of the component information with little computational overhead is on our agenda for future R&D work.

A second avenue for practical improvements is given by a fine tuning of the priority queue used for storing future events. It does not come as a surprise that not all real events computed and registered in the priority queue do indeed correspond to topological changes of the actual wavefront at the times when they are predicted to occur. As a consequence, we spend time on ordering future events even if only a fraction of them will actually happen. Hence, our tests suggest that it might be better to use some form of lazy priority queue. (Standard lazy binomial heaps or Fibonacci heaps still would result in $\mathcal{O}(\log n)$ many comparisons for a delete-min operation.) After all, we do not really require a proper min-heap that obeys the heap property on every layer. Rather, all we need to know is the next event.

An alternative to constructing collapse times and then comparing them is developing predicates for ordering triangle collapses. However, this might be more involved than it appears at first glance: A triangle consists of kinetic vertices that move along the (weighted) bisectors of pairs of input edges, which, in general, uniquely define the position of their wavefront vertex as a function of time. However, in the case of parallel, collinear wavefront edges stemming from one or more parallel, collinear input edges, their bisector is not a uniquely defined one-dimensional line. Rather, the exact trajectory of the kinetic vertex depends on the wavefront propagation that has happened so far; see Figure 10. Nevertheless, having and using exact predicates and resorting to explicit exact computations and comparisons of the collapse times only in degenerate cases might speed up computing the straight skeleton.

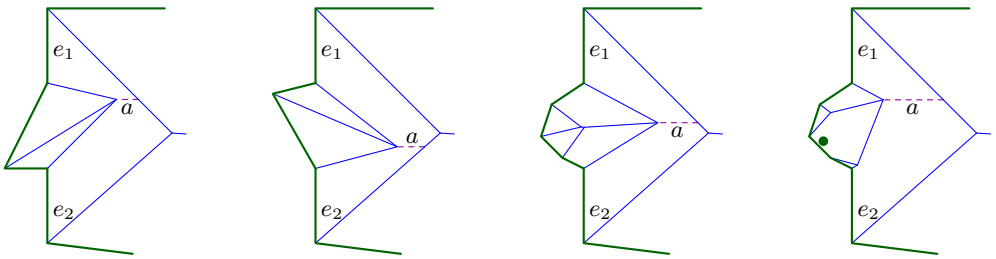


Figure 10 The (supporting line of) straight skeleton arc a (dashed, purple) depends not only on e_1, e_2 but also on the weights (e.g., •), positions, and orientations of other, non-incident input edges.

8 Source code

Our source codes are provided on GitHub and can be used freely under the GPL(v3) license: `https://github.com/cgalab/monos` and `https://github.com/cgalab/surfer2`.

References

1 Oswin Aichholzer and Franz Aurenhammer. Straight Skeletons for General Polygonal Figures in the Plane. In *Voronoi's Impact on Modern Sciences II*, volume 21, pages 7–21. Institute of Mathematics of the National Academy of Sciences of Ukraine, 1998. doi:10.1007/3-540-61332-3_144.

2 Oswin Aichholzer, Franz Aurenhammer, David Alberts, and Bernd Gärtner. A Novel Type of Skeleton for Polygons. *Journal of Universal Computer Science*, 1(12):752–761, 1995. doi:10.1007/978-3-642-80350-5_65.

3 Thomas Auer and Martin Held. Heuristics for the Generation of Random Polygons. In *Proceedings of the 8th Canadian Conference on Computational Geometry (CCCG)*, pages 38–44, 1996.

4 Therese Biedl, Martin Held, Stefan Huber, Dominik Kaaser, and Peter Palfrader. A Simple Algorithm for Computing Positively Weighted Straight Skeletons of Monotone Polygons. *Information Processing Letters*, 115(2):243–247, 2015. doi:10.1016/j.ipl.2014.09.021.

5 Therese Biedl, Martin Held, Stefan Huber, Dominik Kaaser, and Peter Palfrader. Weighted Straight Skeletons in the Plane. *Computational Geometry: Theory and Applications*, 48(2):120–133, 2015. doi:10.1016/j.comgeo.2014.08.006.

6 Therese Biedl, Stefan Huber, and Peter Palfrader. Planar Matchings for Weighted Straight Skeletons. In *Proceedings of the 25th International Symposium on Algorithms and Computation (ISAAC)*, pages 117–127, 2014. doi:10.1007/978-3-319-13075-0_10.

7 Ulrik Brandes, Markus Eiglsperger, Ivan Herman, Michael Himsolt, and M. Scott Marshall. GraphML Progress Report Structural Layer Proposal. In *Proceedings of the 9th International Symposium on Graph Drawing*, pages 501–512, 2001. URL: http://graphml.graphdrawing.org/.

8 Fernando Cacciola. Private email correspondence, 2010.

9 Fernando Cacciola. 2D Straight Skeleton and Polygon Offsetting. In *CGAL User and Reference Manual*. CGAL Editorial Board, 5.0 edition, 2019. URL: https://doc.cgal.org/5.0/Manual/packages.html#PkgStraightSkeleton2.

10 Siu-Wing Cheng, Liam Mencel, and Antoine Vigneron. A Faster Algorithm for Computing Straight Skeletons. *ACM Transactions on Algorithms*, 12(3):44:1–44:21, 2016. doi:10.1145/2898961.

11 Günther Eder, Martin Held, SteinÞór Jasonarson, Philipp Mayer, and Peter Palfrader. On Generating Polygons: Introducing the Salzburg Database. In *Proceedings of the 36th European Workshop on Computational Geometry*, pages 75:1–7, 2020.

12 Günther Eder and Martin Held. Computing Positively Weighted Straight Skeletons of Simple Polygons based on Bisector Arrangement. *Information Processing Letters*, 132:28–32, 2018. doi:10.1016/j.ipl.2017.12.001.

13 David Eppstein and Jeff Erickson. Raising Roofs, Crashing Cycles, and Playing Pool: Applications of a Data Structure for Finding Pairwise Interactions. *Discrete & Computational Geometry*, 22(4):569–592, 1999. doi:10.1145/276884.276891.

14 Petr Felkel and Š. Obdržálek. Straight Skeleton Implementation. In *Proceedings of the 14th Spring Conference on Computer Graphics (SCCG)*, pages 210–218, 1998.

15 Martin Held and Peter Palfrader. Straight Skeletons with Additive and Multiplicative Weights and Their Application to the Algorithmic Generation of Roofs and Terrains. *Computer-Aided Design*, 92(1):33–41, 2017. doi:10.1016/j.cad.2017.07.003.

16 Martin Held and Peter Palfrader. Skeletal Structures for Modeling Generalized Chamfers and Fillets in the Presence of Complex Miters. *Computer-Aided Design and Applications*, 16(4):620–627, 2019. doi:10.14733/cadaps.2019.620-627.

17 Stefan Huber. *Computing Straight Skeletons and Motorcycle Graphs: Theory and Practice*. Shaker Verlag, 2012. ISBN 978-3-8440-0938-5.

18 Stefan Huber and Martin Held. Theoretical and Practical Results on Straight Skeletons of Planar Straight-line Graphs. In *SoCG '11: Proceedings of the twenty-seventh Annual Symposium on Computational Geometry*, 2011. doi:10.1145/1998196.1998223.

19 Tom Kelly and Peter Wonka. Interactive Architectural Modeling with Procedural Extrusions. *ACM Transactions on Graphics*, 30(2):14:1–14:15, April 2011. doi:10.1145/1944846.1944854.

20 Lutz Kettner. Halfedge Data Structures. In *CGAL User and Reference Manual*. CGAL Editorial Board, 5.0 edition, 2019. URL: https://doc.cgal.org/5.0/Manual/packages.html#PkgHalfedgeDS.

21 Peter Palfrader, Martin Held, and Stefan Huber. On Computing Straight Skeletons by Means of Kinetic Triangulations. In *Proceedings of the 20th Annual European Symposium on Algorithms (ESA)*, pages 766–777, 2012. doi:10.1007/978-3-642-33090-2_66.

22 The CGAL Project. *CGAL User and Reference Manual*. CGAL Editorial Board, 5.0 edition, 2019. URL: https://doc.cgal.org/5.0/Manual/packages.html.

23 Antoine Vigneron and Lie Yan. A Faster Algorithm for Computing Motorcycle Graphs. *Discrete & Computational Geometry*, 52(3):492–514, 2014. doi:10.1007/00454-014-9625-2.

24 Evgeny Yakersberg. *Morphing Between Geometric Shapes Using Straight-Skeleton-Based Interpolation*. MSc thesis, CS Dept., Technion, Haifa, Israel, 2004.

25 Mariette Yvinec. 2D Triangulation. In *CGAL User and Reference Manual*. CGAL Editorial Board, 5.0 edition, 2019. URL: https://doc.cgal.org/5.0/Manual/packages.html#PkgTriangulation2.

Removing Connected Obstacles in the Plane Is FPT

Eduard Eiben
Department of Computer Science, Royal Holloway, University of London, UK
eduard.eiben@rhul.ac.uk

Daniel Lokshtanov
Department of Computer Science, UC Santa Barbara, CA, USA
daniello@ucsb.edu

Abstract

Given two points in the plane, a set of obstacles defined by closed curves, and an integer k, does there exist a path between the two designated points intersecting at most k of the obstacles? This is a fundamental and well-studied problem arising naturally in computational geometry, graph theory, wireless computing, and motion planning. It remains NP-hard even when the obstacles are very simple geometric shapes (e.g., unit-length line segments). In this paper, we show that the problem is fixed-parameter tractable (FPT) parameterized by k, by giving an algorithm with running time $k^{O(k^3)}n^{O(1)}$. Here n is the number connected areas in the plane drawing of all the obstacles.

2012 ACM Subject Classification Theory of computation → Parameterized complexity and exact algorithms; Theory of computation → Computational geometry; Theory of computation → Design and analysis of algorithms; Theory of computation → Graph algorithms analysis

Keywords and phrases parameterized complexity and algorithms, planar graphs, motion planning, barrier coverage, barrier resilience, colored path, minimum constraint removal

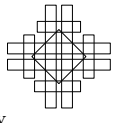
Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.39

Related Version A full version of the paper is available at `https://arxiv.org/abs/2002.01218`.

1 Introduction

In the Connected Obstacle Removal problem we are given as input a source point s and a target point t in the plane, and our goal is to move from the source to the target along a continuous curve. The catch is that the plane is also littered with obstacles – each obstacle is represented by a bounded closed connected subset of the plane, and the goal is to get from the source to the target while intersecting as few of the obstacles as possible. Equivalently we can ask for the minimum number of obstacles that have to be removed so that one can move from s to t without touching any of the remaining one. The problem has a wealth of applications, and has been studied under different names, such as Barrier Coverage or Barrier Resilience in networking and wirless computing [1, 3, 15, 16, 17, 18], or Minimum Constraint Removal in planning [7, 10, 13, 14]. The problem is NP-hard even when the obstacles are restricted to simple geometric shapes, such as line segments (e.g., see [1, 17, 18]). On the other hand, for unit-disk obstacles in a restricted setting, the problem can be solved in polynomial time [16]. Whether Connected Obstacle Removal can be solved in polynomial time for unit-disk obstacles remains open. The problem is known to be APX-hard [2], and also no factor $o(n)$-approximation is known. For restricted inputs (such as unit disc or rectangle obstacles) better approximation algorithms are known [2, 3].

In this paper we approach the general Connected Obstacle Removal problem from the perspective of parameterized algorithms (see [4] for an introduction). In particular it is easy to see that the problem is solvable in time $n^{k+O(1)}$ if the solution curve is to intersect at most k obstacles. Here n is the number of connected regions in the plane defined by



the simultaneous drawing of all the obstacles. If k is considered a constant then this is polynomial time, however the exponent of the polynomial grows with the parameter k. A natural problem is whether the algorithm can be improved to a *Fixed Parameter Tractable* (FPT) one, that is an algorithm with running time $f(k)n^{O(1)}$. In this paper we give the first FPT algorithm for the problem. Our algorithm substantially generalizes previous work by Kumar et al. [16] as well as the first author and Kanj [8].

▶ **Theorem 1.1.** *There is an algorithm for* Connected Obstacle Removal *with running time* $k^{O(k^3)}n^{O(1)}$.

Our arguments and the relation between our results and previous work are more conveniently stated in terms of an equivalent graph problem, which we now discuss. Given a graph G, a set $C \subset \mathbb{N}$ (interpreted as a set of *colors*), and a function $\chi : V(G) \to 2^C$ that assigns a set of colors to every vertex of v, a vertex set S *uses* the color set $\bigcup_{v \in S} \chi(v)$. In the Colored Path problem input consists of G, s, t, χ and k, and the goal is to find an $s - t$ path P that uses at most k colors. Note, that to obtain computational results for the problem, we assume that the regions and intersections formed by the obstacles can be computed and enumerated in polynomial time. We do not assume that the obstacles are simply-connected, however we assume that the boundary of each obstacle is union of finite number of disjoint simple closed curves. We may also assume that s and t are not on a boundary of any obstacle. It is easy to see that Connected Obstacle Removal reduces to Colored Path (see also Figure 1). In particular, we let the vertices of G be the connected components, called regions, of the plane minus the union of the boundaries of the obstacles and we put an edge between two vertices if their boundaries have a curve of positive length in common. The color set is exactly the set of obstacles and the color set of a vertex is the set of obstacles containing the region associated with the vertex. The equivalence of the instances is rather straightforward. One way, the sequence of (closures of) regions a path in the plane intersects when traversing it from s to t, determines an s-t walk in G. On the other hand, we can easily define an s-t path in plane from a path in the graph that intersects precisely the regions associated with the vertices of the path and crosses between consecutive regions in the common boundary. Of course, reducing from Connected Obstacle Removal in this way can not produce all possible instances of Colored Path: the graph G is always a planar graph, and for every color $c \in C$ the set $\chi^{-1}(c) = \{v \in V(G) : c \in \chi(v)\}$ induces a connected subgraph of G. We shall denote the Colored Path problem restricted to instances that satisfy the two properties above by Colored Path*. With these additional restrictions it is easy to reduce back (we can just take the dual of G and let each obstacle be the closure of the union of the faces containing the associated color), and therefore Connected Obstacle Removal and Colored Path* are, for all practical purposes, different formulations of the same problem.

Related Work in Parameterized Algorithms, and Barriers to Generalization. Korman et al. [15] initiated the study of Connected Obstacle Removal from the perspective of parameterized complexity. They show that Connected Obstacle Removal is FPT parameterized by k for unit-disk obstacles, and extended this result to similar-size fat-region obstacles with a constant *overlapping number*, which is the maximum number of obstacles having nonempty intersection. Eiben and Kanj [8] generalize the results of Korman et al. [15] by giving algorithms for Colored Path* with running time $f(k,t)n^{O(1)}$ and $g(k,\ell)n^{O(1)}$ where t is the treewidth of the input graph G, and ℓ is an upper bound on the number of *vertices* on the shortest solution path P.

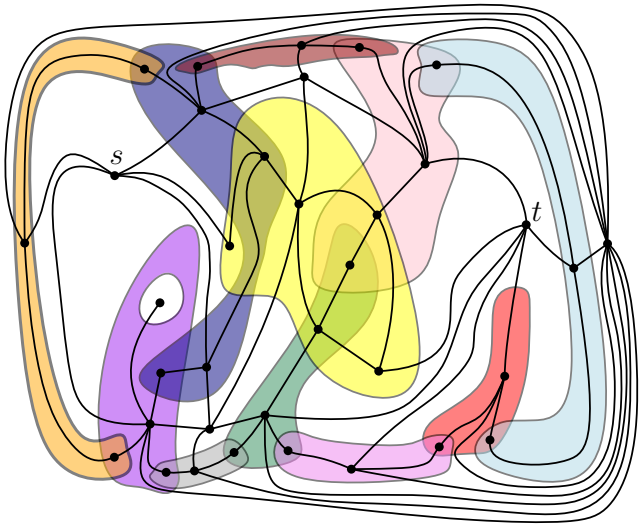


Figure 1 The figure shows an instance of Connected Obstacle Removal and the graph G of an equivalent instance of Colored Path. Every obstacle corresponds to a color, and the color set of a vertex are the obstacles that contain the vertex in their interior.

Eiben and Kanj [8] leave open the existence of an FPT algorithm for Colored Path* - Theorem 1.1 provides such an algorithm. Interestingly, Eiben and Kanj [8] also show that if an FPT algorithm for Colored Path* were to exist, then in many ways it would be the best one can hope for. More concretely, for each of the most natural ways to generalize Thoerem 1.1, Eiben and Kanj [8] provide evidence of hardness. Specifically, the Colored Path* problem imposes two constraints on the input – the graph G has to be planar and the color sets need to be connected. Eiben and Kanj [8] show that lifting *either one* of these constraints results in a W[1]-hard problem (i.e. one that is not FPT assuming plausible complexity theoretic hypotheses) *even* if the treewidth of the input graph G is a small constant, *and* the length of the a solution path (if one exists) is promised to be a function of k.

Algorithms that determine the existence of a path can often be adapted to algorithms that find the *shortest* such path. Eiben and Kanj [8] show that for Colored Path*, this can not be the case. Indeed, they show that an algorithm with running time $f(k)n^{O(1)}$ that given a graph G, color function χ and integers k and ℓ determines whether there exists an $s-t$ path of length at most ℓ using at most k colors, would imply that FPT = W[1]. Thus, unless FPT = W[1] the algorithm of Theorem 1.1 can not be adapted to an FPT algorithm that finds a *shortest* path through k obstacles.

1.1 Overview of the Algorithm

The naive $n^{k+O(1)}$ time algorithm enumerates all choices of a set S on at most k colors in the graph, and then decides in polynomial time whether S is a feasible color set, in other words whether there exists a solution path that only uses colors from S. At a very high level our algorithm does the same thing, but it only computes sets S that can be obtained as a union of colors of at most k vertices and additionally it performs a pruning step so that not all n^k choices for S are enumerated.

In FPT algorithms such a pruning step is often done by clever *branching*: when choosing the i'th vertex defining S one would show that there are only $f(k)$ viable choices that could possibly lead to a solution. We are not able to implement a pruning step in this way. Instead, our pruning step is inspired by algorithms based on representative sets [12].

In particular, our algorithm proceeds in k rounds. In each round we make a family $\mathcal{P}_i$ of color sets of size at most i, with the following properties. First, $|\mathcal{P}_i| \leq k^{O(k^3)}n^{O(1)}$. Second, if there exists a solution path, then there exists a solution such that the set containing the *first i visited* colors is in $\mathcal{P}_i$.

In each round i the algorithm does two things: first it *extends* the already computed families $\mathcal{P}_0, \ldots \mathcal{P}_{i-1}$ by going over every set $S \in \bigcup_{j=0}^{i-1} \mathcal{P}_j$ and every vertex $v \in V(G)$ and inserting $S \cup \chi(v)$ into the new family $\hat{\mathcal{P}}_i$ if $|S \cup \chi(v)| = i$. It is quite easy to see that $\hat{\mathcal{P}}_i$ satisfies the second property - however it is a factor of n larger than the union of previous $\mathcal{P}_j$'s. If we keep extending $\hat{\mathcal{P}}_i$ in this way then after a super-constant number of steps we will break the first requirement that the family size should be at most $k^{O(k^3)}n^{O(1)}$. For this reason the algorithm also performs an *irrelevant set* step: as long as $\hat{\mathcal{P}}_i$ is "too large" we show that one can identify a set $S \in \hat{\mathcal{P}}_i$ that can be removed from $\hat{\mathcal{P}}_i$ without breaking the second property. We repeat this irrelevant set step until $\hat{\mathcal{P}}_i$ is sufficiently small. At this point we declare that this is our i'th family $\mathcal{P}_i$ and proceed to step $i+1$.

The most technically involved part of our argument is the proof of correctness for the irrelevant set step, see Section 3.3. This argument crucially exploits the structure of a large set of paths in a planar graph that start and end in the same vertex.

2 Preliminaries

For integers n, m with $n \leq m$, we let $[n, m] := \{n, n+1, \ldots, m\}$ and $[n] := [1, n]$. Let $\mathcal{F}$ be a family of subsets of a universe U. A *sunflower* in $\mathcal{F}$ is a subset $\mathcal{F}' \subseteq \mathcal{F}$ such that all pairs of elements in $\mathcal{F}'$ have the same intersection.

▶ **Lemma 2.1** ([9, 11]). *Let $\mathcal{F}$ be a family of subsets of a universe U, each of cardinality exactly b, and let $a \in \mathbb{N}$. If $|\mathcal{F}| \geq b!(a-1)^b$, then $\mathcal{F}$ contains a sunflower $\mathcal{F}'$ of cardinality at least a. Moreover, $\mathcal{F}'$ can be computed in time polynomial in $|\mathcal{F}|$.*

We assume familiarity with the basic notations and terminologies in graph theory and parameterized complexity. We refer the reader to the standard books [4, 5, 6] for more information on these subjects.

Graphs. All graphs in this paper are simple (i.e., loop-less and with no multiple edges). Let G be an undirected graph. For an edge $e = uv$ in G, *contracting* e means removing the two vertices u and v from G, replacing them with a new vertex w, and for every vertex y in the neighborhood of v or u in G, adding an edge wy in the new graph, not allowing multiple edges. Given a connected vertex-set $S \subseteq V(G)$, *contracting* S means contracting the edges between the vertices in S to obtain a single vertex at the end. For a set of edges $E' \subseteq E(G)$, the subgraph of G induced by E' is the graph whose vertex-set is the set of endpoints of the edges in E', and whose edge-set is E'.

A graph is *planar* if it can be drawn in the plane without edge intersections (except at the endpoints). A *plane graph* is a planar graph together with a fixed drawing. Each maximal connected region of the plane minus the drawing is an open set; these are the *faces*.

Let $W_1 = (u_1, \ldots, u_p)$ and $W_2 = (v_1, \ldots, v_q)$, $p, q \in \mathbb{N}$, be two walks such that $u_p = v_1$. Define the *gluing* operation $\circ$ that when applied to W_1 and W_2 produces that walk $W_1 \circ W_2 = (u_1, \ldots, u_p, v_2, \ldots, v_q)$. For a path $P = (v_1, \ldots, v_q)$, $q \in \mathbb{N}$ and $i \in [q]$, we let $\mathbf{pre}(P, v_i)$ be the *prefix* of the P ending at v_i, that is the path $(v_1, v_2, \ldots v_i)$. Similarly, we let $\mathbf{suf}(P, v_i)$ be the *suffix* of the P starting at v_i, that is the path $(v_i, v_{i+1}, \ldots v_q)$.

For a graph G and two vertices $u, v \in V(G)$, we denote by $d_G(u, v)$ the *distance* between u and v in G, which is the length (number of edges) of a shortest path between u and v in G.

Parameterized Complexity. A *parameterized problem* Q is a subset of $\Omega^* \times \mathbb{N}$, where Ω is a fixed alphabet. Each instance of the parameterized problem Q is a pair (x, k), where $k \in \mathbb{N}$ is called the *parameter*. We say that the parameterized problem Q is *fixed-parameter tractable* (FPT) [6], if there is a (parameterized) algorithm, also called an *FPT-algorithm*, that decides whether an input (x, k) is a member of Q in time $f(k) \cdot |x|^{O(1)}$, where f is a computable function. Let FPT denote the class of all fixed-parameter tractable parameterized problems. By *FPT-time* we denote time of the form $f(k) \cdot |x|^{O(1)}$, where f is a computable function and $|x|$ is the input instance size.

Colored Path and Colored Path*. For a set S, we denote by 2^S the power set of S. Let $G = (V, E)$ be a graph, let $C \subset \mathbb{N}$ be a finite set of colors, and let $\chi : V \longrightarrow 2^C$. A vertex v in V is *empty* if $\chi(v) = \emptyset$. A color c *appears on*, or is *contained in*, a subset S of vertices if $c \in \bigcup_{v \in S} \chi(v)$. For $u, v \in V(G)$, $\ell \in \mathbb{N}$, a u-v walk $W = (u = v_0, \ldots, v_r = v)$ in G is *ℓ-valid* if $|\bigcup_{i=0}^{r} \chi(v_i)| \leq \ell$; i.e., if the total number of colors appearing on the vertices of W is at most ℓ. A color c is *connected* in G, or simply *connected*, if $\bigcup_{c \in \chi(v)} \{v\}$ induces a connected subgraph of G. The graph G is *color-connected*, if for every $c \in C$, c is connected in G.

For an instance (G, C, χ, s, t, k) of Colored Path*, if s and t are nonempty vertices, we can remove their colors and decrement k by $|\chi(s) \cup \chi(t)|$ because their colors appear on every s-t path. If afterwards k becomes negative, then there is no k-valid s-t path in G. Moreover, if s and t are adjacent, then the path (s, t) is a path with the minimum number of colors among all s-t paths in G. Therefore, we will assume:

▶ **Assumption 2.2.** *For an instance (G, C, χ, s, t, k) of Colored Path or Colored Path*, we can assume that s and t are nonadjacent empty vertices.*

▶ **Definition 2.3.** Let s, t be two designated vertices in G, and let x, y be two adjacent vertices in G such that $\chi(x) = \chi(y)$. We define the following operation to x and y, referred to as a *color contraction* operation, that results in a graph G', a color function χ', and two designated vertices s', t' in G', obtained as follows:

- G' is the graph obtained from G by contracting the edge xy, which results in a new vertex z;
- $s' = s$ (resp. $t' = t$) if $s \notin \{x, y\}$ (resp. $t \notin \{x, y\}$), and $s' = z$ (resp. $t' = z$) otherwise;
- $\chi' : V(G') \longrightarrow 2^C$ is defined as $\chi'(w) = \chi(w)$ if $w \neq z$, and $\chi'(z) = \chi(x) = \chi(y)$.

G is *irreducible* if there does not exist two vertices in G to which the color contraction operation is applicable.

▶ **Observation 2.4.** *Let G be a color-connected plane graph, C a color set, $\chi : V \longrightarrow 2^C$, $s, t \in V(G)$, and $k \in \mathbb{N}$. Suppose that the color contraction operation is applied to two vertices x, y in G to obtain G', χ', s', t', as described in Definition 2.3. For any two vertices $u, v \in V(G)$ and $p \subseteq C$ there is a u-v walk W with $\chi(W) = p$ in G if and only if there is a u'-v' walk W' with $\chi(W') = p$, where $u' = u$ (resp. $v' = v$) if $u \notin \{x, y\}$ (resp. $v \notin \{x, y\}$), and $u' = z$ (resp. $v' = z$) otherwise.*

3 FPT algorithm for Colored Path*

Given an instance (G, C, χ, s, t, k) and a vertex $v \in V(G)$, we say that a vertex u is *reachable* from a vertex v by a color set $p \subseteq C$ if there exists a v-u path p with $\chi(P) \subseteq p$. Furthermore, we say that a color set $p \subseteq C$ is *v-opening* if there is a vertex $u \in V(G)$ such that u is reachable from v by p, but not by any proper subset of p. Note that necessarily $\chi(v) \subseteq p$. A set of colors p *completes* a v-t walk Q if there is an s-v path P with $\chi(P) = p$, $|p \cup \chi(Q)| \leq k$, and v is the only vertex on Q reachable from s by p. We say p *minimally completes* a v-t

walk Q, if p completes Q and there is no s-v path P' with $\chi(P') \subsetneq p$. We say that an s-t path P is *nice*, if for every prefix $\mathbf{pre}(P, u)$ of P ending at the vertex $u \in V(G)$ there is no s-u path P' with $\chi(P') \subsetneq \chi(\mathbf{pre}(P, u))$.

▶ **Observation 3.1.** *There is a k-valid s-t path if and only if there is a nice k-valid s-t path.*

▶ **Definition 3.2** (k-representation)**.** *Given an instance (G, C, χ, s, t, k) of Colored Path*, a vertex $v \in V(G)$, and two families $\mathcal{P}$ and $\mathcal{P}'$ of s-opening subsets of C of size $\ell \leq k$, we say that $\mathcal{P}'$ k-represents $\mathcal{P}$ w.r.t. v if for every $p \in \mathcal{P}$ and every v-t walk Q such that p minimally completes Q, there is a set $p' \in \mathcal{P}'$ such that $|p' \cup \chi(Q)| \leq k$, $p' \cap \chi(Q) \supseteq p \cap \chi(Q)$, and there is an s-v path P' with $\chi(P') = p'$.*

The main technical result of this paper is then the following theorem stating that if a family $\mathcal{P}$ of color sets is large, then we can find an irrelevant color set in $\mathcal{P}$.

▶ **Lemma 3.3.** *Let (G, C, χ, s, t, k) be an instance of Colored Path*. Given a family $\mathcal{P}$ of s-opening color sets of set of size $\ell \leq k$ and a vertex $v \in V(G)$, if $|\mathcal{P}| > f(k)$, $f(k) = k^{\mathcal{O}(k^3)}$, then we can in time polynomial in $|\mathcal{P}| + |V(G)|$ find a set $p \in \mathcal{P}$ such that $\mathcal{P} \setminus \{p\}$ k-represents $\mathcal{P}$ w.r.t. v.*

3.1 Algorithm assuming Lemma 3.3

In this subsection, we show how to get an FPT-algorithm for Colored Path* assuming Lemma 3.3 is true. The whole algorithm is relatively simple and is given in Algorithm 1. The main goal of the subsection is to show that the algorithm is correct and runs in FPT-time.

While the definition of k-representation is not the most intuitive definition of representation (for example it is not transitive), we show that it is sufficient to preserve a path of some specific form. Let P be a k-valid s-t path. For $i \in [0, k]$ let $v_i(P)$ be the last vertex on P such that $|\chi(\mathbf{pre}(P, v_i(P)))| \leq i$ and let $\ell_i(P)$ be the length, i.e., number of edges, of $\mathbf{suf}(P, v_i(P))$. If the path P is clear from the context, we write v_i and ℓ_i instead of $v_i(P)$ and $\ell_i(P)$. For example, we write $\mathbf{pre}(P, v_i)$ instead of $\mathbf{pre}(P, v_i(P))$. Note that for a k-valid s-t path P, $\ell_k(P) = 0$ and since G is irreducible w.r.t. color contraction, $\ell_0(P)$ is precisely the length of P. For two vectors $(a_0, a_1, a_2, \ldots, a_k), (b_0, b_1, b_2, \ldots, b_k)$ we say $(a_0, \ldots, a_k) < (b_0, \ldots, b_k)$ if there exists $i \in [0, k]$ such that $a_i < b_i$ and for all $j > i$ $a_j = b_j$. For a k-valid s-t path, we call the vector $\vec{\ell}(P) = (\ell_0(P), \ldots, \ell_k(P))$ the *characteristic* vector of P (see also Figure 2).

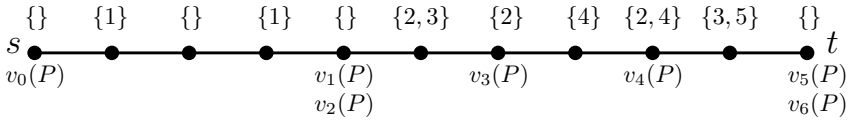


Figure 2 Figure depicting the definition of $v_i(P)$ for $k = 6$ and a path using 5 colors. The characteristic vector $\vec{\ell}(P) = (\ell_0(P), \ldots, \ell_6(P))$ is $(10, 6, 6, 4, 2, 0, 0)$.

▶ **Lemma 3.4.** *Let P be a k-valid s-t path with characteristic vector $\vec{\ell}(P)$, then there exists a nice k-valid s-t path P' with characteristic vector $\vec{\ell}(P')$ such that $\vec{\ell}(P') \leq \vec{\ell}(P)$.*

The following technical lemma will help us later show that replacing a prefix of a path P with $\chi(\mathbf{pre}(P, v_i)) \in \mathcal{P}$ by its representative will always lead to a path P' with $\vec{\ell}(P') \leq \vec{\ell}(P)$.

▶ **Lemma 3.5.** *Let P be an s-t path, $w \in V(P)$, let* $\mathrm{pre} = \mathbf{pre}(P, w)$, $\mathrm{suf} = \mathbf{suf}(P, w)$, *and let* pre' *be an s-w path such that $|\chi(\mathrm{pre}') \cup (\chi(\mathrm{pre}) \cap \chi(\mathrm{suf}))| \leq |\chi(\mathrm{pre})|$ and $|\chi(\mathrm{pre}')| < |\chi(\mathrm{pre})|$. Then $\vec{\ell}(\mathrm{pre}' \circ \mathrm{suf}) < \vec{\ell}(P)$.*

Algorithm 1 The algorithm for Colored Path*.

Data: An instance (G, C, χ, s, t, k) of Colored Path*
Result: A k-valid s-t path or NO, if such a path does not exists

$\mathcal{P}_0 = \{\emptyset\}$;
for $i \in [k]$ **do**
  $\hat{\mathcal{P}}_i = \emptyset$
  for $v \in V(G)$ **do**
    for $p \in \bigcup_{j\in[0,i-1]} \mathcal{P}_j$ **do**
      if $|\chi(v) \cup p| = i$ **then**
        if *there is a k-valid s-t path P with $\chi(P) \subseteq \chi(v) \cup p$* **then**
          Output P and stop
        end
        $\hat{\mathcal{P}}_i = \hat{\mathcal{P}}_i \cup \{\chi(v) \cup p\}$
      end
    end
  end
  for $v \in V(G)$ **do**
    $\mathcal{P}_i^v = \hat{\mathcal{P}}_i$
    while $|\mathcal{P}_i^v| > f(k)$ **do**
      Compute $p \in \mathcal{P}_i^v$ such that $\mathcal{P}_i^v \setminus \{p\}$ k-represents $\mathcal{P}_i^v$ w.r.t. v (by Lemma 3.3)
      $\mathcal{P}_i^v = \mathcal{P}_i^v \setminus \{p\}$
    end
  end
  $\mathcal{P}_i = \bigcup_{v\in V(G)} \mathcal{P}_i^v$
end
Output NO

Next, we show that k-representativity preserve in a sense a representation of a k-valid paths with minimal characteristic vector. Before we state the next lemma we introduce the following notation. We say that a set of colors p *i-captures* a s-t path P if $|\chi(\mathbf{pre}(P, v_i)| = |p|$, p completes $\mathbf{suf}(P, v_i)$, and p contains $\chi(\mathbf{pre}(P, v_i)) \cap \chi(\mathbf{suf}(P, v_i))$.

▶ **Lemma 3.6.** *Let (G, C, χ, s, t, k) be a* YES*-instance, P a nice k-valid path minimizing $\vec{\ell}(P)$, and $\mathcal{P}'$ and $\mathcal{P}$ two families of s-opening subsets of C of size $i \leq k$. If $|\chi(\mathbf{pre}(P, v_i))| = i$, $\mathcal{P}'$ k-represents $\mathcal{P}$ w.r.t. $v_i = v_i(P)$, and there is $p \in \mathcal{P}$ such that p i-captures P. Then there is $p' \in \mathcal{P}'$ such that p' i-captures P.*

Proof. Since $|p| = |\mathbf{pre}(P, v_i)| = i$ and p completes $\operatorname{suf} Pv_i$, it follows from the choice of P and Lemma 3.5 that p minimally completes P. Because, $\mathcal{P}'$ k-represents $\mathcal{P}$ w.r.t. v_i, it follows that there exists $p' \in \mathcal{P}'$ such that $|p' \cup \chi(\operatorname{suf} Pv_i)|$, there is a s-v_i path P' with $\chi(P') = p'$ and $p' \cap \chi(\mathbf{suf}(P, v_i)) \supseteq p \cap \chi(\mathbf{suf}(P, v_i)) \supseteq \chi(\mathbf{pre}(P, v_i)) \cap \chi(\mathbf{suf}(P, v_i))$. Where the second containment follows, because p i-captures P. Therefore p' contains $\chi(\mathbf{pre}(P, v_i)) \cap \chi(\mathbf{suf}(P, v_i))$. To finish the proof it only remains to show that no vertex on $\mathbf{suf}(P, v_i)$ other than v_i is reachable from s by p'. Assume otherwise and let $w \in V(\mathbf{suf}(P, v_i)) \setminus \{v_i\}$ be the last vertex that is reachable by p'. We show that $|p' \cup (\chi(\mathbf{pre}(P, w)) \cap \chi(\mathbf{suf}(P, w)))| \leq |\chi(\mathbf{pre}(P, w))|$. Since clearly $|p'| = i < |\chi(\mathbf{pre}(P, w))|$, the lemma then follows by applying Lemma 3.5 and from the choice of P. ◀

▶ **Lemma 3.7.** *Let (G, C, χ, s, t, k) be a* YES*-instance, P a nice k-valid s-t path minimizing the vector $\vec{\ell}(P)$. Moreover, let $\mathcal{P}_0 = \emptyset$ and $\mathcal{P}_1, \ldots, \mathcal{P}_k$ the color sets created in the step on line 21 of Algorithm 1. Then for all $i \in [0, k]$ such that $|\chi(\mathbf{pre}(P, v_i))| = i$, there is $p_i \in \mathcal{P}_i$ such that p_i i-captures P.*

Proof. We will prove the lemma by induction. Since $\mathcal{P}_0$ contains $\emptyset$ and $\chi(s) = \emptyset$, it is easy to see that the lemma is true for $i = 0$ and that $\chi(\mathbf{pre}(P, v_0)) = 0$. Let us assume that the lemma is true for all $j < i$. If $v_i = v_{i-1}$,[1] then the statement is true for i, because $|\chi(\mathbf{pre}(P, v_i))| \leq i - 1$. Hence, we assume for the rest of the proof that $v_i \neq v_{i-1}$. Let $j \in [0, i-1]$ be such that $v_{j-1} \neq v_{i-1}$ but $v_j = v_{i-1}$ and let u be the vertex on P just after v_j. It follows from definition of v_{j-1}, v_j, and v_{i-1} that $|\chi(\mathbf{pre}(P, v_j))| = j$ and $|\chi(\mathbf{pre}(P, u))| = i$. By the induction hypothesis there is $p_j \in \mathcal{P}_j$ such that p_j i-captures P. In particular v_j is the last vertex on $\mathbf{suf}(P, v_j)$ reachable from s by p_j and $p_j \supseteq \chi(\mathbf{pre}(P, v_j)) \cap \chi(\mathbf{suf}(P, v_j))$.

▷ Claim 3.8. $|p_j \cup \chi(u)| = i$ and $p_j \cup \chi(u)$ minimally completes $\mathbf{suf}(P, v_i)$.

From the above claim, it follows that $\hat{\mathcal{P}}_i$ contains a color set $\hat{p} = p_j \cup \chi(u)$ such that $|\hat{p}| = i$ minimally completes $\mathbf{suf}(P, v_i)$. Moreover, $\hat{p} \supseteq \chi(\mathbf{pre}(P, v_i)) \cap \chi(\mathbf{suf}(P, v_i))$ and $\hat{p}$ i-captures P. The rest of the proof follows by applying Lemma 3.6 in every loop between the steps on lines 16 and 19 for $v = v_i$. ◀

Now, if the nice k-valid s-t path P minimizing the vector $\vec{\ell}(P)$ contains $i \leq k$ colors, then $v_i(P)$ is a singleton path (t). Since by Lemma 3.7 there is $p \in \mathcal{P}_i$ that i-captures P, it means that t is reachable from s by p and Algorithm 1 outputs a s-t path using only the colors in p. Moreover, whenever it outputs a path it check whether it is k-valid. Therefore after analyzing the running time of Algorithm 1 we obtain the following theorem.

▶ **Theorem 3.9.** *There is an algorithm that given an instance (G, C, χ, s, t, k) of* Colored Path* *either outputs k-valid s-t path or decides that no such path exists, in time $\mathcal{O}(k^{\mathcal{O}(k^3)} \cdot |V(G)|^{\mathcal{O}(1)})$.*

Note that by the reduction from Connected Obstacle Removal to Colored Path* discussed in the introduction, Theorem 3.9 implies also an algorithm for Connected Obstacle Removal with the asymptotically same running time and hence Theorem 1.1.

3.2 Proof of Lemma 3.3

▶ **Observation 3.10.** *Let $\mathcal{P}$ be a family of s-opening subsets of C of size $\ell \leq k$, $v \in V(G)$, and $p \in \mathcal{P}$. If there is an s-v path P with $\chi(P) \subsetneq p$, then $\mathcal{P} \setminus \{p\}$ k-represents $\mathcal{P}$.*

For the rest of the section we will fix $v \in V(G)$, $\ell \in [k]$, and we let $\mathcal{P}$ be a family of s-opening color sets of size ℓ such that, for every $p \in \mathcal{P}$, v is reachable from s by p but is not reachable from s by any proper subset of p. Our goal in the remainder of the section is to show that if $|\mathcal{P}| > f(k)$, $f(k) = k^{\mathcal{O}(k^3)}$, then we can find in FPT-time a color set $p \in \mathcal{P}$ such that $\mathcal{P} \setminus \{p\}$ k-represents $\mathcal{P}$ w.r.t. v. We refer to such p also as an *irrelevant* color set.

[1] Throughout the proof, to improve readability we write v_i instead of $v_i(P)$.

3.2.1 Sketch of the Proof

The main idea is to show that if the family $\mathcal{P}$ is large, in our case of size at least $k^{\mathcal{O}(k^3)}$, then we can find a subfamily of $\mathcal{P}$ that is structured and this structure makes it easier to find an irrelevant color set that can be always represented within the structured subfamily. We can first apply sunflower lemma and restrict our search to a subfamily of size at least $k^{\mathcal{O}(k^2)}$ whose color sets pairwise intersect in the same color sets c, but are otherwise pairwise color-disjoint. Now we can remove colors in c from the graph and apply the color contraction operation to newly created neighbors with the same color (see Subsection 3.2.3).

In the rest of the proof, we can restrict our search for an irrelevant color set to a family $\mathcal{P}$ whose color sets are pairwise color disjoint. Moreover, we assume the graph is irreducible w.r.t. color contraction. Now for each $p_i \in \mathcal{P}$ we compute an s-v path P_i such that $\chi(P_i) = p_i$, by Observation 3.10 this is simply done by finding an s-v path in the subgraph induced on vertices with colors in p_i. The goal is to further restrict the search for an irrelevant path to a set of paths $\mathbf{P}$ such that there is a small set of vertices U, $|U| \leq 2k$, such that all the paths in $\mathbf{P}$ visit all vertices of U in the same order, but every vertex in $V(G) \setminus (U \cup \{s, v\})$ appears on at most $\frac{|\mathbf{P}|}{f(k)}$ paths. This is simply done by finding a vertex that appear on the most paths in $\mathbf{P}$, including the vertex in U if the vertex appears on at least $\frac{|\mathbf{P}|}{|U|! \cdot f(k)}$ paths, and restricting $\mathbf{P}$ to the paths containing the vertex. Otherwise, we stop. We show in Lemma 3.13 that because each path in $\mathbf{P}$ has at most k colors, we stop after including at most $2k$ vertices into U. To get the paths that visit U in the same order, we just go through all $|U|!$ orderings of U and pick the one most paths adhere to. To finish the proof, we show that thanks to the structure of paths in $\mathbf{P}$, for any two consecutive vertices in U, there is a large set of paths that are pairwise vertex disjoint between the two consecutive vertices of U (Lemma 3.16). Hence, we get into the situation similar to the one in Figure 3. Any v-t path (walk) that contains at most k colors and does not contain vertices in U can only interact with a few of these paths between the two consecutive vertices. Hence, because $\mathcal{P}$ was large and because of the structure of paths in $\mathbf{P}$, we find a path that cannot share a color with any v-t walk with at most k colors (Lemma 3.17). But the color set of such a path is then represented by any other color set in $\mathcal{P}$, as they have the same size.

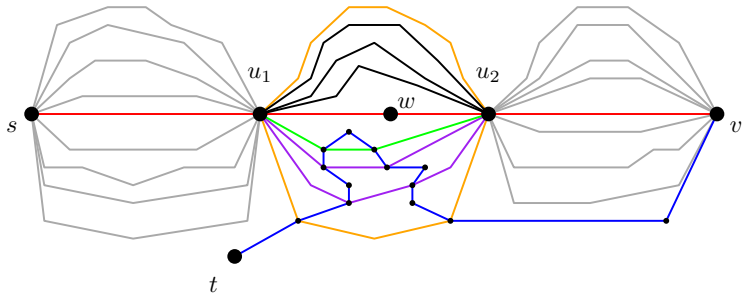


Figure 3 A set of pairwise color-disjoint paths that intersects exactly in u_1 and u_2 in the same order. If a path P from v to t do not contain s, u_1, nor u_2 but it shares a color with some vertex w on the part of the red. Then P has to cross at least 4 of the color-disjoint path and hence it has to contain at least 3 colors. For example for the blue path are vertices outside of the orange region, inside the purple region, and the region between red and green path pairwise color-disjoint. In each of these regions the blue path contains at least 2 consecutive vertices, hence at least one is not empty.

3.2.2 The Color-Disjoint Case

The goal of this subsection is to show that Lemma 3.3 is true for a special case when the color sets in $\mathcal{P}$ are pairwise color-disjoint and the input graph is irreducible w.r.t. color contraction. This is the most difficult and technical part of the proof. For the rest of the subsection we will have the following assumption:

▶ **Assumption 3.11.** *For an instance (G, C, χ, s, t, k) of* Colored Path* *and family $\mathcal{P}$ of color sets each of size $\ell \leq k$, we assume that G is irreducible w.r.t. color contraction and the sets in $\mathcal{P}$ are pairwise color-disjoint.*

In this subsection, it will be more convenient to work with a set of paths instead of a set of color sets. Given a set $\mathcal{P} = \{p_1, \ldots, p_{|\mathcal{P}|}\}$ of color-disjoint color sets such that v is reachable by each $p \in \mathcal{P}$ from s but not by any proper subset of p, we will construct a set of paths $\mathbf{P} = \{P_1, \ldots, P_{|\mathcal{P}|}\}$ such that $\chi(P_i) = p_i$ for all $i \in [|\mathcal{P}|]$. Note that, since v is not reachable from s by any proper subset of p_i, this can be simply done by finding a shortest s-v path in the graph obtained from G by removing all vertices containing a color not in p_i.

Now we restrict our attention to a subset of paths $\mathbf{Q}$ constructed by Algorithm 2.

Algorithm 2 Refining the set of important s-v paths.

Data: A set of pairwise color-disjoint paths $\mathbf{P}$ in a graph G
Result: A subset $\mathbf{Q}$ of $\mathbf{P}$ and $U \subseteq V(G)$ such that $|\mathbf{Q}| > \frac{|\mathbf{P}|}{((|U|+1)!\cdot(8k^2+8k+2))^{|U|}}$, all paths in $\mathbf{Q}$ contains all the vertices in U, and for every vertex $w \in V(G) \setminus U$ at most $\frac{|\mathbf{Q}|}{(|U|+1)!\cdot(8k^2+8k+2)}$ paths in $\mathbf{Q}$ contains w.

1 $U = \emptyset$ and $\mathbf{Q} = \mathbf{P}$
2 let u be a vertex in $V(G) \setminus U$ contained by the highest number of paths in $\mathbf{Q}$
3 **if** *u is contained in more than $\frac{|\mathbf{Q}|}{(|U|+1)!\cdot(8k^2+8k+3)}$ paths* **then**
4 $U = U \cup \{u\}$
5 restrict $\mathbf{Q}$ to contain only the paths containing u
6 go to the step on line 2
7 **end**

▶ **Lemma 3.12.** *Let G a color-connected plane graph that is irreducible w.r.t. color contraction, s, u_1, u_2, u_3, v be vertices in G and let $\mathbf{P} = \{P_1, \ldots, P_{|\mathbf{P}|}\}$ be pairwise color-disjoint s-v paths all going through the vertices u_1, u_2, and u_3 in the same order. Then there are at most two paths $P_i \in \mathcal{P}$ such that if w^i_j, $j \in [3]$, denotes the vertex on P_i immediately after u_j then $\chi(w^i_1) \cap \chi(w^i_3) \neq \emptyset$.*

Now we can show that if $|U| \geq 2k + 1$, then at the point when Algorithm 2 adds $2k + 1$-st element to U, we can find $k^2 + k + 1$ paths in $\mathbf{Q}$ that visit the first $2k + 1$ vertices of U in the same order. Lemma 3.12 then implies that there is a path $P_i \in \mathbf{P}$ such that $\chi(w^i_j) \cap \chi(w^i_{j'}) = \emptyset$ for all $j \neq j'$, $j, j' \in \{1, 3, 5, \ldots, 2k + 1\}$, where w^i_j denotes the vertex on P_i immediately after u_j. Then $|\chi(P_i)| \geq k + 1$ which contradicts definition of $\mathbf{P}$.

▶ **Lemma 3.13.** *If $|\mathbf{P}| \geq f(k)$, $f(k) = k^{\mathcal{O}(k^2)}$, then when Algorithm 2 terminates, it holds that $|U| < 2k + 1$.*

We can now fix an ordering $\tau = (u_1, u_2, \ldots, u_{|U|})$ of vertices in U which maximizes the number of paths in $\mathbf{Q}$ that visit U in the same order as τ and let $\mathbf{Q}'$ be the restriction of $\mathbf{Q}$

to the paths that are consistent with this ordering. Clearly $|\mathbf{Q}| \leq |\mathbf{Q}'| \cdot (2k)!$ and it suffice to show that we can find an irrelevant path in $\mathbf{Q}'$ if $|\mathbf{Q}'|$ is large. The agenda for the rest of the proof is as follows. Because $|U| \leq 2k$ and intersection number of each vertex outside $|U|$ is small compared to the size of $\mathbf{Q}'$, only "few" paths can share a color with any k-valid v-t walk that do not contain a vertex in U hence we can find an irrelevant path. The color set of this irrelevant path is then the irrelevant color set in $\mathcal{P}$.

Recall that due to Assumption 3.11, we assume that the graph G is color contracted and no two neighbors have the same color set. Moreover, the paths in $\mathbf{Q}'$ are color-disjoint, so the vertices in $U \cup \{s, v\}$ are all empty and each neighbor of these vertices belongs to at most one path in $\mathbf{Q}'$. The goal in the following few technical lemmas is to show that for any two consecutive vertices u_i and u_{i+1} in U we can find a large (of size at least $4k+1$) subsets of paths in $\mathbf{Q}'$ that pairwise do not intersect between u_i and u_{i+1}.

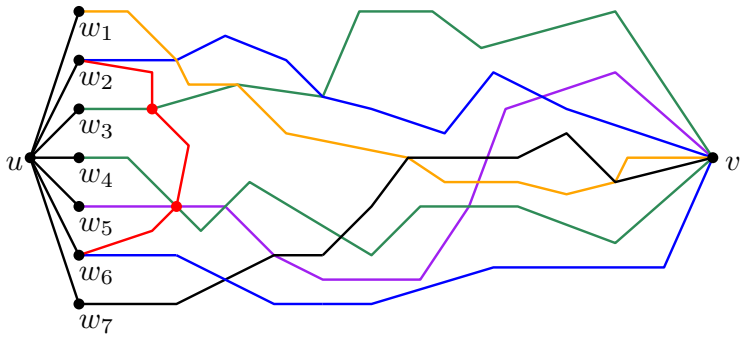


Figure 4 Situation in Lemma 3.14. On the picture are seven u-v paths, no 3 of them intersecting in the same vertex. The red w_2-w_6 path on the picture intersects the three paths containing w_3, w_4, and w_5, respectively. Any such path has to contain at least 2 vertices, else the only vertex on the path would be the intersection of 3 u-v paths.

▶ **Lemma 3.14.** *Given an instance (G, C, χ, s, t, k) which is irreducible w.r.t. color contraction, two vertices u, v, $b \in \mathbb{N}$ and a set $\mathbf{P}$ of k-valid u-v paths such that no b paths intersect in the same vertex. Let $w_1, \ldots, w_r$ be the neighbors of u, each the second vertex of a different path in $\mathbf{P}$, in counterclockwise order. For $i \in [r]$ let P_i denote the path in $\mathbf{P}$ containing w_i. Let $1 \leq i < j \leq r$, then the shortest curve σ from w_i to w_j that intersects G only in vertices of $V(G) \setminus \{u, v\}$ contains at least $\frac{\min\{j-i, r+i-j\}-1}{b}$ vertices on paths in $\mathcal{P} \setminus \{P_i, P_j\}$.*

Proof. See an example of the situation in Figure 4. Given a curve σ, we can easily find a closed curve σ' that intersect G in u, w_i, w_j and the vertices that are intersected by σ. The vertices on σ' are then the vertex separator separating v from either $w_{i+1}, \ldots, w_{j-1}$ or from $w_1, \ldots, w_{i-1}$ and $w_{j+1}, \ldots, w_r$. If the vertices on σ' are the vertex separator separating v from $w_{i+1}, \ldots, w_{j-1}$, then all the paths $P_{i+1}, \ldots, P_{j-1}$ has to pass a vertex on σ different than w_i or w_j. Since no b paths intersect in the same vertex, we get that σ contains at least $\frac{j-i-1}{b}$ vertices in this case. The case when the vertices on σ' are the vertex separator separating v from $w_1, \ldots, w_{i-1}$ and $w_{j+1}, \ldots, w_r$ is symmetric and the lemma follows. ◀

▶ **Lemma 3.15.** *Let (G, C, χ, s, t, k) be an instance of Colored Path* such that G is irreducible w.r.t. color contraction, H a subgraph of G, and P a k-valid u-v path with $u, v \in V(H)$ and $\chi(P) \cap \chi(H) = \emptyset$. Then P intersects at most k faces of H.*

The combination of the two above lemmas immediately yields the following:

▶ **Lemma 3.16.** *Given an instance (G, C, χ, s, t, k) which is irreducible w.r.t. color contraction, two vertices u, v, an integer $b \in \mathbb{N}$ and a set $\mathbf{P}$ of k-valid pairwise color-disjoint u-v paths such that no b paths intersect in the same vertex. Let $w_1, \ldots, w_r$ be the neighbors of u, each the second vertex of a different path in $\mathcal{P}$, in counterclockwise order. Let $1 \leq i < j \leq r$ and let P_i and P_j be the two paths in $\mathcal{P}$ containing w_i and w_j, respectively. If $\min\{j - i, r + i - j\} > 2k \cdot b$, then P_i and P_j do not intersect.*

▶ **Lemma 3.17.** *If no b paths in $\mathbf{Q}'$ intersect in the same vertex in $V(G) \setminus (U \cup \{s, v\})$ and $|\mathbf{Q}'| > (8k^2 + 8k + 2) \cdot (|U| + 1) \cdot b$, then we can in polynomial time find a path $P \in \mathbf{Q}'$ such that for every k-valid v-t walk Q that does not contain a vertex in U holds $\chi(P) \cap \chi(Q) = \emptyset$.*

Proof. For the convenience let us denote s by u_0 and v by $u_{|U|+1}$. We will show that for every $i \in \{0, \ldots, |U|\}$, every k-valid v-t walk can intersect at most $(8k^2 + 8k + 2) \cdot b$ paths in a vertex on the path between u_i and u_{i+1}. For a path $P \in \mathbf{Q}'$ let P^i denote the subpath between u_i and u_{i+1} and let $\mathbf{Q}^i = \{P^i \mid P \in \mathbf{Q}'\}$. Clearly, the paths in $\mathbf{Q}^i$ are color-disjoint u_i-u_{i+1} each containing at most $\ell \leq k$ colors and no b paths in $\mathbf{Q}^i$ intersect in the same vertex beside u_i and u_{i+1}. Now let H^i be the subgraph of G induced by the edges on paths in $\mathbf{Q}^i$. Since G is color contracted, u_i is an empty vertex, and the paths in $\mathbf{Q}^i$ are colored disjoint, each neighbor of u_i appears on a unique path in $\mathbf{Q}^i$. Let $w_1, w_2, \ldots, w_{|\mathbf{Q}^i|}$ be the neighbors of u_i in H^i in counterclockwise order and let P^i_j be the path in $\mathbf{Q}^i$ that contains w_j. Clearly, t is in the interior of some face f of H^i and there is at least one path that contains an edge incident on f in H^i. Without loss of generality let P^i_1 be such path (note that we can always choose a counterclockwise order around u_i for which this is true).

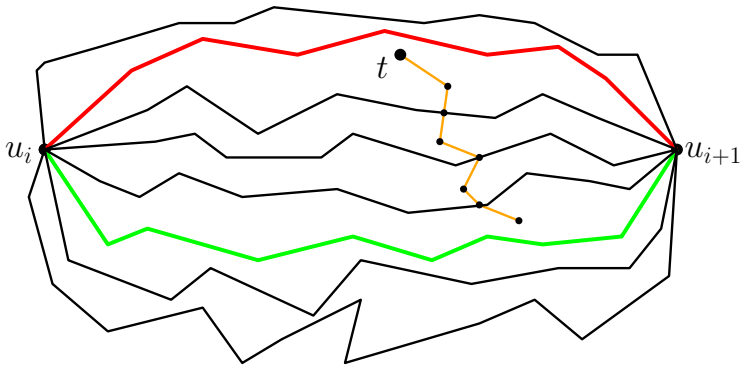


Figure 5 Any path that starts in a face incident on the red path and finishes in a face incident on the green path that does not contain u_i nor u_{i+1} has to appear in at least 4 different faces. Since the paths are color-disjoint, only the consecutive faces can share colors and hence any such path contains at least 2 colors.

▷ Claim 3.18. Let $j \in [|\mathbf{Q}^i|]$. If $(2k + 1)(2k + 1) \cdot b < j < |\mathbf{Q}^i| - (2k + 1)(2k + 1) \cdot b$, k-valid v-t walk Q that does not contain u_i nor u_{i+1} in the interior holds $\chi(P^i_j) \cap \chi(Q) = \emptyset$.

Proof. Consider the following set of paths: $P^i_1, P^i_{2k+2}, P^i_{4k+3}, \ldots, P_{4k^2+4k+1}, P^i_j$, P^i_{j+2k+1}, $P^i_{j+4k+2}, \ldots, P^i_{j+4k^2+4k}$. By Lemma 3.16, these paths are pairwise non-intersecting. Hence, we are in the situation as depicted in Figure 5. Since the paths in $\mathbf{Q}^i$ are pairwise color-disjoint, the colors of P^i_j are only on vertices of G inside the region bounded by P_{2k^2+k+1} and P^i_{j+2k+1}. Therefore, if $\chi(Q) \cap P^i_j \neq \emptyset$ for some v-t walk Q, then Q contains a vertex w inside the region bounded by P_{2k^2+k+1} and P^i_{j+2k+1}. Moreover, Q does not contain u_i nor u_{i+1} as an inner vertex then it either crosses all the paths in $\mathbf{P}_1 = \{P^i_{2k+2}, P^i_{4k+3}, \ldots, P_{4k^2+4k+1}\}$ or all the

paths in $\mathbf{P}_2 = \{P^i_{j+2k+1}, P^i_{j+4k+2}, \ldots, P^i_{j+4k^2+2k}\}$. Without loss of generality, let us assume that Q crosses all the paths in $\mathbf{P}_1$. The other case is symmetric. As G is color contracted, no two consecutive vertices of P are empty. Hence, Q either crosses a path in $\mathbf{P}_1$ in a colored vertex or there is a colored vertex on Q between two consecutive paths in $\mathbf{P}_1$ (resp. $\mathbf{P}_2$). Let us partition the paths in $\mathbf{P}_1 \cup \{P_1, P_j\}$ into $k+1$ group of two consecutive pairs. that is we partition $\mathbf{P}_1$ into groups $\{P_1, P_{2k+2}\}$, $\{P_{4k+3}, P_{6k+4}\}$,...,$\{P_{4k^2-1}, P_{4k^2+2k}\}$,$\{P_{4k^2+4k+1}, P_j\}$. If the walk Q crosses all paths in $\mathbf{P}_1$, it has to contains a colored vertex in each of the $k+1$ groups. However, each two groups are separated by color-disjoint paths. Therefore, two colored vertices in two different groups have to be color-disjoint. But then $\chi(Q)$ contains at least $k+1$ colors, this is however not possible, because Q is k-valid. ◁

The lemma then follows by marking for each of $|U|+1$ consecutive pairs $2(2k+1)^2 \cdot b$ paths that can share a color with some Q and outputting any non-marked path. ◀

Since $\chi(P) \cap \chi(Q) = \emptyset$, $\chi(P)$ can be replaced by any other color set of $|\chi(P)|$ colors and we can safely remove it from $\mathcal{P}$. Since we chose $\mathbf{Q}'$ such that no $\frac{|\mathbf{Q}|}{(|U|+1)!\cdot(8k^2+8k+3)} = \frac{|\mathbf{Q}'|}{(|U|+1)\cdot(8k^2+8k+3)}$ paths intersect in $\mathbf{Q}'$, we get the following main result of this subsection.

▶ **Lemma 3.19.** *Let (G, C, χ, s, t, k) be an instance of* Colored Path* *such that G is irreducible w.r.t. color contraction. Given a family $\mathcal{P}$ of pairwise color-disjoint s-reachable color sets of set of size $\ell \leq k$ and a vertex $v \in V(G)$, if $|\mathcal{P}| > 2^{\mathcal{O}(k^2 \log(k))}$, then we can in time polynomial in $|\mathcal{P}| + |V(G)|$find a set $p \in \mathcal{P}$ such that $\mathcal{P} \setminus \{p\}$ k-represents $\mathcal{P}$ w.r.t. v.*

3.2.3 Finishing the Proof

Proof of Lemma 3.3. Since each set in $\mathcal{P}$ has precisely $\ell \leq k$ colors, if $|\mathcal{P}| > \ell! \cdot (g(k))^{\ell+1}$, $g(k) = k^{\mathcal{O}(k^2)}$ then, by Lemma 2.1 we can, in time polynomial in $|\mathcal{P}|$, find a set $\mathcal{Q}$ of $g(k)+1$ sets in $\mathcal{P}$ such that there is a color set $c \subseteq C$ and for any two distinct sets p_1, p_2 in $\mathcal{Q}$ it holds $p_1 \cap p_2 = c$. Now let $(G, C', \chi', s, t, k-|c|)$ be the instance of Colored Path* such that $C' = C \setminus c$ and for every $v \in V(G)$, $\chi'(v) = \chi(v) \setminus c$ and let $\mathcal{Q}' = \{p \setminus c \mid p \in \mathcal{Q}\}$.

▷ Claim 3.20. For all $p \in \mathcal{Q}$, $\mathcal{Q}' \setminus \{p \setminus c\}$ $(k-|c|)$-represents $\mathcal{Q}'$ w.r.t. v in $(G, C', \chi', s, t, k-|c|)$ if and only if $\mathcal{Q}^v \setminus \{p\}$ k-represents $\mathcal{Q}^v$ w.r.t. v in (G, C, χ, s, t, k).

Removing the colors in c from G can result in an instance that is not irreducible w.r.t. color contraction. However, in our algorithm for color-disjoint case, we crucially rely on the fact that G is irreducible w.r.t. color contraction. Now let $G_0 = G$, $\chi_0 = \chi'$, $s_0 = s$, $t_0 = t$, $v_0 = v$ and for $i \geq 1$ let $(G_i, C, \chi_i, s_i, t_i, k-|c|)$ be an instance we obtain from $(G_{i-1}, C, \chi_{i-1}, s_{i-1}, t_{i-1}, k-|c|)$ by a single color contraction of vertices x_i and y_i into a vertex z_i and let $v_i = z_i$ if $v_{i-1} \in \{x_i, y_i\}$ and $v_i = v_{i-1}$ otherwise.

▷ Claim 3.21. For all $p \in \mathcal{P}$, if the set $\mathcal{P} \setminus p$ $(k-|c|)$-represents $\mathcal{P}$ w.r.t. v_i in $(G_i, C, \chi_i, s_i, t_i, k-|c|)$, then $\mathcal{P} \setminus p$ $(k-|c|)$-represents $\mathcal{P}$ w.r.t. v in $(G_{i+1}, C, \chi_{i+1}, s_{i+1}, t_{i+1}, k-|c|)$.

Let $(G_i, C, \chi_i, s_i, t_i, k-|c|)$ be the instance obtained from $(G, C', \chi', s, t, k-|c|)$ by repeating color contraction operation until G_i is irreducible w.r.t. color contraction and let v_i be the image of v. Since G_i is irreducible w.r.t. color contraction, the sets in $\mathcal{Q}'$ are pairwise color-disjoint, and $|\mathcal{Q}'| = g(k)+1 > g(k-|c|)$, we can use Lemma 3.19 to find in time polynomial in $|\mathcal{Q}'| + |V(G)|$ a set $p \in \mathcal{Q}'$ such that $\mathcal{Q}' \setminus \{p\}$ $(k-|c|)$-represents $\mathcal{Q}'$ w.r.t. v_i in $(G_i, C, \chi_i, s_i, t_i, k-|c|)$. By Claim 3.21, it follows that $\mathcal{Q}' \setminus \{p\}$ $(k-|c|)$-represents $\mathcal{Q}'$ w.r.t. v in $(G, C', \chi', s, t, k-|c|)$ and by Claim 3.20 $\mathcal{Q} \setminus \{p \cup c\}$ k-represents $\mathcal{Q}$ in (G, C, χ, s, t, k). Finally, since for all $p' \in \mathcal{P} \setminus \mathcal{Q}$ is $p' \in \mathcal{P} \setminus \{p \cup c\}$ it follows that $\mathcal{P} \setminus \{p \cup c\}$ k-represents $\mathcal{P}$. ◀

References

1 H. Alt, S. Cabello, P. Giannopoulos, and C. Knauer. Minimum cell connection in line segment arrangements. *International Journal of Computational Geometry and Applications*, 27(3):159–176, 2017.

2 Sayan Bandyapadhyay, Neeraj Kumar, Subhash Suri, and Kasturi R. Varadarajan. Improved approximation bounds for the minimum constraint removal problem. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX-/RANDOM 2018, August 20-22, 2018 - Princeton, NJ, USA*, pages 2:1–2:19, 2018.

3 S. Bereg and D. Kirkpatrick. Approximating barrier resilience in wireless sensor networks. In *Proceedings of ALGOSENSORS*, pages 29–40, 2009.

4 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.

5 Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.

6 Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013.

7 Eduard Eiben, Jonathan Gemmell, Iyad A. Kanj, and Andrew Youngdahl. Improved results for minimum constraint removal. In Sheila A. McIlraith and Kilian Q. Weinberger, editors, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 6477–6484. AAAI Press, 2018.

8 Eduard Eiben and Iyad A. Kanj. How to navigate through obstacles? In Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella, editors, *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, volume 107 of *LIPIcs*, pages 48:1–48:13. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018. Full version avaiable at arXiv:1712.04043v1.

9 Paul Erdös and Richard Rado. Intersection theorems for systems of sets. *Journal of the London Mathematical Society*, 1(1):85–90, 1960.

10 L. Erickson and S. LaValle. A simple, but NP-hard, motion planning problem. In *Proceedings of AAAI*. AAAI Press, 2013.

11 Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*, volume XIV of *Texts in Theoretical Computer Science. An EATCS Series*. Springer, Berlin, 2006.

12 Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, and Saket Saurabh. Efficient computation of representative families with applications in parameterized and exact algorithms. *J. ACM*, 63(4):29:1–29:60, 2016.

13 A. Gorbenko and V. Popov. The discrete minimum constraint removal motion planning problem. In *Proceedings of the American Institute of Physics*, volume 1648. AIP Press, 2015.

14 K. Hauser. The minimum constraint removal problem with three robotics applications. *International Journal of Robotics Research*, 33(1):5–17, 2014.

15 M. Korman, M. Löffler, R. Silveira, and D. Strash. On the complexity of barrier resilience for fat regions and bounded ply. *Computational Geometry*, 72:34–51, 2018.

16 S. Kumar, T. Lai, and A. Arora. Barrier coverage with wireless sensors. *Wireless Networks*, 13(6):817–834, 2007.

17 K. Tseng and D. Kirkpatrick. On barrier resilience of sensor networks. In *Proceedings of ALGOSENSORS*, pages 130–144, 2012.

18 S. Yang. *Some Path Planning Algorithms in Computational Geometry and Air Traffic Management*. PhD thesis, University of New Yort at Stony Brook. Available at: https://dspace.sunyconnect.suny.edu/handle/1951/59927, 2012.

A Toroidal Maxwell-Cremona-Delaunay Correspondence

Jeff Erickson
University of Illinois, Urbana-Champaign, IL, USA
jeffe@illinois.edu

Patrick Lin
University of Illinois, Urbana-Champaign, IL, USA
plin15@illinois.edu

Abstract

We consider three classes of geodesic embeddings of graphs on Euclidean flat tori:

- A torus graph G is *equilibrium* if it is possible to place positive weights on the edges, such that the weighted edge vectors incident to each vertex of G sum to zero.
- A torus graph G is *reciprocal* if there is a geodesic embedding of the dual graph G^* on the same flat torus, where each edge of G is orthogonal to the corresponding dual edge in G^*.
- A torus graph G is *coherent* if it is possible to assign weights to the vertices, so that G is the (intrinsic) weighted Delaunay graph of its vertices.

The classical Maxwell-Cremona correspondence and the well-known correspondence between convex hulls and weighted Delaunay triangulations imply that the analogous concepts for plane graphs (with convex outer faces) are equivalent. Indeed, all three conditions are equivalent to G being the projection of the 1-skeleton of the lower convex hull of points in $\mathbb{R}^3$. However, this three-way equivalence does not extend directly to geodesic graphs on flat tori. On any flat torus, reciprocal and coherent graphs are equivalent, and every reciprocal graph is equilibrium, but not every equilibrium graph is reciprocal. We establish a weaker correspondence: Every equilibrium graph on any flat torus is affinely equivalent to a reciprocal/coherent graph on *some* flat torus.

2012 ACM Subject Classification Mathematics of computing → Graphs and surfaces

Keywords and phrases combinatorial topology, geometric graphs, homology, flat torus, spring embedding, intrinsic Delaunay

Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.40

Related Version A full version of the paper is available at `https://arxiv.org/abs/2003.10057` [33].

Funding Portions of this work were supported by NSF grant CCF-1408763.

Acknowledgements We thank the anonymous reviewers for their helpful comments and suggestions.

1 Introduction

The Maxwell-Cremona correspondence is a fundamental theorem establishing an equivalence between three different structures on straight-line graphs G in the plane:

- An *equilibrium stress* on G is an assignment of non-zero weights to the edges of G, such that the weighted edge vectors around every interior vertex p sum to zero:

$$\sum_{p\,:\,pq\in E} \omega_{pq}(p-q) = \begin{pmatrix}0\\0\end{pmatrix}$$

- A *reciprocal diagram* for G is a straight-line drawing of the dual graph G^*, in which every edge e^* is orthogonal to the corresponding primal edge e.

- A *polyhedral lifting* of G assigns z-coordinates to the vertices of G, so that the resulting lifted vertices in $\mathbb{R}^3$ are not all coplanar, but the lifted vertices of each face of G are coplanar.

Building on earlier seminal work of Varignon [76], Rankine [62, 61], and others, Maxwell [52, 51, 50] proved that any straight-line planar graph G with an equilibrium stress has both a reciprocal diagram and a polyhedral lifting. In particular, positive and negative stresses correspond to convex and concave edges in the polyhedral lifting, respectively. Moreover, for any equilibrium stress ω on G, the vector $1/\omega$ is an equilibrium stress for the reciprocal diagram G^*. Finally, for any polyhedral liftings of G, one can obtain a polyhedral lifting of the reciprocal diagram G^* via projective duality. Maxwell's analysis was later extended and popularized by Cremona [25, 26] and others; the correspondence has since been rediscovered several times in other contexts [3, 39]. More recently, Whiteley [77] proved the converse of Maxwell's theorem: every reciprocal diagram and every polyhedral lift corresponds to an equilibrium stress; see also Crapo and Whiteley [24]. For modern expositions of the Maxwell-Cremona correspondence aimed at computational geometers, see Hopcroft and Kahn [38], Richter-Gebert [64, Chapter 13], or Rote, Santos, and Streinu [66].

If the outer face of G is convex, the Maxwell-Cremona correspondence implies an equivalence between equilibrium stresses in G that are *positive* on every interior edge, *convex* polyhedral liftings of G, and reciprocal *embeddings* of G^*. Moreover, as Whiteley *et al.* [78] and Aurenhammer [3] observed, the well-known equivalence between convex liftings and weighted Delaunay complexes [5, 4, 13, 32] implies that all three of these structures are equivalent to a fourth:

- A *Delaunay weighting* of G is an assignment of weights to the vertices of G, so that G is the (power-)weighted Delaunay graph [4, 7] of its vertices.

Among many other consequences, combining the Maxwell-Cremona correspondence [77] with Tutte's spring-embedding theorem [75] yields an elegant geometric proof of Steinitz's theorem [70, 69] that every 3-connected planar graph is the 1-skeleton of a 3-dimensional convex polytope. The Maxwell-Cremona correspondence has been used for scene analysis of planar drawings [24, 74, 3, 5, 39], finding small grid embeddings of planar graphs and polyhedra [31, 15, 59, 64, 63, 67, 30, 40], and several linkage reconfiguration problems [22, 29, 73, 72, 60].

It is natural to ask how or whether these correspondences extend to graphs on surfaces other than the Euclidean plane. Lovász [47, Lemma 4] describes a spherical analogue of Maxwell's polyhedral lifting in terms of Colin de Verdière matrices [17, 20]; see also [44]. Izmestiev [42] provides a self-contained proof of the correspondence for planar frameworks, along with natural extensions to frameworks in the sphere and the hyperbolic plane. Finally, and most closely related to the present work, Borcea and Streinu [11], building on their earlier study of rigidity in infinite periodic frameworks [10, 9], develop an extension of the Maxwell-Cremona correspondence to infinite periodic graphs in the plane, or equivalently, to geodesic graphs on the Euclidean flat torus. Specifically, Borcea and Streinu prove that *periodic* polyhedral liftings correspond to *periodic* stresses satisfying an additional homological constraint.[1]

[1] Phrased in terms of toroidal frameworks, Borcea and Streinu consider only equilibrium stresses for which the corresponding reciprocal toroidal framework contains no essential cycles.

1.1 Our Results

In this paper, we develop a different generalization of the Maxwell-Cremona-Delaunay correspondence to geodesic embeddings of graphs on Euclidean flat tori. Our work is inspired by and uses Borcea and Streinu's recent results [11], but considers a different aim. Stated in terms of infinite periodic planar graphs, Borcea and Streinu study periodic equilibrium stresses, which necessarily include both positive and negative stress coefficients, that include periodic *polyhedral lifts*; whereas, we are interested in periodic *positive* equilibrium stresses that induce periodic reciprocal *embeddings* and periodic *Delaunay weights*. This distinction is aptly illustrated in Figures 8–10 of Borcea and Streinu's paper [11].

Recall that a Euclidean flat torus $\mathbb{T}$ is the metric space obtained by identifying opposite sides of an arbitrary parallelogram in the Euclidean plane. A *geodesic* graph G in the flat torus $\mathbb{T}$ is an embedded graph where each edge is represented by a "line segment". Equilibrium stresses, reciprocal embeddings, and weighted Delaunay graphs are all well-defined in the intrinsic metric of the flat torus. We prove the following correspondences for any geodesic graph G on any flat torus $\mathbb{T}$.

- Any equilibrium stress for G is also an equilibrium stress for the affine image of G on any other flat torus $\mathbb{T}'$ (Lemma 2.2). Equilibrium depends only on the common *affine* structure of all flat tori.
- Any reciprocal embedding G^* on $\mathbb{T}$ – that is, any geodesic embedding of the dual graph such that corresponding edges are orthogonal – defines unique equilibrium stresses in both G and G^* (Lemma 3.1).
- G has a reciprocal embedding if and only if G is coherent. Specifically, each reciprocal diagram for G induces an essentially unique set of Delaunay weights for the vertices of G (Theorem 4.5). Conversely, each set of Delaunay weights for G induces a *unique* reciprocal diagram G^*, namely the corresponding weighted Voronoi diagram (Lemma 4.1). Thus, a reciprocal diagram G^* may not be a weighted Voronoi diagram of the vertices of G, but some unique translation of G^* is.
- Unlike in the plane, G may have equilibrium stresses that are not induced by reciprocal embeddings; more generally, not every equilibrium graph on $\mathbb{T}$ is reciprocal (Theorem 3.2). Unlike equilibrium, reciprocality depends on the *conformal* structure of $\mathbb{T}$, which is determined by the shape of its fundamental parallelogram. We derive a simple geometric condition that characterizes which equilibrium stresses are reciprocal on $\mathbb{T}$ (Lemma 5.4).
- More generally, we show that for any equilibrium stress on G, there is a flat torus $\mathbb{T}'$, unique up to rotation and scaling of its fundamental parallelogram, such that the same equilibrium stress is reciprocal for the affine image of G on $\mathbb{T}'$ (Theorem 5.7). In short, every equilibrium stress for G is reciprocal on *some* flat torus. This result implies a natural toroidal analogue of Steinitz's theorem (Theorem 6.1): Every essentially 3-connected torus graph G is homotopic to a weighted Delaunay graph on some flat torus.

Due to space limitations, we defer several proofs to the full version of the paper [33].

1.2 Other Related Results

Our results rely on a natural generalization (Theorem 2.3) of Tutte's spring-embedding theorem to the torus, first proved (in much greater generality) by Colin de Verdière [18], and later proved again, in different forms, by Delgado-Friedrichs [28], Lovász [48, Theorem 7.1][49, Theorem 7.4], and Gortler, Gotsman, and Thurston [36]. Steiner and Fischer [68] and Gortler *et al.* [36] observed that this toroidal spring embedding can be computed by solving the Laplacian linear system defining the equilibrium conditions. We describe this result

and the necessary calculation in more detail in Section 2. Equilibrium and reciprocal graph embeddings can also be viewed as discrete analogues of harmonic and holomorphic functions [49, 48].

Our weighted Delaunay graphs are (the duals of) *power diagrams* [4, 6] in the intrinsic metric of the flat torus. Toroidal Delaunay triangulations are commonly used to generate finite-element meshes for simulations with periodic boundary conditions, and several efficient algorithms for constructing these triangulations are known [53, 37, 14, 8]. Building on earlier work of Rivin [65] and Indermitte *et al.* [41], Bobenko and Springborn [7] proved that on any piecewise-linear surface, intrinsic Delaunay triangulations can be constructed by an intrinsic incremental flipping algorithm, mirroring the classical planar algorithm of Lawson [46]; their analysis extends easily to intrinsic weighted Delaunay graphs. Weighted Delaunay complexes are also known as *regular* or *coherent* subdivisions [79, 27].

Finally, equilibrium and reciprocal embeddings are closely related to the celebrated Koebe-Andreev circle-packing theorem: Every planar graph is the contact graph of a set of interior-disjoint circular disks [43, 1, 2]; see Felsner and Rote [34] for a simple proof, based in part on earlier work of Brightwell and Scheinerman [12] and Mohar [54]. The circle-packing theorem has been generalized to higher-genus surfaces by Colin de Verdière [16, 19] and Mohar [55, 56]. In particular, Mohar proves that any well-connected graph G on the torus is *homotopic* to an essentially unique circle packing *for a unique Euclidean metric* on the torus. This disk-packing representation immediately yields a weighted Delaunay graph, where the areas of the disks are the vertex weights. We revisit this result in Section 6.

Discrete harmonic and holomorphic functions, circle packings, and intrinsic Delaunay triangulations have numerous applications in discrete differential geometry; we refer the reader to monographs by Crane [23], Lovász [49], and Stephenson [71].

2 Background and Definitions

2.1 Flat Tori

A **flat torus** is the metric surface obtained by identifying opposite sides of a parallelogram in the Euclidean plane. Specifically, for any nonsingular 2×2 matrix $M = \left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right)$, let $\mathbb{T}_M$ denote the flat torus obtained by identifying opposite edges of the **fundamental parallelogram** $\Diamond_M$ with vertex coordinates $\binom{0}{0}$, $\binom{a}{c}$, $\binom{b}{d}$, and $\binom{a+b}{c+d}$. In particular, the *square* flat torus $\mathbb{T}_\square = \mathbb{T}_I$ is obtained by identifying opposite sides of the Euclidean unit square $\square = \Diamond_I = [0,1]^2$. The linear map $M\colon \mathbb{R}^2 \to \mathbb{R}^2$ naturally induces a homeomorphism from $\mathbb{T}_\square$ to $\mathbb{T}_M$.

Equivalently, $\mathbb{T}_M$ is the quotient space of the plane $\mathbb{R}^2$ with respect to the lattice Γ_M of translations generated by the columns of M; in particular, the square flat torus is the quotient space $\mathbb{R}^2/\mathbb{Z}^2$. The quotient map $\pi_M\colon \mathbb{R}^2 \to \mathbb{T}_M$ is called a *covering* map or **projection**. A **lift** of a point $p \in \mathbb{T}_M$ is any point in the preimage $\pi_M^{-1}(p) \subset \mathbb{R}^2$. A **geodesic** in $\mathbb{T}_M$ is the projection of any line segment in $\mathbb{R}^2$; we emphasize that geodesics are *not* necessarily shortest paths.

2.2 Graphs and Embeddings

We regard each edge of an undirected graph G as a pair of opposing **darts**, each directed from one endpoint, called the **tail** of the dart, to the other endpoint, called its **head**. For each edge e, we arbitrarily label the darts e^+ and e^-; we call e^+ the **reference dart** of e. We explicitly allow graphs with loops and parallel edges. At the risk of confusing the reader, we often write $p{\to}q$ to denote an arbitrary dart with tail p and head q, and $q{\to}p$ for the reversal of $p{\to}q$.

A **drawing** of a graph G on a torus $\mathbb{T}$ is any continuous function from G (as a topological space) to $\mathbb{T}$. An **embedding** is an injective drawing, which maps vertices of G to distinct points and edges to interior-disjoint simple paths between their endpoints. The **faces** of an embedding are the components of the complement of the image of the graph; we consider only *cellular* embeddings, in which all faces are open disks. (Cellular graph embeddings are also called *maps*.) We typically do not distinguish between vertices and edges of G and their images in any embedding; we will informally refer to any embedded graph on any flat torus as a **torus graph**.

In any embedded graph, $\mathit{left}(d)$ and $\mathit{right}(d)$ denote the faces immediately to the left and right of any dart d. (These are possibly the same face.)

The **universal cover** $\widetilde{G}$ of an embedded graph G on any flat torus $\mathbb{T}_M$ is the unique infinite periodic graph in $\mathbb{R}^2$ such that $\pi_M(\widetilde{G}) = G$; in particular, each vertex, edge, or face of $\widetilde{G}$ projects to a vertex, edge, or face of G, respectively. A torus graph G is **essentially simple** if its universal cover $\widetilde{G}$ is simple, and **essentially 3-connected** if $\widetilde{G}$ is 3-connected [55, 56, 57, 58, 35]. We emphasize that essential simplicity and essential 3-connectedness are features of *embeddings*; see Figure 1.

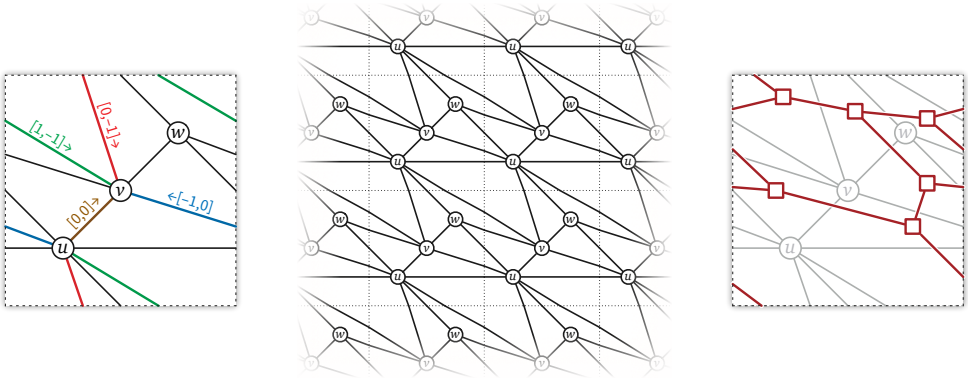


Figure 1 An essentially simple, essentially 3-connected geodesic graph on the square flat torus (showing the homology vectors of all four darts from u to v), a small portion of its universal cover, and its dual graph.

2.3 Homology, Homotopy, and Circulations

For any embedding of a graph G on the square flat torus $\mathbb{T}_\square$, we associate a **homology vector** $[d] \in \mathbb{Z}^2$ with each dart d, which records how the dart crosses the boundary edges of the unit square. Specifically, the first coordinate of $[d]$ is the number of times d crosses the vertical boundary rightward, minus the number of times d crosses the vertical boundary leftward; and the second coordinate of $[d]$ is the number of times d crosses the horizontal boundary upward, minus the number of times d crosses the horizontal boundary downward. In particular, reversing a dart negates its homology vector: $[e^+] = -[e^-]$. Again, see Figure 1. For graphs on any other flat torus $\mathbb{T}_M$, homology vectors of darts are similarly defined by how they crosses the edges of the fundamental parallelogram $\Diamond_M$.

The **(integer) homology class** $[\gamma]$ of a directed cycle γ in G is the sum of the homology vectors of its forward darts. A cycle is *contractible* if its homology class is $\binom{0}{0}$ and *essential* otherwise. In particular, the boundary cycle of each face of G is contractible.

Two cycles on a torus $\mathbb{T}$ are **homotopic** if one can be continuously deformed into the other, or equivalently, if they have the same integer homology class. Similarly, two drawings of the same graph G on the same flat torus $\mathbb{T}$ are homotopic if one can be continuously deformed into the other. Two drawings of the same graph G on the same flat torus $\mathbb{T}$ are homotopic if and only if every cycle has the same homology class in both embeddings [45, 21].

A **circulation** ϕ in G is a function from the darts of G to the reals, such that $\phi(p{\to}q) = -\phi(q{\to}p)$ for every dart $p{\to}q$ and $\sum_{p\to q} \phi(p{\to}q) = 0$ for every vertex p. We represent circulations by column vectors in $\mathbb{R}^E$, indexed by the edges of G, where $\phi_e = \phi(e^+)$. Let Λ denote the $2 \times E$ matrix whose columns are the homology vectors of the reference darts in G. The homology class of a circulation is the matrix-vector product

$$[\phi] \;=\; \Lambda\phi \;=\; \sum_{e\in E} \phi(e^+) \cdot [e^+].$$

(This identity directly generalizes our earlier definition of the homology class $[\gamma]$ of a cycle γ.)

2.4 Geodesic Drawings and Embeddings

A **geodesic drawing** of G on any flat torus $\mathbb{T}_M$ is a drawing that maps edges to geodesics; similarly, a **geodesic embedding** is an embedding that maps edges to geodesics. Equivalently, an embedding is geodesic if its universal cover $\widetilde{G}$ is a straight-line plane graph.

A geodesic drawing of G in $\mathbb{T}_M$ is uniquely determined by its **coordinate representation**, which consists of a coordinate vector $\langle p\rangle \in \Diamond_M$ for each vertex p, together with the homology vector $[e^+] \in \mathbb{Z}^2$ of each edge e.

The **displacement vector** Δ_d of any dart d is the difference between the head and tail coordinates of any lift of d in the universal cover $\widetilde{G}$. Displacement vectors can be equivalently defined in terms of vertex coordinates, homology vectors, and the shape matrix M as follows:

$$\Delta_{p\to q} := \langle q\rangle - \langle p\rangle + M\,[p{\to}q].$$

Reversing a dart negates its displacement: $\Delta_{q\to p} = -\Delta_{p\to q}$. We sometimes write Δx_d and Δy_d to denote the first and second coordinates of Δ_d. The **displacement matrix** Δ of a geodesic drawing is the $2 \times E$ matrix whose columns are the displacement vectors of the reference darts of G. Every geodesic drawing on $\mathbb{T}_M$ is determined *up to translation* by its displacement matrix.

On the *square* flat torus, the integer homology class of any directed cycle is also equal to the sum of the *displacement* vectors of its darts:

$$[\gamma] \;=\; \sum_{p\to q\in\gamma} [p{\to}q] \;=\; \sum_{p\to q\in\gamma} \Delta_{p\to q}.$$

In particular, the total displacement of any contractible cycle is zero, as expected. Extending this identity to circulations by linearity gives us the following useful lemma:

▶ **Lemma 2.1.** *Fix a geodesic drawing of a graph G on $\mathbb{T}_\square$ with displacement matrix Δ. For any circulation ϕ in G, we have $\Delta\phi = \Lambda\phi = [\phi]$.*

2.5 Equilibrium Stresses and Spring Embeddings

A **stress** in a geodesic torus graph G is a real vector $\omega \in \mathbb{R}^E$ indexed by the edges of G. Unlike circulations, homology vectors, and displacement vectors, stresses can be viewed as *symmetric* functions on the *darts* of G. An **equilibrium stress** in G is a stress ω that satisfies the following identity at every vertex p:

$$\sum_{p\to q} \omega_{pq}\Delta_{p\to q} = \begin{pmatrix}0\\0\end{pmatrix}.$$

Unlike Borcea and Streinu [11, 10, 9], we consider only **positive** equilibrium stresses, where $\omega_e > 0$ for every edge e. It may be helpful to imagine each stress coefficient ω_e as a linear spring constant; intuitively, each edge pulls its endpoints inward, with a force equal to the length of e times the stress coefficient ω_e.

Recall that the linear map $M\colon \mathbb{R}^2 \times \mathbb{R}^2$ associated with any nonsingular 2×2 matrix induces a homeomorphism $\underline{M}\colon \mathbb{T}_\square \to \mathbb{T}_M$. In particular, applying this homeomorphism to a geodesic graph in $\mathbb{T}_\square$ with displacement matrix Δ yields a geodesic graph on $\mathbb{T}_M$ with displacement matrix $M\Delta$. Routine definition-chasing now implies the following lemma.

▶ **Lemma 2.2.** *Let G be a geodesic graph on the square flat torus $\mathbb{T}_\square$. If ω is an equilibrium stress for G, then ω is also an equilibrium stress for the image of G on any other flat torus $\mathbb{T}_M$.*

Our results rely on the following natural generalization of Tutte's spring embedding theorem to flat torus graphs.

▶ **Theorem 2.3** (Colin de Verdiére [18]; see also [28, 48, 36])**.** *Let G be any essentially simple, essentially 3-connected embedded graph on any flat torus $\mathbb{T}$, and let ω be any positive stress on the edges of G. Then G is homotopic to a geodesic embedding in $\mathbb{T}$ that is in equilibrium with respect to ω; moreover, this equilibrium embedding is unique up to translation.*

Theorem 2.3 implies the following sufficient condition for a displacement matrix to describe a geodesic embedding on the square torus.

▶ **Lemma 2.4.** *Fix an essentially simple, essentially 3-connected graph G on $\mathbb{T}_\square$, a $2 \times E$ matrix Δ, and a positive stress vector ω. Suppose for every directed cycle (and therefore any circulation) ϕ in G, we have $\Delta\phi = \Lambda\phi = [\phi]$. Then Δ is the displacement matrix of a geodesic **drawing** on $\mathbb{T}_\square$ that is homotopic to G. If in addition ω is an equilibrium stress for that drawing, the drawing is an embedding.*

Proof. A result of Ladegaillerie [45] implies that two embeddings of a graph on the same surface are homotopic if the images of each directed cycle are homotopic. Since homology and homotopy coincide on the torus, the assumption $\Delta\phi = \Lambda\phi = [\phi]$ for every directed cycle immediately implies that Δ is the displacement matrix of a geodesic drawing that is homotopic to G.

If ω is an equilibrium stress for that drawing, then the uniqueness clause in Theorem 2.3 implies that the drawing is in fact an embedding. ◀

Following Steiner and Fischer [68] and Gortler, Gotsman, and Thurston [36], given the coordinate representation of any geodesic graph G on the square flat torus, with any positive stress vector $\omega > 0$, we can compute an isotopic equilibrium embedding of G by solving the linear system

$$\sum_{p \to q} \omega_{pq}\big(\langle q\rangle - \langle p\rangle + [p{\to}q]\big) = \begin{pmatrix}0\\0\end{pmatrix} \qquad \text{for every vertex } q$$

for the vertex locations $\langle p\rangle$, treating the homology vectors $[p{\to}q]$ as constants. Alternatively, Lemma 2.4 implies that we can compute the displacement vectors of every isotopic equilibrium embedding directly, by solving the linear system

$$\sum_{p\to q} \omega_{pq}\Delta_{p\to q} = \begin{pmatrix}0\\0\end{pmatrix} \qquad \text{for every vertex } q$$

$$\sum_{left(d)=f} \Delta_d = \begin{pmatrix}0\\0\end{pmatrix} \qquad \text{for every face } f$$

$$\sum_{d\in\gamma_1} \Delta_d = [\gamma_1]$$

$$\sum_{d\in\gamma_2} \Delta_d = [\gamma_2]$$

where γ_1 and γ_2 are any two directed cycles with independent non-zero homology classes.

2.6 Duality and Reciprocality

Every embedded torus graph G defines a **dual graph** G^* whose vertices correspond to the faces of G, where two vertices in G are connected by an edge for each edge separating the corresponding pair of faces in G. This dual graph G^* has a natural embedding in which each vertex f^* of G^* lies in the interior of the corresponding face f of G, each edge e^* of G^* crosses only the corresponding edge e of G, and each face p^* of G^* contains exactly one vertex p of G in its interior. We regard any embedding of G^* to be *dual* to G if and only if it is homotopic to this natural embedding. Each dart d in G has a corresponding dart d^* in G^*, defined by setting $head(d^*) = left(d)^*$ and $tail(d^*) = right(d^*)$; intuitively, the dual of a dart in G is obtained by rotating the dart counterclockwise.

It will prove convenient to treat vertex coordinates, displacement vectors, homology vectors, and circulations in any dual graph G^* as *row* vectors. For any vector $v \in \mathbb{R}^2$ we define $v^\perp := (Jv)^T$, where $\mathbf{J} := \left(\begin{smallmatrix}0 & -1\\ 1 & 0\end{smallmatrix}\right)$ is the matrix for a 90° counterclockwise rotation. Similarly, for any $2 \times n$ matrix A, we define $A^\perp := (JA)^T = -A^T J$.

Two dual geodesic graphs G and G^* on the same flat torus $\mathbb{T}$ are **reciprocal** if every edge e in G is orthogonal to its dual edge e^* in G^*.

A **cocirculation** in G a *row* vector $\theta \in \mathbb{R}^E$ whose transpose describes a circulation in G^*. The *cohomology* class $[\theta]^*$ of any cocirculation is the transpose of the homology class of the circulation θ^T in G^*. Recall that Λ is the $2 \times E$ matrix whose columns are homology vectors of edges in G. Let λ_1 and λ_2 denote the first and second rows of Λ. The following lemma is illustrated in Figure 2; we defer the proof to the full version of the paper [33].

▶ **Lemma 2.5.** *The row vectors λ_1 and λ_2 describe cocirculations in G with cohomology classes $[\lambda_1]^* = (0\ 1)$ and $[\lambda_2]^* = (-1\ 0)$.*

2.7 Coherent Subdivisions

Let G be a geodesic graph in $\mathbb{T}_M$, and fix arbitrary real weights π_p for every vertex p of G. Let $p{\to}q$, $p{\to}r$, and $p{\to}s$ be three consecutive darts around a common tail p in clockwise order. Thus, $left(p{\to}q) = right(p{\to}r)$ and $left(p{\to}r) = right(p{\to}s)$. We call the edge pr **locally Delaunay** if the following determinant is positive:

$$\begin{vmatrix} \Delta x_{p\to q} & \Delta y_{p\to q} & \frac{1}{2}|\Delta_{p\to q}|^2 + \pi_p - \pi_q \\ \Delta x_{p\to r} & \Delta y_{p\to r} & \frac{1}{2}|\Delta_{p\to r}|^2 + \pi_p - \pi_r \\ \Delta x_{p\to s} & \Delta y_{p\to s} & \frac{1}{2}|\Delta_{p\to s}|^2 + \pi_p - \pi_s \end{vmatrix} > 0. \tag{2.1}$$

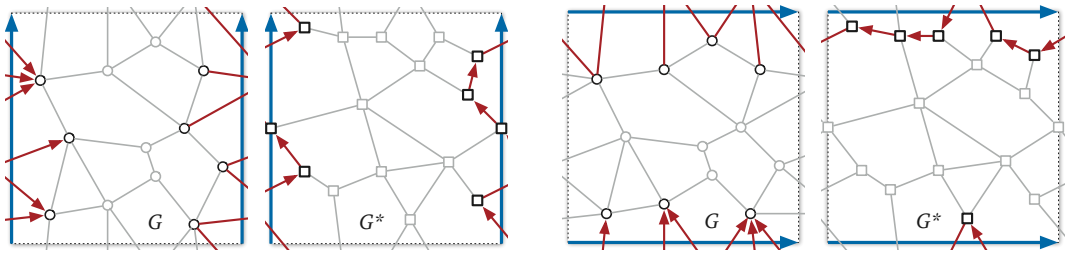


Figure 2 Proof of Lemma 2.5: The darts in G crossing either boundary edge of the fundamental square dualize to a closed walk in G^* parallel to that boundary edge.

This inequality follows by elementary row operations and cofactor expansion from the standard determinant test for appropriate lifts of the vertices p, q, r, s to the universal cover:

$$\begin{vmatrix} 1 & x_p & y_p & \frac{1}{2}(x_p^2 + y_p^2) - \pi_p \\ 1 & x_q & y_q & \frac{1}{2}(x_q^2 + y_q^2) - \pi_q \\ 1 & x_r & y_r & \frac{1}{2}(x_r^2 + y_r^2) - \pi_r \\ 1 & x_s & y_s & \frac{1}{2}(x_s^2 + y_s^2) - \pi_s \end{vmatrix} > 0. \tag{2.2}$$

(The factor $1/2$ simplifies our later calculations, and is consistent with Maxwell's construction of polyhedral liftings and reciprocal diagrams.) Similarly, we say that an edge is **locally flat** if the corresponding determinant is zero. Finally, G is the **weighted Delaunay graph** of its vertices if every edge of G is locally Delaunay and every diagonal of every non-triangular face is locally flat.

One can easily verify that this condition is equivalent to G being the projection of the weighted Delaunay graph of the lift $\pi_M^{-1}(V)$ of its vertices V to the universal cover. Results of Bobenko and Springborn [7] imply that any finite set of weighted points on any flat torus has a unique weighted Delaunay graph. We emphasize that weighted Delaunay graphs are *not* necessarily either simple or triangulations; however, every weighted Delaunay graphs on any flat torus is both essentially simple and essentially 3-connected. The dual **weighted Voronoi graph** of P, also known as its *power diagram* [4, 6], can be defined similarly by projection from the universal cover.

Finally, a geodesic torus graph is **coherent** if it is the weighted Delaunay graph of its vertices, with respect to some vector of weights.

3 Reciprocal Implies Equilibrium

▶ **Lemma 3.1.** *Let G and G^* be reciprocal geodesic graphs on some flat torus $\mathbb{T}_M$. The vector ω defined by $\omega_e = |e^*|/|e|$ is an equilibrium stress for G; symmetrically, the vector ω^* defined by $\omega^*_{e^*} = 1/\omega_e = |e|/|e^*|$ is an equilibrium stress for G^*.*

Proof. Let $\omega_e = |e^*|/|e|$ and $\omega^*_{e^*} = 1/\omega_e = |e|/|e^*|$ for each edge e. Let Δ denote the displacement matrix of G, and let Δ^* denote the (transposed) displacement matrix of G^*. We immediately have $\Delta^*_{e^*} = \omega_e \Delta_e^\perp$ for every edge e of G. The darts leaving each vertex p of G dualize to a facial cycle around the corresponding face p^* of G^*, and thus

$$\left(\sum_{q\colon pq \in E} \omega_{pq} \Delta_{p\to q} \right)^\perp = \sum_{q\colon pq\in E} \omega_{pq} \Delta^\perp_{p\to q} = \sum_{q\colon pq\in E} \Delta^*_{(p\to q)^*} = (0\ 0)\,.$$

We conclude that ω is an equilibrium stress for G, and thus (by swapping the roles of G and G^*) that ω^* is an equilibrium stress for G^*. ◀

A stress vector ω is a **reciprocal stress** for G if there is a reciprocal graph G^* on the same flat torus such that $\omega_e = |e^*|/|e|$ for each edge e. Thus, a geodesic torus graph is reciprocal if and only if it has a reciprocal stress.

▶ **Theorem 3.2.** *Not every positive equilibrium stress for G is a reciprocal stress. More generally, not every equilibrium graph on $\mathbb{T}$ is reciprocal/coherent on $\mathbb{T}$.*

Proof. Let G_1 be the geodesic triangulation in the flat square torus $\mathbb{T}_\square$ with a single vertex p and three edges, whose reference darts have displacement vectors $\binom{1}{0}$, $\binom{1}{1}$, and $\binom{2}{1}$. Every stress ω in G is an equilibrium stress, because the forces applied by each edge cancel out. The weighted Delaunay graph of a single point is identical for all weights, so it suffices to verify that G_1 is not an intrinsic Delaunay triangulation. We easily observe that the longest edge of G_1 is not Delaunay. See Figure 3.

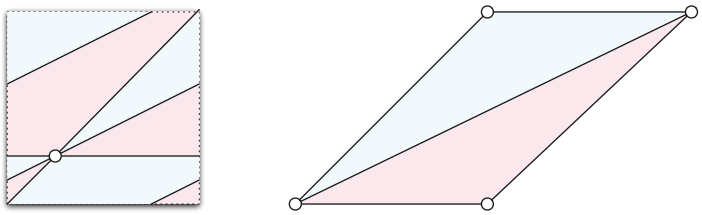


Figure 3 A one-vertex triangulation G_1 on the square flat torus, and a lift of its faces to the universal cover. Every stress in G_1 is an equilibrium stress, but G_1 is not a (weighted) intrinsic Delaunay triangulation.

More generally, for any positive integer k, let G_k denote the $k \times k$ covering of G_1. The vertices of G_k form a regular $k \times k$ square toroidal lattice, and the edges of G_k fall into three parallel families, with displacement vectors $\binom{1/k}{1/k}$, $\binom{2/k}{1/k}$, and $\binom{1/k}{0}$. Every positive stress vector where all parallel edges have equal stress coefficients is an equilibrium stress.

For the sake of argument, suppose G_k is coherent. Let $p \to r$ be any dart with displacement vector $\binom{2/k}{1/k}$, and let q and s be the vertices before and after r in clockwise order around p. The local Delaunay determinant test implies that the weights of these four vertices satisfy the inequality $\pi_p + \pi_r + 1 < \pi_q + \pi_s$. Every vertex of G_k appears in exactly four inequalities of this form – twice on the left and twice on the right – so summing all k^2 such inequalities and canceling equal terms yields the obvious contradiction $1 < 0$. ◀

Every equilibrium stress on any graph G on any flat torus induces an equilibrium stress on the universal cover $\widetilde{G}$, which in turn induces a reciprocal diagram $(\widetilde{G})^*$, which is periodic. Typically, however, for almost all equilibrium stresses, $(\widetilde{G})^*$ is periodic with respect to a different lattice than $\widetilde{G}$. We describe a simple necessary and sufficient condition for an equilibrium stress to be reciprocal in Section 5.

4 Coherent iff Reciprocal

Unlike in the previous and following sections, the equivalence between coherent graphs and graphs with reciprocal diagrams generalizes fully from the plane to the torus.

4.1 Notation

In this section we fix a non-singular matrix $M = (u\ v)$ where $u, v \in \mathbb{R}^2$ are column vectors and $\det M > 0$. We primarily work with the universal cover $\widetilde{G}$ of G; if we are given a reciprocal embedding G^*, we also work with its universal cover $\widetilde{G}^*$ (which is reciprocal to $\widetilde{G}$). Vertices in $\widetilde{G}$ are denoted by the letters p and q and treated as column vectors

in $\mathbb{R}^2$. A generic face in $\widetilde{G}$ is denoted by the letter f; the corresponding dual vertex in $\widetilde{G}^*$ is denoted f^* and interpreted as a row vector. To avoid nested subscripts when edges are indexed, we write $\Delta_i = \Delta_{e_i}$ and $\omega_i = \omega_{e_i}$, and therefore by Lemma 3.1, $\Delta_i^* = \omega_i \Delta_i^\perp$. For any integers a and b, the translation $p + au + bv$ of any vertex p of $\widetilde{G}$ is another vertex of $\widetilde{G}$, and the translation $f + au + bv$ of any face f of $\widetilde{G}$ is another face of $\widetilde{G}$.

4.2 Results

The following lemma follows directly from the definitions of weighted Delaunay graphs and their dual weighted Voronoi diagrams; see, for example, Aurenhammer [4, 6].

▶ **Lemma 4.1.** *Let G be a weighted Delaunay graph on some flat torus $\mathbb{T}$, and let G^* be the corresponding weighted Voronoi diagram on $\mathbb{T}$. Every edge e of G is orthogonal to its dual e^*. In short, every coherent torus graph is reciprocal.*

Maxwell's theorem implies a convex polyhedral lifting $z\colon \mathbb{R}^2 \to \mathbb{R}$ of the universal cover $\widetilde{G}$ of G, where the gradient vector $\nabla z|_f$ within any face f is equal to the coordinate vector of the dual vertex f^* in $\widetilde{G}^*$. To make this lifting unique, we fix a vertex o of $\widetilde{G}$ to lie at the origin $\binom{0}{0}$, and we require $z(o) = 0$.

Define the weight of each vertex $p \in \widetilde{G}$ as $\pi_p := \frac{1}{2}|p|^2 - z(p)$. The determinant conditions (2.1) and (2.2) for an edge to be locally Delaunay are both equivalent to interpreting $\frac{1}{2}|p|^2 - \pi_p$ as a z-coordinate and requiring that the induced lifting be locally convex at said edge. Because z is a convex polyhedral lifting, $\widetilde{G}$ is the intrinsic weighted Delaunay graph of its vertex set with respect to these weights.

To compute $z(q)$ for any point $q \in \mathbb{R}^2$, we choose an arbtirary face f containing q and identify the equation of the plane through the lift of f, that is, $z|_f(q) = \eta q + c$ where η is a row vector and $c \in \mathbb{R}$. Borcea and Streinu [11] give a calculation for η and c, which for our setting can be written as follows:

▶ **Lemma 4.2** ([11, Eq. 7])**.** *For $q \in \mathbb{R}^2$, let f be a face containing q. The function $z|_f$ can be explicitly computed as follows:*

- *Pick an arbitrary* ***root*** *face f_0 incident to o.*
- *Pick an arbitrary path from f_0^* to f^* in $\widetilde{G}^*$, and let $e_1^*, \ldots, e_\ell^*$ be the dual edges along this path. By definition, $f^* = f_0^* + \sum_{i=1}^{\ell} \Delta_i^*$. Set $C(f) = z(o) + \sum_{i=1}^{\ell} \omega_i \, |p_i \; q_i|$, where $e_i = p_i \to q_i$ and $|p_i \; q_i| = \det(p_i \; q_i)$.*
- *Set $\eta = f^*$ and $c = C(f)$, implying that $z|_f(q) = f^* q + C(f)$. In particular, $C(f)$ is the intersection of this plane with the z-axis.*

Reciprocality of $\widetilde{G}^*$ implies that the actual choice of root face f_0^* and the path to f^* do not matter. We use this explicit computation to establish the existence of a translation of G^* such that $\pi_o = \pi_u = \pi_v = 0$. We then show that after this translation, every lift of the same vertex of G has the same Delaunay weight.

▶ **Lemma 4.3.** *There is a unique translation of $\widetilde{G}^*$ such that $\pi_u = \pi_v = 0$. Specifically, this translation places the dual vertex of the root face f_0 at the point*

$$f_0^* = \left(-\tfrac{1}{2}\left(|u|^2 \;\; |v|^2\right) - \left(C(f_0 + u) \;\; C(f_0 + v)\right)\right) M^{-1}.$$

Proof. Lemma 4.2 implies that

$$z(u) \;=\; (f_0 + u)^* u + C(f_0 + u) \;=\; f_0^* u + |u|^2 + C(f_0 + u),$$

and by definition, $\pi_u = 0$ if and only if $z(u) = \frac{1}{2}|u|^2$. Thus, $\pi_u = 0$ if and only if $f_0^* u = -\frac{1}{2}|u|^2 - C(f_0 + u)$. A symmetric argument implies $\pi_v = 0$ if and only if $f_0^* v = -\frac{1}{2}|v|^2 - C(f_0 + v)$. ◀

We defer the proof of the following lemma to the full version of the paper [33].

▶ **Lemma 4.4.** *If $\pi_o = \pi_u = \pi_v = 0$, then $\pi_p = \pi_{p+u} = \pi_{p+v}$ for all $p \in V(\widetilde{G})$. In other words, all lifts of any vertex of G have equal weight.*

The previous two lemmas establish the existence of a set of periodic weights with respect to which $\widetilde{G}$ is the weighted Delaunay complex of its point set, and a unique translation of $\widetilde{G}^*$ that is the corresponding intrinsic weighted Voronoi diagram. Projecting from the universal cover back to the torus, we conclude:

▶ **Theorem 4.5.** *Let G and G^* be reciprocal geodesic graphs on some flat torus $\mathbb{T}_M$. G is a weighted Delaunay complex, and a unique translation of G^* is the corresponding weighted Voronoi diagram. In short, every reciprocal torus graph is coherent.*

5 Equilibrium Implies Reciprocal, Sort Of

In this section, we will fix a positive equilibrium stress ω. It will be convenient to represent ω as the $E \times E$ diagonal stress *matrix* Ω whose diagonal entries are $\Omega_{e,e} = \omega_e$.

Let G be an essentially simple, essentially 3-connected geodesic graph on the *square* flat torus $\mathbb{T}_\square$, and let Δ be its $2 \times E$ displacement matrix. Our results are phrased in terms of the covariance matrix $\Delta\Omega\Delta^T = \left(\begin{smallmatrix}\alpha & \gamma \\ \gamma & \beta\end{smallmatrix}\right)$, where

$$\alpha = \sum_e \omega_e \Delta x_e^2, \qquad \beta = \sum_e \omega_e \Delta y_e^2, \qquad \gamma = \sum_e \omega_e \Delta x_e \Delta y_e. \tag{5.1}$$

Recall that $A^\perp = (JA)^T$.

5.1 The Square Flat Torus

Before considering arbitrary flat tori, as a warmup we first establish necessary and sufficient conditions for ω to be a reciprocal stress for G on the *square* flat torus $\mathbb{T}_\square$, in terms of the parameters α, β, and γ.

▶ **Lemma 5.1.** *If ω is a reciprocal stress for G on $\mathbb{T}_\square$, then $\Delta\Omega\Delta^T = \left(\begin{smallmatrix}1 & 0 \\ 0 & 1\end{smallmatrix}\right)$.*

Proof. Suppose ω is a reciprocal stress for G on $\mathbb{T}_\square$. Then there is a geodesic embedding of the dual graph G^* on $\mathbb{T}_\square$ where $e \perp e^*$ and $|e^*| = \omega_e |e|$ for every edge e of G. Let $\Delta^* = (\Delta\Omega)^\perp$ denote the $E \times 2$ matrix whose rows are the displacement row vectors of G^*.

Recall from Lemma 2.5 that the first and second rows of Λ describe cocirculations of G with cohomology classes $(0\ 1)$ and $(-1\ 0)$, respectively. Applying Lemma 2.1 to G^* implies $\theta\Delta^* = [\theta]^*$ for any cocirculation θ in G. It follows immediately that $\Lambda\Delta^* = \left(\begin{smallmatrix}0 & 1 \\ -1 & 0\end{smallmatrix}\right) = -J$.

Because the rows of Δ^* are displacement vectors of G^*, for every vertex p of G we have

$$\sum_{q\colon pq \in E} \Delta^*_{(p\to q)^*} = \sum_{d\colon tail(d)=p} \Delta^*_{d^*} = \sum_{d\colon left(d^*)=p^*} \Delta^*_{d^*} = (0\ 0)\,. \tag{5.2}$$

It follows that the *columns* of Δ^* describe circulations in G. Lemma 2.1 now implies that $\Delta\Delta^* = -J$. We conclude that $\Delta\Omega\Delta^T = \Delta\Delta^* J = \left(\begin{smallmatrix}1 & 0 \\ 0 & 1\end{smallmatrix}\right)$. ◀

▶ **Lemma 5.2.** *Fix an $E \times 2$ matrix Δ^*. If $\Lambda\Delta^* = -J$, then Δ^* is the displacement matrix of a geodesic drawing on $\mathbb{T}_\square$ that is dual to G. Moreover, if that drawing has an equilibrium stress, it is actually an embedding.*

Proof. Let λ_1 and λ_2 denote the rows of Λ. Rewriting the identity $\Lambda\Delta^* = -J$ in terms of these row vectors gives us $\sum_e \Delta^*_e \lambda_{1,e} = (0\ 1) = [\lambda_1]^*$ and $\sum_e \Delta^*_e \lambda_{2,e} = (-1\ 0) = [\lambda_2]^*$. Because $[\lambda_1]^*$ and $[\lambda_2]^*$ are linearly independent, we have $\sum_e \Delta^*_e \theta_e = [\theta]^*$ for any cocirculation θ in G^*. The result follows from Lemma 2.4. ◀

▶ **Lemma 5.3.** *If $\Delta\Omega\Delta^T = \left(\begin{smallmatrix}1 & 0\\ 0 & 1\end{smallmatrix}\right)$, then ω is a reciprocal stress for G on $\mathbb{T}_\square$.*

Proof. Set $\Delta^* = (\Delta\Omega)^\perp$. Because ω is an equilibrium stress in G, for every vertex p of G we have

$$\sum_{q\colon pq\in E} \Delta^*_{(p\to q)^*} \;=\; \sum_{q\colon pq\in E} \omega_{pq}\Delta_{p\to q} \;=\; \begin{pmatrix}0\\0\end{pmatrix}. \tag{5.3}$$

It follows that the columns of Δ^* describe circulations in G, and therefore Lemma 2.1 implies $\Lambda\Delta^* = \Delta\Delta^* = \Delta(\Delta\Omega)^\perp = \Delta\Omega\Delta^T J^T = -J$.

Lemma 5.2 now implies that Δ^* is the displacement matrix of an drawing G^* dual to G. Moreover, the stress vector ω^* defined by $\omega^*_{e^*} = 1/\omega_e$ is an equilibrium stress for G^*: under this stress vector, the darts leaving any dual vertex f^* are dual to the clockwise boundary cycle of face f in G. Thus G^* is in fact an embedding. By construction, each edge of G^* is orthogonal to the corresponding edge of G. ◀

5.2 Arbitrary Flat Tori

In the full version of the paper [33], we generalize our previous analysis to graphs on the flat torus $\mathbb{T}_M$ defined by an arbitrary non-singular matrix $M = \left(\begin{smallmatrix}a & b\\ c & d\end{smallmatrix}\right)$. These results are stated in terms of the covariance parameters α, β, and γ, which are still defined in terms of $\mathbb{T}_\square$.

▶ **Lemma 5.4.** *If ω is a reciprocal stress for the affine image of G on $\mathbb{T}_M$, then $\alpha\beta - \gamma^2 = 1$; in particular, if $M = \left(\begin{smallmatrix}a & b\\ c & d\end{smallmatrix}\right)$, then*

$$\alpha = \frac{b^2+d^2}{ad-bc}, \qquad \beta = \frac{a^2+c^2}{ad-bc}, \qquad \gamma = \frac{-(ab+cd)}{ad-bc}.$$

▶ **Corollary 5.5.** *If ω is a reciprocal stress for the image of G on $\mathbb{T}_M$, then $M = \sigma R\left(\begin{smallmatrix}\beta & -\gamma\\ 0 & 1\end{smallmatrix}\right)$ for some 2×2 rotation matrix R and some real number $\sigma > 0$.*

▶ **Lemma 5.6.** *If $\alpha\beta - \gamma^2 = 1$ and $M = \sigma R\left(\begin{smallmatrix}\beta & -\gamma\\ 0 & 1\end{smallmatrix}\right)$ for any 2×2 rotation matrix R and any real number $\sigma > 0$, then ω is a reciprocal stress for the image G on $\mathbb{T}_M$.*

▶ **Theorem 5.7.** *Let G be a geodesic graph on $\mathbb{T}_\square$ with positive equilibrium stress ω. Let α, β, and γ be defined as in Equation (5.1). If $\alpha\beta - \gamma^2 = 1$, then ω is a reciprocal stress for the image of G on the flat torus $\mathbb{T}_M$ if and only if $M = \sigma R\left(\begin{smallmatrix}\beta & -\gamma\\ 0 & 1\end{smallmatrix}\right)$ for some (in fact any) rotation matrix R and real number $\sigma > 0$. On the other hand, if $\alpha\beta - \gamma^2 \neq 1$, then ω is not a reciprocal stress for G on any flat torus $\mathbb{T}_M$.*

Theorem 5.7 immediately implies that every equilibrium graph on any flat torus has a coherent affine image on *some* flat torus. The requirement $\alpha\beta - \gamma^2 = 1$ is a necessary scaling condition: Given any equilibrium stress ω, the scaled equilibrium stress $\omega/\sqrt{\alpha\beta-\gamma^2}$ satisfies the requirement.

6 A Toroidal Steinitz Theorem

Finally, Theorem 2.3 and Theorem 5.7 immediately imply a natural generalization of Steinitz's theorem to graphs on the flat torus.

▶ **Theorem 6.1.** *Let G be any essentially simple, essentially 3-connected embedded graph on the square flat torus $\mathbb{T}_\square$, and let ω be **any** positive stress on the edges of G. Then G is homotopic to a geodesic embedding in $\mathbb{T}_\square$ whose image in some flat torus $\mathbb{T}_M$ is coherent.*

As we mentioned in the introduction, Mohar's generalization [55] of the Koebe-Andreev circle packing theorem already implies that every essentially simple, essentially 3-connected torus graph G is homotopic to *one* coherent homotopic embedding on *one* flat torus. In contrast, Lemma 3.1 and Theorem 6.1 characterize *all* coherent homotopic embeddings of G on *all* flat tori; *every* positive vector $\omega \in \mathbb{R}^E$ corresponds to such an embedding.

References

1 E. M. Andreev. Convex polyhedra in Lobačevskiĭ space. *Mat. Sbornik*, 10(3):413–440, 1970. doi:10.1070/SM1970v010n03ABEH001677.

2 E. M. Andreev. On convex polyhedra of finite volume in Lobačevskiĭ space. *Mat. Sbornik*, 12(2):270–259, 1970. doi:10.1070/SM1970v012n02ABEH000920.

3 Franz Aurenhammer. A criterion for the affine equivalence of cell complexes in R^d and convex polyhedra in R^{d+1}. *Discrete Comput. Geom.*, 2(1):49–64, 1987. doi:10.1007/BF02187870.

4 Franz Aurenhammer. Power diagrams: Properties, algorithms and applications. *SIAM J. Comput.*, 16(1):78–96, 1987. doi:10.1137/0216006.

5 Franz Aurenhammer. Recognising polytopical cell complexes and constructing projection polyhedra. *J. Symb. Comput.*, 3(3):249–255, 1987. doi:10.1016/S0747-7171(87)80003-2.

6 Franz Aurenhammer and Hiroshi Imai. Geometric relations among Voronoi diagrams. *Geom. Dedicata*, 27(1):65–75, 1988. doi:10.1007/BF00181613.

7 Alexander I. Bobenko and Boris A. Springborn. A discrete Laplace-Beltrami operator for simplicial surfaces. *Discrete Comput. Geom.*, 38(4):740–756, 2007. doi:10.1007/s00454-007-9006-1.

8 Mikhail Bogdanov, Monique Teillaud, and Gert Vegter. Delaunay triangulations on orientable surfaces of low genus. In Sándor Fekete and Anna Lubiw, editors, *Proc. 32nd Int. Symp. Comput. Geom.*, number 51 in Leibniz Int. Proc. Informatics, pages 20:1–20:17, 2016. doi:10.4230/LIPIcs.SoCG.2016.20.

9 Ciprian Borcea and Ileana Streinu. Periodic frameworks and flexibility. *Proc. Royal Soc. A*, 466(2121):2633–2649, 2010. doi:10.1098/rspa.2009.0676.

10 Ciprian Borcea and Ileana Streinu. Minimally rigid periodic graphs. *Bull. London Math. Soc.*, 43(6):1093–1103, 2011. doi:10.1112/blms/bdr044.

11 Ciprian Borcea and Ileana Streinu. Liftings and stresses for planar periodic frameworks. *Discrete Comput. Geom.*, 53(4):747–782, 2015. doi:10.1007/s00454-015-9689-7.

12 Graham R. Brightwell and Edward R. Scheinerman. Representations of planar graphs. *SIAM J. Discrete Math.*, 6(2):214–229, 1993. doi:10.1137/0406017.

13 Kevin Q. Brown. Voronoi diagrams from convex hulls. *Inform. Process. Lett.*, 9(5):223–228, 1979. doi:10.1016/0020-0190(79)90074-7.

14 Manuel Caroli and Monique Teillaud. Delaunay triangulations of closed Euclidean d-orbifolds. *Discrete Comput. Geom.*, 55(4):827–853, 2016. doi:10.1007/s00454-016-9782-6.

15 Marek Chrobak, Michael T. Goodrich, and Roberto Tamassia. Convex drawings of graphs in two and three dimensions (preliminary version). In *Proc. 12th Ann. Symp. Comput. Geom.*, pages 319–328, 1996. doi:10.1145/237218.237401.

16 Yves Colin de Verdière. Empilements de cercles: Convergence d'une méthode de point fixe. *Forum Math.*, 1(1):395–402, 1989. doi:10.1515/form.1989.1.395.

17 Yves Colin de Verdière. Sur un nouvel invariant des graphes et un critère de planarité. *J. Comb. Theory Ser. B*, 50(1):11–21, 1990. In French, English translation in [20]. doi:10.1016/0095-8956(90)90093-F.

18 Yves Colin de Verdière. Comment rendre géodésique une triangulation d'une surface? *L'Enseignment Mathématique*, 37:201–212, 1991. doi:10.5169/seals-58738.

19 Yves Colin de Verdière. Un principe variationnel pour les empilements de cercles. *Invent. Math.*, 104(1):655–669, 1991. doi:10.1007/BF01245096.

20 Yves Colin de Verdière. On a new graph invariant and a criterion for planarity. In Neil Robertson and Paul Seymour, editors, *Graph Structure Theory*, number 147 in Contemporary Mathematics, pages 137–147. Amer. Math. Soc., 1993. English translation of [17] by Neil Calkin.

21 Éric Colin de Verdière and Arnaud de Mesmay. Testing graph isotopy on surfaces. *Discrete Comput. Geom.*, 51(1):171–206, 2014. doi:10.1007/s00454-013-9555-4.

22 Robert Connelly, Erik D. Demaine, and Günter Rote. Infinitesimally locked self-touching linkages with applications to locked trees. In Jorge Calvo, Kenneth Millett, and Eric Rawdon, editors, *Physical Knots: Knotting, Linking, and Folding of Geometric Objects in* $\mathbb{R}^3$, pages 287–311. Amer. Math. Soc., 2002.

23 Keenan Crane. Discrete differential geometry: An applied introduction, 2019. URL: http://www.cs.cmu.edu/~kmcrane/Projects/DDG/paper.pdf.

24 Henry Crapo and Walter Whiteley. Plane self stresses and projected polyhedra I: The basic pattern. *Topologie structurale / Structural Topology*, 20:55–77, 1993. URL: http://hdl.handle.net/2099/1091.

25 Luigi Cremona. *Le figure reciproche nella statica grafica.* Tipografia di Giuseppe Bernardoni, 1872. English translation in [26]. URL: http://www.luigi-cremona.it/download/Scritti_matematici/1872_statica_grafica.pdf.

26 Luigi Cremona. *Graphical Statics.* Oxford Univ. Press, 1890. English translation of [25] by Thomas Hudson Beare. URL: https://archive.org/details/graphicalstatic02cremgoog.

27 Jesús De Loera, Jörg Rambau, and Francisco Santos. *Triangulations: Structures for Algorithms and Applications.* Number 25 in Algorithms and Computation in Mathematics. Springer, 2010. doi:10.1007/978-3-642-12971-1.

28 Olaf Delgado-Friedrichs. Equilibrium placement of periodic graphs and convexity of plane tilings. *Discrete Comput. Geom.*, 33(1):67–81, 2004. doi:10.1007/s00454-004-1147-x.

29 Erik D. Demaine and Joseph O'Rourke. *Geometric Folding Algorithms: Linkages, Origami, Polyhedra.* Cambridge Univ. Press, 2007.

30 Erik D. Demaine and André Schulz. Embedding stacked polytopes on a polynomial-size grid. *Discrete Comput. Geom.*, 57(4):782–809, 2017. doi:10.1007/s00454-017-9887-6.

31 Peter Eades and Patrick Garvan. Drawing stressed planar graphs in three dimensions. In *Proc. 2nd Symp. Graph Drawing*, number 1027 in Lecture Notes Comput. Sci., pages 212–223, 1995. doi:10.1007/BFb0021805.

32 Herbert Edelsbrunner and Raimund Seidel. Voronoi diagrams and arrangements. *Discrete Comput. Geom.*, 1(1):25–44, 1986. doi:10.1007/BF02187681.

33 Jeff Erickson and Patrick Lin. A toroidal Maxwell-Cremona-Delaunay correspondence. Prepint, March 2020.

34 Stefan Felsner and Günter Rote. On Primal-Dual Circle Representations. In *Proc. 2nd Symp. Simplicity in Algorithms*, volume 69 of *OpenAccess Series in Informatics (OASIcs)*, pages 8:1–8:18. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018. doi:10.4230/OASIcs.SOSA.2019.8.

35 Daniel Gonçalves and Benjamin Lévêque. Toroidal maps: Schnyder woods, orthogonal surfaces and straight-line representations. *Discrete Comput. Geom.*, 51(1):67–131, 2014. doi:10.1007/s00454-013-9552-7.

36 Steven J. Gortler, Craig Gotsman, and Dylan Thurston. Discrete one-forms on meshes and applications to 3D mesh parameterization. *Comput. Aided Geom. Design*, 23(2):83–112, 2006. doi:10.1016/j.cagd.2005.05.002.

37 Clara I. Grima and Alberto Márquez. *Computational Geometry on Surfaces*. Springer, 2001.

38 John E. Hopcroft and Peter J. Kahn. A paradigm for robust geometric algorithms. *Algorithmica*, 7(1–6):339–380, 1992. doi:10.1007/BF01758769.

39 David Huffman. A duality concept for the analysis of polyhedral scenes. In Edward W. Elcock and Donald Michie, editors, *Machine Intelligence*, volume 8, pages 475–492. Ellis Horwood Ltd. and John Wiley & Sons, 1977.

40 Alexander Igambardiev and André Schulz. A duality transform for constructing small grid embeddings of 3d polytopes. *Comput. Geom. Theory Appl.*, 56:19–36, 2016. doi:10.1016/j.comgeo.2016.03.004.

41 Clause Indermitte, Thomas M. Liebling, Marc Troyanov, and Heinz Clemençon. Voronoi diagrams on piecewise flat surfaces and an application to biological growth. *Theoret. Comput. Sci.*, 263(1–2):263–274, 2001. doi:10.1016/S0304-3975(00)00248-6.

42 Ivan Izmestiev. Statics and kinematics of frameworks in Euclidean and non-Euclidean geometry. In Vincent Alberge and Athanase Papadopoulos, editors, *Eighteen Essays in Non-Euclidean Geometry*, number 29 in IRMA Lectures in Mathematics and Theoretical Physics. Europ. Math. Soc., 2019. doi:10.4171/196-1/12.

43 Paul Koebe. Kontaktprobleme der Konformen Abbildung. *Ber. Sächs. Akad. Wiss. Leipzig, Math.-Phys. Kl.*, 88:141–164, 1936.

44 Andrew Kotlov, László Lovász, and Santosh Vempala. The Colin de Verdière number and sphere representations of a graph. *Combinatorica*, 17(4):483–521, 1997. doi:10.1007/BF01195002.

45 Yves Ladegaillerie. Classes d'isotopie de plongements de 1-complexes dans les surfaces. *Topology*, 23(3):303–311, 1984.

46 Charles L. Lawson. Transforming triangulations. *Discrete Math.*, 3(4):365–372, 1972. doi:10.1016/0012-365X(72)90093-3.

47 László Lovász. Representations of polyhedra and the Colin de Verdière number. *J. Comb. Theory Ser. B*, 82(2):223–236, 2001. doi:10.1006/jctb.2000.2027.

48 László Lovász. Discrete analytic functions: An exposition. In Alexander Grigor'yan and Shing-Tung Yau, editors, *Eigenvalues of Laplacians and other geometric operators*, volume 9 of *Surveys in Differential Geometry*, pages 241–273. Int. Press, 2004. doi:10.4310/SDG.2004.v9.n1.a7.

49 László Lovász. *Graphs and Geometry*. Number 69 in Colloquium Publications. Amer. Math. Soc., 2019.

50 James Clerk Maxwell. On reciprocal figures and diagrams of forces. *Phil. Mag. (Ser. 4)*, 27(182):250–261, 1864. doi:10.1080/14786446408643663.

51 James Clerk Maxwell. On the application of the theory of reciprocal polar figures to the construction of diagrams of forces. *Engineer*, 24:402, 1867.

52 James Clerk Maxwell. On reciprocal figures, frames, and diagrams of forces. *Trans. Royal Soc. Edinburgh*, 26(1):1–40, 1870. doi:10.1017/S0080456800026351.

53 Maria Mazón and Tomás Recio. Voronoi diagrams on orbifolds. *Comput. Geom. Theory Appl.*, 8(5):219–230, 1997. doi:10.1016/S0925-7721(96)00017-X.

54 Bojan Mohar. A polynomial time circle packing algorithm. *Discrete Math.*, 117(1–3):257–263, 1993. doi:10.1016/0012-365X(93)90340-Y.

55 Bojan Mohar. Circle packings of maps—The Euclidean case. *Rend. Sem. Mat. Fis. Milano*, 67(1):191–206, 1997. doi:10.1007/BF02930499.

56 Bojan Mohar. Circle packings of maps in polynomial time. *Europ. J. Combin.*, 18(7):785–805, 1997. doi:10.1006/eujc.1996.0135.

57 Bojan Mohar and Pierre Rosenstiehl. Tessellation and visibility representations of maps on the torus. *Discrete Comput. Geom.*, 19(2):249–263, 1998. doi:10.1007/PL00009344.

58 Bojan Mohar and Alexander Schrijver. Blocking nonorientability of a surface. *J. Comb. Theory Ser. B*, 87(1):2–16, 2003.

59 Shmuel Onn and Bernd Sturmfels. A quantitative Steinitz' theorem. *Beitr. Algebra Geom.*, 35(1):125–129, 1994. URL: `https://www.emis.de/journals/BAG/vol.35/no.1/`.

60 David Orden, Günter Rote, Fransisco Santos, Brigitte Servatius, Herman Servatius, and Walter Whiteley. Non-crossing frameworks with non-crossing reciprocals. *Discrete Comput. Geom.*, 32(4):567–600, 2004. `doi:10.1007/s00454-004-1139-x`.

61 W. J. Macquorn Rankine. Principle of the equilibrium of polyhedral frams. *London, Edinburgh, and Dublin Phil. Mag J. Sci.*, 27(180):92, 1864. `doi:10.1080/14786446408643629`.

62 William John Macquorn Rankine. *A Manual of Applied Mechanics*. Richard Griffin and Co., 1858. URL: `https://archive.org/details/manualappmecha00rankrich`.

63 Ares Ribó Mor, Günter Rote, and André Schulz. Small grid embeddings of 3-polytopes. *Discrete Comput. Geom.*, 45(1):65–87, 2011. `doi:10.1007/s00454-010-9301-0`.

64 Jürgen Richter-Gebert. *Realization spaces of polytopes*. Number 1643 in Lecture Notes Math. Springer, 1996. `doi:10.1007/BFb0093761`.

65 Igor Rivin. Euclidean structures on simplicial surfaces and hyperbolic volume. *Ann. Math.*, 139:553–580, 1994. `doi:10.2307/2118572`.

66 Günter Rote, Fransisco Santos, and Ileana Streinu. Pseudo-triangulations—a survey. In Jacob E. Goodman, János Pach, and Richard Pollack, editors, *Essays on Discrete and Computational Geometry: Twenty Years Later*, number 453 in Contemporary Mathematics, pages 343–410. Amer. Math. Soc., 2008.

67 André Schulz. Drawing 3-polytopes with good vertex resolution. *J. Graph Algorithms Appl.*, 15(1):33–52, 2011. `doi:10.7155/jgaa.00216`.

68 Dvir Steiner and Anath Fischer. Planar parameterization for closed 2-manifold genus-1 meshes. In *Proc. 9th ACM Symp. Solid Modeling Appl.*, pages 83–91, 2004.

69 Ernst Steinitz. Polyeder und Raumeinteilungen. *Enzyklopädie der mathematischen Wissenschaften mit Einschluss ihrer Anwendungen*, III.AB(12):1–139, 1916.

70 Ernst Steinitz and Hans Rademacher. *Vorlesungen über die Theorie der Polyeder: unter Einschluß der Elemente der Topologie*, volume 41 of *Grundlehren der mathematischen Wissenschaften*. Springer-Verlag, 1934. Reprinted 1976.

71 Kenneth Stephenson. *Introduction to Circle Packing: The Theory of Discrete Analytic Functions*. Cambridge Univ. Press, 2005.

72 Ileana Streinu. Erratum to "Pseudo-triangulations, rigidity and motion planning". *Discrete Comput. Geom.*, 35(2):358, 2006. `doi:10.1007/s00454-006-3300-1`.

73 Ileana Streinu. Pseudo-triangulations, rigidity and motion planning. *Discrete Comput. Geom.*, 34(4):587–635, 2006. Publisher's erratum in [72]. `doi:10.1007/s00454-005-1184-0`.

74 Kokichi Sugihara. Realizability of polyhedrons from line drawings. In Godfried T. Toussaint, editor, *Computational Morphology: A Computational Geometric Approach to the Analysis Of Form*, number 6 in Machine Intelligence and Pattern Recognition, pages 177–206, 1988.

75 William T. Tutte. How to draw a graph. *Proc. London Math. Soc.*, 13(3):743–768, 1963.

76 Pierre Varignon. *Nouvelle mechanique ou statique, dont le projet fut donné en M.DC.LXXVII*. Claude Jombert, Paris, 1725. URL: `https://gallica.bnf.fr/ark:/12148/bpt6k5652714w.texteImage`.

77 Walter Whiteley. Motion and stresses of projected polyhedra. *Topologie structurale / Structural Topology*, 7:13–38, 1982. URL: `http://hdl.handle.net/2099/989`.

78 Walter Whiteley, Peter F. Ash, Ethan Poiker, and Henry Crapo. Convex polyhedra, Dirichlet tesselations, and spider webs. In Marjorie Senechal, editor, *Shaping Space: Exploring Polyhedra in Nature, Art, and the Geometrical Imagination*, chapter 18, pages 231–251. Springer, 2013. `doi:10.1007/978-0-387-92714-5_18`.

79 Günter M. Ziegler. *Lectures on Polytopes*. Number 152 in Graduate Texts in Mathematics. Springer, 1995.

Combinatorial Properties of Self-Overlapping Curves and Interior Boundaries

Parker Evans
Department of Mathematics, Rice University, Houston, TX, USA
Parker.G.Evans@rice.edu

Brittany Terese Fasy
School of Computing and Department of Mathematical Sciences,
Montana State University, Bozeman, MT, USA
brittany@cs.montana.edu

Carola Wenk
Department of Computer Science, Tulane University, New Orleans, LA, USA
cwenk@tulane.edu

Abstract

We study the interplay between the recently-defined concept of *minimum homotopy area* and the classical topic of *self-overlapping curves*. The latter are plane curves that are the image of the boundary of an immersed disk. Our first contribution is to prove new sufficient combinatorial conditions for a curve to be self-overlapping. We show that a curve γ with Whitney index 1 and without any self-overlapping subcurves is self-overlapping. As a corollary, we obtain sufficient conditions for self-overlappingness solely in terms of the Whitney index of the curve and its subcurves. These results follow from our second contribution, which shows that any plane curve γ, modulo a basepoint condition, is transformed into an *interior boundary* by wrapping around γ with Jordan curves. In fact, we show that $n+1$ wraps suffice, where γ has n vertices. Our third contribution is to prove the equivalence of various definitions of self-overlapping curves and interior boundaries, often implicit in the literature. We also introduce and characterize *zero-obstinance curves*, a further generalization of interior boundaries defined by optimality in minimum homotopy area.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Self-overlapping curves, interior boundaries, minimum homotopy area, immersion

Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.41

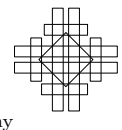
Related Version This paper is based on the honors thesis of the first author [5]. A full version of this paper [7] is available at `https://arxiv.org/abs/2003.13595`.

Supplementary Material An accompanying computer program that can determine whether a plane curve is self-overlapping, compute its minimum homotopy area, and display the self-overlapping decomposition associated with a minimum homotopy is available for download [6], `http://www.cs.tulane.edu/~carola/research/code.html`. Figure 10 was created with this program.

Funding *Parker Evans*: Supported by NSF grant CCF-1618469 and by a Goldwater Scholarship from the Goldwater Foundation.
Brittany Terese Fasy: Supported by NSF-CCF 1618605.
Carola Wenk: Supported by NSF grant CCF-1618469.

1 Introduction

Classically, a curve $\gamma : \mathbb{S}^1 \to \mathbb{R}^2$ is called **self-overlapping** if there is an orientation-preserving immersion $F : \mathbb{D}^2 \to \mathbb{R}^2$ of the unit disk $\mathbb{D}^2$, a map of full rank on the entire unit disk $\mathbb{D}^2$, such that $F|_{\partial\mathbb{D}^2} = \gamma$. One can think of such an immersion as distorting a unit disk



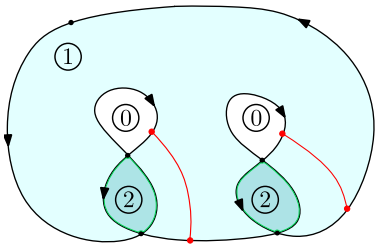


Figure 1 A self-overlapping curve γ with winding numbers for the faces circled. The Blank cuts, shown in red, slice γ into a collection of simple positively oriented (counterclockwise) Jordan curves.

that lies flat in the plane and stretching and pulling it continuously without leaving the plane and without twisting or pinching it [15]. If the disk is painted blue on top and pink on the bottom, then one only sees blue. If we also imagine the disk being semi-transparent, then the blue will appear darker in the regions where it overlaps itself; see Figure 1. That means, any self-overlapping curve γ must have non-negative winding numbers, $wn(x, \gamma) \geq 0$ for every $x \in \mathbb{R}^2$. We call this condition **positive consistent**. Another simple and intuitive view originates from Blank [1]: The curve is self-overlapping when we can cut it along simple curves into simple positively oriented Jordan curves, i.e., a collection of blue topological disks. **Interior boundaries** are generalizations of self-overlapping curves that are defined similarly, except that F is an interior map which allows finitely many branch points [12]. Interior boundaries are composed of multiple self-overlapping curves (of the same orientation); see Figure 2 for an example. In this paper, all curves $\gamma : [0, 1] \to \mathbb{R}^2$ are assumed to be closed, immersed, and generic, i.e., with only finitely many intersection points, each of which are transverse double points. We also assume $\gamma'(t)$ exists and is nonzero for all $t \in [0, 1]$. We show new combinatorial properties of self-overlapping curves and interior boundaries by revealing new connections to the minimum homotopy area of curves.

1.1 Related Work

Self-Overlapping Curves and Interior Boundaries. Self-overlapping curves and interior boundaries have a rather rich history, and have been studied under the lenses of analysis, topology, geometry, combinatorics, and graph theory [1, 4, 10, 12–15, 17, 19]. In the 1960s, Titus [19] provided the first algorithm to test whether a curve is self-overlapping (or an interior boundary), by defining a set of cuts that must cut the curve into smaller subcurves that are self-overlapping (or interior boundaries). In a 1967 PhD thesis [1], Blank proved that a curve is self-overlapping iff there is a sequence of cuts (different from Titus cuts) that completely decompose the curve into simple pieces. He represents plane curves with words and showed that one can determine the existence of a cut decomposition by looking for algebraic decompositions of the word. In the 1970s, Marx [13] extended Blank's work to give an algorithm to test if a curve is an interior boundary. In the 1990s, Shor and Van Wyk [17] expedited Blank's algorithm to run in $O(N^3)$ time for a polygonal curve with N line segments. Their dynamic programming algorithm is currently the fastest algorithm to test for self-overlappingness. It is not known whether this runtime bound is tight or whether a faster runtime might be achievable. In distantly related work, Eppstein and Mumford [4] showed that it is NP-complete to determine whether a fixed self-overlapping curve γ is the 2D projection of an immersed surface in $\mathbb{R}^3$ defined over a compact two-manifold with boundary. Graver and Cargo [10] approached the problem from a graph-theoretical perspective using

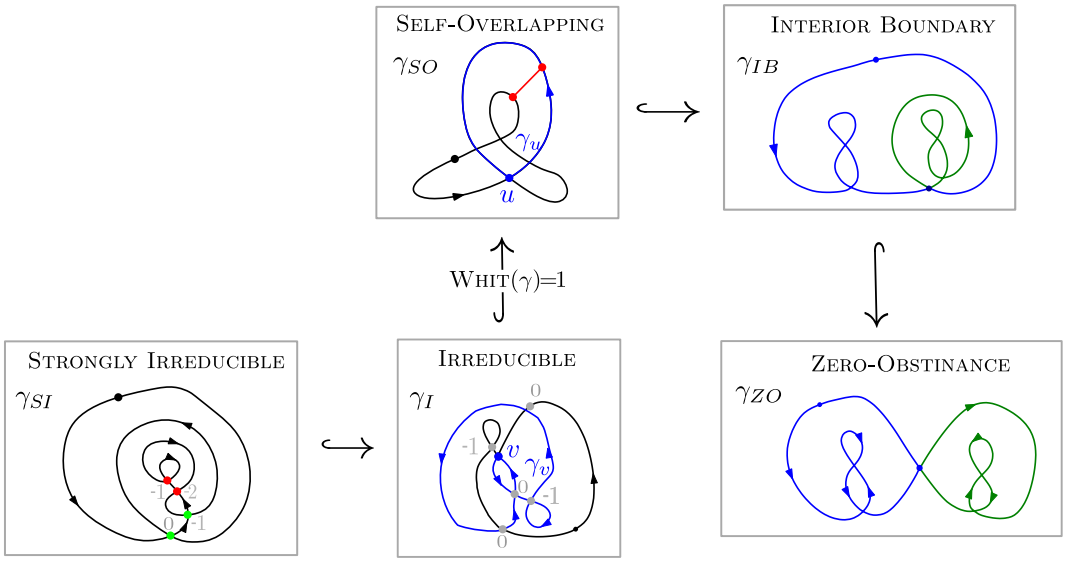


Figure 2 Example curves of different curve classes and inclusion relationships between the classes. γ_{SO} is self-overlapping as indicated by the Blank cuts in red. γ_{IB} is an interior boundary consisting of two self-overlapping curves (of the same orientation), one in blue the other in green. The bottom row shows curve classes that are introduced in this paper: γ_{SI} is strongly irreducible as can be seen from the non-positive Whitney indices (shown in gray) of its direct split subcurves. Similarly, γ_I is irreducible; note that γ_v has Whitney index 1 but is not self-overlapping. Also note that γ_{SO} is not irreducible since γ_u is self-overlapping. γ_{ZO} also consists of two self-overlapping curves but of different orientation and is therefore not an interior boundary, but it has zero obstinance.

so-called covering graphs. All of these algorithms also compute the number of inequivalent immersions. Another fact we glean from Blank is that any self-overlapping curve γ necessarily makes one full turn, i.e., it has Whitney index $\text{WHIT}(\gamma) = 1$. The necessity of Whitney index 1 and positive consistency to be self-overlapping are well-known and date back to [19].

Minimum Homotopy Area. The **minimum homotopy area $\sigma(\gamma)$** is the infimum of areas swept out by nullhomotopies of a closed plane curve γ. The key link between minimum area homotopies and self-overlapping curves arose in [8, 11], where the authors showed that any curve γ has a minimum area homotopy realized by a sequence of nullhomotopies of self-overlapping subcurves (direct split subcurves; see Section 3.1 for the definition). The minimum homotopy area was introduced by Chambers and Wang [3] as a more robust metric for curve comparison than homotopy width (i.e., Fréchet distance or one of its variants) or homotopy height [2]. The minimum homotopy area can be computed in $O(N^2 \log N)$ time for consistent curves [3]. For general curves, Nie gave an algorithm to compute $\sigma(\gamma)$ based on an algebraic interpretation of the problem that runs in $O(N^6)$ time, while the self-overlapping decomposition result of [8] yields an exponential-time algorithm.

The **winding area $W(\gamma)$** is the integral over all winding numbers in the plane. A simple argument shows that $\sigma(\gamma) \geq W(\gamma)$; see [3]. Both self-overlapping curves and interior boundaries are characterized by positive consistency and optimality in minimum homotopy area, $\sigma(\gamma) = W(\gamma)$. A curve possessing both of these properties is self-overlapping when $\text{WHIT}(\gamma) = 1$ and an interior boundary when $\text{WHIT}(\gamma) \geq 1$.

1.2 New Results

We ask the following question: what are the sufficient combinatorial conditions for a plane curve to be self-overlapping? Such conditions provide novel mathematical foundations that could pave the way for speeding up algorithms for related problems, such as deciding self-overlappingness or computing the minimum homotopy area of a curve. The first contribution of this paper is to answer this question in the affirmative (Theorems 16 and 17 in Section 4): We show that a curve γ with Whitney index 1 and without any self-overlapping subcurves is self-overlapping, and we obtain sufficient conditions for a curve to be self-overlapping solely in terms of the Whitney index of the curve and its subcurves. Here, we only consider **direct split** subcurves γ_v that traverse γ between the first and second appearance of vertex v in the plane graph induced by γ. Our results apply to (strongly) irreducible curves; see Figure 2: We call γ **irreducible**, if every (proper) direct split is not self-overlapping; if the Whitney index of each such direct split is non-positive, then we call γ **strongly irreducible**.

These results follow from our second contribution (Theorems 13 and 14 in Section 4), which shows that any plane curve γ is transformed into an interior boundary by wrapping around γ with Jordan curves. Equivalently, this means that the minimum homotopy area of γ is reduced to the minimal possible threshold, namely the winding area, through wrapping. See Figure 3 for an example of wrapping. Of course, we can make a curve positive consistent with repeated wrapping, since a single wrap increases the winding numbers of each face by one. However, our result shows a new and non-trivial connection between wrapping and the minimum homotopy area.

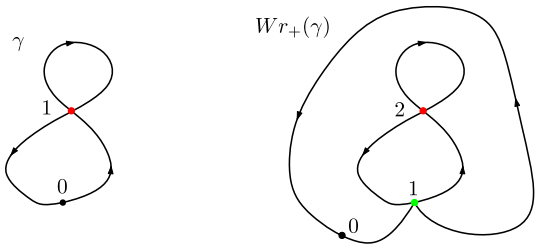


Figure 3 The curve γ is not self-overlapping, but its wrap $\mathrm{Wr}_+(\gamma)$ is self-overlapping.

The final contribution of this paper (in Section 3) is to unite the various definitions and perspectives on self-overlapping curves and interior boundaries. We prove the equivalence of five definitions of self-overlapping curves and four of interior boundaries (Theorems 10 and 9). To this end, we define the new concept of **obstinance** of a curve γ as $\mathrm{obs}(\boldsymbol{\gamma}) = \sigma(\gamma) - W(\gamma) \geq 0$, and characterize **zero-obstinance** curves (Theorem 11), see Figure 2. Rephrasing our earlier characterization, self-overlapping curves and interior boundaries are positive-consistent curves with zero-obstinance and positive Whitney index.

2 Preliminaries

2.1 Regular and Generic Curves

We work with regular, generic, closed plane curves $\gamma : [0, 1] \to \mathbb{R}^2$ with basepoint $\gamma(0) = \gamma(1)$. Let $\mathscr{C}$ denote the set of such curves. A curve γ is **regular** if $\gamma'(t)$ exists and is non-zero for all t; a curve is **generic** (or normal) if the embedding has only a finite number of intersection points, each of which are transverse crossings. Being generic is a weak restriction, as normal curves are dense in the space of regular curves [20]. Viewing a generic

curve γ by its image $[\boldsymbol{\gamma}] \subseteq \mathbb{R}^2$, we can treat γ as a directed plane multigraph $\boldsymbol{G(\gamma)} = (V(\gamma), E(\gamma))$. Here, $\boldsymbol{V(\gamma)} = \{p_0, p_1, \ldots, p_n\}$ is the set of ordered **vertices** (points) of γ, with basepoint $p_0 = \gamma(0)$ regarded as a vertex as well. An **edge** (p_i, p_j) corresponds to a simple path along γ between p_i and p_j. The **faces** of $G(\gamma)$ are the path-connected components of $\mathbb{R}^2 \setminus [\gamma]$. Each $\gamma \in \mathscr{C}$ has exactly one unbounded face, the exterior face $\boldsymbol{F_{ext}}$. See Figure 4. Two curves are combinatorially equivalent when their planar multigraphs are isomorphic. We may therefore define a curve just by its image, orientation, and basepoint. A curve is **simple** if it has no intersection points. We notate $\boldsymbol{|\gamma|} = |V(\gamma) \setminus \{p_0\}|$ as the complexity of γ.

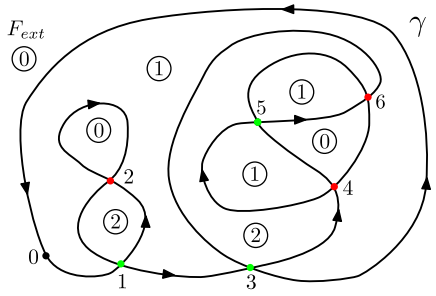


Figure 4 A curve γ that is self-overlapping. The winding numbers of each face are enclosed by circles. The signed intersection sequence of γ is $0, 1_+, 2_-, 2_+, 1_-, 3_+, 4_-, 5_+, 6_-, 4_+, 5_-, 6_+, 3_-, 0$; vertex labels are shown, and the sign of each vertex is indicated with green (positive) or red (negative). The combinatorial relations are: $p_2 \subset p_1$; $p_4, p_5, p_6 \subset p_3$; p_1, p_2 S p_3, p_4, p_5, p_6; p_4, p_5 L p_6; p_4 L p_5.

For any $x \in \mathbb{R}^2 \setminus [\gamma]$, the **winding number** $\boldsymbol{wn(x, \gamma)} = \sum_i a_i$ is defined using a simple path P from x to F_{ext} that avoids the intersection points of γ. Here, $a_i = +1$ if P crosses γ from left to right at the i-th intersection of P with γ, and $a_i = -1$ otherwise. Since this number is independent of the path chosen and is constant over each face F of of $G(\gamma)$, we write $\boldsymbol{wn(F, \gamma)}$. If $wn(F, \gamma) \geq 0$ for every face F on $G(\gamma)$, then we call γ **positive consistent**. If $wn(F, \gamma) \leq 0$ for every face, then γ is **negative consistent**. See Figure 4. The **winding area** of a curve γ is given by $\boldsymbol{W(\gamma)} = \int_{\mathbb{R}^2} |wn(x, \gamma)|\, dx = \sum_F A(F)\, |wn(F, \gamma)|$, where $A(F)$ is the area of the face F and $wn(x, \gamma) = 0$ for $x \in [\gamma]$. The **Whitney index** $\boldsymbol{\mathrm{WHIT}(\gamma)}$ is the winding number of the derivative γ' about the origin. A curve γ is **positively oriented** if $\mathrm{WHIT}(\gamma) > 0$ and **negatively oriented** if $\mathrm{WHIT}(\gamma) < 0$.

A basepoint $p_0 = \gamma(0)$ is a **positive outer basepoint** if p_0 is incident to the two faces F_{ext} and F, and $wn(F, \gamma) = 1$. If $wn(F, \gamma) = -1$, then p_0 is a negative outer basepoint. Several of our results require γ to have a positive outer basepoint. A curve $\gamma : \mathbb{S}^1 \to \mathbb{R}^2$ is **(positive) self-overlapping** when there is an orientation-preserving immersion $F : \mathbb{D}^2 \to \mathbb{R}^2$, a map of full rank, extending γ to a map on the entire two-dimensional unit disk $\mathbb{D}^2$. If the reversal $\overline{\gamma}$ of a curve is self-overlapping, then we call γ **negative self-overlapping**.

2.2 Combinatorial Relations and Intersection Sequences

Following Titus [18], we now describe how the intersection points of a curve $\gamma \in \mathscr{C}$ relate to each other; see Figure 4. Let $p_i, p_j \in V(\gamma)$ be two vertices such that $p_i = \gamma(t_i) = \gamma(t_i^*)$ and $p_j = \gamma(t_j) = \gamma(t_j^*)$ with $t_i < t_i^*$ and $t_j < t_j^*$. Then, one of the following must hold:

- p_i **links** p_j, or p_i **L** p_j, iff $t_i < t_j < t_i^* < t_j^*$ or $t_j < t_i < t_j^* < t_i^*$
- p_i is **separate** from p_j, or p_i **S** p_j, iff $t_i < t_i^* < t_j < t_j^*$ or $t_j < t_j^* < t_i < t_i^*$
- p_i is **contained in** p_j, or $p_i \subset p_j$, iff $t_j \leq t_i < t_i^* \leq t_j^*$.

To define the intersection sequence of γ, the vertices are labeled in the order they appear on γ, starting with 0 for the basepoint $\gamma(0)$, and increasing by one each time an unlabeled vertex is encountered. The **signed intersection sequence** consists of the sequence of all

vertex labels along γ starting at the basepoint; the first time vertex p_i is visited, the label is augmented with $\text{sgn}(p_i)$, and the second time with $-\text{sgn}(p_i)$. Here, $\text{sgn}(p_i) = \text{sgn}(p_i, \gamma)$ is the **sign** of vertex $p_i = \gamma(t_i) = \gamma(t_i^*)$, and is 1 if the vector γ' rotates clockwise from t_i to t_i^*, and -1 otherwise. Note that $\text{sgn}(p_i)$ depends on the basepoint of the curve. Interior boundaryness is invariant with respect to signed intersection sequences [19].

2.3 Minimum Homotopies

A **homotopy** between two generic curves γ and γ' is a continuous function $H : [0,1]^2 \to \mathbb{R}^2$ such that $H(0,\cdot) = \gamma$ and $H(1,\cdot) = \gamma'$. In $\mathbb{R}^2$, any curve is null-homotopic, i.e., homotopic to a constant map. Given a sequence of homotopies $(H_i)_{i=1}^k$, we notate the concatenation of these homotopies in order as $\sum_{i=1}^{k} \boldsymbol{H_i}$. We use the notation $\overline{H}$ for the reversal $\overline{H}(i,t) = H(1-i,t)$ of a homotopy. If $H(0,\cdot) = \gamma$ and $H(1,\cdot) = \gamma'$, we may write $\boldsymbol{\gamma \overset{H}{\rightsquigarrow} \gamma'}$.

Homotopy moves are basic local alterations to a curve defined by their action on $G(\gamma)$. These moves come in three pairs [8]; see Figure 5: The I-moves destroy/create an empty loop, II-moves destory/create a bigon, and III-moves flip a triangle. We denote the moves that remove vertices as $\mathbf{I_a}$ and $\mathbf{II_a}$, and moves that create vertices as $\mathbf{I_b}$ and $\mathbf{II_b}$. See Figure 5. It is well-known that any homotopy such that each intermediate curve is piecewise regular and generic, or almost generic, can be achieved by a sequence of homotopy moves. Thus, without loss of generality, we assume that each time the curve $H(i,\cdot)$ combinatorially changes is through a single homotopy move.

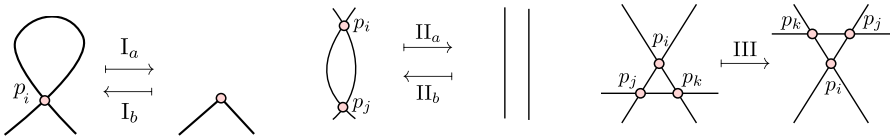


Figure 5 All three homotopy moves and their reversals. Figure from [8].

Let $\gamma \in \mathscr{C}$ and H be a nullhomotopty of γ. Define $E_H(x)$ as the number of connected components of $H^{-1}(x)$. Intuitively, this counts the number of times that H sweeps over x. The **minimum homotopy area** of γ is defined as $\boldsymbol{\sigma(\gamma)} = \inf_H\{\int_{\mathbb{R}^2} E_H(x)\, dx \mid H \text{ is a nullhomotopy of } \gamma\}$. The following was shown in [3, 8]:

▶ **Lemma 1** (Homotopy Area ≥ Winding Area). *Let $\gamma \in \mathscr{C}$. Then $\sigma(\gamma) \geq W(\gamma)$.*

We call a homotopy **left (right) sense-preserving** if $H(i+\epsilon, t)$ lies on or to the left (right) of the oriented curve $H(i,\cdot)$ for any $i, t \in [0,1]$ and any $\epsilon > 0$. The following two lemmas provide useful properties about sense-preserving homotopies; the first was proven in [3], the second in [8].

▶ **Lemma 2** (Monotonicity of Winding Numbers). *Let H be a homotopy. If H is left (right) sense-preserving, then for any $x \in \mathbb{R}^2$ the function $a(i) = wn(x, H(i,\cdot))$ is monotonically decreasing (increasing).*

▶ **Lemma 3** (Sense-Preserving Homotopies are Optimal). *Let $\gamma \in C$ be consistent. Then a nullhomotopy H of γ is optimal if and only if it is sense-preserving.*

3 Equivalences

In this section, we show the equivalence of different characterizations of interior boundaries (Theorem 9) and of self-overlapping curves (Theorem 10). Our analysis of curve classes hinges around the concept of obstinance. In Theorem 11 we classify zero obstinance curves, which are generalizations of interior boundaries and of self-overlapping curves.

3.1 Direct Splits

Let $\gamma \in \mathscr{C}$ and $p_i \in V(\gamma)$ with $p_i = \gamma(t_i) = \gamma(t_i^*)$ and $t_i < t_i^*$. Then, γ can be split into two subcurves at p_i: The **direct split** is the curve with image $[\gamma|_{[t_i,t_i^*]}]$ with basepoint p_i, and the **indirect split** is the curve with image $[\gamma|_{[t_i^*,1]}] \cup [\gamma|_{[0,t_i]}]$ with basepoint $\gamma(0)$. We endow both of these curves with the same orientation as γ. Given a direct (or indirect) split $\widetilde{\gamma}$ on a curve γ, we write $\gamma \setminus \widetilde{\gamma}$ for the indirect (or direct) split complementary to $\widetilde{\gamma}$. We call a direct split **proper** if it is not the entire curve γ. See Figure 6. If $v = p_i \in V(\gamma)$, we may notate the direct split as γ_i or γ_v. When removing multiple splits iteratively, we write $\gamma \setminus (\cup_{i=1}^{n} \gamma_i)$, where we require that γ_i is a direct split of $\gamma \setminus \left(\cup_{j=1}^{i-1} \gamma_j\right)$. Being a direct split of a curve is a transitive property. I.e., if γ_i is a direct split on γ, and γ_j is a direct split on γ_i, then γ_j is a direct split on γ. The parallel statement on indirect splits, however, is false.

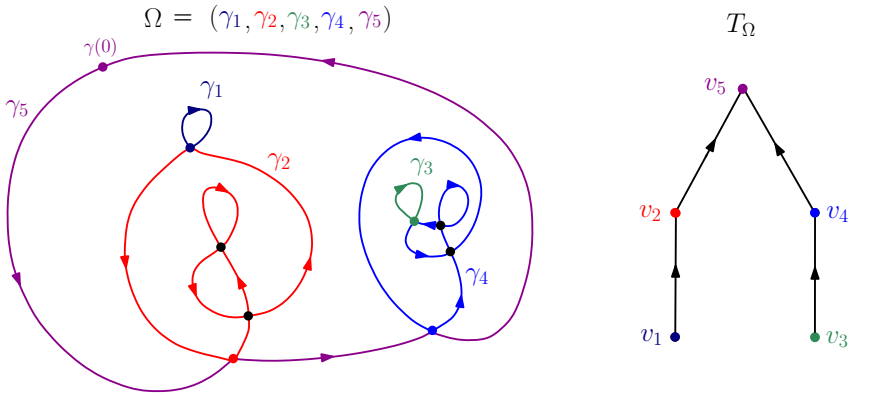


Figure 6 A self-overlapping decomposition of a self-overlapping curve γ. Here, γ_1 and γ_3 are (proper) direct splits of γ, while γ_2, γ_4, and γ_5 are neither direct nor indirect splits of γ.

3.2 Decompositions and Loops

A curve $\gamma \in \mathscr{C}$ can be entirely decomposed by iteratively removing direct splits. For a sequence of subcurves $\Omega = (\gamma_i)_{i=1}^k$, define $C_0 = \gamma$ and inductively $C_i = C_{i-1} \setminus \gamma_i$ for $i \geq 1$; the basepoint of γ_i is $v_i = C_i \cap \gamma_i$. We call Ω a **direct split decomposition** if γ_i is a direct split of C_{i-1}, for all $i \in \{1, 2, \ldots, k\}$, and $\gamma_k = C_{k-1}$. Given a direct split decomposition $\Omega = (\gamma_i)_{i=1}^k$, we write $\boldsymbol{V(\Omega)}$ for the set of basepoints of all $\gamma_i \in \Omega$. See Figure 6. Observe that no two vertices $v_i, v_j \in V(\Omega)$ may be linked. Hence, we obtain a partial order $\prec$ on $V(\Omega)$ by declaring $v_i \prec v_j$ whenever $v_i \subset v_j$. We define T_Ω to be the tree with vertex set $V(T_\Omega) = V(\Omega)$ and edges $e = (v_i, v_j)$ whenever $v_i \subset v_j$ and there is no other vertex $v_k \neq v_i, v_j$ such that $v_i \subset v_k \subset v_j$. We consider two subcurve decompositions Ω, Γ equivalent, $\Omega \sim \Gamma$, when $T_\Omega = T_\Gamma$. This means that Ω and Γ contain the same set of subcurves, just in a different order. If every γ_i is self-overlapping, we call Ω a **self-overlapping decomposition**; it may contain self-overlapping subcurves of positive and negative orientations. We now observe that the vertex set of a decomposition already determines the subcurves in the decomposition:

▶ **Observation 4.** *Given a curve $\gamma \in \mathscr{C}$ and a subset $S \subset V(\gamma)$ such that $p_0 \in S$ and no two vertices in S are linked, there is a unique equivalence class $\mathscr{E}$ of direct split decompositions with $V(\Omega) = S$ for all $\Omega \in \mathscr{E}$.*

The observation below follows directly from the definition of winding numbers.

▶ **Observation 5.** *Let Ω be a direct split decomposition of a curve $\gamma \in \mathscr{C}$. Then for any face F in the plane multigraph $G(\gamma)$, $wn(F, \gamma) = \sum_{\gamma_i \in \Omega} wn(F, \gamma_i)$.*

We define a **loop** as a simple direct split γ_v of a curve $\gamma \in \mathscr{C}$. Intersection points of γ may lie on γ_v, but none occur as intersections of γ_v with itself. Every non-simple plane curve has a loop; e.g., the direct split γ_w, where w is the highest index vertex on γ in the signed intersection sequence. A loop γ_v is **empty** if v links no vertex $w \in V(\gamma)$. Let $\text{int}(\gamma_v)$ denote its interior. We call γ_v an **outwards loop** if the edges e_1, e_4, that are incident on v and lie on $\gamma \setminus \gamma_v$, both lie outside $\text{int}(\gamma_v)$. Otherwise γ_v is an **inwards loop**. See Figure 7.

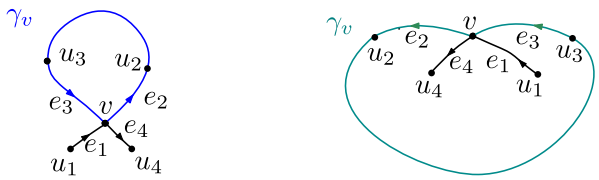


Figure 7 An outwards loop (left) and an inwards loop (right).

The lemma below follows from [9, 16] and is proven in [7].

▶ **Lemma 6** (Whitney Index Through Decompositions). *Let $\gamma \in \mathscr{C}$ and Ω be a direct split decomposition of γ. Then $\textit{WHIT}(\gamma) = \sum_{C \in \Omega} \textit{WHIT}(C)$.*

A consequence of Lemma 6 is that iteratively removing loops and summing ± 1 for their signs allows one to quickly compute Whitney indices. Assuming γ is given as a directed plane multigraph, one can adapt a depth-first traversal to compute such a loop decomposition of γ in $O(|\gamma|)$ time, which yields the following corollary:

▶ **Corollary 7** (Compute Whitney Index). *Let $\gamma \in \mathscr{C}$ be of complexity $n = |\gamma| = |V(\gamma)|$. One can compute a loop decomposition of γ, and $\textit{WHIT}(\gamma)$, in $O(n)$ time.*

Now, let H be a nullhomotopy of a curve γ, and consider all the points $A = \{v_i\}_{i=1}^k$ of $\mathbb{R}^2$ such that H performs a I_a move to contract a loop to that point. All such points are called **anchor points** of the homotopy H. Following [8] we call a homotopy H **well-behaved** when the anchor points A of H satisfy $A \subseteq V(\gamma)$, i.e., H only contracts loops to vertices of the original curve. The theorem below from [8] shows that computing minimum homotopy area is reduced to finding an optimal self-overlapping decomposition. The homotopy H guaranteed in the following theorem is well-behaved.

▶ **Theorem 8** (Minimum Homotopy Decompositions). *Let $\gamma \in \mathscr{C}$. Then there is a self-overlapping decomposition $\Omega = (\gamma_i)_{i=1}^k$ of γ as well as an associated minimum homotopy H_Ω of γ such that $H_\Omega = \sum_{i=1}^k H_i$ and each H_i is a nullhomotopy of γ_i. In particular, $\sigma(\gamma) = \min_{\Omega \in \mathscr{D}(\gamma)} \sum_{C \in \Omega} W(C)$, where $\mathscr{D}(\gamma)$ is the set of all self-overlapping decompositions of γ.*

3.3 Equivalence of Interior Boundaries

In this section, we unify different definitions and characterizations of interior boundaries by showing their equivalence. We call a curve γ a **k-interior boundary** when (1) $\text{obs}(\gamma) = 0$, (2) $\text{WHIT}(\gamma) = k > 0$, and (3) γ is positive consistent. We call γ a **($-$k)-interior boundary**

when its reversal $\overline{\gamma}$ is a k-interior boundary. In accordance with Titus [19], we call a curve $\zeta : [0,1] \to \mathbb{R}^2$ a **Titus interior boundary** if there exists a map $F : \mathbb{D}^2 \to \mathbb{R}^2$ such that F is continuous, **light** (defined as: pre-images are totally disconnected), open, orientation-preserving, and $F|_{\partial \mathbb{D}^2} = \zeta$. The map F is called **properly interior**.

▶ **Theorem 9** (Equivalence of Interior Boundaries). *Let $\gamma \in \mathscr{C}$ and suppose $WHIT(\gamma) = k > 0$. Then, the following are equivalent:*

1. *γ is an interior boundary.*
2. *γ is a Titus interior boundary.*
3. *γ admits a self-overlapping decomposition $\Omega = (\gamma_i)_{i=1}^k$, where each γ_i is positive self-overlapping.*
4. *γ admits a well-behaved left sense-preserving nullhomotopy H with exactly k I_a-moves.*

Proof.

1 ⇒ 3: Let γ be an interior boundary. By Theorem 8, we have an optimal self-overlapping decomposition $\Omega = (\gamma_i)_{i=1}^j$ of γ. Suppose, by contradiction, that there exists an $l \leq j$ such that γ_l is negative self-overlapping. Let F be any face contained in the interior $\text{int}(\gamma_l)$. We know by Observation (5) that $wn(F,\gamma) = \sum_{i=1}^j wn(F,\gamma_i)$, and since γ is positive consistent $wn(F,\gamma) \geq 0$. Thus there must exist a positive self-overlapping curve $\gamma_i \in \Omega$ with $F \subseteq \text{int}(\gamma_i)$. Consider the nullhomotopies H_l and H_i that are part of the canonical optimal homotopy H_Ω. Then H_l contracts γ_l and is right sense-preserving, while H_i contracts H_i and is left sense-preserving. Thus by Lemma 2, H_l increases the winding number on F and H_i decreases the winding number, which means F is swept more than $W(F)$ times, a contradiction. Thus, no negative self-overlapping subcurve γ_l may exist in Ω. Since $\text{WHIT}(C) = 1$ for any positive self-overlapping subcurve and $\text{WHIT}(\gamma) = \sum_{i=1}^k \text{WHIT}(\gamma_i)$ by Lemma 6, we we must have $k = j$.

1 ⇔ 4: If γ has a well-behaved left sense-preserving nullhomotopy H with exactly k I_a-moves, then H comes naturally with an associated self-overlapping decomposition Ω of γ with $|\Omega| = k$, and $\text{WHIT}(\gamma) = k > 0$ by Lemma 6. We now show that $\sigma(\gamma) = W(\gamma)$. Consider the reversal $\overline{H}$ from the constant curve $\gamma_{p_0}(t) = p_0$ to γ. Then, $\overline{H}$ is right sense-preserving and by Lemma 2 the function $a(i) = wn(x, \overline{H}(i,\cdot))$ is monotonically increasing for any $x \in \mathbb{R}^2$. Since $wn(x,\gamma_{p_0}) = 0$ for all $x \in \mathbb{R}^2$, we have that $wn(x,\gamma) \geq 0$ for all $x \in \mathbb{R}^2$. Thus, γ is an interior boundary. Conversely, if γ is a positive interior boundary, then $\text{obs}(\gamma) = 0$ and by Lemma 3, and since γ is positive, H is left sense-preserving. Again, by Lemma 6, $\text{WHIT}(\gamma) = j$, where j is the number of I_a-moves in any well-behaved nullhomotopy H of γ. Hence, we must have $j = k$. The remaining cases are proved in [7]. ◀

3.4 Equivalences of Self-Overlapping Curves

In this section, we study different characterizations of self-overlapping curves and show their equivalence. First we describe a geometric formulation of self-overlappingness, inspired by the work of Blank and Marx [1, 13]. Let $\gamma \in \mathscr{C}$ be self-overlapping. Let $P : [0,1] \to \mathbb{R}^2$ be a simple path so that $P(0) = q = \gamma(t_q)$ and $P(1) = p = \gamma(t_p)$ lie on $[\gamma]$ but are not vertices of γ. Without loss of generality, assume $t_q < t_p$. Let $\widetilde{P} = \gamma|_{[t_q,t_p]}$, and suppose that (1) $P \cap \widetilde{P} = \{p,q\}$, (2) $C = \widetilde{P} * \overline{P}$ is a simple closed curve, and (3) C is positively oriented; see Figure 8. Then we call P a **Blank cut** of γ. By cutting along P, γ is split into two curves of strictly smaller complexity, γ_1 and C. We call a sequence $(P_i)_{i=1}^k$ of Blank cuts a **Blank cut decomposition** if the final curve is a simple positively oriented curve.

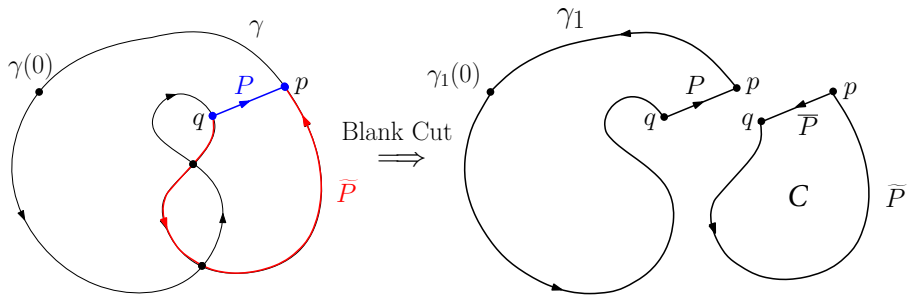


Figure 8 A Blank cut on a small self-overlapping curve.

▶ **Theorem 10** (Equivalent Characterizations of Self-Overlapping Curves). *Let $\gamma \in \mathscr{C}$. Then the following are equivalent:*

1. *(Analysis) There is an immersion $F : D^2 \to \mathbb{R}^2$ so that $F|_{\partial D^2} = \gamma$.*
2. *(Geometry) γ admits a Blank cut decomposition.*
3. *(Geometry/Topology) γ is a $(+1)$-interior boundary, i.e., self-overlapping.*
4. *(Topology) γ admits a left-sense preserving nullhomotopy H with exactly one I_a-move.*
5. *(Analysis) γ is a Titus interior boundary with $WHIT(\gamma) = 1$.*

Proof. By property 3 in Theorem 9, self-overlapping curves are 1-interior boundaries, since any self-overlapping curve γ has the trivial self-overlapping decomposition $\Omega = (\gamma)$. Thus, we have already established $1 \Leftrightarrow 3 \Leftrightarrow 4 \Leftrightarrow 5$ in Theorem 9. We now prove $2 \Leftrightarrow 4$. Any Blank cut P can be performed by a left sense-preserving homotopy that deforms $\widetilde{P}$ to P. Hence the Blank cut decomposition corresponds to a left sense-preserving homotopy to a simple positively oriented curve. Finally we perform a single I_a-move to complete a left sense-preserving nullhomotopy of γ. Conversely, let γ have a left sense-preserving nullhomotopy H. From $3 \Leftrightarrow 4$ we know that every intermediary curve $\gamma_i = H(i, \cdot)$ is self-overlapping since the subhomotopy $H_i = H|_{[i,1]\times[0,1]}$ is a left sense-preserving nullhomotopy of γ_i with one I_a-move. As H ends with a I_a-move, we may select a subhomotopy H' such that $\gamma \overset{H'}{\rightsquigarrow} C$, where C is a simple self-overlapping curve. Moreover, we see that $H = H' + H''$, where the unique I_a-move of H occurs during H''. Thus, H' is regular, i.e., consists of a sequence of homotopy moves only of types II_a, II_b, or III, which deform γ to C. Each of these homotopy moves can be performed by a Blank cut, as shown in Figure 9. Since all of the intermediary curves are self-overlapping, this induces a Blank cut decomposition. ◀

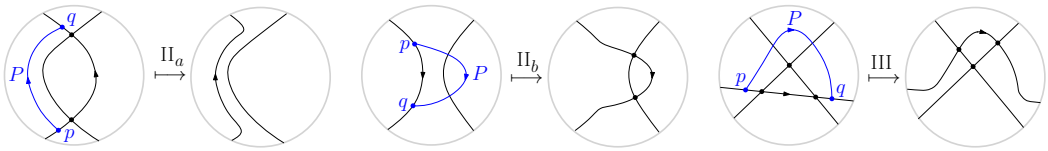


Figure 9 Homotopy moves II_a, II_b, and III each correspond to a Blank cut (shown in blue).

3.5 Zero Obstinance Curves

In this section, we classify curves $\gamma \in \mathscr{C}$ with **zero obstinance**, $\text{obs}(\gamma) = \sigma(\gamma) - W(\gamma) = 0$. See Figures 2 and 10 for examples of zero-obstinance curves. We show that just as interior boundaries can be decomposed into self-overlapping curves, so too can zero-obstinance curves be decomposed into interior boundaries.

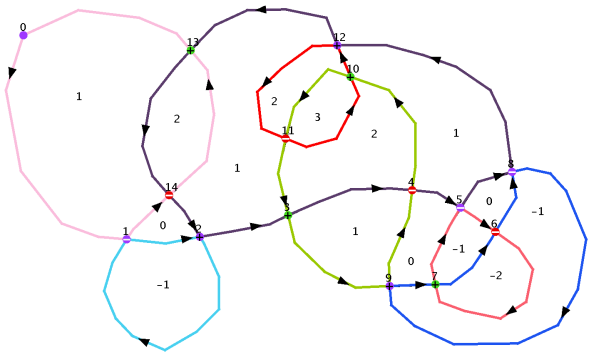


Figure 10 A zero obstinance curve, with its minimum homotopy decomposition, and winding numbers shown. Each curve in the decomposition is self-overlapping and shown in a different color. The vertices with labels $1, 2, 5, 8, 9$ are sign-changing.

If a curve γ has zero obstinance, then there is a nullhomotopy H which sweeps each face F on $G(\gamma)$ exactly $wn(F, \gamma)$ times. Note that such a homotopy H is necessarily minimal by Lemma 1. Intuitively, this implies that the homotopy H should be locally sense-preserving. We expect it to sweep leftwards on positive consistent regions and rightwards on negative consistent regions. Hence, we might expect regions of the curve where the winding numbers change from positive to negative to be especially problematic. Indeed, let $v \in V(\gamma)$ be incident to the faces $\{F_1, F_2, F_3, F_4\}$. We call v **sign-changing** when, as a multiset, $\{wn(\gamma, F_1), wn(\gamma, F_2), wn(\gamma, F_3), wn(\gamma, F_4)\} = \{-1, 0, 0, 1\}$; see Figures 10 and 11.

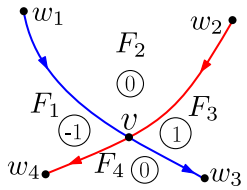


Figure 11 A sign-changing vertex v. The winding numbers of the incident faces are -1, 0, 1, 0.

▶ **Theorem 11** (Zero Obstinance Characterization). *Let $\gamma \in \mathscr{C}$ and let $\mathscr{S}$ be the sign-changing vertices of γ. Then $obs(\gamma) = 0$ iff no two vertices in $\mathscr{S}$ are linked and any direct split subcurve decomposition Ω with vertex set $V(\Omega) = \mathscr{S} \cup \{p_0\}$ contains only interior boundaries.*

The proof is available in [7].

4 Wraps and Irreducability

In this section, we show (Theorems 13 and 14) that wrapping around a curve γ until its obstinance is reduced to zero results in an interior boundary. This key result is used to prove sufficient combinatorial conditions for a curve to be self-overlapping based on the Whitney index of the curve and its direct splits (Theorems 16 and 17).

4.1 Wraps

Let $\gamma \in \mathscr{C}$, and let I be its signed intersection sequence. Form I' by incrementing each label by one and removing the occurrences of 0 corresponding to the basepoint. If γ has a positive outer basepoint $\gamma(0)$, then its **(positive) wrap $Wr_+(\gamma)$** is the unique (class of) curve with

signed intersection sequence $0, 1_+, I', 1_-, 0$. This corresponds to gluing a simple positively oriented curve α to γ at $\gamma(0)$, where the interior $\text{int}(\alpha) \supseteq [\gamma]$; the new basepoint is on α. See Figure 12. The **negative wrap $Wr_-(\gamma)$** is defined analogously if γ has a negative outer basepoint. We write $Wr_+^k(\gamma)$ for the curve achieved from γ by wrapping k times.

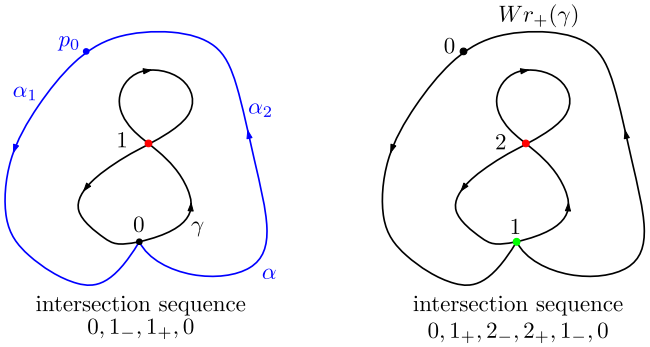


Figure 12 A curve γ with positive outer basepoint and its *positive* wrap $Wr_+(\gamma)$.

To wrap a curve in the direction opposed to the sign of the basepoint, we must be more careful. Without loss of generality, we describe the construction of $Wr_-(\gamma)$ when γ has a positive outer basepoint. Perform a I_b-move to add a simple loop $\tilde{\gamma}$ of the opposite orientation tangent to the basepoint $\gamma(0)$. Let γ' be the curve after the I_b-move, with a basepoint chosen to lie on $\tilde{\gamma}$. We then define $Wr_-(\gamma) = Wr_-(\gamma')$. See Figure 13.

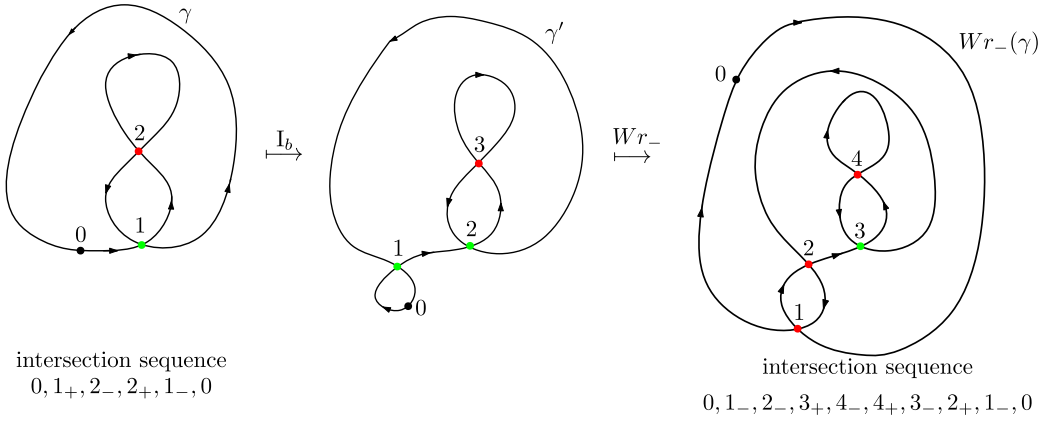


Figure 13 A curve γ with positive outer basepoint and its transformation into its *negative* wrap $Wr_-(\gamma)$. First, we perform a I_b-move and then wrap normally on γ'.

4.2 Wrapping Resolves Obstinance

First, we prove a simple lemma:

▶ **Lemma 12** (Existence of an Outwards Loop). *Let $\gamma \in \mathscr{C}$ have an outer basepoint. Then, if γ is non-simple, it has an outwards loop.*

Proof. Let v be the first self-intersection of γ. Then γ_v is a loop. Write $\gamma^{-1}(v) = \{t, t^*\}$, where $t < t_*$. Since $\gamma(0)$ lies outside of $\text{int}(\gamma_v)$, as an outer basepoint, we note that if γ_v were inwards, the path $P = \gamma_{[0,t_1]}$ would cross $[\gamma_v]$ to get from outside the simple curve to

inside it. This is then a contradiction, for if the crossing occurred at a point q on $[\gamma_v]$, then q would be the first self-intersection of γ. Indeed, we would reach q a second time before we reach v a second time. Thus, γ_v is outwards. ◀

Now, let $t_1 < t_1^* \in [0,1]$ be the smallest value so that $\gamma_1 = \gamma_{[t_1,t_1^*]}$ is a loop. We call this the **first loop** of γ. Since t_1^* is the first time that γ self-intersects, we know that such a loop is outwards by the argument in Lemma 12.

▶ **Theorem 13** (Wrapping Resolves Obstinance). *Let $\gamma \in \mathscr{C}$ have positive outer basepoint. Then there is a positive integer k so that $obs(Wr_+^k(\gamma)) = 0$. Moreover, $Wr_+^k(\gamma)$ is a positive interior boundary.*

Proof. Let l be the number of negative vertices in $V(\gamma)$. Set $k = l+1$. We claim that $\mathrm{Wr}_+^k(\gamma)$ is an interior boundary. We will show this by iteratively constructing a left sense-preserving nullhomotopy H for γ. By property 4 of Theorem 9 it then follows that γ is a positive interior boundary and $\mathrm{obs}(\gamma) = 0$.

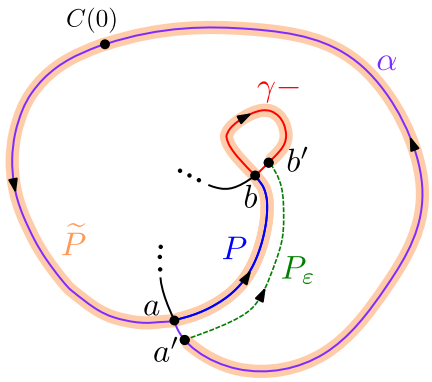


Figure 14 The combinatorial structure necessary to apply balanced loop deletion: a wrapped curve, with outer wrap α and a negatively oriented loop γ_- as first loop of C.

We first introduce a trick that we call **balanced loop deletion**. See Figure 14, where all of the following objects are shown. Suppose that $C \in \mathscr{C}$ is a curve that is positively wrapped, $C = Wr_+(C')$ for some curve $C' \in \mathscr{C}$, and suppose that the first loop γ_- of C, shown in red, is negatively oriented. Let $b = C(t_b) = C(t_b^*)$, with $t_b < t_b^*$, be the basepoint of γ_-. Balanced loop deletion performs a left sense-preserving homotopy H so that $C \overset{H}{\rightsquigarrow} C \setminus (\alpha \cup \gamma_-)$, where α, shown in purple, is the positive wrap on C. Let P, shown in blue, be the simple subpath of C from $a = C(t_a) = C(t_a^*)$ to b, where a is the unique outer intersection point on $[C]$, i.e., the basepoint of the wrap α, and $t_a < t_a^*$. For $\varepsilon > 0$ sufficiently small, let $a' = C(t_a^* + \varepsilon)$ and $b' = C(t_b^* - \varepsilon)$ and let P_ε, shown in dashed green, be a simple path between a' and b' that is ε-close to P in Hausdorff distance. Let $\widetilde{P} = C|_{[t_a^*+\varepsilon, t_b^*-\varepsilon]}$ (shown in thick beige) be the simple subpath of C from a' to b'. Then, $\widetilde{P}$ is the concatenation of (i) the path from a' to a along α, (ii) the path P from a to b, and (iii) the path from b to b' along γ_-. The path $\widetilde{P}$ is simple, because each of these subpaths are simple and none of them intersect each other since b is the first self-intersection point of the curve. Observe that $\widetilde{P} * P_\varepsilon$ is a simple, positively oriented, closed curve. It follows that we can perform a Blank cut along P_ε that replaces $\widetilde{P}$ on C with the path P_ϵ. The effect of this cut on C is that both the outer wrap α and the negatively oriented loop γ_- are deleted, and the path P is replaced by P_ε. This Blank cut can be performed by a left sense-preserving homotopy, so we have established the existence of left sense-preserving balanced loop deletion.

Now we construct a left sense-preserving nullhomotopy H of $\mathrm{Wr}_+^k(\gamma) = \gamma_1$ by iteratively concatenating several left sense-preserving subhomotopies, so $H = \sum_i H_i$. We proceed inductively as follows. Suppose $H_1, \ldots, H_{i-1}$ have been defined and γ_i is the current curve. Consider the first loop C_i of γ_i. If C_i is positively oriented we let H_i be the left sense-preserving nullhomotopy that contracts this loop. Otherwise C_i is negatively oriented and we let H_i be the homotopy performing balanced loop deletion.

We claim that there is a wrap available to perform this balanced loop deletion. Each homotopy H_j for $j = 1 \ldots i-1$ deletes one direct split and at most one indirect split of γ_j. Therefore the signs of the remaining intersection points are not affected. Observe that if a vertex v is the basepoint of an outwards loop γ_v, then $\mathrm{sgn}(v) = 1$ iff γ_v is positively oriented, and $\mathrm{sgn}(v) = -1$ iff γ_v is negatively oriented; see Figure 15. By definition of l, we have

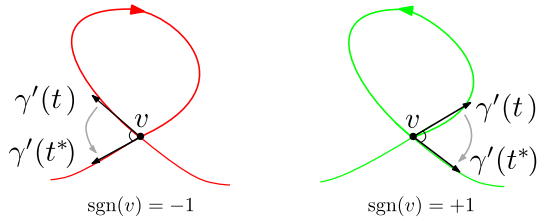


Figure 15 Two outwards loops; negatively oriented (left) and positively oriented (right).

$n_i \leq l$, where n_i is the number of negative vertices on γ_i. Therefore there can be at most l distinct integers $i_1, \ldots, i_l$ such that the first loop on γ_{i_ν} is negatively oriented, by Lemma 12 the first loop is always outwards. Since $k = l + 1$, there is a wrap available on $Wr_+^k(\gamma)$.

The process of constructing homotopies H_i never gets stuck, and $|\gamma_{i+1}| < |\gamma_i|$. Therefore we must eventually reach a point when the current curve γ_m has $|\gamma_m| = 0$. We now show that this final curve γ_m is positively oriented. Note that if γ is simple then $\mathrm{Wr}_+^k(\gamma)$ is trivially a positive k-interior boundary. So, assume γ is not simple.

By definition, the intersection sequence of $\mathrm{Wr}_+^k(\gamma)$ has the form $0, 1+, 2+, \ldots, k+, I', k-, \ldots, 1-, 0$, where I' is obtained from the signed intersection sequence of γ by incrementing each label by k and removing occurrences of 0. The basepoint of the first loop on $\gamma_1 = \mathrm{Wr}_+^k(\gamma)$ must be a vertex from γ. Then H_1 modifies the intersection sequence by removing two labels from I' corresponding to this loop. If the loop is negatively oriented, then balanced loop deletion also removes a pair $a+ \ldots a-$ for the wrap. The same modification happens for each homotopy H_i until I' is empty, and the homotopies after that contract wraps $a+ \ldots a-$ which are all positively oriented loops. We know that γ has at most $l = k - 1$ negative vertices, hence there can only be $k - 1$ balanced loop deletions, but there are k wraps. Thus, γ_m must be a wrap that $Wr_+^k(\gamma)$ added to γ, so γ_m is a positively oriented loop which can be contracted to its basepoint using a final left sense-preserving homotopy H_m. This shows that $H = \sum_{i=1}^m H_i$ is a left sense-preserving nullhomotopy of $\mathrm{Wr}_+^k(\gamma)$ as desired. ◀

The example in Figure 16 shows that the number of wraps used in Theorem 13 is nearly tight. We now show that wrapping resolves obstinance in either direction of wrapping. The proof is straight-forward and given in [7].

▶ **Theorem 14** (Wrapping Resolves Obstinance (General)). *Let $\gamma \in \mathscr{C}$ with outer basepoint and set $n = |\gamma|$. Then there are constants $k_-, k_+ \leq n+2$ so that $obs(Wr_-^{k_-}(\gamma)) = obs(Wr_+^{k_+}(\gamma)) = 0$, $Wr_-^{k_-}(\gamma)$ is a negative interior boundary, and $Wr_+^{k_+}(\gamma)$ is a positive interior boundary.*

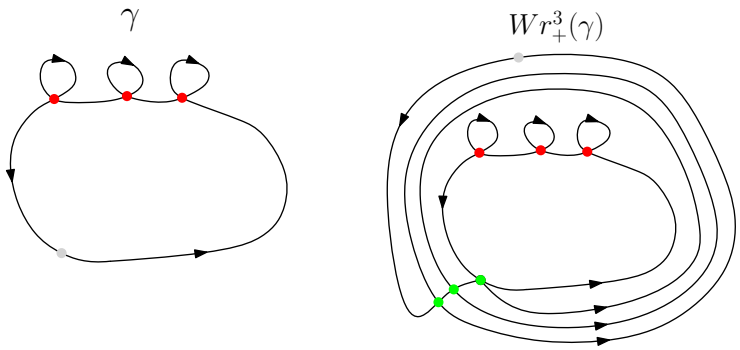


Figure 16 An example of a family of curves that require $k = l$ wraps to resolve obstinance, where l is the number of negative vertices in $V(\gamma)$. Here, $k = l = 3$.

Let us make a simple observation: once $Wr^k_\pm(\gamma)$ is an interior boundary, so too is $Wr^j_\pm(\gamma)$ for any integer $j \geq k$. This holds because we can simply add the extra $j - k$ wraps to the self-overlapping decomposition Ω of $Wr^k_+(\gamma)$.

4.3 Irreducible and Strongly Irreducible Curves

We are now ready to apply Theorem 14 to prove sufficient combinatorial conditions for a curve γ to be self-overlapping based on $\text{WHIT}(\gamma)$ and properties of its direct splits. If $\gamma \in \mathscr{C}$ has no proper positive self-overlapping direct splits, we call γ **irreducible**. A special case of irreducibility is of particular interest to us: If $\text{WHIT}(\gamma_v) \leq 0$ for all proper direct splits, we call γ **strongly irreducible**. See Figures 4 and 6 for examples of strongly irreducible curves. Note that a strongly irreducible curve is irreducible since any positive self-overlapping curve γ has $\text{WHIT}(\gamma) = 1$.

▶ **Lemma 15** (Existence of a Direct Split). *Let $\gamma \in \mathscr{C}$ and Ω be a direct split decomposition of γ, with $|\Omega| \geq 2$. Then Ω contains a proper direct split.*

Proof. A leaf v_i in the tree T_Ω necessarily corresponds to the basepoint of a direct split γ_i in the decomposition Ω. Since $|\Omega| \geq 2$, this direct split γ_i must be proper. ◀

▶ **Theorem 16** (Irreducible Curves are Self-Overlapping). *Assume γ has $WHIT(\gamma) = 1$ and positive outer basepoint. If γ is irreducible, then it is self-overlapping.*

Proof. Apply Theorem 13 to find a $k \in \mathbb{Z}$ such that $Wr^k_+(\gamma)$ is a positive interior boundary. We know from property 3 of Theorem 9 that there is a self-overlapping decomposition Ω of $Wr^k_+(\gamma)$ into positive self-overlapping subcurves. By Lemma 15 we know that Ω must have a self-overlapping direct split of $Wr^k_+(\gamma)$, and we will show that γ is the only direct split of $Wr^k_+(\gamma)$ that can be self-overlapping.

Let w_i be the vertex created by the i^{th} wrap. The intersection sequence of $Wr^k_+(\gamma)$ therefore has the prefix $w_k, w_{k-1}, \ldots, w_1$. Then the direct split $Wr^k_+(\gamma)_{w_i}$ at w_i on $Wr^k_+(\gamma)$ has $\text{WHIT}(Wr^k_+(\gamma)_{w_i}) = 1 + (i-1) = i$ by Lemma 6, and is therefore not self-overlapping for $i \geq 2$. And any direct split $Wr^k_+(\gamma)$ at a vertex of γ which is also a proper direct split on γ cannot be self-overlapping since γ is irreducible. Note that by our notation w_1 is the vertex corresponding to the original basepoint $\gamma(0)$. And this is the only vertex at which the direct split $Wr^k_+(\gamma)_{w_1} = \gamma$ could potentially be self-overlapping. Thus, it follows with Lemma 15 that γ is self-overlapping. ◀

We now have a nice corollary: conditions on the Whitney indices of a curve and its subcurves alone can be sufficient for self-overlappingness.

▶ **Theorem 17** (Strongly Irreducible Curves are Self-Overlapping). *Assume γ has $WHIT(\gamma) = 1$ and positive outer basepoint. If γ is strongly irreducible, then it is self-overlapping.*

Note that strongly irreducible curves are a proper subset of irreducible curves; see γ_I in Figure 2. And Theorem 17 is false without the basepoint assumption; see Figure 17.

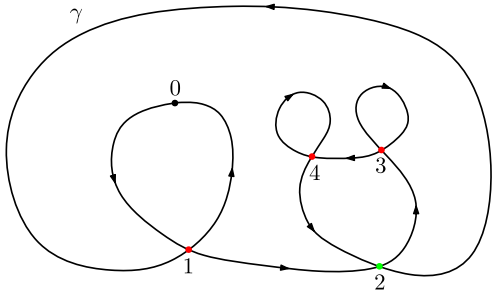


Figure 17 This curve γ does not have an outer basepoint. It is not self-overlapping, yet γ is strongly irreducible due to the empty positively oriented loop on the indirect split γ_{1^*}.

One can decide whether a piecewise linear curve γ is (strongly) irreducible by checking the required condition for each direct split. Let N be the number of line segments of γ and $n = |\gamma| = |V(\gamma)| \in O(N^2)$. Then irreducibility can be tested in $O(nN^3)$ time, using Shor and Van Wyk's algorithm to test for self-overlappingness in $O(N^3)$ time [17]. Strong irreducibility can be decided in $O(n^2)$ time by applying Corollary 7 to each direct split of γ.

5 Discussion

We introduced new curve classes (zero-obstinance, irreducable, and strongly irreducable curves; see Figure 2), which help us understand self-overlapping curves and interior boundaries. We proved combinatorial results and showed that wrapping a curve resolves obstinance. These new mathematical foundations for self-overlapping curves and interior boundaries could pave the way for related algorithmic questions. For example, is it possible to decide whether a curve is self-overlapping in $o(N^3)$ time? How fast can one decide self-overlappingness of a curve on the sphere? Can one decide irreducability in $o(n^2)$ time, even in the presence of a large number of linked subcurves?

References

1 Samuel J. Blank. *Extending Immersions of the Circle*. PhD thesis, Brandeis University, 1967.
2 Erin Chambers and David Letscher. On the height of a homotopy. In *21st Canadian Conference on Computational Geometry*, pages 103–106, 2009.
3 Erin Chambers and Yusu Wang. Measuring similarity between curves on 2-manifolds via homotopy area. In *Proc. Proc. 29th Symposium on Computational Geometry*, pages 425–434, 2013.
4 David Eppstein and Elena Mumford. Self-overlapping curves revisited. *SODA '09 Proceedings of the twentieth annual ACM-SIAM Symposium on Discrete Algorithms*, 2009.
5 Parker Evans. On self-overlapping curves, interior boundaries, and minimum area homotopies. Undergraduate Thesis, Tulane University, 2018.

6 Parker Evans, Andrea Burns, and Carola Wenk. Homotopy visualizer. `http://www.cs.tulane.edu/~carola/research/code.html`, 2016.

7 Parker Evans and Carola Wenk. Combinatorial properties of self-overlapping curves and interior boundaries, 2020. `arXiv:2003.13595`.

8 Brittany Terese Fasy, Selcuk Karakoc, and Carola Wenk. On minimum area homotopies of normal curves in the plane, 2017. `arXiv:1707.02251`.

9 Carl Friedrich Gauß. Zur Geometria Situs. In *Nachlass. I.* Teubner-Archiv zur Mathematik, ca. 1900.

10 Jack Graver and Gerald Cargo. When does a curve bound a distorted disk? *SIAM Journal on Discrete Mathematics*, 25(1):280–305, 2011.

11 Selcuk Karakoc. *On Minimum Homotopy Areas.* PhD thesis, Tulane University, 2017.

12 Morris Marx. The branch point structure of extensions of interior boundaries. *Transactions of the American Mathematical Society*, 131(1):79–98, 1968.

13 Morris Marx. Extending immersions of $\mathbb{S}^1$ to $\mathbb{R}^2$. *Transactions of the American Mathematical Society*, 187:309–326, 1974.

14 Uddipan Mukherjee. Self-overlapping curves: Analaysis and applications. *2013 SIAM Conference on Geometric and Physical Modeling*, 2013.

15 Uddipan Mukherjee, M. Gopi, and Jarek Rossignac. Immersion and embedding of self-crossing loops. *Proc. ACM/Eurographics Symposium on Sketch-Based Interfaces and Modeling*, pages 31–38, 2011.

16 Herbert Seifert. *Konstruktion dreidimensionaler geschlossener Raume.* PhD thesis, Saxon Academy of Sciences Leipzig, 1931.

17 Peter W. Shor and Christopher Van Wyk. Detecting and decomposing self-overlapping curves. *Computational Geometry: Theory and Applications*, 2(1):31–50, 1992.

18 Charles Titus. A theory of normal curves and some applications. *Pacific J. Math*, 10(3):1083–1096, 1960.

19 Charles Titus. The combinatorial topology of analytic functions of the boundary of a disk. *Acta Mathematica*, 106(1):45–64, 1961.

20 Hassler Whitney. On regular closed curves in the plane. *Compositio Math. 4*, pages 276–284, 1937.

Worst-Case Optimal Covering of Rectangles by Disks

Sándor P. Fekete
Department of Computer Science, TU Braunschweig, Germany
s.fekete@tu-bs.de

Utkarsh Gupta
Department of Computer Science & Engineering, IIT Bombay, India
utkarshgupta149@gmail.com

Phillip Keldenich
Department of Computer Science, TU Braunschweig, Germany
p.keldenich@tu-bs.de

Christian Scheffer
Department of Computer Science, TU Braunschweig, Germany
scheffer@ibr.cs.tu-bs.de

Sahil Shah
Department of Computer Science & Engineering, IIT Bombay, India
sahilshah00199@gmail.com

Abstract

We provide the solution for a fundamental problem of geometric optimization by giving a complete characterization of worst-case optimal disk coverings of rectangles: For any $\lambda \geq 1$, the critical covering area $A^*(\lambda)$ is the minimum value for which any set of disks with total area at least $A^*(\lambda)$ can cover a rectangle of dimensions $\lambda \times 1$. We show that there is a threshold value $\lambda_2 = \sqrt{\sqrt{7}/2 - 1/4} \approx 1.035797\ldots$, such that for $\lambda < \lambda_2$ the critical covering area $A^*(\lambda)$ is $A^*(\lambda) = 3\pi\left(\frac{\lambda^2}{16} + \frac{5}{32} + \frac{9}{256\lambda^2}\right)$, and for $\lambda \geq \lambda_2$, the critical area is $A^*(\lambda) = \pi(\lambda^2 + 2)/4$; these values are tight. For the special case $\lambda = 1$, i.e., for covering a unit square, the critical covering area is $\frac{195\pi}{256} \approx 2.39301\ldots$. The proof uses a careful combination of manual and automatic analysis, demonstrating the power of the employed interval arithmetic technique.

2012 ACM Subject Classification Theory of computation → Packing and covering problems; Theory of computation → Computational geometry

Keywords and phrases Disk covering, critical density, covering coefficient, tight worst-case bound, interval arithmetic, approximation

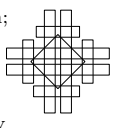
Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.42

Related Version A full version of this paper can be found at `https://arxiv.org/abs/2003.08236` [18].

Supplementary Material The code of the automatic prover can be found at `https://github.com/phillip-keldenich/circlecover`. Furthermore, there is a video contribution [20], video available at `https://www.ibr.cs.tu-bs.de/users/fekete/Videos/Cover_full.mp4`, illustrating the algorithm and proof presented in this paper.

Acknowledgements We thank Sebastian Morr and Arne Schmidt for helpful discussions. Major parts of the research by Utkarsh Gupta and Sahil Shah were carried out during a stay at TU Braunschweig.

36th International Symposium on Computational Geometry (SoCG 2020).
Editors: Sergio Cabello and Danny Z. Chen; Article No. 42; pp. 42:1–42:23



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

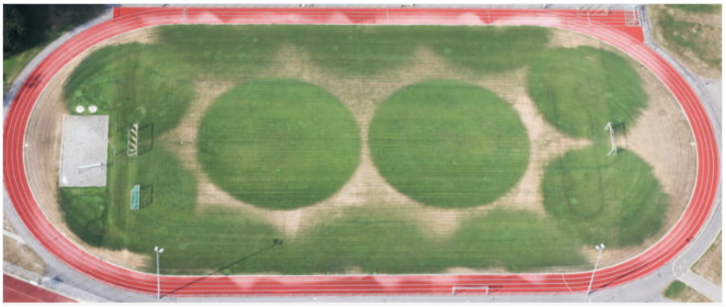


Figure 1 An incomplete covering of a rectangle by disks: Sprinklers on a soccer field during a drought. (Source: dpa [16].)

1 Introduction

Given a collection of (not necessarily equal) disks, is it possible to arrange them so that they completely cover a given region, such as a square or a rectangle? Covering problems of this type are of fundamental theoretical interest, but also have a variety of different applications, most notably in sensor networks, communication networks, wireless communication, surveillance, robotics, and even gardening and sports facility management, as shown in Fig. 1.

If the total area of the disks is small, it is clear that completely covering the region is impossible. On the other hand, if the total disk area is sufficiently large, finding a covering seems easy; however, for rectangles with large aspect ratio, a major fraction of the covering disks may be useless, so a relatively large total disk area may be required. The same issue is of clear importance for applications: What fraction of the total cost of disks can be put to efficient use for covering? This motivates the question of characterizing a critical threshold: For any given λ, find the minimum value $A^*(\lambda)$ for which any collection of disks with total area at least $A^*(\lambda)$ can cover a rectangle of dimensions $\lambda \times 1$. What is the critical covering area of $\lambda \times 1$ rectangles? In this paper we establish a complete and tight characterization.

1.1 Related Work

Like many other packing and covering problems, disk covering is typically quite difficult, compounded by the geometric complications of dealing with irrational coordinates that arise when arranging circular objects. This is reflected by the limitations of provably optimal results for the largest disk, square or triangle that can be covered by n unit disks, and hence, the "thinnest" disk covering, i.e., a covering of optimal density. As early as 1915, Neville [37] computed the optimal arrangement for covering a disk by five unit disks, but reported a wrong optimal value; much later, Bezdek[6, 7] gave the correct value for $n = 5, 6$. As recently as 2005, Fejes Tóth [45] established optimal values for $n = 8, 9, 10$. The question of incomplete coverings was raised in 2008 by Connelly, who asked how one should place n small disks of radius r to cover the largest possible area of a disk of radius $R > r$. Szalkai [44] gave an optimal solution for $n = 3$. For covering rectangles by n unit disks, Heppes and Mellissen [28] gave optimal solutions for $n \leq 5$; Melissen and Schuur [34] extended this for $n = 6, 7$. See Friedman [25] for the best known solutions for $n \leq 12$. Covering equilateral triangles by n unit disks has also been studied. Melissen [33] gave optimality results for $n \leq 10$, and conjectures for $n \leq 18$; the difficulty of these seemingly small problems is illustrated by the fact that Nurmela [38] gave conjectured optimal solutions for $n \leq 36$, improving the conjectured optimal covering for $n = 13$ of Melissen. Carmi et al. [11] considered algorithms

for covering point sets by unit disks at fixed locations. There are numerous other related problems and results; for relevant surveys, see Fejes Tóth [17] (Section 8), Fejes Tóth [46] (Chapter 2), Brass et al. [10] (Chapter 2) and the book by Böröczky [9].

Even less is known for covering by non-uniform disks, with most previous research focusing on algorithmic aspects. Alt et al. [3] gave algorithmic results for minimum-cost covering of point sets by disks, where the cost function is $\sum_j r_j^\alpha$ for some $\alpha > 1$, which includes the case of total disk area for $\alpha = 2$. Agnetis et al. [2] discussed covering a line segment with variable radius disks. Abu-Affash et al. [1] studied covering a polygon minimizing the sum of areas; for recent improvements, see Bhowmick et al. [8]. Bánhelyi et al. [4] gave algorithmic results for the covering of polygons by variable disks with prescribed centers.

For relevant applications, we mention the survey by Huang and Tseng [29] for wireless sensor networks, the work by Johnson et al. [30] on covering density for sensor networks, the algorithmic results for placing a given number of base stations to cover a square [13] and a convex region by Das et al. [14]. For minimum-cost sensor coverage of planar regions, see Xu et al. [47]; for wireless communication coverage of a square, see Singh and Sengupta [42], and Palatinus and Bánhelyi [40] for the context of telecommunication networks.

The analogous question of *packing* unit disks into a square has also attracted attention. For $n = 13$, the optimal value for the densest square covering was only established in 2003 [24], while the optimal value for 14 unit disks is still unproven; densest packings of n disks in equilateral triangles are subject to a long-standing conjecture by Erdős and Oler from 1961 [39] that is still open for $n = 15$. Other mathematical work on densely packing relatively small numbers of identical disks includes [26, 32, 22, 23], and [41, 31, 27] for related experimental work. The best known solutions for packing equal disks into squares, triangles and other shapes are published on Specht's website `http://packomania.com` [43].

Establishing the critical packing density for (not necessarily equal) disks in a square was proposed by Demaine, Fekete, and Lang [15] and solved by Morr, Fekete and Scheffer [36, 21]. Using a recursive procedure for cutting the container into triangular pieces, they proved that the critical packing density of disks in a square is $\frac{\pi}{3+2\sqrt{2}} \approx 0.539$. The critical density for (not necessarily equal) disks in a disk was recently proven to be 1/2 by Fekete, Keldenich and Scheffer [19]; see the video [5] for an overview and various animations. The critical packing density of (not necessarily equal) squares was established in 1967 by Moon and Moser [35], who used a shelf-packing approach to establish the value of 1/2 for packing into a square.

1.2 Our Contribution

We show that there is a threshold value $\lambda_2 = \sqrt{\sqrt{7}/2 - 1/4} \approx 1.035797\ldots$, such that for $\lambda < \lambda_2$ the critical covering area $A^*(\lambda)$ is $A^*(\lambda) = 3\pi\left(\frac{\lambda^2}{16} + \frac{5}{32} + \frac{9}{256\lambda^2}\right)$, and for $\lambda \geq \lambda_2$, the critical area is $A^*(\lambda) = \pi(\lambda^2+2)/4$. These values are tight: For any λ, any collection of disks of total area $A^*(\lambda)$ can be arranged to cover a $\lambda \times 1$-rectangle, and for any $a(\lambda) < A^*(\lambda)$, there is a collection of disks of total area $a(\lambda)$ such that a $\lambda \times 1$-rectangle cannot be covered. (See Fig. 2 for a graph showing the (normalized) critical covering density, and Fig. 3 for examples of worst-case configurations.) The point $\lambda = \lambda_2$ is the unique real number greater than 1 for which the two bounds $3\pi\left(\frac{\lambda^2}{16} + \frac{5}{32} + \frac{9}{256\lambda^2}\right)$ and $\pi\frac{\lambda^2+2}{4}$ coincide; see Fig. 2. At this so-called *threshold value*, the worst case changes from three identical disks to two disks – the circumcircle $r_1^2 = \frac{\lambda^2+1}{4}$ and a disk $r_2^2 = \frac{1}{4}$; see Fig. 3. For the special case $\lambda = 1$, i.e., for covering a unit square, the critical covering area is $\frac{195\pi}{256} \approx 2.39301\ldots$.

The proof uses a careful combination of manual and automatic analysis, demonstrating the power of the employed interval arithmetic technique.

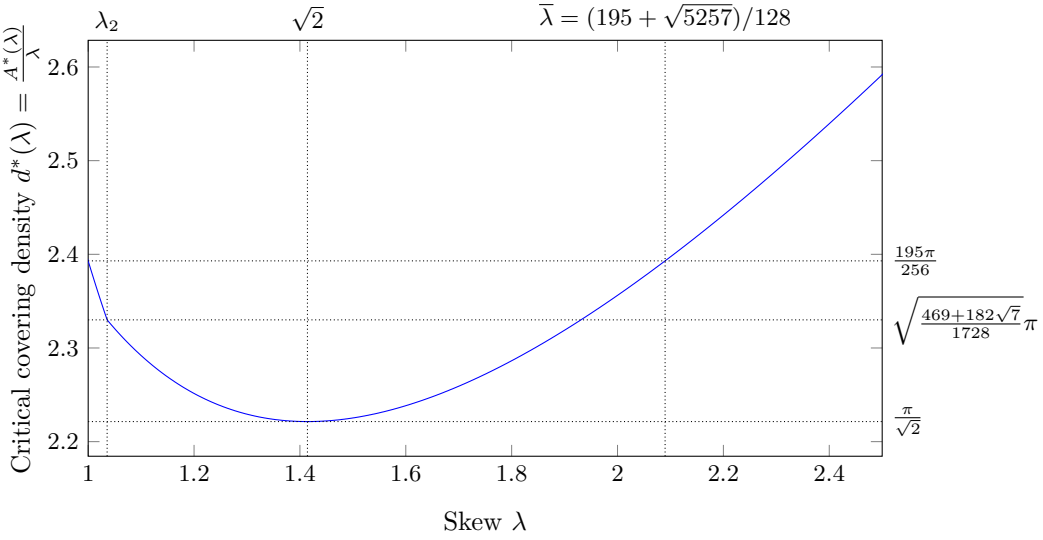


Figure 2 The critical covering density $d^*(\lambda)$ depending on λ and its values at the threshold value λ_2, the global minimum $\sqrt{2}$ and the skew $\overline{\lambda}$ at which the density becomes as bad as for the square.

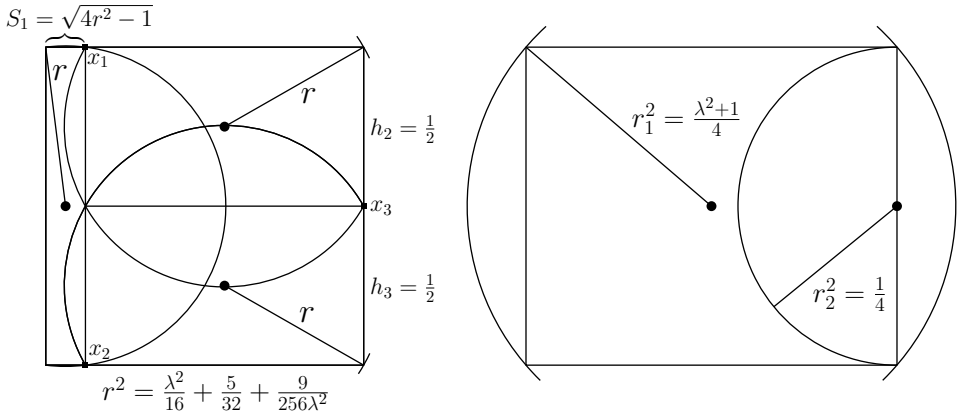


Figure 3 Worst-case configurations for small $\lambda \leq \lambda_2$ (left) and for large skew $\lambda \geq \lambda_2$ (right). Shrinking r or r_1 by any $\varepsilon > 0$ in either configuration leads to an instance that cannot be covered.

2 Preliminaries

We are given a rectangular container $\mathcal{R}$, which we assume w.l.o.g. to have height 1 and some width $\lambda \geq 1$, which is called the *skew* of $\mathcal{R}$. For a collection $D = \{r_1, \ldots, r_n\}$ of radii $r_1 \geq r_2 \geq \cdots \geq r_n$, we want to decide whether there is a placement of n closed disks with radii $r_1, \ldots, r_n$ on $\mathcal{R}$, such that every point $x \in \mathcal{R}$ is covered by at least one disk. Because we are only given radii and not center points, in a slight abuse of notation, we identify the disks with their radii and use r_i to refer to both the disk and the radius.

For any set D of disks, the *total disk area* is $A(D) \coloneqq \pi \sum_{r \in D} r^2$. The *weight* of a disk of radius r is r^2, and $W(D) \coloneqq \frac{A(D)}{\pi}$ is the *total weight* of D. For any rectangle $\mathcal{R}$, the *critical covering area* $A^*(\mathcal{R})$ of $\mathcal{R}$ is the minimum value for which any set D of disks with total area at least $A(D) \geq A^*(\mathcal{R})$ can cover $\mathcal{R}$. The *critical covering weight* of $\mathcal{R}$ is $W^*(\mathcal{R}) \coloneqq \frac{A^*(\mathcal{R})}{\pi}$. For $\lambda \geq 1$, we define $A^*(\lambda) \coloneqq A^*(\mathcal{R})$ and $W^*(\lambda) \coloneqq W^*(\mathcal{R})$ for a $\lambda \times 1$ rectangle $\mathcal{R}$.

For a placement $\mathcal{P}$ of the disks in D fully covering some area A, the *covering coefficient* of $\mathcal{P}$ is the ratio $\frac{W(D)}{A}$. For $\lambda \geq 1$, the amount $E^*(\lambda) := \frac{W^*(\lambda)}{\lambda}$ of total disk weight per unit of rectangle area that is necessary for guaranteeing a possible covering is the *(critical) covering coefficient* of λ. Analogously, $d^*(\lambda) := \frac{A^*(\lambda)}{\lambda}$ is the *(critical) covering density* of λ.

For proving our result, we use Greedy Splitting for partitioning a collection of disks into two parts whose weight differs by at most the weight of the smallest disk in the heavier part: After sorting the disks by decreasing radius, we start with two empty lists and continue to place the next disk in the list with smaller total weight.

3 High-Level Description

Now we present and describe our main result: a theorem that characterizes the worst case for covering rectangles with disks. This theorem gives a closed-form solution for the *critical covering area* $A^*(\lambda)$ for any $\lambda \geq 1$; in other words, for any given rectangle $\mathcal{R}$, we determine the total disk area that is (1) sometimes necessary and (2) always sufficient to cover $\mathcal{R}$.

▶ **Theorem 1.** *Let $\lambda \geq 1$ and let $\mathcal{R}$ be a rectangle of dimensions $\lambda \times 1$. Let*

$$\lambda_2 = \sqrt{\frac{\sqrt{7}}{2} - \frac{1}{4}} \approx 1.035797\ldots, \text{ and } A^*(\lambda) = \begin{cases} 3\pi\left(\frac{\lambda^2}{16} + \frac{5}{32} + \frac{9}{256\lambda^2}\right), & \text{if } \lambda < \lambda_2, \\ \pi\frac{\lambda^2+2}{4}, & \text{otherwise.} \end{cases}$$

(1) *For any $a < A^*(\lambda)$, there is a set D^- of disks with $A(D^-) = a$ that cannot cover $\mathcal{R}$.*
(2) *Let $D = \{r_1, \ldots, r_n\} \subset \mathbb{R}$, $r_1 \geq r_2 \geq \ldots \geq r_n > 0$ be any collection of disks identified by their radii. If $A(D) \geq A^*(\lambda)$, then D can cover $\mathcal{R}$.*

The critical covering area does not depend linearly on the area λ of the rectangle; it also depends on the rectangle's skew. Fig. 2 shows a plot of the dependency of the covering density $d(\lambda)$ on λ. In the following, to simplify notation, we factor out π if possible; instead of working with the areas $A(D)$ or $A^*(\lambda)$ of the disks, we use their *weight*, i.e., their area divided by π. Similarly, we work with the covering coefficient $E^*(\lambda)$ instead of the density $d^*(\lambda)$; a lower covering coefficient corresponds to a more efficient covering.

As shown in Fig. 2, the critical covering coefficient $E^*(\lambda)$ is monotonically decreasing from $\lambda = 1$ to $\sqrt{2}$ and monotonically increasing for $\lambda > \sqrt{2}$. For a square, $E^*(1) = \frac{195}{256}$; the point $\lambda > 1$ for which the covering coefficient becomes as bad as for the square is $\overline{\lambda} := \frac{195+\sqrt{5257}}{128} \approx 2.08988\ldots$; for all $\lambda \leq \overline{\lambda}$, the covering coefficient is at most $\frac{195}{256}$.

3.1 Proof Components

The proof of Theorem 1 uses a number of components. First is a lemma that describes the worst-case configurations and shows tightness, i.e., claim (1), of Theorem 1 for all λ.

▶ **Lemma 2.** *Let $\lambda \geq 1$ and let $\mathcal{R}$ be a rectangle of dimensions $\lambda \times 1$. (1) Two disks of weight $r_1^2 = \frac{\lambda^2+1}{4}$ and $r_2^2 = \frac{1}{4}$ suffice to cover $\mathcal{R}$. (2) For any $\varepsilon > 0$, two disks of weight $r_1^2 - \varepsilon$ and r_2^2 do not suffice to cover $\mathcal{R}$. (3) Three identical disks of weight $r^2 = \frac{\lambda^2}{16} + \frac{5}{32} + \frac{9}{256\lambda^2}$ suffice to cover a rectangle $\mathcal{R}$ of dimensions $\lambda \times 1$. (4) For $\lambda \leq \lambda_2$ and any $\varepsilon > 0$, three identical disks of weight $r_-^2 := r^2 - \varepsilon$ do not suffice to cover $\mathcal{R}$.*

For large λ, the critical covering coefficient $E^*(\lambda)$ of Theorem 1 becomes worse, as large disks cannot be used to cover the rectangle efficiently. If the weight of each disk is bounded by some $\sigma \geq r_1^2$, we provide the following lemma achieving a better covering coefficient $E(\sigma)$ with $E^*(\overline{\lambda}) \leq E(\sigma) \leq E^*(\lambda)$. This coefficient is independent of the skew of $\mathcal{R}$.

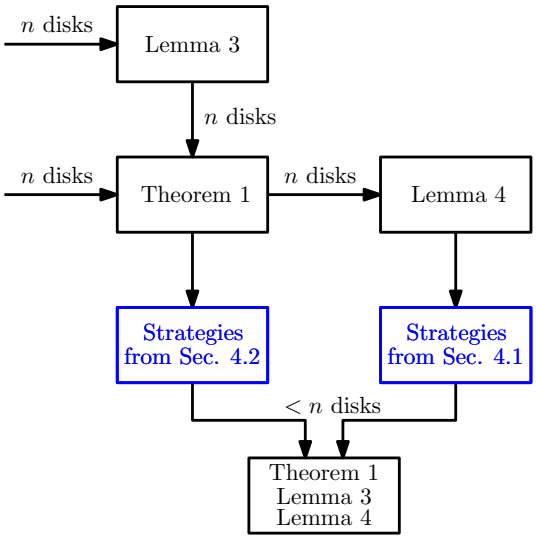


Figure 4 The inductive structure of the proof; the blue parts are computer-aided.

▶ **Lemma 3.** *Let* $\hat{\sigma} \coloneqq \frac{195\sqrt{5257}}{16384} \approx 0.8629$. *Let* $\sigma \geq \hat{\sigma}$ *and* $E(\sigma) \coloneqq \frac{1}{2}\sqrt{\sqrt{\sigma^2+1}+1}$. *Let* $\lambda \geq 1$ *and* $D = \{r_1, \ldots, r_n\}$ *be any collection of disks with* $\sigma \geq r_1^2 \geq \ldots \geq r_n^2$ *and* $W(D) = \sum_{i=1}^{n} r_i^2 \geq E(\sigma)\lambda$. *Then* D *can cover a rectangle* $\mathcal{R}$ *of dimensions* $\lambda \times 1$.

Note that $E(\hat{\sigma}) = \frac{195}{256}$, i.e. the best covering coefficient established by Lemma 3, coinciding with the critical covering coefficient of the square established by Theorem 1. Thus, we can cover any rectangle with covering coefficient $\frac{195}{256}$ if the largest disk satisfies $r_1^2 \leq \hat{\sigma}$.

The final component is the following Lemma 4, which also gives a better covering coefficient if the size of the largest disk is bounded. The bound required for Lemma 4 is smaller than for Lemma 3; in return, the covering coefficient that Lemma 4 yields is better. Note that the result of Lemma 4 is not tight.

▶ **Lemma 4.** *Let* $\lambda \geq 1$ *and let* $\mathcal{R}$ *be a rectangle of dimensions* $\lambda \times 1$. *Let* $D = \{r_1, \ldots, r_n\}$, $0.375 \geq r_1 \geq \ldots \geq r_n > 0$ *be a collection of disks. If* $W(D) \geq 0.61\lambda$, *or equivalently* $A(D) \geq 0.61\pi\lambda \approx 1.9164\lambda$, *then* D *suffices to cover* $\mathcal{R}$.

3.2 Proof Overview

The proofs of Theorem 1 and Lemmas 3 and 4 work by induction on the number of disks. For proving Lemma 3 for n disks, we use Theorem 1 for n disks. For proving Theorem 1 for n disks, we use Lemma 4 for n disks; Lemma 3 is only used for fewer than n disks; see Fig. 4. For proving Lemma 4 for n disks, we only use Theorem 1 and Lemma 3 for fewer than n disks. Therefore, there are no cyclic dependencies in our argument; however, we have to perform the induction for Theorem 1 and Lemmas 3 and 4 simultaneously.

Routines. The proofs of Theorem 1 and Lemma 4 are constructive; they are based on an efficient recursive algorithm that uses a set of simple *routines*. We go through the list of routines in some fixed order. For each routine, we check a sufficient criterion for the routine to work. We call these criteria *success criteria*. They only depend on the total available weight and a constant number of largest disks. If we cannot guarantee that a routine works

by its success criterion, we simply disregard the routine; this means that our algorithm does not have to backtrack. We prove that, regardless of the distribution of the disks' weight, at least one success criterion is met, implying that we can always apply at least one routine. The number of routines and thus success criteria is large; this is where the need for automatic assistance comes from.

Recursion. Typical routines are recursive; they consist of splitting the collection of disks into smaller parts, splitting the rectangle accordingly, and recursing, or recursing after fixing the position of a constant number of large disks.

In the entire remaining proof, the criterion we use to guarantee that recursion works is as follows. Given a collection $D' \subsetneq D$ and a rectangular region $\mathcal{R}' \subsetneq \mathcal{R}$, we check whether the preconditions of Theorem 1 or Lemma 3 or 4 are met after appropriately scaling and rotating $\mathcal{R}'$ and the disks. Note that, due to the scaling, the radius bounds of Lemmas 3 and 4 depend on the length of the shorter side of $\mathcal{R}'$. In some cases where we apply recursion, we have more weight than necessary to satisfy the weight requirement for recursion according to Lemma 3 or 4, but these lemmas cannot be applied due to the radius bound. In that case, we also check whether we can apply Lemma 3 or 4 after increasing the length of the shorter side of $\mathcal{R}'$ as far as the disk weight allows. This excludes the case that we cannot recurse on $\mathcal{R}'$ due to the radius bound, but there is some $\mathcal{R}'' \supset \mathcal{R}'$ on which we could recurse.

3.3 Interval Arithmetic

We use interval arithmetic to prove that there always is a successful routine. In interval arithmetic, operations like addition, multiplication or taking a square root are performed on intervals $[a, b] \subset \mathbb{R}$ instead of numbers. Arithmetic operations on intervals are derived from their real counterparts as follows. The result of an operation $\circ$ in interval arithmetic is

$$[a_1, b_1] \circ [a_2, b_2] \coloneqq \left[\min_{x_1 \in [a_1, b_1], x_2 \in [a_2, b_2]} x_1 \circ x_2, \max_{x_1 \in [a_1, b_1], x_2 \in [a_2, b_2]} x_1 \circ x_2 \right].$$

Thus, the result of an operation is the smallest interval that contains all possible results of $x \circ y$ for $x \in [a_1, b_1], y \in [a_2, b_2]$. Unary operations are defined analogously. For square roots, division or other operations that are not defined on all of $\mathbb{R}$, a result is undefined iff the input interval(s) contain values for which the real counterpart of the operation is undefined.

Truth values. In interval arithmetic, inequalities such as $[a_1, b_1] \leq [a_2, b_2]$ can have three possible truth values. An inequality can be *definitely true*; this means that the inequality holds for any value of $x \in [a_1, b_1], y \in [a_2, b_2]$. In the example $[a_1, b_1] \leq [a_2, b_2]$, this is the case if $b_1 \leq a_2$. An inequality can be *indeterminate*; this means that there are some values $x, x' \in [a_1, b_1], y, y' \in [a_2, b_2]$ such that the inequality holds for x, y and does not hold for x', y'. In the example $[a_1, b_1] \leq [a_2, b_2]$, this is the case if $a_1 \leq b_2$ and $b_1 > a_2$. Otherwise, an inequality is *definitely false*. An inequality that is either *definitely true* or *indeterminate* is called *possibly true*; an inequality that is either *indeterminate* or *definitely false* is called *possibly false*. These truth values can also be interpreted as intervals $[0, 0], [0, 1], [1, 1]$.

Using interval arithmetic. We apply interval arithmetic in our proof as follows. Recall that for each routine, we have a *success criterion*. These criteria only consider $\lambda \geq 1$ and the largest $k \in \mathcal{O}(1)$ disks $r_1 \geq \cdots \geq r_k$ as well as the remaining weight $R_{k+1} \coloneqq \sum_{i=k+1}^{n} r_i^2$, which can be computed from λ and $r_1, \ldots, r_k$, assuming w.l.o.g. that the total disk weight $W(D)$ is exactly $W^*(\lambda)$.

If we can manually perform induction base and induction step of our result for all $\lambda \geq \hat{\lambda}$ for some finite value $\hat{\lambda}$, we can also provide an upper bound $\hat{r}_1$ for r_1 such that all cases that remain to be considered (in our induction base and induction step) correspond to a point in the $(k+1)$-dimensional space Ψ given by

$$\lambda \in [1, \hat{\lambda}], r_1 \in [0, \hat{r}_1], r_2 \in [0, r_1], \ldots, r_k \in [0, r_{k-1}], \sum_{i=1}^{k} r_i^2 \leq W^*(\lambda).$$

This is due to the fact that there is nothing to prove if r_1 can cover $\mathcal{R}$ on its own; r_1 can have no more than the total disk weight $W(D)$ and $r_k \leq \cdots \leq r_2 \leq r_1$. Furthermore, observe that the induction base is just a special case with $r_i = r_{i+1} = \cdots = 0$ for some $1 < i \leq k$.

This allows subdividing (a superset of) Ψ into a large finite number of *hypercuboids* by splitting the range of each of the variables $\lambda, r_1, \ldots, r_k$ into a number of smaller intervals. For each hypercuboid, we then use interval arithmetic to verify that there is a routine whose success criterion is met. If we find such a routine, we have eliminated all points in that hypercuboid from further consideration. Hypercuboids for which this does not succeed are called *critical* and must be resolved manually; note that, in particular, hypercuboids containing (tight) worst-case configurations cannot be handled by interval arithmetic. The restriction to critical hypercuboids makes the overall analysis feasible, while a manual analysis of the entire space is impractical due to the large number of routines and variables.

Implementation. We implemented the subdivision outlined above and all success criteria of our routines using interval arithmetic[1]. Because most of our success criteria use the squared radii r_i^2 instead of the radii r_i, we use λ and r_i^2 instead of r_i as variables. Moreover, for efficiency reasons, instead of the simple grid-like subdivision outlined above, we use a search-tree-like subdivision strategy where we begin by subdividing the range of λ, continue by subdividing r_1^2, followed by r_2^2, and so on. Whenever a success criterion only needs the first $i < k$ disks, we can check this criterion farther up in the tree, thus potentially avoiding visits to large parts of the search tree; see Fig. 5 for a sketch of this procedure. Even with this pruning in place, the number of hypercuboids that we have to consider is still very large; this is a result of the fact that, depending on the claim at stake, we have 5 or even 8 dimensions. Therefore, we implemented the checks for our success criteria on a CUDA-capable GPU to perform them in a massively parallel fashion.

Moreover, to provide a finer subdivision where necessary, we run our search in several *generations* (our proof uses 11 generations). Each generation yields a set of critical hypercuboids that could not be handled automatically. After each generation, for each subinterval of λ, we collect all critical hypercuboids and merge those for which the r_1^2-subintervals are overlapping by taking the smallest hypercuboid containing all points in the merged hypercuboids. This procedure typically yields only 1-3 hypercuboids per subinterval of λ. The next generation is run on each of these, starting with the bounds given by these hypercuboids.

Numerical issues. When performing computations on a computer with limited-precision floating-point numbers instead of real numbers, there can be rounding errors, underflow errors and overflow errors. Our implementation of interval arithmetic performs all operations using appropriate rounding modes; this technique is also used by the implementation of interval

[1] The source code of the implementation is available online: `https://github.com/phillip-keldenich/circlecover` .

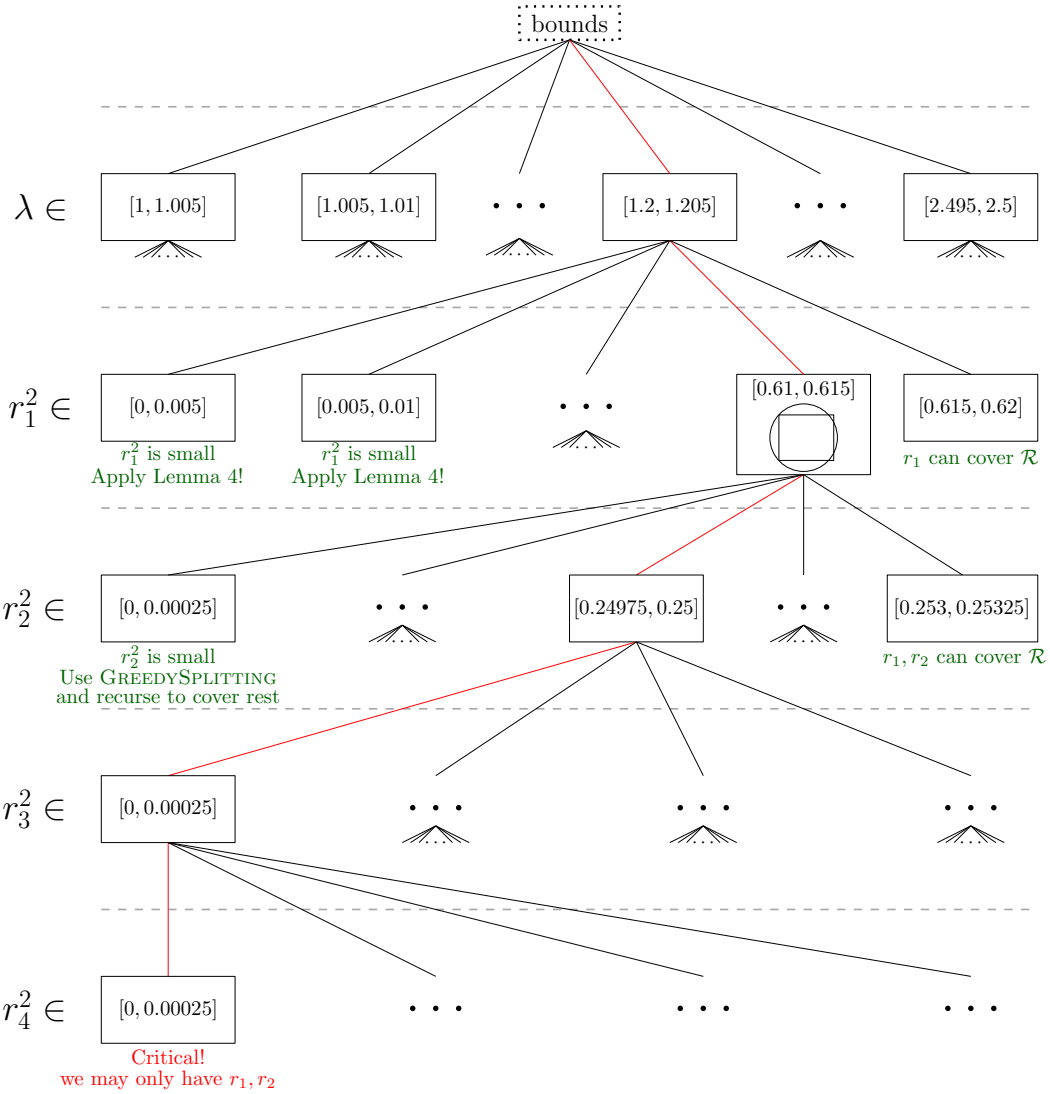


Figure 5 Sketch of our interval arithmetic-based search procedure. The red edges denote a path leading to a critical cuboid containing a tight two-disk worst-case configuration. Green text indicates that the children of the corresponding node do not have to be considered.

arithmetic in the well-known Computational Geometry Algorithms Library (CGAL) [12]. This means that any operation $\circ$ on two intervals A, B yields an interval $I \supseteq A \circ B$ to ensure that the result of any operation contains all values that are possible outcomes of $x \circ y$ for $x, y \in A, B$. This guarantees soundness of our results in the presence of numerical errors.

4 Proof Structure

In this section, we give an overview of the structure of the proofs of Theorem 1 and Lemmas 2, 3 and 4. For the proof of Lemma 2, we refer to the full version [18] of our paper. Lemma 3 is proven in Section 4.3 using a simple recursive algorithm; basically, we show that we can always split the disks using Greedy Splitting, split the rectangle accordingly, and recurse using Theorem 1. The proofs of Theorem 1 and Lemma 4 involve a larger number of routines and make use of an automatic prover based on interval arithmetic as described in Section 3.3.

4.1 Proof Structure for Lemma 4

Proving Lemma 4 means proving that, for any skew λ, any collection D of disks of radius $r_1 \leq 0.375$ and with total weight $W(D) = E\lambda$ suffices to cover $\mathcal{R}$, where $E = 0.61$ is the covering coefficient guaranteed by Lemma 4. We first reduce the number of cases that we have to consider in our induction base and induction step to a finite number. As described in Section 3.3, this requires handling the case of arbitrarily large skew λ. Finding a bound $\hat{\lambda}$ and reducing Lemma 4 for $\lambda \geq \hat{\lambda}$ to the case of $\lambda < \hat{\lambda}$ yields bounds for λ and $r_1, \ldots, r_k$ that allow a reduction to finitely many cases using interval arithmetic.

▶ **Lemma 5.** *Let $\hat{\lambda} = 2.5$. Given disks D according to the preconditions of Lemma 4 and $\lambda \geq \hat{\lambda}$, we can cover $\mathcal{R}$ using a simple recursive routine.*

Proof. The routine works as follows. We build a list of disks D_1 by adding disks in decreasing order of radius until $W(D_1) \geq E$. Due to the radius bound, this procedure always stops before all disks are used, i.e., $D_1 \subsetneq D$. Let $D_2 \coloneqq D \setminus D_2$ be the remaining disks. We then place a vertical rectangular strip $\mathcal{R}_1$ of height 1 and width $\beta_{\mathcal{R}_1} \coloneqq \frac{W(D_1)}{E} \geq 1$ at the left side of $\mathcal{R}$. By induction, we can recurse on $\mathcal{R}_1$ using Lemma 4 and the disks from D_1, because both side lengths are at least 1 and the efficiency we require is *exactly* E. Note that, due to adapting the width $\beta_{\mathcal{R}_1}$ according to the actual weight $W(D_1)$, we actually achieve an efficiency of E; in other words, there is no *waste* of disk weight. This means that we also require an efficiency of exactly E on the remaining rectangle $\mathcal{R}_2 \coloneqq \mathcal{R} \setminus \mathcal{R}_1$. Therefore, provided that the largest disk in D_2 satisfies the size bound of Lemma 4, we can inductively apply Lemma 4 to $\mathcal{R}_2$ and D_2 and are done. This can be guaranteed by proving that the shorter side of $\mathcal{R}_2$ is at least 1 as well. We have $W(D_1) \leq E + r_1^2 \leq E + 0.375^2$ which implies $\beta_{\mathcal{R}_1} \leq 1 + \frac{0.375^2}{E} < 1.5$; therefore, $\lambda \geq 2.5$ ensures that the width of $\mathcal{R}_2$ is at least 1. ◀

As outlined in Section 3.3, the remainder of the proof of Lemma 4 is based on a list of simple covering routines and their success criteria. We prove that there always is a working routine in that list using an automatic prover based on interval arithmetic, as described in Section 3.3. This automatic prover considers the 8-dimensional space spanned by the variables λ and $r_1^2, \ldots, r_7^2$ and subdivides it into a total of more than 2^{46} hypercuboids in order to prove that there always is a working routine, i.e., no critical hypercuboids remain to be analyzed manually; this only works because the result of Lemma 4 is not tight.

In the following, we give a brief description of the routines that we use. Due to space constraints, for a detailed description of the routines, we refer to the full version of our paper [18].

Recursive splitting. Routines (S-I.1) and (S-I.2) work by splitting D into two parts, splitting $\mathcal{R}$ accordingly, and recursing on the two sub-rectangles. This split is either performed as balanced as possible using Greedy Splitting, or in an unbalanced manner; in the latter case, we choose an unbalanced split to accommodate large disks that violate the radius bound of Lemma 4 w.r.t. a rectangle of half the width of $\mathcal{R}$.

Building a strip. Routine (S-II.1) works by either covering the left or the bottom side of a rectangular strip $\mathcal{R}$; see Fig. 6. This strip uses a subset of the largest six disks and tries several configurations for placing the disks. The remaining area is covered by recursion.

Wall building. Routines (S-III.1) and (S-IV.1) are based on the idea of covering a rectangular strip of fixed length ℓ and variable width b with covering coefficient exactly E. We call this *wall building*. To achieve this covering coefficient, we stack disks of similar size on top

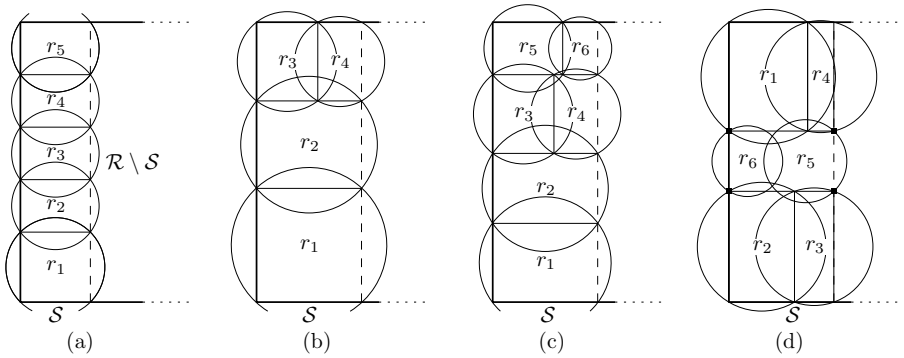


Figure 6 Some placements considered by Routine S-II.1 to build a vertical strip; horizontal strips are analogous. (a) Simply stacking a subset T of the six largest disks on top of each other. (b) Stacking r_1, r_2 on top of each other, and placing r_3, r_4 horizontally next to each other on top. (c) Same as (b), but with an additional row built from r_5, r_6. (d) Building two rows at the top and the bottom consisting of r_1, r_4 and r_2, r_3, and covering the remaining region by r_5, r_6. The points on the boundary defining the position of r_5 and r_6 are marked by squares. Note that r_5 and r_6 are not big enough to cover the entire rectangular area between the top and the bottom row.

of (or horizontally next to) each other; each disk placed in this way covers a rectangle of variable height, but width b. We provide sufficient conditions for this procedure to result in a successful covering of a strip of length ℓ. Routine (S-III.1) uses this idea to build a column of stacked disks at the left side of $\mathcal{R}$; see Fig. 7. Routine (S-IV.1) uses this idea by placing r_1 in the bottom-left corner of $\mathcal{R}$ and filling the area above r_1 with horizontal rows of disks; see Fig. 8. Intuitively speaking, these routines are necessary to handle cases in which there are large disks that interfere with recursion, but small disks, for which we do not know the weight distribution, significantly contribute to the total weight.

Using the two largest disks. Routine (S-V.1) places the two largest disks in diagonally opposite corners, each disk covering its inscribed square; see Fig. 9. The remaining area is subdivided into three rectangular regions; we cover these regions recursively, considering several ways to split the remaining disks.

Using the three largest disks. Routines (S-VI.1) and (S-VI.2) consider two different placements of the largest three disks as shown in Fig. 10.

Using the four largest disks. Routines (S-VII.1)–(S-VII.3) consider different placements of the four largest disks and recursion to cover $\mathcal{R}$; see Fig. 11.

Using the five largest disks. Routines (S-VIII.1) and (S-VIII.2) consider different placements of the five largest disks and recursion to cover $\mathcal{R}$; see Fig. 12.

Using the six largest disks. Routines (S-IX.1)–(S-IX.3) consider different placements of the six largest disks and recursion to cover $\mathcal{R}$; see Fig. 13.

Using the seven largest disks. Routines (S-X.1)–(S-X.8) consider different placements of the seven largest disks, together with recursion, to cover $\mathcal{R}$; see Figs. 14, 15 and 16.

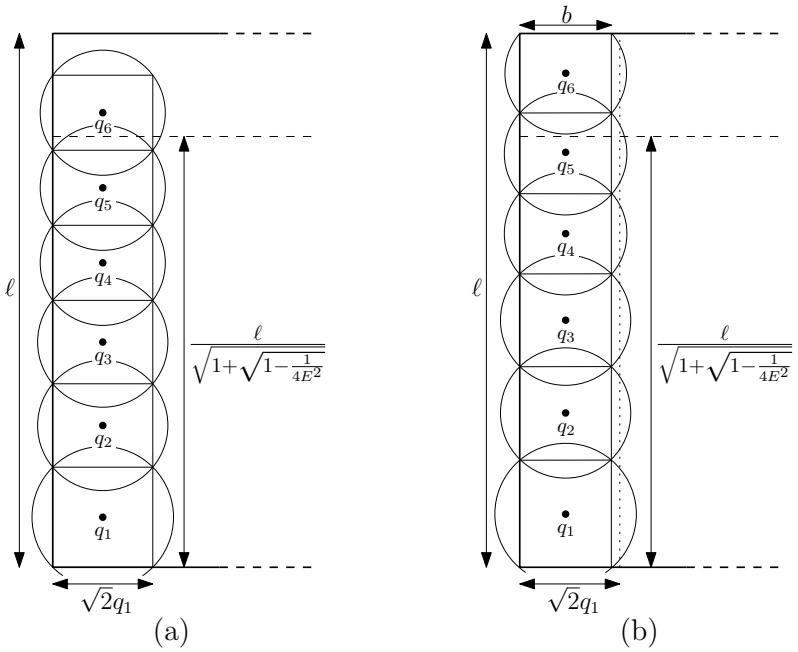


Figure 7 The wall-building procedure. (a) Using an initial guess of $b = \sqrt{2}q_1$ as width, where q_1 is the largest disk that we use, we stack disks until they exceed a certain fraction of the length ℓ. (b) We decrease b until the disks exactly cover a strip of length ℓ.

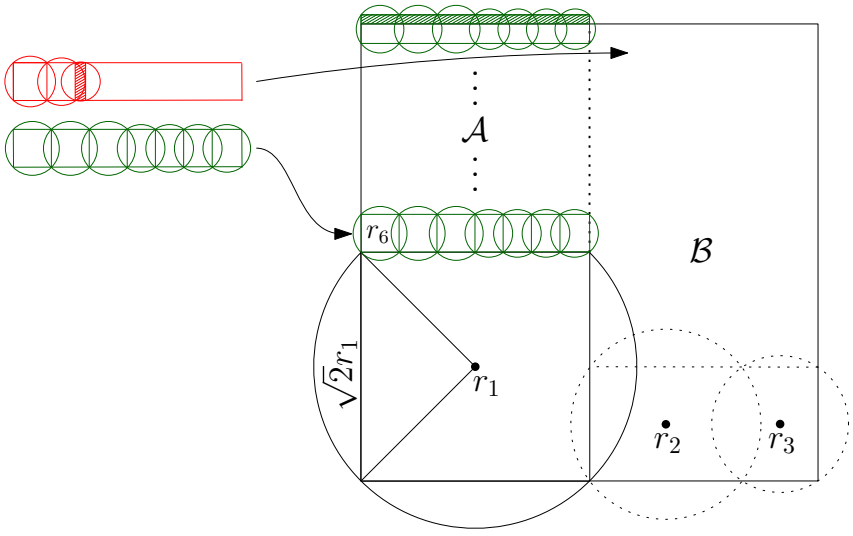


Figure 8 Routine S-IV.1 places r_1 in the bottom-left corner and tries to cover $\mathcal{A}$ using either recursion or wall building. In the latter case, whenever the disk radius drops too much while building a row of length $\ell = \sqrt{2}r_1$, we move the disks constituting this incomplete row to $\mathcal{B}$ (red). Otherwise, a complete row is built (green) and we continue with the next row. This process stops once the entire area $\mathcal{A}$ is covered, including some potential overhead (shaded green region). We compensate for the overhead by the area gained by placing r_1 covering a square. In case r_2 does not fit into $\mathcal{B}$ recursively, we try placing r_2, r_3 (or r_2, r_3, r_4) at the bottom of $\mathcal{B}$ (dotted outline).

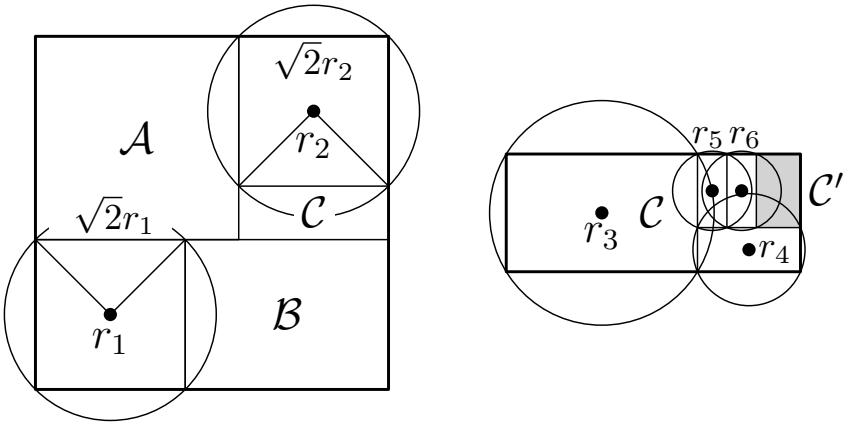


Figure 9 The routine S-V.1 places r_1 and r_2 in diagonally opposite corners, each covering a square. We cover three remaining rectangular areas $\mathcal{A}, \mathcal{B}, \mathcal{C}$ using the remaining disks (left). Regions $\mathcal{A}, \mathcal{B}$ and $\mathcal{C}$ are covered by recursion; we also consider using disks $r_3, \ldots, r_6$ to reduce $\mathcal{C}$ to $\mathcal{C}'$ (light gray) before recursing.

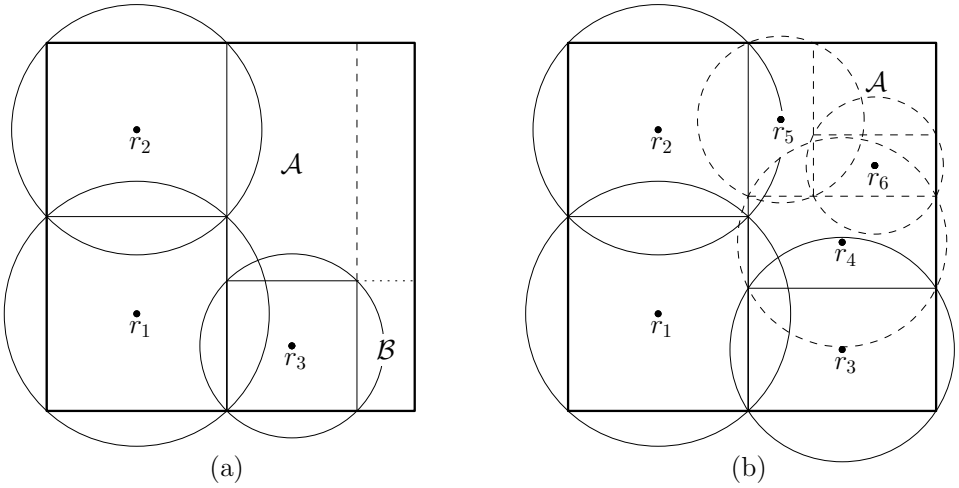


Figure 10 Two routines based on using the three largest disks. We use r_1 and r_2 to cover a vertical strip of height 1 and maximal width. (a) In Routine S-VI.1, we place r_3 to the right of r_1, covering its inscribed square at the lower left corner of the remaining rectangle; the remaining region can be subdivided into two rectangles $\mathcal{A}, \mathcal{B}$ in two ways (dashed and dotted line). (b) In Routine S-VI.2, we cover a horizontal strip of the remaining rectangle using r_3; we either recurse on the remaining rectangle directly or place some of the disks r_4, r_5, r_6 to cut off pieces of the longer side of the remaining rectangle (dashed outlines).

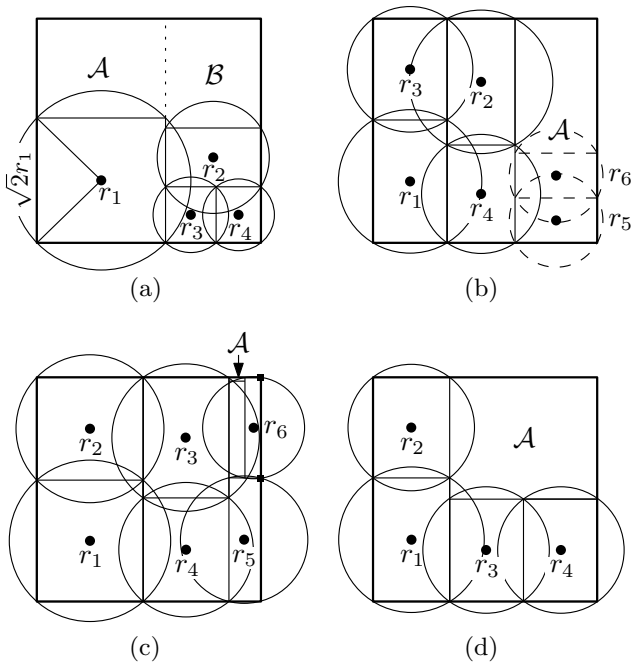


Figure 11 Covering routines that mainly rely on the four largest disks to cover $\mathcal{R}$. (a) In Routine S-VII.1, we place r_1 in the bottom-left corner, covering its inscribed square; use r_2, r_3 and r_4 to cover as much of the vertical strip remaining to the left of r_1, and recurse on $\mathcal{A}$ and $\mathcal{B}$. (b) In Routine S-VII.2, in the first case, we cover a rectangular strip using $r_1, \ldots, r_4$. Either use recursion immediately on the remainder $\mathcal{A}$, or recurse after placing r_5 and possibly r_6 covering a rectangle at the bottom of $\mathcal{A}$. (c) In Routine S-VII.2, in the second case, we cover a rectangular strip using $r_1, \ldots, r_4$ and place r_5 at the bottom of the remainder as in (b); however, we change the placement of r_6 to cover the remaining part of the right side of $\mathcal{R}$. The points that determine the position of r_6 are marked by black squares in the figure. We use recursion to cover the bounding box $\mathcal{A}$ of the area that remains uncovered. (d) In Routine S-VII.3, we cover an L-shaped region of $\mathcal{R}$ using the four largest disks, and recurse on the remaining region $\mathcal{A}$.

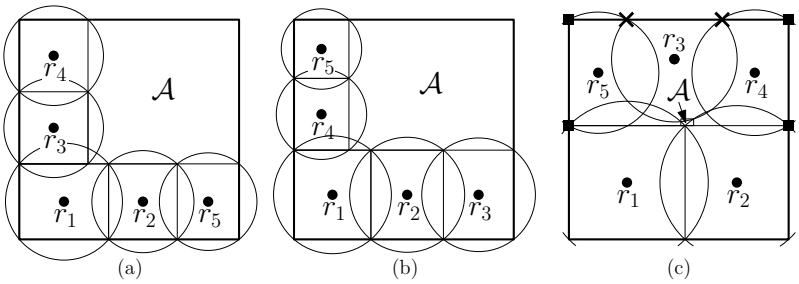


Figure 12 Routines for covering $\mathcal{R}$ using the five largest disks and recursion. According to Routine S-VIII.1, in (a) and (b), we first cover a horizontal strip of maximum height using three disks (r_1, r_2, r_3 or r_1, r_2, r_5) and then cover a vertical strip using the other two disks. (c) Routine S-VIII.2 places the five largest disks such that everything but a small region $\mathcal{A}$ is covered. The points that define the placement of r_4 and r_5 in are marked by boxes; those that define the placement of r_3 are marked $\times$. All three routines use recursion to cover $\mathcal{A}$.

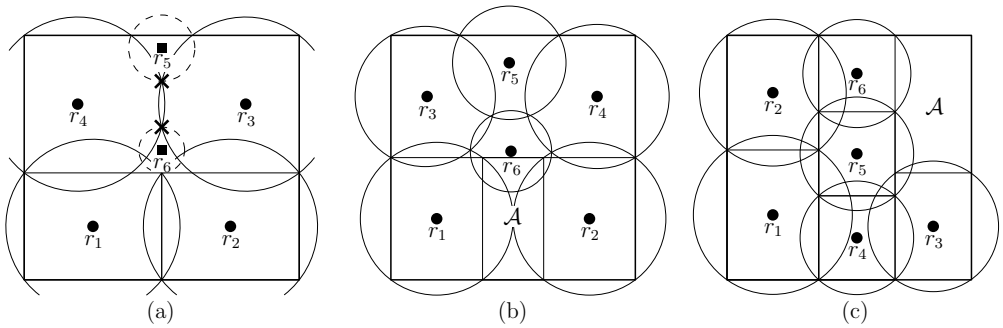


Figure 13 (a) Routine S-IX.1 covers $\mathcal{R}$ using the six largest disks. (b) In Routine S-IX.2, we also use recursion on the remaining disks to cover an additional rectangular region $\mathcal{A}$. (c) In Routine S-IX.3, we cover two vertical strips using r_1, r_2 and r_4, r_5, r_6, using r_3 and recursion to cover the remaining strip.

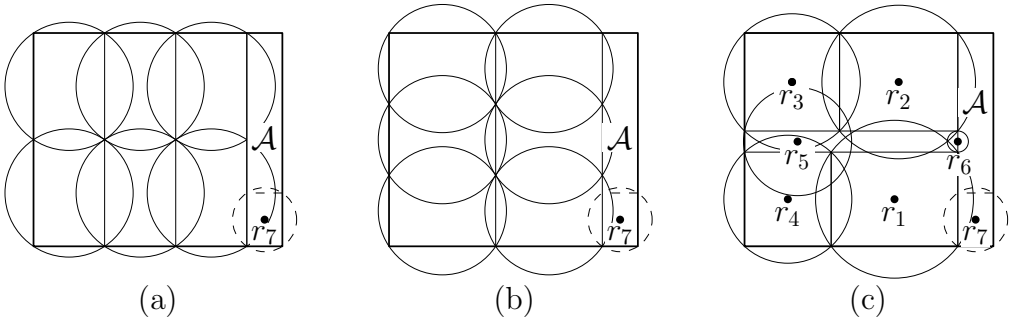


Figure 14 Routine S-X.1 considers the following three configurations to cover a strip of maximal width w. (a) Using any partition of $r_1, \ldots, r_6$ into three groups of two disks, each covering a strip of height 1 and maximal width, (b) using any partition of $r_1, \ldots, r_6$ into two groups of three disks, each covering a strip of height 1 and maximal width, or (c) using the disks r_1, r_4 and r_2, r_3 to cover strips of width w and maximal height and covering the uncovered pockets using r_5 and r_6.

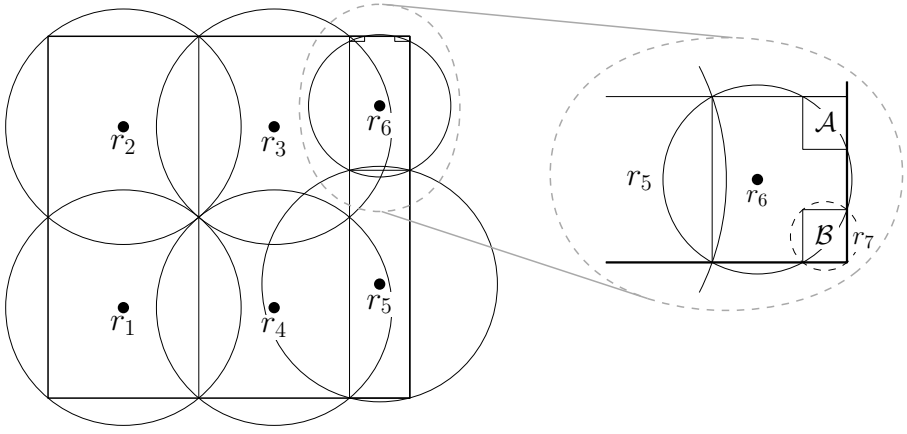


Figure 15 Routine S-X.2 covers as much width as possible using disks $r_1, \ldots, r_4$, using r_5, r_6 and r_7 on the remaining strip.

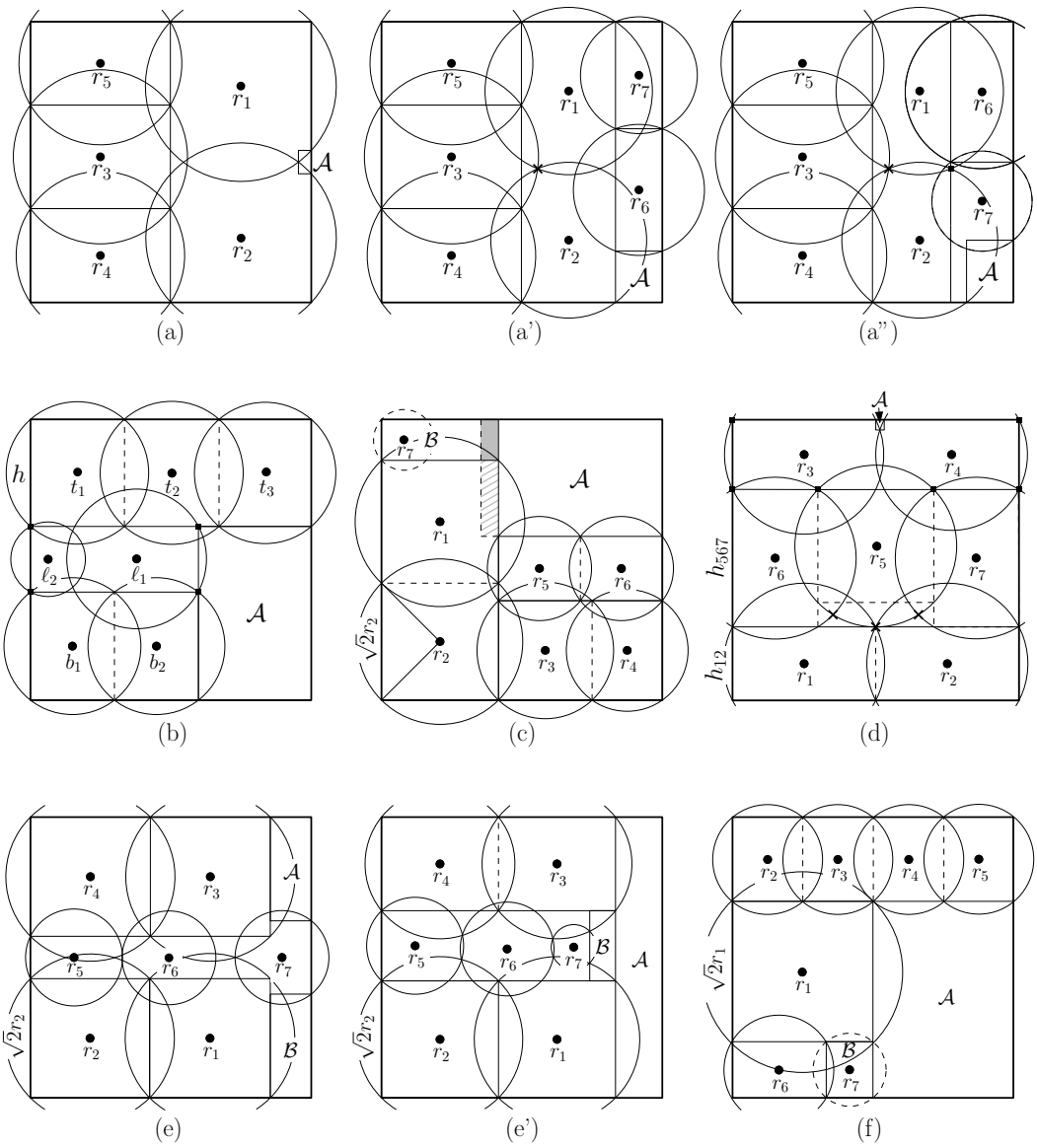


Figure 16 Routines S-X.3–S-X.8 using disks $r_1, \ldots, r_7$ and recursion to cover $\mathcal{R}$.

4.2 Proof Structure for Theorem 1

Tightness of the result claimed by Theorem 1 is proved by Lemma 2. Therefore, proving Theorem 1 means proving that, for any skew λ, any collection of disks D with $A(D) = A^*(\lambda)$ suffices to cover $\mathcal{R}$. As in the proof of Lemma 4, we begin by reducing the number of cases we have to consider to a finite number. Again, we begin by proving our result for all rectangles with sufficiently large skew.

▶ **Lemma 6.** *Let $\lambda \geq \overline{\lambda}$ and let D be a collection of disks with $W(D) = W^*(\lambda)$. We can cover $\mathcal{R}$ using the disks from D.*

Due to space restrictions, for the full proof we refer to the full version [18] of our paper. The proof is manual and uses the two simple routines Split Cover (W-I.1) and Large Disk (W-I.2); see Fig. 17. Intuitively speaking, if r_1 is small, we split D using Greedy Splitting, split $\mathcal{R}$ accordingly, and recurse on the two resulting regions. On the other hand, if r_1 is big, we cover the left side of $\mathcal{R}$ using r_1 and recurse on the remaining region.

The remainder of the proof of Theorem 1 is again based on a list of simple covering routines, which our algorithm tries to apply until it finds a working routine. We prove that there always is a working routine in the list using an automatic prover based on interval arithmetic as described in Section 3.3. After automatic analysis, several critical cases remain. We complete our proof by manually analyzing these critical cases. In the following, we give a brief description of the routines we use. Due to space constraints, for details, we refer to the full version of our paper [18].

Small disks. Because the covering coefficient guaranteed by Lemma 4 is always better than $E^*(\lambda)$, Routine (W-II.1) attempts to apply Lemma 4 directly; this works if the largest disk is not too big.

Using the largest disk. Routines (W-III.1)–(W-III.3) try several placements for the largest disk r_1; see Fig. 18.

Using the two largest disks. Routines (W-IV.1) and (W-IV.2) try several placements for the largest two disks r_1, r_2; see Fig. 19.

Using the three largest disks. Routines (W-V.1)–(W-V.5) consider several placements for the largest three disks; see Figs. 20, 21, 22, and 23.

Using the four largest disks. Routines (W-VI.1)–(W-VI.3) consider several placements for the largest four disks; see Fig. 24.

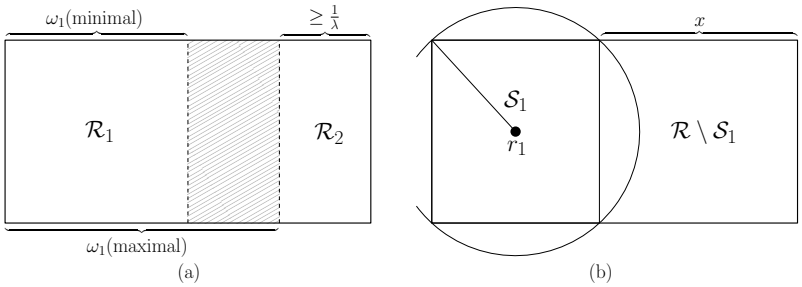


Figure 17 (a) The routine Split Cover (W-I.1) applies Greedy Splitting to the input disks, splits $\mathcal{R}$ into $\mathcal{R}_1, \mathcal{R}_2$ according to the split and recurses. The resulting split must not be too unbalanced for this routine to succeed. (b) The routine Large Disk (W-I.2) places r_1 covering a rectangle $\mathcal{S}_1$ at the right border of $\mathcal{R}$ and recurses on the remaining rectangle.

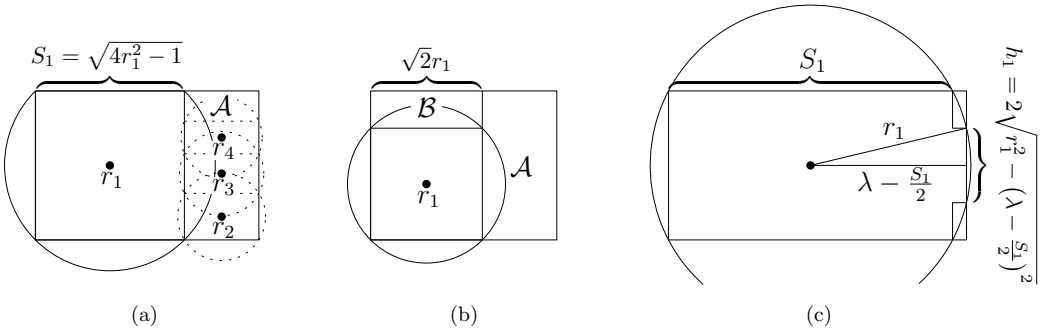


Figure 18 (a) In Routine W-III.1, we place r_1 covering a strip at the left side of $\mathcal{R}$ and try to recurse on $\mathcal{A}$. If that does not work, we also try to place r_2, r_3 and potentially r_4 covering horizontal strips at the bottom of the remaining rectangle before we try recursing. (b) In Routine W-III.2, we place r_1 covering its inscribed square at the bottom-left corner of $\mathcal{R}$, covering the two remaining regions $\mathcal{A}, \mathcal{B}$ recursively. (c) In Routine W-III.3, we place r_1 covering a strip at the left side of $\mathcal{R}$; if placed like this, r_1 intersects the right border of $\mathcal{R}$, only leaving two small uncovered pockets.

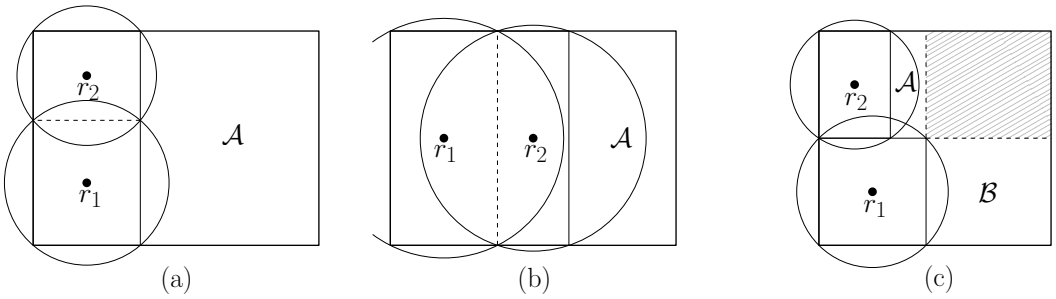


Figure 19 (a) and (b) depict Routine W-IV.1. The two largest disks are used to cover as wide a strip as possible at the left side of $\mathcal{R}$; the remaining disks are used for recursion on $\mathcal{A}$. (c) Routine W-IV.2 places r_1 covering its inscribed square and covers the remaining part of $\mathcal{R}$'s left boundary using r_2. Two regions $\mathcal{A}$ and $\mathcal{B}$ remain. The shaded area can be added to either $\mathcal{A}$ or $\mathcal{B}$; we try both options.

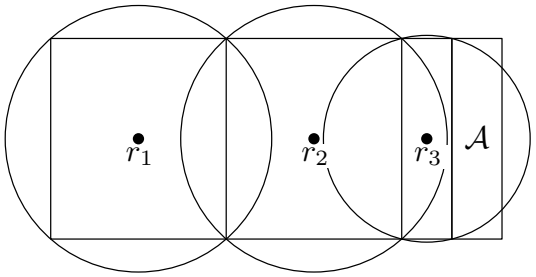


Figure 20 Routine W-V.1 places the three largest disks next to each other, each covering a vertical strip of height 1. If this does not cover the entire rectangle, we recurse on the bounding box $\mathcal{A}$ of the remaining area.

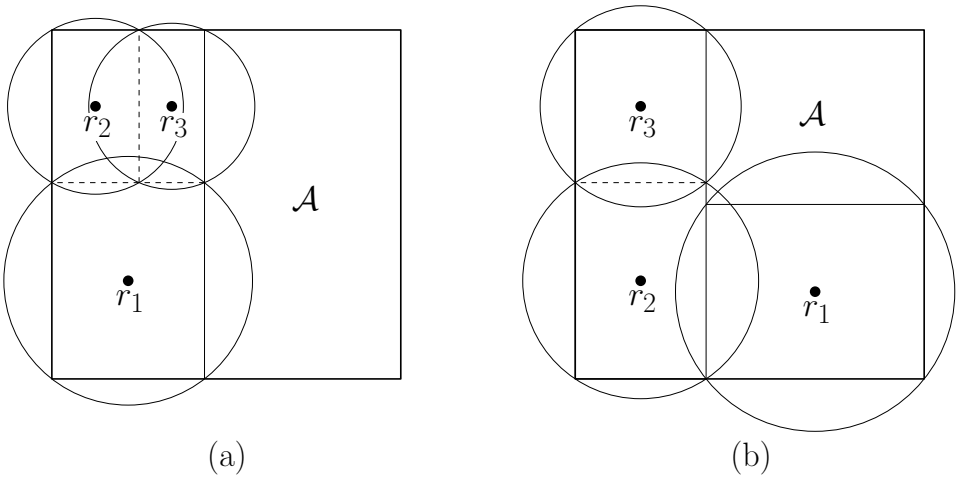


Figure 21 (a) Routine W-V.2 builds a strip of maximum possible width by placing r_1 at the bottom and r_2 besides r_3 on top. (b) Routine W-V.3 builds a vertical strip of maximum possible width by placing r_2, r_3 on top of each other, and covers the remaining part of the lower boundary using r_1.

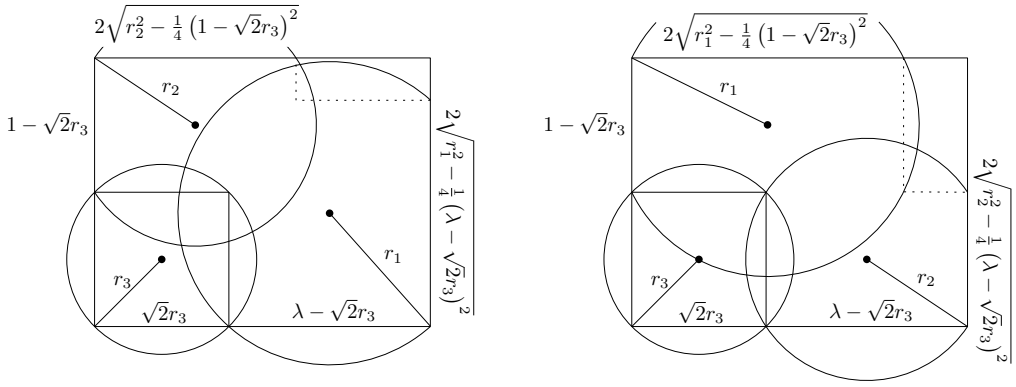


Figure 22 Routine W-V.4 covers the rectangle using the third-largest disk to cover a square at the bottom-left corner. The remaining rectangle that we recurse on is drawn with dashed outline. **Left:** Placing the largest disk to the right of the third-largest disk and the second-largest disk on top of the third-largest disk. **Right:** Placing the largest disk on top of the third-largest disk and the second-largest disk to the right of the third-largest disk.

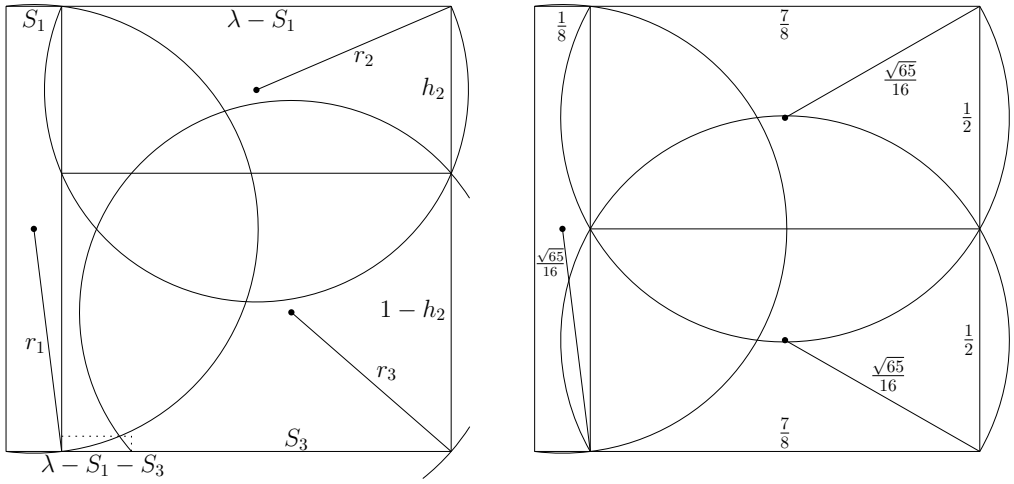


Figure 23 Left: Routine W-V.5 covers the rectangle using the largest disk to cover a strip of width S_1, using the second- and third-largest disks to cover the remaining corners. The bounding box of the uncovered pocket between the largest and third-largest disk is drawn with dashed outline. **Right:** The worst-case example for a square, consisting of three equal disks with radius $\frac{\sqrt{65}}{16}$. The covering of Routine W-V.5 converges to this covering for disks converging to this worst-case example.

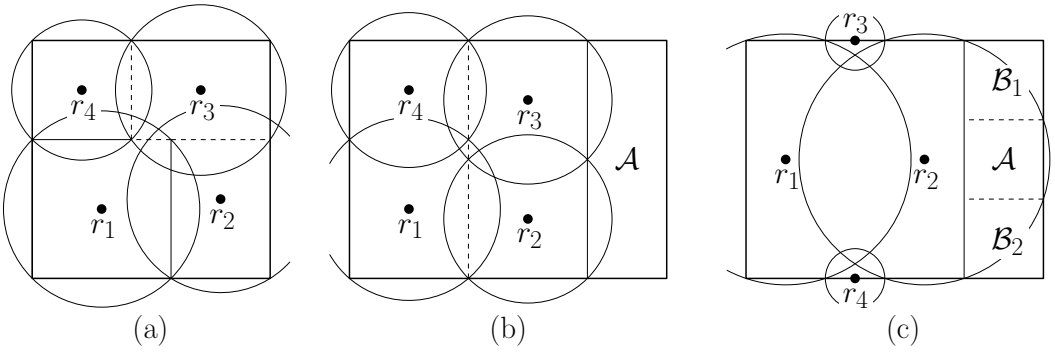


Figure 24 (a) Routine W-VI.1 covers $\mathcal{R}$ using only the four largest disks. The dashed outline depicts the rectangle that r_3 has to be able to cover. (b) Routine W-VI.2 covers a strip of maximum possible width using two groups of two disks and recurses on the remaining rectangle $\mathcal{A}$. (c) Routine W-VI.3 covers $\mathcal{R}$ by placing the two largest disks besides each other, filling the gaps between the disks using r_3, r_4. If this does not cover $\mathcal{R}$, we either recurse on the remaining strip $\mathcal{A}$ or on the bounding box of two pockets $\mathcal{B}_1, \mathcal{B}_2$ if r_2 intersects $\mathcal{R}$'s right border.

4.3 Proof of Lemma 3

In this section, we give a proof of Lemma 3.

▶ **Lemma 3.** *Let* $\hat{\sigma} \coloneqq \frac{195\sqrt{5257}}{16384} \approx 0.8629$. *Let* $\sigma \geq \hat{\sigma}$ *and* $E(\sigma) \coloneqq \frac{1}{2}\sqrt{\sqrt{\sigma^2+1}+1}$. *Let* $\lambda \geq 1$ *and* $D = \{r_1, \ldots, r_n\}$ *be any collection of disks with* $\sigma \geq r_1^2 \geq \ldots \geq r_n^2$ *and* $W(D) = \sum_{i=1}^{n} r_i^2 \geq E(\sigma)\lambda$. *Then* D *can cover a rectangle* $\mathcal{R}$ *of dimensions* $\lambda \times 1$.

Proof. In the following, let $E \coloneqq E(\sigma)$; we assume w.l.o.g. that $W(D) = E\lambda$. First, we observe that $\sigma \geq \hat{\sigma}$ implies $E \geq \frac{195}{256} = E^*(\bar{\lambda})$. Because $E^*(\lambda)$ for $\lambda \geq \bar{\lambda}$ is continuous and strictly monotonically increasing, there is a unique $\Lambda(E) \geq \bar{\lambda}$ such that $E^*(\Lambda(E)) = E$, given by $\Lambda(E) \coloneqq 2E + \sqrt{4E^2 - 2}$. Similarly, we observe that $\sigma(E) = E \cdot \left(\Lambda(E) - \frac{2}{\Lambda(E)}\right)$ is the inverse function of $E(\sigma)$. If $\lambda \leq \Lambda(E)$, we have $E \geq E^*(\lambda)$ and the result immediately follows from Theorem 1.

Otherwise, we apply Greedy Splitting to D. This yields a partition into two groups D_1, D_2; w.l.o.g., let D_1 be the heavier one. We split $\mathcal{R}$ into two rectangles $\mathcal{R}_1, \mathcal{R}_2$ such that $\frac{W(D_1)}{W(D_2)} = \frac{|\mathcal{R}_1|}{|\mathcal{R}_2|}$ by dividing the longer side (w.l.o.g., the width) of $\mathcal{R}$ in that ratio. After the split, we have $E = \frac{W(D)}{|\mathcal{R}|} = \frac{W(D_1)}{|\mathcal{R}_1|} = \frac{W(D_2)}{|\mathcal{R}_2|}$ and $|\mathcal{R}_2| = \frac{W(D_2)}{E}$.

If the resulting width of any $\mathcal{R}_i$ is greater than $\Lambda(E)$, we use D_i to inductively apply Lemma 3 to it. Otherwise, we apply Theorem 1; in order to do so, we must show that the skew of the narrower rectangle $\mathcal{R}_2$ is at most $\Lambda(E)$, which means proving that its width is at least $\frac{1}{\Lambda(E)}$. Because of $W(D_1) - W(D_2) \leq r_1^2 \leq \sigma$, we have $W(D_2) \geq \frac{W(D)-\sigma}{2} = \frac{E\lambda-\sigma(E)}{2}$. This implies that the area, and thus the width, of $\mathcal{R}_2$ is $\frac{W(D_2)}{E} \geq \frac{\Lambda(E)-\sigma(E)/E}{2} = \frac{1}{\Lambda(E)}$. ◀

5 Conclusion

We have given a tight characterization of the critical covering density for arbitrary rectangles. This gives rise to numerous followup questions and extensions.

As discussed (and shown in Fig. 3), the worst-case values correspond to instances with only 2 or 3 relatively large disks; if we have an upper bound R on the size of the largest disk, this gives rise to the critical covering area $A_R^*(\lambda)$ for $\lambda \times 1$-rectangles. Both from a theoretical and a practical point of view, getting some tight bounds on $A_R^*(\lambda)$ would be interesting and useful. Our results of Lemma 3 and Lemma 4 indicate possible progress in that direction; just like for unit disks, tighter results will require considerably more effort.

Establishing the critical covering density for disks and triangles is also open. We are optimistic that an approach similar to the one of this paper can be used for a solution.

Finally, *computing* optimal coverings by disks appears to be quite difficult. However, while deciding whether a given collection of disks can be packed into a unit square is known to be NP-hard [15], the complexity of deciding whether a given set of disks can be used to cover a unit square is still open. Ironically, it is the higher practical difficulty of covering by disks that makes it challenging to apply a similar idea in a straightforward manner.

References

1 A Karim Abu-Affash, Paz Carmi, Matthew J. Katz, and Gila Morgenstern. Multi cover of a polygon minimizing the sum of areas. *International Journal of Computational Geometry & Applications*, 21(06):685–698, 2011.

2 Alessandro Agnetis, Enrico Grande, Pitu B. Mirchandani, and Andrea Pacifici. Covering a line segment with variable radius discs. *Computers & Operations Research*, 36(5):1423–1436, 2009.

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 *Proc. 22nd Annu. ACM Sympos. Comput. Geom.*, pages 449–458, 2006. doi:10.1145/1137856.1137922.

4 Balázs Bánhelyi, Endre Palatinus, and Balázs L. Lévai. Optimal circle covering problems and their applications. *Central European Journal of Operations Research*, 23(4):815–832, 2015.

5 Aaron T. Becker, Sándor P. Fekete, Phillip Keldenich, Sebastian Morr, and Christian Scheffer. Packing Geometric Objects with Optimal Worst-Case Density (Multimedia Exposition). In *Proceedings 35th International Symposium on Computational Geometry (SoCG)*, pages 63:1–63:6, 2019. Video available at https://www.ibr.cs.tu-bs.de/users/fekete/Videos/PackingCirclesInSquares.mp4. doi:10.4230/LIPIcs.SoCG.2019.63.

6 K. Bezdek. *Körök optimális fedései (Optimal covering of circles)*. PhD thesis, Eötvös Lorand University, 1979.

7 Käroly Bezdek. Über einige optimale Konfigurationen von Kreisen. *Ann. Univ. Sci. Budapest Rolando Eötvös Sect. Math*, 27:143–151, 1984.

8 Santanu Bhowmick, Kasturi R. Varadarajan, and Shi-Ke Xue. A constant-factor approximation for multi-covering with disks. *JoCG*, 6(1):220–234, 2015. doi:10.20382/jocg.v6i1a9.

9 Károly Böröczky Jr. *Finite packing and covering*, volume 154. Cambridge University Press, 2004.

10 Peter Brass, William O.J. Moser, and János Pach. Density problems for packings and coverings. *Research Problems in Discrete Geometry*, pages 5–74, 2005.

11 Paz Carmi, Matthew J. Katz, and Nissan Lev-Tov. Covering points by unit disks of fixed location. In *Proc. International Symposium on Algorithms and Computation (ISAAC)*, pages 644–655. Springer, 2007.

12 CGAL, Computational Geometry Algorithms Library. http://www.cgal.org.

13 Gautam K. Das, Sandip Das, Subhas C. Nandy, and Bhabani P. Sinha. Efficient algorithm for placing a given number of base stations to cover a convex region. *Journal of Parallel and Distributed Computing*, 66(11):1353–1358, 2006.

14 Gautam K. Das, Sasanka Roy, Sandip Das, and Subhas C. Nandy. Variations of base-station placement problem on the boundary of a convex region. *International Journal of Foundations of Computer Science*, 19(02):405–427, 2008.

15 Erik D. Demaine, Sándor P. Fekete, and Robert J. Lang. Circle packing for origami design is hard. In *Origami^5: 5th International Conference on Origami in Science, Mathematics and Education*, AK Peters/CRC Press, pages 609–626, 2011. arXiv:1105.0791.

16 dpa. Rasensprenger zeichnet Kreise auf Fußballfeld, 2018.

17 Gábor Fejes Tóth. Recent progress on packing and covering. *Contemporary Mathematics*, 223:145–162, 1999.

18 Sándor P. Fekete, Utkarsh Gupta, Phillip Keldenich, Christian Scheffer, and Sahil Shah. Worst-case optimal covering of rectangles by disks. *arXiv:2003.08236 [cs.CG]*, 2020. arXiv:2003.08236.

19 Sándor P. Fekete, Phillip Keldenich, and Christian Scheffer. Packing Disks into Disks with Optimal Worst-Case Density. In *Proceedings 35th International Symposium on Computational Geometry (SoCG 2019)*, pages 35:1–35:19, 2019. doi:10.4230/LIPIcs.SoCG.2019.35.

20 Sándor P. Fekete, Phillip Keldenich, and Christian Scheffer. Covering rectangles by disks: The video. In *Proceedings of the 36th International Symposium on Computational Geometry (SoCG)*, 2020. To appear.

21 Sándor P. Fekete, Sebastian Morr, and Christian Scheffer. Split packing: Algorithms for packing circles with optimal worst-case density. *Discrete & Computational Geometry*, 2018. doi:10.1007/s00454-018-0020-2.

22 F. Fodor. The densest packing of 19 congruent circles in a circle. *Geometriae Dedicata*, 74:139–145, 1999.

23 F. Fodor. The densest packing of 12 congruent circles in a circle. *Beiträge zur Algebra und Geometrie (Contributions to Algebra and Geometry)*, 41:401–409, 2000.

24 F. Fodor. The densest packing of 13 congruent circles in a circle. *Beiträge zur Algebra und Geometrie (Contributions to Algebra and Geometry)*, 44:431–440, 2003.

25 E. Friedman. Circles covering squares web page, 2014. URL: `http://www2.stetson.edu/~efriedma/circovsqu/`.

26 M. Goldberg. Packing of 14, 16, 17 and 20 circles in a circle. *Mathematics Magazine*, 44:134–139, 1971.

27 R.L. Graham, B.D. Lubachevsky, K.J. Nurmela, and P.R.J. Östergøard. Dense packings of congruent circles in a circle. *Discrete Mathematics*, 181:139–154, 1998.

28 Aladár Heppes and Hans Melissen. Covering a rectangle with equal circles. *Periodica Mathematica Hungarica*, 34(1-2):65–81, 1997.

29 Chi-Fu Huang and Yu-Chee Tseng. A survey of solutions for the coverage problems in wireless sensor networks. *Journal of Internet Technology*, 6(1):1–8, 2005.

30 Matthew P. Johnson, Deniz Sariöz, Amotz Bar-Noy, Theodore Brown, Dinesh Verma, and Chai W. Wu. More is more: the benefits of denser sensor deployment. *ACM Transactions on Sensor Networks (TOSN)*, 8(3):22, 2012.

31 B.D. Lubachevsky and R.L. Graham. Curved hexagonal packings of equal disks in a circle. *Discrete & Computational Geometry*, 18:179–194, 1997.

32 H. Melissen. Densest packing of eleven congruent circles in a circle. *Geometriae Dedicata*, 50:15–25, 1994.

33 Hans Melissen. Loosest circle coverings of an equilateral triangle. *Mathematics Magazine*, 70(2):118–124, 1997.

34 Johannes Bernardus Marinus Melissen and Peter Cornelis Schuur. Covering a rectangle with six and seven circles. *Discrete Applied Mathematics*, 99(1-3):149–156, 2000.

35 John W. Moon and Leo Moser. Some packing and covering theorems. In *Colloquium Mathematicae*, volume 17, pages 103–110. Institute of Mathematics, Polish Academy of Sciences, 1967.

36 Sebastian Morr. Split packing: An algorithm for packing circles with optimal worst-case density. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 99–109, 2017.

37 Eric H. Neville. On the solution of numerical functional equations. *Proceedings of the London Mathematical Society*, 2(1):308–326, 1915.

38 Kari J. Nurmela. Conjecturally optimal coverings of an equilateral triangle with up to 36 equal circles. *Experimental Mathematics*, 9(2):241–250, 2000.

39 Norman Oler. A finite packing problem. *Canadian Mathematical Bulletin*, 4:153–155, 1961.

40 Endre Palatinus and Balázs Bánhelyi. Circle covering and its applications for telecommunication networks. In *8 th International Conference on Applied Informatics*, page 255, 2010.

41 G.E. Reis. Dense packing of equal circles within a circle. *Mathematics Magazine*, issue 48:33–37, 1975.

42 Williamjeet Singh and Jyotsna Sengupta. An efficient algorithm for optimizing base station site selection to cover a convex square region in cell planning. *Wireless personal communications*, 72(2):823–841, 2013.

43 Eckard Specht. Packomania, 2015. URL: `http://www.packomania.com/`.

44 Balázs Szalkai. Optimal cover of a disk with three smaller congruent disks. *Advances in Geometry*, 16(4):465–476, 2016.

45 Gábor Fejes Tóth. Thinnest covering of a circle by eight, nine, or ten congruent circles. *Combinatorial and computational geometry*, 52(361):59, 2005.

46 Gábor Fejes Tóth. Packing and covering. In *Handbook of Discrete and Computational Geometry, Third Edition*, pages 27–66. Chapman and Hall/CRC, 2017.

47 Xiaochun Xu, Sartaj Sahni, and Nageswara S.V. Rao. Minimum-cost sensor coverage of planar regions. In *FUSION*, pages 1–8, 2008.

Minimum Scan Cover with Angular Transition Costs

Sándor P. Fekete
Department of Computer Science, TU Braunschweig, Germany
s.fekete@tu-bs.de

Linda Kleist
Department of Computer Science, TU Braunschweig, Germany
l.kleist@tu-bs.de

Dominik Krupke
Department of Computer Science, TU Braunschweig, Germany
d.krupke@tu-bs.de

Abstract

We provide a comprehensive study of a natural geometric optimization problem motivated by questions in the context of satellite communication and astrophysics. In the problem Minimum Scan Cover with Angular Costs (msc), we are given a graph G that is embedded in Euclidean space. The edges of G need to be *scanned*, i.e., probed from both of their vertices. In order to scan their edge, two vertices need to face each other; changing the heading of a vertex takes some time proportional to the corresponding turn angle. Our goal is to minimize the time until all scans are completed, i.e., to compute a schedule of minimum makespan.

We show that msc is closely related to both graph coloring and the minimum (directed and undirected) cut cover problem; in particular, we show that the minimum scan time for instances in 1D and 2D lies in $\Theta(\log \chi(G))$, while for 3D the minimum scan time is not upper bounded by $\chi(G)$. We use this relationship to prove that the existence of a constant-factor approximation implies $P = NP$, even for one-dimensional instances. In 2D, we show that it is NP-hard to approximate a minimum scan cover within less than a factor of $3/2$, even for bipartite graphs; conversely, we present a $9/2$-approximation algorithm for this scenario. Generally, we give an $O(c)$-approximation for k-colored graphs with $k \leq \chi(G)^c$. For general metric cost functions, we provide approximation algorithms whose performance guarantee depend on the arboricity of the graph.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms; Theory of computation → Computational geometry

Keywords and phrases Graph scanning, graph coloring, angular metric, complexity, approximation, scheduling

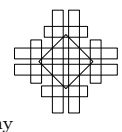
Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.43

Related Version A full version of the paper is available at `https://arxiv.org/abs/2003.08816`, [15].

Acknowledgements We thank Phillip Keldenich, Irina Kostitsyna, Christian Rieck, and Arne Schmidt for helpful algorithmic discourse, Kenny Cheung (NASA) and Christian Schurig (European Space Agency) for joint work on intersatellite communication, and Karl-Heinz Glaßmeier for discussions of astrophysical aspects.

36th International Symposium on Computational Geometry (SoCG 2020).
Editors: Sergio Cabello and Danny Z. Chen; Article No. 43; pp. 43:1–43:18

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



1 Introduction

Many problems of geometric optimization arise from questions of communication, where different locations need to be connected. For physical networks, the cost of a connection corresponds to the geometric distance between the involved vertices, e.g., the length of an electro-optic link. Often wireless transmissions may be used instead; however, for ultra-long distances such as in space, this requires focused transmission, e.g., by communication partners facing each other with directional, paraboloid antennas or laser beams. This makes it impossible to exchange information with multiple partners at once; moreover, a change of communication partner requires a change of heading, which is costly in the context of space missions with limited resources, making it worthwhile to invest in a good schedule.

With the advent of satellite swarms of ever-growing size, problems of this type are of increasingly practical importance for ensuring communication between spacecraft. They also come into play when astro- and geophysical measurements are to be performed, in which groups of spacecraft can determine physical quantities not just at their current locations, but also along their common line of sight.

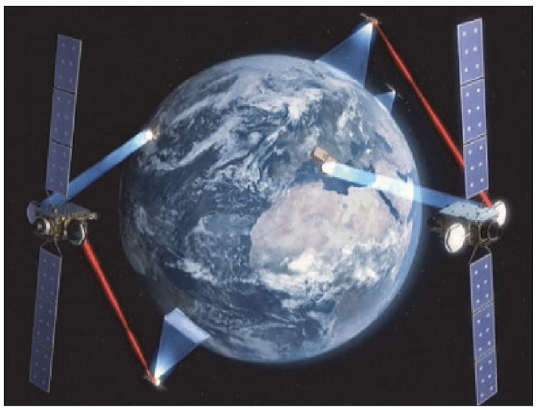


Figure 1 Artist's rendition of the European Data Relay Satellite constellation architecture. Note the intersatellite links shown in red. (Image credit: ESA).

We consider an optimization problem arising from this context: How can we schedule a given set of intersatellite communications, such that the overall timetable is as efficient as possible? In particular, we study the question of a Minimum Scan Cover with Angular Costs (msc), in which we need to establish a collection of connections between a given set of locations, described by a graph $G = (V, E)$ that is embedded in space. For any connection (or *scan*) of an edge, the two involved vertices need to face each other; changing the heading of a vertex to cover a different connection takes an amount of time proportional to the corresponding turn angle. Our goal is to minimize the time until all tasks are completed, i.e., compute a geometric schedule of minimum makespan.

In this paper, we provide a comprehensive study of this problem. We show that msc is closely related to both graph coloring and the minimum (directed and undirected) cut cover problem. We also provide a number of hardness results and approximation algorithms for a variety of geometric scenarios; see Section 1.2 for an overview.

1.1 Problem definition: Minimum Scan Cover

In the *abstract version* of Minimum Scan Cover, denoted by A-MSC, we are given a simple graph $G = (V, E)$ and a metric cost function $\alpha : \{(e, e') \in E \times E \mid e \cap e' \neq \emptyset\} \to \mathbb{R}^+$ that describes the cost for switching between two incident edges uv, vw for the common vertex v. A *scan cover* is an assignment $S : E \to \mathbb{R}^+$, such that for every vertex v and every pair of incident edges uv and vw it holds that

$$|S(uv) - S(vw)| \geq \alpha(uv, vw).$$

This condition provides sufficient time for the vertices to face each other. We seek a scan cover that minimizes the *scan time* $\max_{e \in E} S(e)$. Note that A-MSC generalizes the Path-TSP (see Observation 8), so the problem becomes intractable if the cost function α is not metric.

Given the practical motivation, our main focus is the (geometric) Minimum Scan Cover Problem (MSC), for which every vertex v corresponds to a point in $\mathbb{R}^d$, $d \in \{1, 2, 3\}$, the turn cost $\alpha(uv, vw)$ of v from uv and vw is the (smaller) angle at v between the segments uv and vw. Figure 2 illustrates a minimum scan cover for a point set in the plane that can be scanned in 300° with eleven discrete time steps.

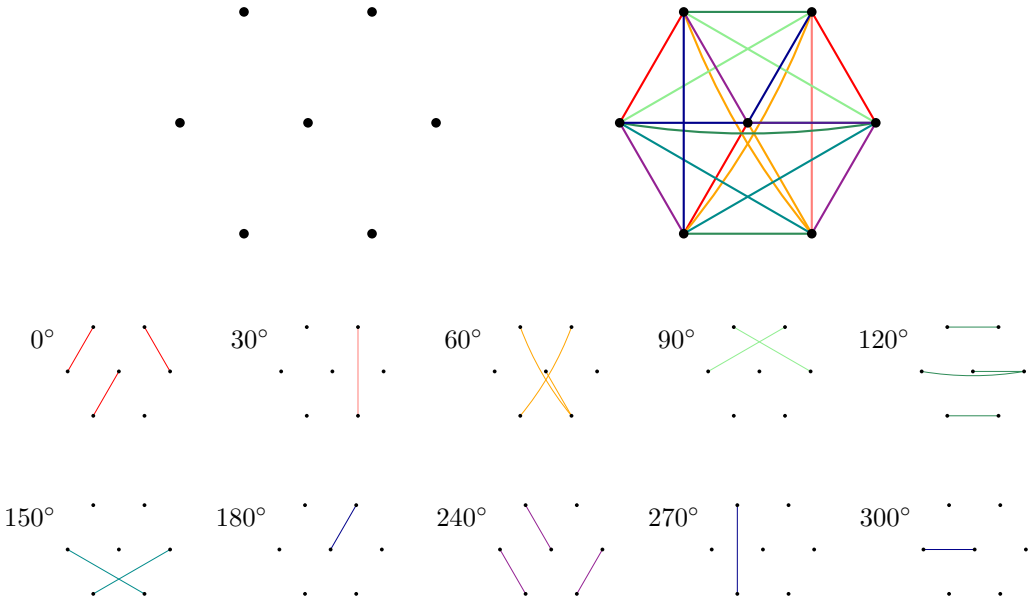


Figure 2 (Left) A set of seven points in $\mathbb{R}^2$, for which the complete graph K_7 needs to be scanned. **(Bottom)** A sequence of edge scans; note how some edges can be scanned in parallel.

In fact, a scan cover is completely determined by an edge order: For each edge sequence $e_1, \ldots, e_m$, the best scan cover that scans the edges in this order can be computed by

$$S(e_1) = 0 \text{ and } S(e_i) = \max\{S(e_j) + \alpha(e_i, e_j) | j : j < i, e_i \cap e_j \neq \emptyset)\} \text{ for } i > 1.$$

In some settings, we may be given an initial heading of each vertex. However, the cost of changing from the initial heading to any other is usually negligible compared to the cost of the remaining schedule. In fact, any C-approximation without initial headings yields a $(C + 1)$-approximation for the variant with initial headings, as turning from the initial heading to any edge via a smallest angle is not more expensive than the minimum makespan.

1.2 Overview of results and organization

In Section 2, we show that the MSC in 1D corresponds to a minimum directed cut cover and has a strong correlation to the chromatic number. We provide an improved upper bound of $\lceil \log_2 \chi(G) + \frac{1}{2} \log_2 \log_2 \chi(G) + 1 \rceil$ (Theorem 1 and Corollary 2) for the minimum directed cut cover number, which is essentially tight in general; even for directed acyclic graphs corresponding to minimum scan covers (Lemma 1) this is in the right order of magnitude. This implies that, unless $P = NP$, there exists no constant-factor approximation even in 1D (Theorem 3). Nevertheless, we show that instances in which the underlying graphs are bipartite or complete graphs can be solved in polynomial time (Observations 3 and 4).

In Section 3, we consider the problem in 2D and show that it is NP-hard to approximate minimum scan covers of bipartite graphs better than $3/2$ (Theorem 4). Furthermore, we provide absolute and relative bounds. On the one hand, every bipartite graph in 2D has a scan cover of length 360° (Theorem 5). On the other hand, we present a $9/2$-approximation algorithm (Theorem 6). More generally, we present an $O(c)$-approximation for a k-colored graph with $k \leq \chi(G)^c$ (Theorem 7). This has immediate consequences for several interesting graph classes, e.g., the scan time of graphs in 1D and 2D lies in $\Theta(\log_2 \chi(G))$ and there exist constant factor approximations (Corollaries 8 and 9).

In Section 4, we consider MSC in 3D and the abstract version A-MSC. We show that in contrast to 2D, the length of a minimum scan cover in 3D may exceed $O(\log_2 n)$ (Observation 11). Complementary to the fact that A-MSC for stars is equivalent to path-TSP and thus NP-hard, we provide a 2.5-approximation of A-MSC for trees (Theorem 10). This yields an $O(A)$-approximation for every graph with arboricity A (Theorem 11).

1.3 Related work

The use of directional antennas has introduced a number of geometric questions. Carmi et al. [9] study the α-MST problem, which arises from finding orientations of directional antennas with α-cones, such that the connectivity graph yields a spanning tree of minimum weight, based on bidirectional communication. They prove that for $\alpha < \pi/3$, a solution may not exist, while $\alpha \geq \pi/3$ always suffices. See Aschner and Katz [7] for more recent hardness proofs and constant-factor approximations for some specific values of α.

Many other geometric optimization problems deal with turn cost. Arkin et al. [5, 6] show hardness of finding an optimal milling tour with turn cost, even in relatively constrained settings, and give a 2.5-approximation algorithm for obtaining a cycle cover, yielding a 3.75-approximation algorithm for tours. The complexity of finding an optimal cycle cover in a 2-dimensional grid graph was stated as *Problem 53* in *The Open Problems Project* [11] and shown to be NP-complete in [12], which also provides constant-factor approximations; practical methods and results are given in [13], and visualized in the video [8].

Finding a fastest roundtrip for a set of points in the plane for which the travel time depends only on the turn cost is called the *Angular Metric Traveling Salesman Problem*. Aggarwal et al. [1] prove hardness and provide an $O(\log n)$ approximation algorithm for cycle covers and tours that works even for distance costs and higher dimensions. For the abstract version on graphs in which "turns" correspond to weighted changes between edges, Fellows et al. [16] show that the problem is fixed-parameter tractable in the number of turns, the treewidth, and the maximum degree. Fekete and Woeginger [14] consider the problem of connecting a set of points by a tour in which the angles of successive edges are constrained.

Our paper also draws connections to other graph optimization problems. In particular, for each point in time, the set of scanned edges induces a bipartite graph. Therefore, one approach for scanning all edges of the given graph is to partition it into a small number of bipartite graphs, each corresponding to the set of edges separated by the *cut* induced by a partition of vertices into two non-trivial sets. This problem is also known as the Minimum Cut Cover Problem: Find the minimum number of cuts to cover all edges of a graph. Loulou [22] shows that for complete graphs, an optimal solution consists of $\lceil \log_2 |V| \rceil$ cuts. Motwani and Naor [23] prove that, unless $P = NP$, the problem on general graphs is not approximable within 1.5 of the optimum, or $OPT + \varepsilon \log |V|$ for some $\varepsilon > 0$ in absolute terms, due to a direct relationship with graph coloring. Hoshino [18] considers practical methods based on integer programming and heuristics for cut covers. Chuzhoy and Khanna [10] show that the directed version of covering a directed graph by the minimum number of directed cuts is also an NP-hard problem.

On the application side, Korth et al. [20] describe the use of tomography (i.e., determining physical phenomena by measuring aggregated effects along a ray between two sensors) in the context of astrophysics. Using multiple sensors (e.g., satellites) for performing efficient measurements is one of the motivations for the algorithmic work in this paper. Scheduling satellite communication has received a growing amount of attention, corresponding to the increasing size of satellite swarms. See Krupke et al. [21] for a recent overview.

In the context of scheduling, Allahverdi et al. [2, 3, 4] provide a nice and comprehensive survey on scheduling variants with sequence-dependent setup costs. Sotskov et al. [25] consider a scheduling variant that can directly be expressed as vertex coloring.

2 One-dimensional point sets

In the one-dimensional case, all vertices lie on a single line L. Therefore, an instance can be described by a graph $G = (V, E)$ and a total order of the vertices $<_L$ on L. We assume this line to be horizontal, so vertices face either left or right when scanning an edge. Moreover, scan times can be restricted to discrete multiples of 180°. This allows us to encode the headings of a vertex v at these time steps by a 0-1-vector $s(v)$, where a right heading is denoted by 0, and a left one by 1; we denote by $s_i(v)$ the ith bit of $s(v)$. Then a scan cover with N steps of $(G, <_L)$ is an assignment $s\colon V \to \{0,1\}^N$, such that for every edge $uv \in E$, $u <_L v$, there exists an index $i \in [N]$ with $s_i(u) = 0$ and $s_i(v) = 1$. The value of such a scan cover is clearly $180°(N-1)$. For an example, consider Figure 3.

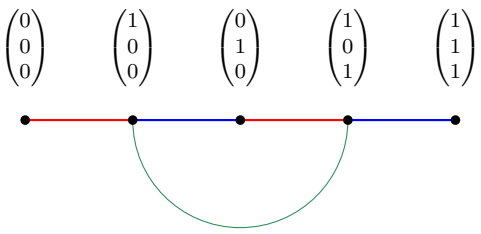


Figure 3 This instance can be scanned in three steps. However, two steps are not sufficient because the edges of a monotone path would need to be scanned in alternating time steps; making it impossible to scan the green edge.

2.1 Bounds based on chromatic number and cut cover number

In the following, we establish a strong relationship between the length of a MSC in 1D and the chromatic number $\chi(G)$, which is closely linked to the cut cover number $c(G)$ of the involved graph $G = (V, E)$, i.e., the size of a smallest partition of the edge set into bipartite graphs. Motwani and Naor [23] show that

$$c(G) = \lceil \log_2 \chi(G) \rceil.$$

Because the scanned edges in each time step form a bipartite graph, a scan cover induces a cut cover. However, the resulting bipartite graphs have the additional property that for each vertex all neighbors are either smaller or larger with respect to $<_L$. Thus, not every cut cover corresponds to a scan cover. However, scan covers correspond to *directed* cut covers of the directed graph, induced by orienting the edges from left to right. Watanabe et al. [26] bound the directed cut cover number $\vec{c}(G)$ of a directed graph G:

$$\vec{c}(G) \leq \lceil \log_2 \chi(G) \rceil + \lceil \log_2 \lceil \log_2 \chi(G) + 1 \rceil \rceil$$

We improve this bound by showing an upper bound for the size of a smallest scan cover in terms of the chromatic number (and the cut cover number); this bound is best possible for the directed cut cover number as we explain later.

▶ **Theorem 1.** *For every graph G with $\chi(G) \geq 2$ and every ordering $<_L$ of the vertices, there exists a scan cover of $(G, <_L)$ with N steps such that*

$$N \leq \lceil \log_2 \chi(G) + \frac{1}{2} \log_2 \log_2 \chi(G) + 1 \rceil$$

Proof. Consider a coloring of G with $C := \chi(G)$ colors and choose an N large enough such that $C \leq \binom{N}{\lfloor N/2 \rfloor}$. For $k := \lfloor \frac{N}{2} \rfloor$, we consider the set of vectors $\{0,1\}^N_k$ of length N with exactly k many 1's. We define a scan cover $s : V \to \{0,1\}^N_k$, such that for all vertices of the same color, we assign the same vector, while vertices of different color obtain different vectors. Such an assignment exists, because the number of vectors, i.e., $\binom{N}{\lfloor N/2 \rfloor}$, is at least as large as the number of colors.

To see that s is a scan cover, consider a fixed but arbitrary edge uv of G. Because the vectors $s(u)$ and $s(v)$ differ but have the same number of 1's, they are *incomparable*, i.e., there exist i and j such that $s_i(u) = 0$, $s_i(v) = 1$ and $s_j(u) = 1, s_j(v) = 0$. Therefore, depending on the ordering of u and v on L, the edge uv is either scanned in step i or j.

It remains to show that defining $N := \lceil \log_2 C + \frac{1}{2} \log_2 \log_2 C + 1 \rceil$ satisfies $C \leq \binom{N}{\lfloor N/2 \rfloor}$. By a variant of Stirling's formula [24], it holds that

$$e^{1/(12n+1)} \leq \frac{n!}{\sqrt{2\pi n}(n/e)^n} \leq e^{1/(12n)}.$$

This implies that $\binom{N}{\lfloor N/2 \rfloor} \geq \sqrt{\frac{2}{\pi N}} \cdot 2^N \cdot e^{\frac{-1}{4N-1}}$, so it suffices to guarantee

$$C \leq \sqrt{\frac{2}{\pi N}} \cdot 2^N \cdot e^{\frac{-1}{4N-1}} \iff \log_2 C \leq N + \frac{1}{2}(1 - \log_2 \pi - \log_2 N) - \frac{1}{4N-1} \log_2 e.$$

If $C \geq 3$, this holds for $N = \lceil \log_2 C + \frac{1}{2} \log_2 \log_2 C + 1 \rceil \geq 3$; in case of $C = 2$, it holds that $N = \lceil \log_2 C + \frac{1}{2} \log_2 \log_2 C + 1 \rceil = 2$, and thus $C \leq \binom{N}{\lfloor N/2 \rfloor}$. ◀

Note that the assigned vectors in the proof of Theorem 1 are pairwise incomparable. Therefore, such an assignment yields a directed cut cover for all edge directions and thus a general bound on the directed cut cover number.

▶ **Corollary 2.** *For every directed graph G, the directed cut cover number is bounded by*

$$\vec{c}(G) \leq \lceil \log_2 \chi(G) + \frac{1}{2} \log_2 \log_2 \chi(G) + 1 \rceil.$$

In fact, the bound in Corollary 2 is best possible for general directed graphs, because a cut cover of the complete bidirected graph corresponds to an assignment of pairwise incomparable vectors (and Sperner's theorem asserts that the used set of vectors is maximal).

Figure 3 illustrates an example of a graph G and an ordering $<_L$ showing that the bound of Theorem 1 and Corollary 2 is also tight for some (directed acyclic) graphs with $\chi(G) = 3$. In the following, we show a general lower bound for our more special setting.

▶ **Lemma 1.** *For every C, there exists a graph G and an ordering $<_L$ such that $\chi(G) > C$ and the number N of steps in every scan cover of $(G, <_L)$ is at least*

$$N \geq \lceil \log_2 \chi(G) + \frac{1}{4} \log_2 \log_2 \chi(G) \rceil.$$

Proof. Let $\ell \geq 4$ be an integer divisible by 4 and $n := 2^\ell$ such that $2^n > C$. We consider the Turan graph G on $n2^n$ vertices partitioned into 2^n independent sets of size n. Because G is a complete 2^n-partite graph, it holds that $\chi(G) = 2^n$. We place the vertices on the line, such that for a fixed $\{1, \ldots, 2^n\}$-coloring of G, there exist n disjoint intervals in which the colors appear in the order $1, \ldots, 2^n$. For an illustration consider Figure 4.

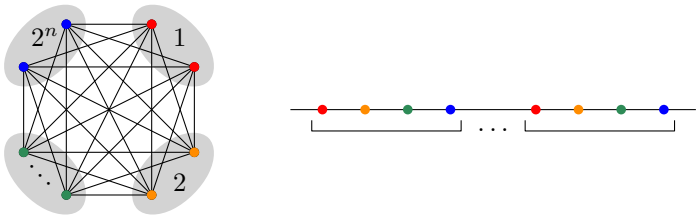


Figure 4 Illustration of G and the ordering $<_L$ of the vertices on L for $n = 2$ ($\ell = 1$).

For a contradiction, suppose that there exists a scan cover $s : V \to \{0,1\}^k$ of $(G, <_L)$ with $k := \lceil \log_2 \chi(G) + \frac{1}{4} \log_2 \log_2 \chi(G) \rceil - 1 = n + \frac{\ell}{4} - 1$ steps. Thus, the number of different vectors is $2^k = 2^{n-1} n^{1/4}$.

Let t denote the number of different color classes in which some vector is used at least $n^{3/4}$ times. We show that $t \geq \frac{1}{2} 2^n$. Clearly, each vector may only appear in one color class, i.e., the color classes induce a partition of the set of vectors. Consider the $2^n - t$ color classes (and their assigned vectors) in which no vector is used $n^{3/4}$ times. Let δ denote the average usage of vectors in these classes. Note that δ is lower bounded by the ratio of the number of vertices, namely $(2^n - t)n$, and the maximum number of remaining vectors, namely $2^k - t$. Consequently, $\delta \geq \frac{n2^n - tn}{2^k - t}$. Moreover, $\delta < n^{3/4}$, because otherwise there exists a further color class for which some vector appears at least $n^{3/4}$ times. Therefore, we obtain the following chain of implications:

$$\delta < n^{3/4} \implies \frac{n2^n - tn}{2^k - t} < n^{3/4} \iff t > 2^n \cdot \frac{1}{2(1 - n^{-1/4})} \implies t > \frac{1}{2} 2^n$$

For each of these t color classes, we choose a vector with a maximal number of appearances and introduce an interval on L from the first to the last occurrence. By the ordering of the vertices, every two vertices of the same color have a distance of at least 2^n, and hence the interval spans at least $d = 2^n n^{3/4}$ vertices. On average, every vertex is contained in the following number of intervals

$$\frac{t \cdot d}{|V|} \geq \frac{\frac{1}{2}2^n \cdot 2^n n^{3/4}}{n2^n} = \frac{2^n}{2n^{1/4}} = 2^{n-1}n^{-1/4}.$$

By the pigeonhole principle, there exists a set S of at least $2^{n-1}n^{-1/4}$ vectors with mutually intersecting intervals. We claim that any two vectors a and b of S are pairwise *incomparable*, i.e., there exist two indices i, j such that $a_i = 0, b_i = 1$ and $a_j = 1, b_j = 0$: Because the intervals intersect, among the four occurrences of a and b on $<_L$, there exist three such that they appear alternating. To scan the corresponding edges, the vectors must be incomparable. Thus, there must exist $2^{n-1}n^{-1/4}$ pairwise incomparable vectors.

However, by Sperner's theorem, every set of vectors of length k contains at most $\binom{k}{\lfloor k/2 \rfloor}$ pairwise incomparable vectors and

$$\binom{k}{\lfloor k/2 \rfloor} \leq \sqrt{\frac{2}{k\pi}}2^k(1 + \frac{1}{11}) \leq 2^k \frac{1}{\sqrt{k}}.$$

It remains to show that the number of necessary incomparable vectors exceeds this:

$$2^k \cdot \frac{1}{\sqrt{k}} < \frac{2^n}{2n^{1/4}} \iff n < k$$

This holds for $\ell > 4$ and yields a contradiction. For $\ell = 4$ it holds that $k = n$. Thus, each color class has a unique vector, all of which need to be incomparable, a contradiction. ◀

2.2 No constant-factor approximation in 1D

Theorem 1 implies the following.

▶ **Lemma 2.** *A C-approximation algorithm for MSC implies a polynomial-time algorithm for computing a coloring of graph G, $k := \chi(G)$, with $4^C \cdot k^C \cdot \sqrt{\log_2(k)}^C$ colors.*

Proof. Let ℓ^* denote the length of a minimum scan cover of G. Then a C-approximation algorithm computes a scan cover of length $\ell \leq C \cdot \ell^*$. Theorem 1 implies that $C \cdot \ell^* \leq C \cdot \lceil \log_2 k + \frac{1}{2} \log_2 \log_2 k + 1 \rceil$, yielding a coloring with 2^ℓ colors. Thus,

$$2^\ell \leq 2^{C(\lceil \log_2 k + \frac{1}{2} \log_2 \log_2 k + 1 \rceil)} \leq 2^{C \cdot \log_2 k} \cdot 2^{\frac{1}{2} \cdot C \cdot \log_2 \log_2 k} \cdot 2^{2C} \leq 4^C \cdot k^C \cdot \sqrt{\log_2(k)}^C. \quad ◀$$

▶ **Theorem 3.** *Even in 1D, a C-approximation for MSC for any $C \geq 1$ implies $P = NP$.*

Proof. Suppose there is a C-approximation for some constant $C > 1$. By Lemma 2, a C-approximation of MSC in 1D implies that there is a polynomial-time algorithm for finding for every k-colorable graph G a coloring with $4^C \cdot k^C \cdot \sqrt{\log_2(k)}^C$ colors. Khot [19] showed that, for sufficiently large k, it is NP-hard to color a k-colorable graph with at most $k^{\log_2(k)/25}$ colors. However, for every C we can find a k such that $4^C \cdot k^C \cdot \sqrt{\log_2(k)}^C < k^{\log_2(k)/25}$. This yields a polynomial-time algorithm for an NP-hard problem, implying that $P = NP$. ◀

2.3 Polynomially solvable cases

Even though there is no constant-factor approximation in general, we would like to note that bipartite and complete graphs in 1D can be solved in polynomial time.

▶ **Observation 3.** *For instances of* MSC *in 1D for which the underlying graph G is bipartite, there exists a polynomial-time algorithm for computing an optimal scan cover.*

Proof. We assume that $\chi(G) = 2$, otherwise there is no edge to scan. If for every vertex, all its neighbors lie either before or after it, G can be scanned within one step, which is clearly optimal. Otherwise, every scan cover needs at least two steps. By Theorem 1, there exists a scan cover with 2 steps. Because bipartite graphs can be colored in polynomial time, the proof of Theorem 1 provides a scan cover. ◀

▶ **Observation 4.** *For instances of* MSC *in 1D for which the underlying graph G is a complete graph, there exists a polynomial-time algorithm for computing an optimal scan cover.*

Proof. Because every scan cover induces a cut cover and $c(G) = \lceil \log_2 n \rceil$, it suffices to provide a scan cover with this number of steps. To this end, we recursively scan the bipartite graphs induced by two vertex sets when split into halves with respect to $<_L$. ◀

3 Two-dimensional point sets

For two-dimensional point sets, we show that even for bipartite graphs, it is hard to approximate MSC better than $3/2$. Conversely, we present a $9/2$-approximation algorithm for these graphs and apply the gained insights to achieve approximations for k-colorable graphs.

3.1 Bipartite graphs

By Theorem 3, we cannot hope for a constant-factor approximation for general graphs. However, bipartite graphs in 1D can be solved in polynomial time. We show that the added geometry of 2D makes the MSC hard to approximate even for bipartite graphs.

3.1.1 No approximation better than 1.5 for bipartite graphs in 2D

As a stepping stone for the geometric case, we establish the following.

▶ **Lemma 5.** *It is NP-hard to approximate* A-MSC *better than $3/2$ even for bipartite graphs.*

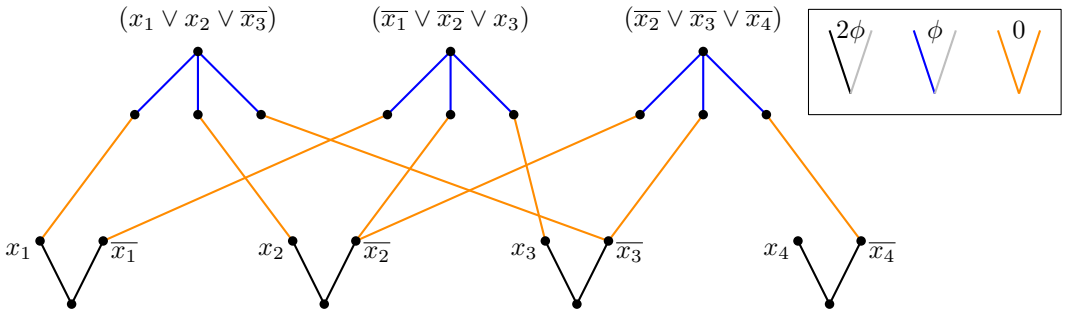


Figure 5 Illustration of the graph G_I for the instance $I = (x_1 \vee x_2 \vee \overline{x_3}) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_3})$.

Proof sketch. The proof is based on a reduction from the problem Not-All-Equal-3-Sat, for which a satisfying assignment fulfills the property that each clause has a `true` and a `false` literal. For every instance I of Not-All-Equal-3-Sat, we construct a graph G_I and a cost function α, such that there exists a scan cover with a scan time of 2ϕ only if I is a satisfiable instance. Otherwise, every scan cover has at least a value of 3ϕ.

The graph G_I, illustrated in Figure 5, is a special variant of a clause-variable-incidence graph and has (blue) *clause*, (black) *variable*, and (orange) *incidence* edges. The transition cost for any edge pair is ϕ if it contains a clause edge, 2ϕ if it contains a variable edge, and 0 otherwise. An example is depicted in Figure 6. For full details, see the long version [15]. ◀

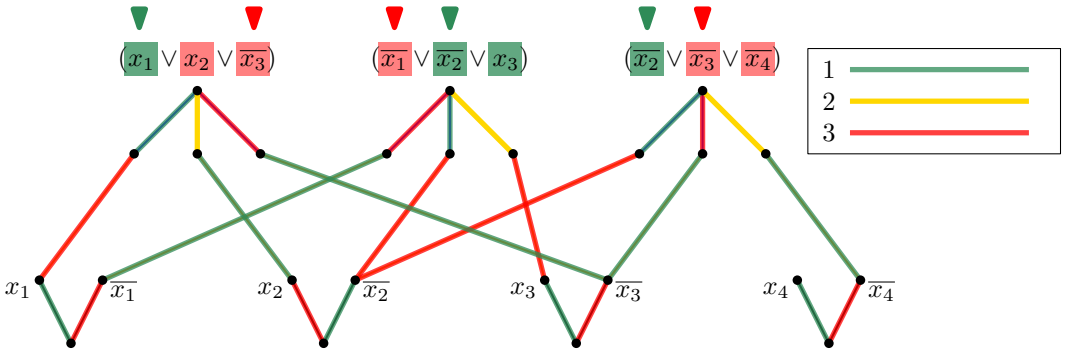


Figure 6 Illustration of a scan cover of the graph G_I. Green edges are scanned in the first, yellow in the second, and red edges in the third step.

We now use Lemma 5 for showing hardness of bipartite graphs in the geometric version.

▶ **Theorem 4.** *Even for bipartite graphs in 2D, a C-approximation for* MSC *for any $C < 3/2$ implies $P = NP$.*

Proof sketch. Suppose that there is a $(3/2 - \varepsilon)$-approximation for some $\varepsilon > 0$. For every instance I of Not-All-Equal-3-Sat, we can construct a graph G_I for MSC in 2D such that it has a scan time of 240° if I is satisfiable, and a scan time of at least $360° - \varepsilon$ otherwise. We essentially use the same reduction as in the proof of Lemma 5. The idea is to embed the constructed graph G_I in the plane on a triangular grid such that the transition costs are reflected by the angle differences. Figure 7 illustrates the gadgets.

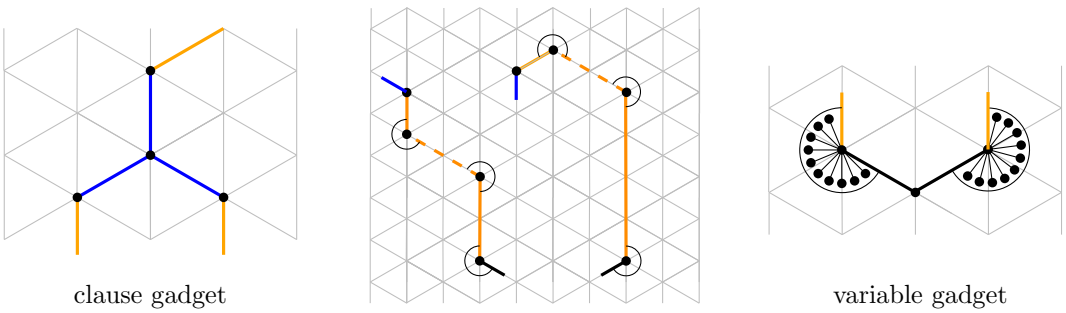


Figure 7 Embedding the graph G_I into the plane by using $\phi = 120°$. Additional leaves are added to force the usage of the larger angle of 240°.

The clause and variable gadgets are connected by paths instead of edges (solid and dashed orange edges). In order to enforce angles of 240°, we insert additional edges and vertices into the 240° angle with an angle difference of ε. If an incident vertex uses the shorter angle of 120°, it still needs to cover the additional edges resulting in an overall turning angle of at least $360° - \varepsilon = 3\phi - \varepsilon$. For full details, see the long version [15]. ◀

3.1.2 4.5-approximation for bipartite graphs in 2D

Conversely, we give absolute and relative performance guarantees for bipartite graphs in 2D.

▶ **Theorem 5.** *Let $I = (P, E)$ be a bipartite instance of* MSC *with vertex classes $P = P_1 \cup P_2$. Then I has a scan cover of time 360°. Moreover, if P_1 and P_2 are separated by a line, there is a scan cover of time 180°.*

Proof. We show that the following strategy yields a scan cover of time 360°: All points turn in clockwise direction, with the points in P_1 starting with heading north and the points in P_2 with heading south; see Figure 8a for an example. Note that the connecting line between any point $p_1 \in P_1$ and any point $p_2 \in P_2$ forms alternate angles with the parallel vertical lines through p_1 and p_2, so both face each other when reaching this angle during their rotation; see Figure 8b. In the case of separated point sets, a rotation of 180° suffices to sweep the other set, as illustrated in Figure 8c. ◀

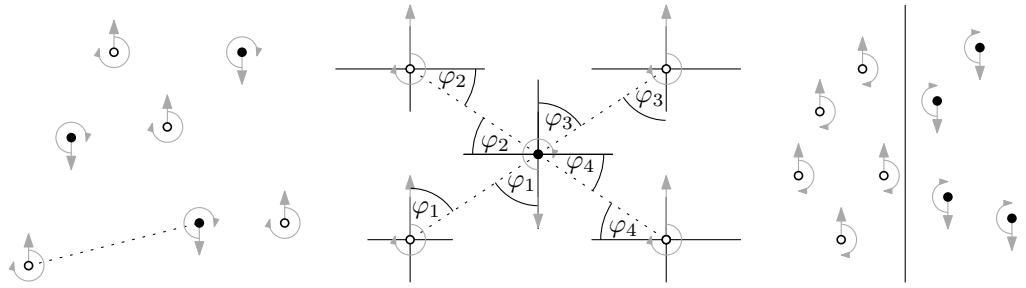


Figure 8 **(a)** The vertices in P_1 and P_2 rotate clockwise and start by heading north and south, respectively. **(b)** Due to alternate angles, vertices of different parts of the vertex partition face each other at the same time. **(c)** When P_1 and P_2 are separated by a line, a scan time of 180° suffices.

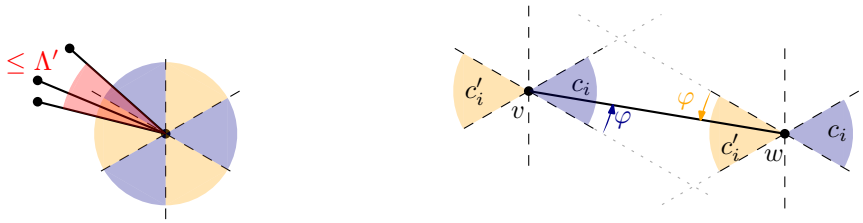
Theorem 5 yields an *absolute* bound for bipartite graphs. Now we give a constant-factor approximation even for small optimal values.

▶ **Theorem 6.** *There is a 4.5-approximation algorithm for* MSC *for bipartite graphs in 2D.*

Proof. Consider an instance I of MSC in 2D and let Λ denote the minimum angle such that for every vertex some Λ-cone contains all its edges. Clearly, Λ is a lower bound on the value OPT of a minimum scan cover of I. We use one of two strategies depending on Λ.

If $\Lambda \geq 90°$, we use the strategy of Theorem 5 which yields a scan cover of at most 360° and hence a 4-approximation.

If $\Lambda < 90°$, we use an adaptive strategy as follows. For each vertex, we partition the set of headings $[0, 360°)$ into $2s$ sectors of size $\Lambda' = {}^{360°}/_{2s}$, see Figure 9a. We choose s maximal (and, thus, Λ' minimal) such that $\Lambda' \geq \Lambda$. This implies that the edges of every vertex are contained in at most two adjacent sectors, see Figure 9a. Note also that $\Lambda' < {}^{3}/_{2}\Lambda$, because $\Lambda > {}^{360°}/_{2(s+1)}$ and $s \geq 2$.



(a) **(b)**

Figure 9 **(a)** Dividing the headings into $2s$ sectors with angle $\Lambda' = {}^{360°}/_{2s}$ by inserting s lines. For every vertex, the incident edges lie in at most two adjacent sectors of size Λ', because $\Lambda \leq \Lambda'$. **(b)** An edge $e = vw$ that lies in c_i for v, lies in $c_i' = c_{i+s}$ for w. If v scans c_1 and w scans c_1' (both counterclockwise), they scan e at the same time φ due to the alternate angles in the parallelogram.

Let the sectors be $c_i = [(i \cdot \Lambda', (i+1) \cdot \Lambda')$, for $i = 0, \ldots, 2s-1$. Moreover, $c_i' := c_{i+s \mod 2s}$ is the sector *opposite* of c_i. Note that an edge $e = vw$ is in the sector c_i of v if and only if e is in the opposite sector c_i' of w, see Figure 9b. Let C_{even} be the set of sectors with even indices, C_{odd} the one with odd indices, and C'_{even} and C'_{odd} the set of opposite sectors, respectively. Because the incident edges of each vertex are contained in at most two adjacent sectors, every vertex has edges in (at most) one sector of C_{even} and one sector of C_{odd}.

This allows the following strategy. Denote the bipartition of the vertex set by $P = P_1 \cup P_2$. In the first phase, the vertices in P_1 scan the sector with edges in C_{even} in clockwise direction, while the vertices in P_2 scan the sector in C'_{even}. In the second phase, vertices in P_1 scan the sector in C_{odd} in counterclockwise direction, while the vertices in P_2 scan the sector in C'_{odd}.

As in Theorem 5, every edge is scanned in the first or second scan phase due to the alternate angles. Clearly, each scan phase needs Λ'. Between the two scan phases, every vertex v needs to turn to change its heading from the end heading of the first scan phase to the start heading of the second. Because both sectors of v are incident and due to the reversed direction, the turning angle is at most Λ'; in particular, either the end heading of the first sector is contained in the boundary of the second sector or the two start headings of both phases coincide. Figure 10 depicts an example scan cover. The resulting scan time is $3\Lambda' \leq 3 \cdot {}^{3}/_{2}\Lambda \leq 4.5 \cdot \text{OPT}$. ◀

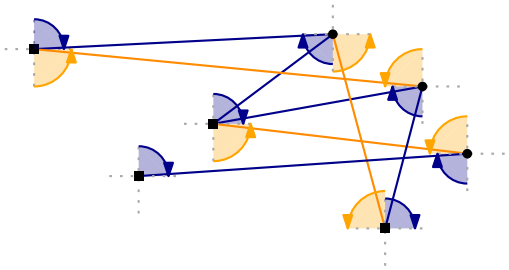


Figure 10 An example with $\Lambda' = 90°$. P_1 and P_2 are indicated by squares and circles, respectively. The blue sectors are scanned in the first scan phase; the orange sectors in the second. Each scan phase and the turning phase costs Λ'.

3.2 Graphs with bounded chromatic number

Like in 1D, the value of a minimum scan cover in 2D has a strong relation to the chromatic number. More specifically, we show that the optimal scan time lies in $\Theta(\log_2 \chi(G))$ and that for a given coloring of the graph G with $\chi(G)^c$ colors, we can provide an $O(c)$-approximation.

▶ **Lemma 6.** *Let $I = (P, E)$ be an instance of MSC in $\mathbb{R}^d, d > 1$. If I has a scan cover of length $T > 0$, then G has a cut cover of size $d \cdot \lceil \frac{T}{90^\circ} \rceil$, i.e., $c(G) \leq d \cdot \lceil \frac{T}{90^\circ} \rceil$.*

Proof. Partition the scan cover into $\lceil \frac{T}{90^\circ} \rceil$ intervals of length at most 90°. For each interval i, we consider the set of edges that are scanned within this interval, inducing a graph G_i. We show that each G_i is 2^d-partite. Because $c(G_i) = \lceil \log_2(\chi(G_i)) \rceil \leq d$, this implies the claim.

We first consider the case $d = 2$. We classify the points of P into four sets, depending on their turning behavior within the interval i. Each point has a quadrant $[0, 90^\circ)$, $[90^\circ, 180^\circ)$, $[180^\circ, 270^\circ)$, or $[270^\circ, 360^\circ)$ to which it is heading at the time 45°; we assign each point this quadrant. Note that every point can only leave its assigned quadrant by less than ±45°. Two points that are assigned to the same quadrant are independent in G_i: When their edge is scanned, the headings of the two points have to be opposite, i.e., they differ by exactly 180°. Thus, the only case in which two point headings could differ by 180° is if one leaves its quadrant by 45° in clockwise and the other by 45° in counterclockwise direction. However, in this case, the points would not have been assigned to the same half-open sector. For an illustration consider Figure 11.

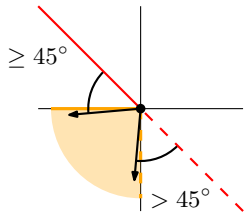


Figure 11 Every vertex is assigned to the (orange) sector to which it is heading at time 45°. The boundary of the reachable headings within 90° is shown in red. Since the sector is half-open, two vertices assigned to the same sector cannot reach opposing headings and thus cannot scan their edge.

For $d \geq 3$, the idea is analogous. To simplify the argument we choose a coordinate system, i.e., an orthonormal basis (ONB), such that, at time 45°, no point heads in a direction that lies on a lower-dimensional subspace spanned by the basis vectors. Let B denote the set of all potential basis vectors in $\mathbb{R}^d$, i.e., $B = \mathbb{R}^d$. For every point, we delete the line spanned by its heading h at time 45° and the $(d-1)$-dimensional subspace orthogonal to h from B. The remaining set B' is a d-dimensional space minus a finite number of lower-dimensional subspaces. It follows by induction that B' contains an ONB.

The points of P are partitioned into 2^d different sets, depending on the orthant in which they are contained at time 45°. Note that if two point headings of the same orthant differ by 180°, their angle difference at time 45° is 90°, i.e., they lie on a lower dimensional subspace spanned by the basis vectors. This is a contradiction to the choice of the ONB. ◀

Because $c(G) = \lceil \log_2 \chi(G) \rceil$, Lemma 6 has the following implication.

▶ **Lemma 7.** *Every instance I of MSC in $\mathbb{R}^d$ needs a scan time T of at least $\Omega(\log_2 \chi(G_I))$, with G_I denoting the underlying graph of I. More precisely, $T \geq \frac{\lceil \log_2 \chi(G) \rceil - d}{d} \cdot 90^\circ$.*

Proof. Because $c(G) = \lceil \log_2 \chi(G) \rceil$, Lemma 6 implies that $\lceil \log_2 \chi(G) \rceil \leq d \lceil \frac{T}{90^\circ} \rceil$. In particular, it holds that $\lceil \log_2 \chi(G) \rceil \leq d \frac{T}{90^\circ} + d \iff \frac{\lceil \log_2 \chi(G) \rceil - d}{d} \cdot 90^\circ \leq T$. ◀

These insights have the following implications.

▶ **Theorem 7.** *For instances of* MSC *in 2D with a k-coloring of the graph $G = (V, E)$, such that $k \leq \chi(G)^c$ for some function c, there is an $O(c)$-approximation.*

Proof. Partition G into $\lceil \log_2 k \rceil$ bipartite graphs G_i. By Theorem 6, each G_i can be scanned in time $\beta \cdot OPT_i$, with $\beta = 4.5$ and OPT_i denoting the optimum of the instance induced by G_i. Clearly, $OPT_i \leq OPT$, and turning from the last scan of one bipartite graph to the next takes at most a time of OPT. Hence, this scan cover needs a time of at most $(\beta + 1)OPT\lceil \log_2 k \rceil$.

If $\chi(G) \leq 4$, then $O(\lceil \log_2 k \rceil) \leq O(\lceil c \cdot \log_2(\chi(G)) \rceil) \leq O(\lceil c \cdot \log_2(4) \rceil) \leq O(\lceil 2c \rceil) \in O(c)$.

If $\chi(G) \geq 5$, then Lemma 7 ensures that $OPT \geq \Omega(\log_2(\chi(G))) > 0$. Therefore, the performance guarantee is in $O\left(\frac{\log_2 k}{\log_2(\chi(G))}\right) = O\left(\frac{c \log_2(\chi(G))}{\log_2(\chi(G))}\right) = O(c)$. ◀

As a direct implication of Theorem 7, we get a spectrum of approximation algorithms for interesting special cases.

▶ **Corollary 8.** *MSC in 2D allows the following approximation factors.*

1. $O(\log_2 n)$ *for all graphs. Furthermore, the minimum scan time lies in* $\Theta(\log_2 \chi(G))$.
2. $O(1)$ *for planar graphs.*
3. $O(\log_2 d)$ *for d-degenerate graphs.*
4. $O(1)$ *for graphs of bounded treewidth.*
5. $O(1)$ *for complete graphs.*

The following bound shows a refined approximation for complete graphs.

▶ **Corollary 9.** *Consider the* MSC *for complete graphs with n vertices in 2D. There is a c-approximation algorithm with $c \to 6$ for $n \to \infty$.*

Proof. We may assume without loss of generality that $n > 4$. By Lemma 7, the minimum scan time is at least $(\lceil \log_2(n) \rceil - 2) \cdot 45° > 0$. For the upper bound, we partition the point set recursively into $\lceil \log_2(n) \rceil$ bipartite graphs by lines (alternating horizontal and vertical). Hence, Theorem 5 allows us to scan each bipartite graph within 180°. The transition between two scan phases is at most 90°. Therefore, the scan time is upper bounded by $\lceil \log_2(n) \rceil 180° + (\lceil \log_2(n) \rceil - 1)90°$. This yields a performance guarantee of

$$\frac{270°(\lceil \log_2(n) \rceil - 2) + 450°}{45°(\lceil \log_2(n) \rceil - 2)} = 6 + \frac{10}{(\lceil \log_2(n) \rceil - 2)}.$$ ◀

The factor in Corollary 9 is $c \leq 8$ when $n \geq 2^7$ and $c \leq 7$ when $n \geq 2^{12}$.

4 Three-dimensional point sets and abstract MSC

In the following, we observe that A-MSC generalizes the Path-TSP.

▶ **Observation 8.** *Let $G = (V, E)$ be a star on $n + 1$ vertices with center v and α a metric transition cost function on $E \times E$. Then, an* A-MSC *of (G, α) corresponds to a TSP-path of the complete graph on $V \setminus \{v\}$ with metric cost $c(u_1, u_2) = \alpha(vu_1, vu_2)$ and vice versa.*

Observation 8 has two immediate consequences. Firstly, because the metric Path-TSP is NP-hard, it follows that

▶ **Observation 9.** *A-MSC is NP-hard even for stars.*

Secondly, the 1.5-approximation for metric Path-TSP by Zenklusen [27] can be applied.

▶ **Observation 10.** *There exists a* 1.5*-approximation algorithm for* A-MSC *for stars.*

In contrast to 1D and 2D, we show that the chromatic number does not provide an upper bound for MSC in 3D and A-MSC.

▶ **Observation 11.** *There are instances of* MSC *in* $3D$ *with* $\chi(G) = 2$ *that need at least* $\Omega(\sqrt{n})$*. There are instances of* MSC *in* nD *with* $\chi(G) = 2$ *that need at least* $\Omega(n)$*.*

Proof. For the first claim consider a geodesic triangular grid on a sphere and embed a star graph such that its leaves are grid points and the center of the star lies in the center of the sphere. The $\Omega(\sqrt{n})$ can be achieved by increasing the resolution of the grid by subdivision, see Figure 12: While the minimum turn cost between two consecutive edges approximately halves, the number of vertices roughly quadruples, doubling the overall costs that is lower bounded by $(n-1) \cdot l$ if l is the minimum edge length.

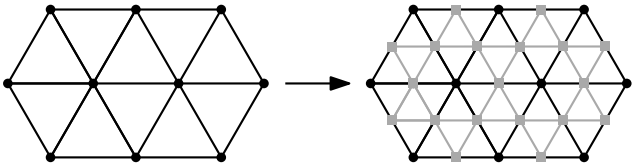


Figure 12 The geodesic grid is refined by subdividing the edges. Because a triangulation with n vertices has $3n - 6$ edges, subdividing roughly quadruples the number of vertices.

The second claim follows from considering a star on n vertices for which each leaf is placed on a different coordinate axis. Therefore, the turn cost between any two edges is 90° and it takes $90°n$ to scan the graph. ◀

The approximation technique for bipartite graphs in 2D relies on alternate angles and fails for $3D$ or A-MSC. Nevertheless, we provide a 2.5-approximation for trees.

▶ **Theorem 10.** *There exists a* 2.5*-approximation algorithm for* A-MSC *for trees.*

Proof. Let $I = (G, \alpha)$ be an instance of A-MSC for which G is a tree, and let OPT be the minimum scan time of I. For every vertex v, we approximate an ordering of minimum cost over all its incident edges E_v. Let $N(v)$ denote the set of neighbors of v. By Observation 8 such an ordering corresponds to a TSP-path. Consequently, we may use the 1.5-approximation algorithm for metric Path-TSP by Zenklusen [27]. Moreover, we enhance the edge ordering to a cyclic ordering by inserting an edge from the last to the first edge; because the cost function is metric, the cost of the additional edge is upper bounded by the minimum cost ordering of the incident edges. Therefore, the scan time ℓ_v of the computed cyclic edge ordering of v is at most $\ell_v \leq (1.5 + 1)OPT$.

We construct a scan cover as follows: Every vertex follows its cyclic edge ordering. The start headings of the vertices are chosen such that the scan time of each edge $e = uv$ is synchronized at the vertices u and v. To this end, we choose some vertex r as the root and denote the parent of each vertex v by $par(v)$ in the tree G with respect to the root r. We scan the edges of r according to the cyclic edge orderings by starting with any heading, see also Figure 13. We then determine the start headings by a tree traversal (e.g., DFS or BFS): Let v be a vertex whose start heading has to be determined, and assume the start heading of $u := par(v)$ is already fixed. When uv is scanned at time t for u, then the cyclic ordering of E_v is shifted, so that v sees uv also at time t. If this time lies between two scans,

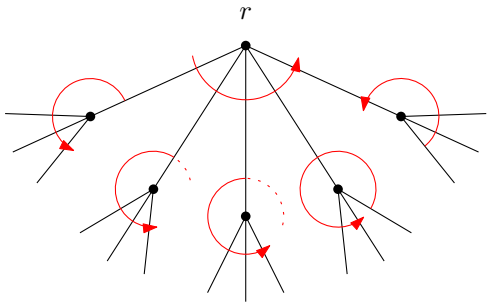


Figure 13 Root r can choose its schedule. The (cyclic) schedules of the children are synchronized with the timing of its parent. Because the graph is a tree, there are no cyclic dependencies.

we simply start at the next incident edge and let the vertex wait for the appropriate time. Because all vertices start at the same time, the resulting scan cover has a scan time of at most $\max_v \ell_v \leq 2.5 \cdot OPT$. ◀

Theorem 10 allows an approximation algorithm in terms of the arboricity of the underlying graph. Recall that the *arboricity* of a graph denotes the minimum number of forests into which its edges can be partitioned.

▶ **Theorem 11.** *There is a* $3.5A$*-approximation for the* A-MSC *for graphs of arboricity* A*.*

Proof. We compute a decomposition into A forests in polynomial time [17]. To obtain a scan cover we use the approximation algorithm of Theorem 10 for each forest and concatenate the resulting scan covers in any order. Because the transition cost between any two forests is upper bounded by the minimum scan time OPT, the resulting scan cover has time of at most $(2.5 + 1)OPT \cdot A$. Consequently, we obtain a $3.5A$-approximation. ◀

5 Conclusion and open problems

We have presented several algorithmic results for the abstract and geometric version of the minimum scan cover problem with a metric angular cost function, which has strong connections to the chromatic number.

There is a spectrum of interesting directions for future work. How can we make use of methods for solving graph coloring problems to compute practical solutions to real-world instances? In many scenarios, this will involve considering satellites on different trajectories, in the presence of a large obstacle: the earth. This gives rise to a variety of generalizations, such as the presence of time windows for possible communication. Other variations of both practical and theoretical interest arise from considering objective functions such as minimizing the total energy of all satellites or minimizing the maximum energy per satellite.

References

1 Alok Aggarwal, Don Coppersmith, Sanjeev Khanna, Rajeev Motwani, and Baruch Schieber. The angular-metric traveling salesman problem. *SIAM J. Comp.*, 29(3):697–711, 1999.

2 Ali Allahverdi. The third comprehensive survey on scheduling problems with setup times/costs. *European Journal of Operational Research*, 246(2):345–378, 2015.

3 Ali Allahverdi, Jatinder N.D. Gupta, and Tariq Aldowaisan. A review of scheduling research involving setup considerations. *Omega*, 27(2):219–239, 1999.

4 Ali Allahverdi, C.T. Ng, T.C. Edwin Cheng, and Mikhail Y. Kovalyov. A survey of scheduling problems with setup times or costs. *European journal of operational research*, 187(3):985–1032, 2008.

5 Esther M. Arkin, Michael A. Bender, Erik D. Demaine, Sándor P. Fekete, Joseph S. B. Mitchell, and Saurabh Sethia. Optimal covering tours with turn costs. In *Proc. 12th ACM-SIAM Symp. Disc. Alg. (SODA)*, pages 138–147, 2001.

6 Esther M. Arkin, Michael A. Bender, Erik D. Demaine, Sándor P. Fekete, Joseph S. B. Mitchell, and Saurabh Sethia. Optimal covering tours with turn costs. *SIAM J. Comp.*, 35(3):531–566, 2005.

7 Rom Aschner and Matthew J. Katz. Bounded-angle spanning tree: modeling networks with angular constraints. *Algorithmica*, 77(2):349–373, 2017.

8 Aaron T. Becker, Mustapha Debboun, Sándor P. Fekete, Dominik Krupke, and An Nguyen. Zapping Zika with a Mosquito-Managing Drone: Computing Optimal Flight Patterns with Minimum Turn Cost. In *Proc. 33rd Symp. Comp. Geom. (SoCG)*, pages 62:1–62:5, 2017.

9 Paz Carmi, Matthew J. Katz, Zvi Lotker, and Adi Rosén. Connectivity guarantees for wireless networks with directional antennas. *Computational Geometry*, 44(9):477–485, 2011.

10 Julia Chuzhoy and Sanjeev Khanna. Hardness of cut problems in directed graphs. In *Proceedings of the Thirty-eighth Annual ACM Symposium on Theory of Computing*, STOC '06, pages 527–536, New York, NY, USA, 2006. ACM.

11 Erik D. Demaine, Joseph S. B. Mitchell, and Joseph O'Rourke. The Open Problems Project. URL: `http://cs.smith.edu/~orourke/TOPP/`.

12 Sándor P. Fekete and Dominik Krupke. Covering tours and cycle covers with turn costs: Hardness and approximation. In *Proceedings of the 11th International Conference on Algorithms and Complexity (CIAC)*, pages 224–236, 2019.

13 Sándor P. Fekete and Dominik Krupke. Practical methods for computing large covering tours and cycle covers with turn cost. In *Proc. 21st SIAM Workshop Alg. Engin. Exp. (ALENEX)*, pages 186–198, 2019.

14 Sándor P. Fekete and Gerhard J. Woeginger. Angle-restricted tours in the plane. *Comp. Geom.*, 8:195–218, 1997.

15 Sándor P. Fekete, Linda Kleist, and Dominik Krupke. Minimum scan cover with angular transition costs, 2020. `arXiv:2003.08816`.

16 Mike Fellows, Panos Giannopoulos, Christian Knauer, Christophe Paul, Frances A. Rosamond, Sue Whitesides, and Nathan Yu. Milling a graph with turn costs: A parameterized complexity perspective. In *Proc 36th Worksh. Graph Theo. Conc. Comp. Sci. (WG)*, pages 123–134, 2010.

17 Harold N. Gabow and Herbert H. Westermann. Forests, frames, and games: Algorithms for matroid sums and applications. *Algorithmica*, 7(1):465, June 1992.

18 Edna Ayako Hoshino. The minimum cut cover problem. *Electronic Notes in Discrete Mathematics*, 37:255–260, 2011. `doi:10.1016/j.endm.2011.05.044`.

19 Subhash Khot. Improved inapproximability results for maxclique, chromatic number and approximate graph coloring. In *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, pages 600–609. IEEE, 2001.

20 Haje Korth, Michelle F. Thomsen, Karl-Heinz Glassmeier, and W. Scott Phillips. Particle tomography of the inner magnetosphere. *Journal of Geophysical Research: Space Physics*, 107(A9):SMP–5, 2002.

21 Dominik Krupke, Volker Schaus, Andreas Haas, Michael Perk, Jonas Dippel, Benjamin Grzesik, Mohamed Khalil Ben Larbi, Enrico Stoll, Tom Haylock, Harald Konstanski, Kattia Flores Pozzo, Mirue Choi, Christian Schurig, and Sándor P. Fekete. Automated data retrieval from large-scale distributed satellite systems. In *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)*, pages 1789–1795. IEEE, 2019.

22 Richard Loulou. Minimal cut cover of a graph with an application to the testing of electronic boards. *Oper. Res. Lett.*, 12(5):301–305, November 1992.

23 Rajeev Motwani and Joseph (Seffi) Naor. On exact and approximate cut covers of graphs. Technical report, Stanford University, Stanford, CA, USA, 1994.

24 Herbert Robbins. A remark on stirling's formula. *The American Mathematical Monthly*, 62(1):26–29, 1955.

25 Yuri N. Sotskov, Alexandre Dolgui, and Frank Werner. Mixed graph coloring for unit-time job-shop scheduling. *International Journal of Mathematical Algorithms*, 2(4):289–323, 2001.

26 Kaoru Watanabe, Masakazu Sengoku, Hiroshi Tamura, and Shoji Shinoda. Cut cover problem in directed graphs. In *IEEE. APCCAS 1998. 1998 IEEE Asia-Pacific Conference on Circuits and Systems. Microelectronics and Integrating Systems. Proceedings (Cat. No.98EX242)*, pages 703–706, 1998.

27 Rico Zenklusen. A 1.5-approximation for path tsp. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '19, pages 1539–1549, Philadelphia, PA, USA, 2019. Society for Industrial and Applied Mathematics.

ETH-Tight Algorithms for Long Path and Cycle on Unit Disk Graphs

Fedor V. Fomin
University of Bergen, Norway
fomin@ii.uib.no

Daniel Lokshtanov
University of California, Santa Barbara, CA, USA
daniello@ucsb.edu

Fahad Panolan
Department of Computer Science and Engineering, IIT Hyderabad, India
fahad@cse.iith.ac.in

Saket Saurabh
Department of Informatics, University of Bergen, Norway
The Institute of Mathematical Sciences, HBNI and IRL 2000 ReLaX, Chennai, India
saket@imsc.res.in

Meirav Zehavi
Ben-Gurion University of the Negev, Beer-Sheva, Israel
meiravze@bgu.ac.il

Abstract

We present an algorithm for the extensively studied Long Path and Long Cycle problems on unit disk graphs that runs in time $2^{\mathcal{O}(\sqrt{k})}(n+m)$. Under the Exponential Time Hypothesis, Long Path and Long Cycle on unit disk graphs cannot be solved in time $2^{o(\sqrt{k})}(n+m)^{\mathcal{O}(1)}$ [de Berg et al., STOC 2018], hence our algorithm is optimal. Besides the $2^{\mathcal{O}(\sqrt{k})}(n+m)^{\mathcal{O}(1)}$-time algorithm for the (arguably) much simpler Vertex Cover problem by de Berg et al. [STOC 2018] (which easily follows from the existence of a $2k$-vertex kernel for the problem), *this is the only known ETH-optimal fixed-parameter tractable algorithm on UDGs*. Previously, Long Path and Long Cycle on unit disk graphs were only known to be solvable in time $2^{\mathcal{O}(\sqrt{k}\log k)}(n+m)$. This algorithm involved the introduction of a new type of a tree decomposition, entailing the design of a very tedious dynamic programming procedure. Our algorithm is substantially simpler: we completely avoid the use of this new type of tree decomposition. Instead, we use a marking procedure to reduce the problem to (a weighted version of) itself on a standard tree decomposition of width $\mathcal{O}(\sqrt{k})$.

2012 ACM Subject Classification Theory of computation → Fixed parameter tractability; Theory of computation → Computational geometry

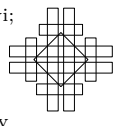
Keywords and phrases Optimality Program, ETH, Unit Disk Graphs, Parameterized Complexity, Long Path, Long Cycle

Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.44

Related Version A full version of the paper is available at [34].

Funding *Fedor V. Fomin*: Research Council of Norway via the project MULTIVAL.
Daniel Lokshtanov: European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant no. 715744), and United States - Israel Binational Science Foundation grant no. 2018302.
Saket Saurabh: European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant no. 819416), and Swarnajayanti Fellowship grant DST/SJF/MSA-01/2017-18.
Meirav Zehavi: Israel Science Foundation grant no. 1176/18, and United States – Israel Binational Science Foundation grant no. 2018302.

36th International Symposium on Computational Geometry (SoCG 2020).
Editors: Sergio Cabello and Danny Z. Chen; Article No. 44; pp. 44:1–44:18
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



1 Introduction

Unit disk graphs are the intersection graphs of disks of radius 1 in the Euclidean plane. That is, given n disks of radius 1, we represent each disk by a vertex, and insert an edge between two vertices if and only if their corresponding disks intersect. Unit disk graphs form one of the best studied graph classes in computational geometry because of their use in modelling optimal facility location [59] and broadcast networks such as wireless, ad-hoc and sensor networks [37, 47, 61]. These applications have led to an extensive study of NP-complete problems on unit disk graphs in the realms of computational complexity and approximation algorithms. We refer the reader to [17, 25, 40] and the citations therein for these studies. However, these problems remain hitherto unexplored in the light of parameterized complexity with exceptions that are few and far between [1, 15, 35, 44, 57].

We study the Long Path (resp. Long Cycle) problem on unit disk graphs. Here, given a graph G and an integer k, the objective is to decide whether G contains a path (resp. cycle) on at least k vertices. To the best of our knowledge, the Long Path problem is among the five most extensively studied problems in Parameterized Complexity [18, 24] (see Section 1.1). One of the best known open problems in Parameterized Complexity was to develop a $2^{\mathcal{O}(k)}n^{\mathcal{O}(1)}$-time algorithm for Long Path on general graphs [55], that is, shaving the $\log k$ factor in the exponent of the previously best $2^{\mathcal{O}(k \log k)}n^{\mathcal{O}(1)}$-time parameterized algorithm for this problem on general graphs [52]. This was resolved in the positive in the seminal work by Alon, Yuster and Zwick on color coding 25 years ago [5], which was recently awarded the IPEC-NERODE prize for the most outstanding research in Parameterized Complexity. In particular, the aforementioned $2^{\mathcal{O}(k)}n^{\mathcal{O}(1)}$-time algorithm for Long Path on general graphs is optimal under the Exponential Time Hypothesis (ETH).

Both Long Path and Long Cycle are NP-hard on unit disk graphs [42], and cannot be solved in time $2^{o(\sqrt{n})}$ (hence neither in time $2^{o(\sqrt{k})}n^{\mathcal{O}(1)}$) on unit disk graphs unless the ETH fails [20]. Our contribution is an *optimal* parameterized algorithm for Long Path (and Long Cycle) on unit disk graphs under the ETH. Specifically, we prove the following.

▶ **Theorem 1.** *Long Path and Cycle are solvable in time* $2^{\mathcal{O}(\sqrt{k})}(n+m)$ *on unit disk graphs.*

Two years ago, a celebrated work by de Berg et al. [20] presented (non-parameterized) algorithms with running time $2^{\mathcal{O}(\sqrt{n})}$ for a number of problems on intersection graphs of so called "fat", "similarly-sized" geometric objects, accompanied by matching lower bounds of $2^{\Omega(\sqrt{n})}$ under the ETH. Only for the Vertex Cover problem this work implies an ETH-tight parameterized algorithm. More precisely, Vertex Cover admits a $2k$-vertex kernel on general graphs [53, 18], hence the $2^{\mathcal{O}(\sqrt{n})}$-time algorithm for Vertex Cover by de Berg et al. [20] is trivially a $2^{\mathcal{O}(\sqrt{k})}n^{\mathcal{O}(1)}$-time fixed-parameter tractable algorithm for this problem. None of the other problems in [20] is known to admit a linear-vertex kernel, and we know of no other work that presents a $2^{\mathcal{O}(\sqrt{k})}n^{\mathcal{O}(1)}$-time algorithm for any basic problem on unit disk graphs. Thus, we present the second known ETH-tight fixed-parameter tractable algorithm for a basic problem on unit disk graphs, or, in fact, on any family of geometric intersection graphs of fat objects. In a sense, our work is the first time where tight ETH-optimality of fixed-parameter tractable algorithms on unit disk graphs is explicitly answered. (The work of de Berg et al. [20] primarily concerned non-parameterized algorithms.) We believe that our work will open a door to the realm to an ETH-tight optimality program for fixed-parameter tractable algorithms on intersection graphs of fat geometric objects.

Prior to our work, Long Path and Long Cycle were known to be solvable in time $2^{\mathcal{O}(\sqrt{k}\log k)}(n+m)$ on unit disk graphs [33]. Thus, we shave the $\log k$ factor in the exponent in the running time, and thereby, in particular, achieve optimality. Our algorithm is substantially

simpler, both conceptually and technically, than the previous algorithm as we explain below. The main tool in the previous algorithm (of [33]) for Long Path (and Long Cycle) on unit disk graphs was a new (or rather refined) type of a tree decomposition. The width of the tree decomposition constructed in [33] is $k^{\mathcal{O}(1)}$, which on its own does not enable to design a subexponential (or even single-exponential) time algorithm. However, each of its bags (of size $k^{\mathcal{O}(1)}$) is equipped with a partition into $\mathcal{O}(\sqrt{k})$ sets such that each of them induces a clique. By establishing a property that asserts the existence of a solution (if at least one solution exists) that crosses these cliques "few" times, the tree decomposition can be exploited. Specifically, this exploitation requires to design a very tedious dynamic programming algorithm (significantly more technical than algorithms over "standard" tree decompositions, that is, tree decompositions of small width) to keep track of the interactions between the cliques in the partitions.

We completely avoid the use of the new type of tree decomposition of [33]. Instead, we use a simple marking procedure to reduce the problem to (a weighted version of) itself on a tree decomposition of width $\mathcal{O}(\sqrt{k})$. Then, the new problem can be solved by known algorithms as black boxes by employing either an essentially trivial $\mathtt{tw}^{\mathcal{O}(\mathtt{tw})}n$-time algorithm, or a more sophisticated $2^{\mathcal{O}(\mathtt{tw})}n$-time algorithm (of [10] or [29]). On a high level, we are able to mark few vertices in certain cliques (which become the cliques in the above mentioned partitions of bags in [33]), so that there exists a solution (if at least one solution exists) that uses only marked vertices as "portals" – namely, it crosses cliques only via edges whose both endpoints are marked. Then, in each clique, we can just replace all unmarked vertices by a single weighted vertex. This reduces the size of each clique to be constant, and yields a tree decomposition of width $\mathcal{O}(\sqrt{k})$. We believe that our idea of identification of portals and replacement of all non-portals by few weighted vertices will find further applications in the design of ETH-tight parameterized algorithms on intersection graphs of fat geometric objects.

Before we turn to briefly survey some additional related works, we would like to stress that shaving off logarithmic factors in the exponent of running times of parameterized algorithms is a major issue in this field. Indeed, when they appear in the exponent, logarithmic factors have a *critical* effect on efficiency that can render algorithms impractical even on small instances. Over the past two decades, most fundamental techniques in Parameterized Complexity targeted not only the objective of eliminating the logarithmic factors, but even improving the base c in running times of the form $c^k n^{\mathcal{O}(1)}$. For example, this includes the aforementioned color coding technique [5] that was developed to shave off the $\log k$ in a previous $2^{\mathcal{O}(k \log k)}n^{\mathcal{O}(1)}$-time algorithm, which further entailed a flurry of research on techniques to improve the base of the exponent (see Section 1.1), and the cut-and-count technique to design parameterized algorithms in time $2^{\mathcal{O}(t)}n^{\mathcal{O}(1)}$ rather than $2^{\mathcal{O}(t \log t)}n^{\mathcal{O}(1)}$ (in fact, for connectivity problems such as Long Path) on graphs of treewidth t [19]. Accompanying this active line of research, much efforts were devoted to prove that problems that have long resisted the design of algorithms without logarithmic factors in the exponent are actually unlikely to admit such algorithms [51].

1.1 Related Works on Long Path and Long Cycle

We now briefly survey some known results in Parameterized Complexity on Long Path and Long Cycle. Clearly, this survey is illustrative rather than comprehensive. The standard parameterization of Long Path and Long Cycle is by the solution size k, and here we will survey only results that concern this parameterization.

Long Path (parameterized by the solution size k) is arguably one of the problems with the richest history in Parameterized Complexity, having parameterized algorithms continuously developed since the early days of the field and until this day. The algorithms developed along

the way gave rise to some of the most central techniques in the field, such as color-coding [5] and its incarnation as divide-and-color [16], techniques based on the polynomial method [49, 50, 60, 8], and matroid based techniques [30]. The first parameterized algorithm for this problem was an $2^{\mathcal{O}(k \log k)}n^{\mathcal{O}(1)}$-time given in 1985 by Monien [52], even before the term "parameterized algorithm" was in known use. Originally in 1994, the logarithmic factor was shaved off [5], resulting in an algorithm with running time $c^k n^{\mathcal{O}(1)}$ for $c = 2e$. After that, a long line of works presenting improvements over c followed [49, 50, 60, 8, 30, 62, 39, 56, 16, 58], where the algorithm with the current best running time is a randomized algorithm whose time complexity is $1.66^k n^{\mathcal{O}(1)}$ [8]. Unless the ETH fails, Long Path (as well as Long Cycle) does not admit any algorithm with running time $2^{o(k)}n^{\mathcal{O}(1)}$ [41].

For a long time, the Long Cycle problem was considered to be significantly harder than Long Path due to the following reason: while the existence of a path of size at least k implies the existence of a path of size exactly k, the existence of a cycle of size at least k does not imply the existence of a cycle of size exactly k – in fact, the only cycle of size at least k in the input graph might be a Hamiltonian cycle. Thus, for this problem, the first parameterized algorithm appeared (originally) only in 2004 [36], and the first parameterized algorithm with running time $2^{\mathcal{O}(k)}n^{\mathcal{O}(1)}$ appeared (originally) only in 2014 [30]. Further improvements on the base of the exponent in the running time were given in [63, 31]. Lastly, we remark that due to their importance, over the past two decades there has been extensive research of Long Path and Long Cycle parameterized by k above some guarantee [7, 27, 45, 28], and the (approximate) counting versions of these problems [26, 6, 2, 4, 3, 13, 9]. Both Long Path and Long Cycle are unlikely to admit a polynomial kernel [11], and in fact, are even conjectured not to admit Turing kernels [38, 46].

While Long Path and Long Cycle remain NP-complete on planar graphs, they admit $2^{\mathcal{O}(\sqrt{k})}n^{\mathcal{O}(1)}$-time algorithms: By combining the bidimensionality theory of Demaine et al. [21] with efficient algorithms on graphs of bounded treewidth [23], Long Path and Long Cycle, can be solved in time $2^{\mathcal{O}(\sqrt{k})}n^{\mathcal{O}(1)}$ on planar graphs. Moreover, the parameterized subexponential "tractability" of Long Path/Cycle can be extended to graphs excluding some fixed graph as a minor [22]. Unfortunately, unit disk graphs are somewhat different than planar graphs and H-minor free graphs – in particular, unlike planar graphs and H-minor free graphs where the maximum clique size is bounded by 5 (for planar graphs) or some other fixed constant (for H-minor free graphs), unit disk graphs can contain cliques of arbitrarily large size and are therefore "highly non-planar". Nevertheless, Fomin et al. [35] were able to obtain subexponential parameterized algorithms of running time $2^{\mathcal{O}(k^{0.75} \log k)}n^{\mathcal{O}(1)}$ on unit disk graphs for Long Path, Long Cycle, Feedback Vertex Set and Cycle Packing. None of these four problems can be solved in time $2^{o(\sqrt{n})}$(and hence also in time $2^{o(\sqrt{k})}n^{\mathcal{O}(1)}$) on unit disk graphs unless the ETH fails [20]. Afterwards (originally in 2017), Fomin et al. [33] obtained improved, yet technically quite tedious, $2^{\mathcal{O}(\sqrt{k} \log k)}n^{\mathcal{O}(1)}$-time algorithms for Long Path, Long Cycle and Feedback Vertex Set and Cycle Packing. Recall that this work was discussed earlier in the introduction. Later, the same set of authors designed $2^{\mathcal{O}(\sqrt{k} \log k)}n^{\mathcal{O}(1)}$ time algorithm for the aforementioned problems on map graphs [32]. We also remark that recently, Panolan et al. [54] proved a contraction decomposition theorem on unit disk graphs and as an application of the theorem, they proved that Min-Bisection on unit disk graphs can be solved in time $2^{\mathcal{O}(k)}n^{\mathcal{O}(1)}$.

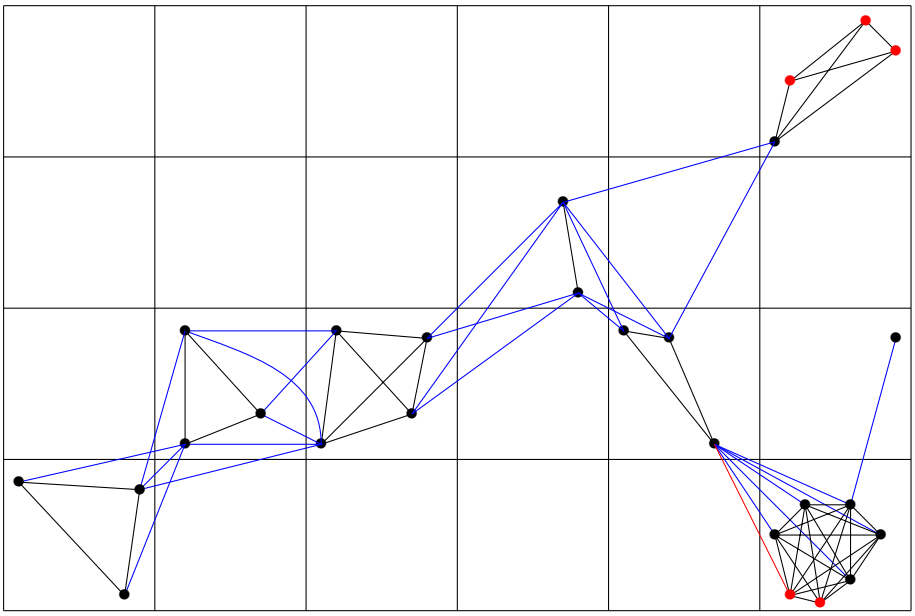


Figure 1 A clique-grid graph G. Marked vertices are colored black. Good and bad edges are colored blue and red, respectively (see Definition 8). For the sake of illustration only, we use the threshold 5 instead of 121 – that is, in phase II of marking let $\mathsf{Mark}_2(v,(i',j'))$ denote a set of 5 vertices in $f^{-1}(i',j')$ that are adjacent to v in G, where if no 5 vertices with this property exist, then let $\mathsf{Mark}_2(v,(i',j'))$ denote the set of all vertices with this property.

2 Preliminaries

For $\ell \in \mathbb{N}$, let $[\ell] = \{1,\ldots,\ell\}$. See [34] for standard graph theoretic terms.

Unit disk graphs. Let $P = \{p_1 = (x_1,y_1), p_2 = (x_2,y_2), \ldots, p_n = (x_n,y_n)\}$ be a set of points in the Euclidean plane. Let $D = \{d_1, d_2, \ldots, d_n\}$ where for every $i \in [n]$, d_i is the disk of radius 1 whose centre is p_i. Then, the unit disk graph of D is the graph G such that $V(G) = D$ and $E(G) = \{\{d_i,d_j\} \mid d_i,d_j \in D, i \neq j, \sqrt{(x_i-x_j)^2+(y_i-y_j)^2} \leq 2\}$. Throughout the paper, we suppose that any given UDG G is accompanied by a corresponding set of disks D, which is critical to our algorithm. We also remark that, given a graph G (without D), the decision of whether G is a UDG is NP-hard [14] (in fact, even $\exists\mathbb{R}$-hard [48]).

Clique-Grids. Intuitively, a clique-grid is a graph whose vertices can be embedded in grid cells (where multiple vertices can be embedded in each cell), so that the each cell induces a clique and "interacts" (via edges incident to its vertices) only with cells at "distance" at most 2 (see Fig. 1).

▶ **Definition 2** (Clique-Grid). *A graph G is a* clique-grid *if there exists a function $f : V(G) \to [t] \times [t]$ for some $t \in \mathbb{N}$, called a* representation*, satisfying the following conditions.*

1. *For all $(i,j) \in [t] \times [t]$, $G[f^{-1}(i,j)]$ is a clique.*
2. *For all $\{u,v\} \in E(G)$, $|i-i'| \leq 2$ and $|j-j'| \leq 2$ where $f(u) = (i,j)$ and $f(v) = (i',j')$.*

We call a pair $(i,j) \in [t] \times [t]$ a *cell*. It is easy to see that a unit disk graph is a clique-grid, and a representation of it, can be computed in linear time. A formal proof can be found in [33] (also see [43] for a similar result). Specifically, we will refer to the following proposition.

▶ **Proposition 3** ([43, 33]). *Let G be the unit disk graph of a set of unit disks D. Then, G is a clique-grid, and a representation of G can be computed in linear time.*

Treewidth. The treewidth of a graph is a standard measure of its "closeness" to a tree, whose formal definition is not explicitly used in this paper and therefore omitted. The treewidth of a graph can be approximated within a constant factor efficiently as follows.

▶ **Proposition 4** ([12]). *Given a graph G and a positive integer k, in time $2^{\mathcal{O}(k)} \cdot n$, we can either decide that $\mathtt{tw}(G) > k$ or output a tree decomposition of G of width $5k$.*

We will need the following proposition to argue that a unit disk graph *of bounded degree* contains a grid minor of dimension *linear* in its treewidth.

▶ **Proposition 5** ([35]). *Let G be a unit disk graph with maximum degree Δ and treewidth $\mathtt{tw}$. Then, G contains a $\frac{\mathtt{tw}}{100\Delta^3} \times \frac{\mathtt{tw}}{100\Delta^3}$ grid as a minor.*

3 Marking Scheme

In this section, we present a marking scheme whose purpose is to mark a constant number of vertices in each cell of a clique-grid G so that, if G has a path (resp. cycle) on at least k vertices, then it also has a path (resp. cycle) on at least k vertices that "crosses" cells only at marked vertices. Then, we further argue that unmarked vertices in a cell can be thought of, in a sense, as a "unit" representable by one weighted vertex. We note that we did not make any attempt to optimize the number of vertices marked, but only make the proof simple.

Marking Scheme. Let G be a clique-grid graph with representation $f : V(G) \to [t] \times [t]$. Then, the marking scheme consists of two phases defined as follows.

Phase I. For each pair of distinct cells $(i,j), (i',j') \in [t] \times [t]$ with $|i-i'| \leq 2$ and $|j-j'| \leq 2$, let M be a maximal matching where each edge has one endpoint in $f^{-1}(i,j)$ and the other endpoint in $f^{-1}(i',j')$; if $|M| \leq 241$, then denote $\mathsf{Mark}_1(\{(i,j),(i',j')\}) = M$, and otherwise choose a subset M' of M of size 241 and let $\mathsf{Mark}_1(\{(i,j),(i',j')\}) = M'$.
For each cell $(i,j) \in [t] \times [t]$, let $\mathsf{Mark}_1(i,j)$ denote the set of all vertices in $f^{-1}(i,j)$ that are endpoints of edges in $\bigcup_{(i',j')} \mathsf{Mark}_1(\{(i,j),(i',j')\})$ where (i',j') ranges over all cells such that $|i-i'| \leq 2$ and $|j-j'| \leq 2$; the vertices belonging to this set are called *marked vertices*.

Phase II. For each ordered pair of distinct cells $(i,j), (i',j') \in [t] \times [t]$ with $|i-i'| \leq 2$ and $|j-j'| \leq 2$ and vertex $v \in \mathsf{Mark}_1(i,j)$, let $\mathsf{Mark}_2(v,(i',j'))$ denote a set of 121 vertices in $f^{-1}(i',j')$ that are adjacent to v in G, where if no 121 vertices with this property exist, then let $\mathsf{Mark}_2(v,(i',j'))$ denote the set of all vertices with this property; the vertices that belong to this set are also called *marked vertices*.

Altogether. For each cell $(i,j) \in [t] \times [t]$, let $\mathsf{Mark}^\star(i,j)$ denote the set of all marked vertices in $f^{-1}(i,j)$.

Clearly, given G and f, $\mathsf{Mark}^\star(i,j)$ is not uniquely defined. Whenever we write $\mathsf{Mark}^\star(i,j)$, we refer to an arbitrary set that can be the result of the scheme above. We remark that "guessing" the endpoints of the sought path and marking them can simplify some arguments ahead, but this will lead to worse time complexity. We have the following simple observation regarding the size of $\mathsf{Mark}^\star(i,j)$ and the computation time.

▶ **Observation 6.** *Let G be a clique-grid with representation $f : V(G) \to [t] \times [t]$. For each cell $(i, j) \in [t] \times [t]$,$|\mathsf{Mark}^\star(i, j)| \leq 10^{10}$. Moreover, the computation of all the sets $\mathsf{Mark}^\star(i, j)$ together can be done in linear time.*

Proof. Consider a cell $(i, j) \in [t] \times [t]$. In the first phase, at most $24 \cdot 241$ vertices in $f^{-1}(i, j)$ are marked. In the second phase, for each of the 24 cells (i', j') such that $|i - i'| \leq 2$ and $|j - j'| \leq 2$, and each of the at most $24 \cdot 241$ marked vertices in $f^{-1}(i', j')$, at most 121 new vertices in $f^{-1}(i, j)$ are marked. Therefore, in total at most $24 \cdot 241 + 24 \cdot (24 \cdot 241) \cdot 121 \leq 10^{10}$ vertices in $f^{-1}(i, j)$ are marked. The claim regarding the computation time is immediate. ◀

As part of the proof that our marking scheme has the property informally stated earlier, we will use the following proposition.

▶ **Proposition 7** ([33])**.** *Let G be a clique-grid with representation f that has a path (resp. cycle) on at least k vertices. Then, G also has a path (resp. cycle) P on at least k vertices with the following property: for every two distinct cells (i, j) and (i', j'), there exist at most 5 edges $\{u, v\} \in E(P)$ such that $f(u) = (i, j)$ and $f(v) = (i', j')$.*

We now formally state and prove the property achieved by our marking scheme. For this purpose, we have the following definition (see Fig. 1) and lemma.

▶ **Definition 8.** *Let G be a clique-grid with representation f. An edge $\{u, v\} \in E(G)$ where $f(u) \neq f(v)$ is a* good *edge if $u \in \mathsf{Mark}^\star(i, j)$ and $v \in \mathsf{Mark}^\star(i', j')$ where $f(u) = (i, j)$ and $f(v) = (i', j')$; otherwise, it is* bad.

Intuitively, the following lemma asserts the existence of a solution (if any solution exists) that crosses different cells only via good edges, that is, via marked vertices.

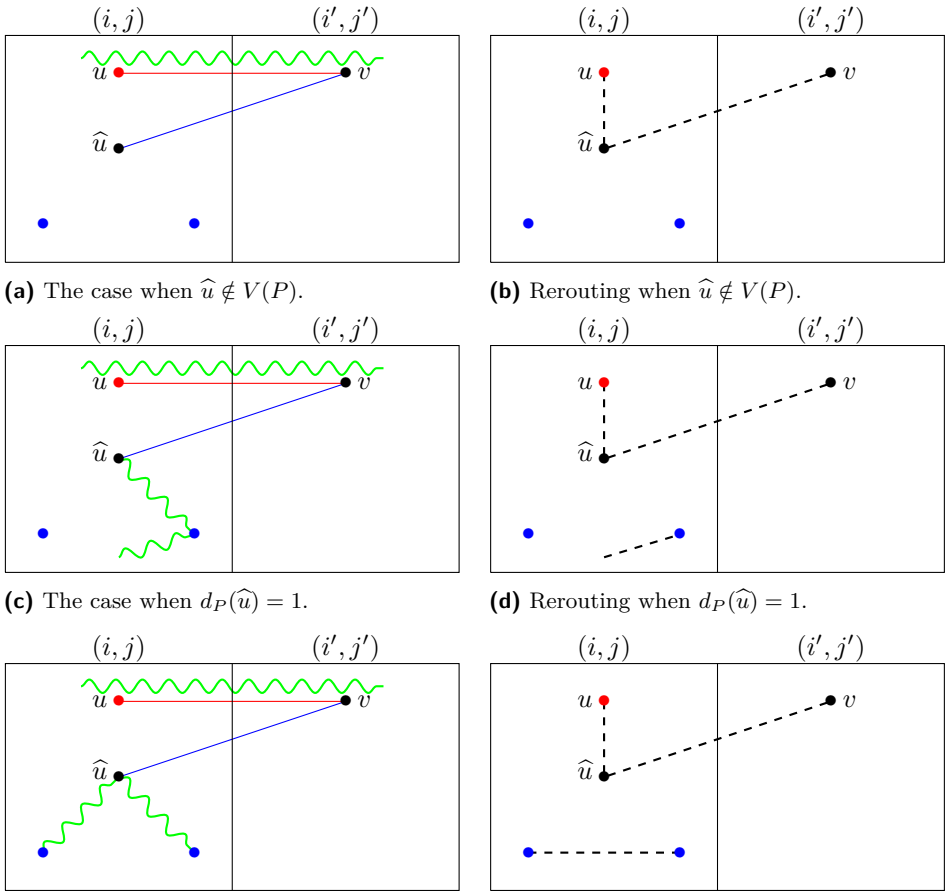
▶ **Lemma 9.** *Let G be a clique-grid with representation f that has a path (resp. cycle) on at least k vertices. Then, G also has a path (resp. cycle) P on at least k vertices with the following property: every edge $\{u, v\} \in E(P)$ where $f(u) \neq f(v)$ is a good edge.*

Proof. By Proposition 7, G has a path (resp. cycle) on at least k vertices with the following property: for every two distinct cells (i, j) and (i', j'), there exist at most 5 edges $\{u, v\} \in E(P)$ such that $f(u) = (i, j)$ and $f(v) = (i', j')$. Among all such paths (resp. cycles), let P be one that minimizes the number of bad edges. The following claim follows immediately from the choice of P and Property 2 in Definition 2.

▷ Claim 10. For each cell $(i, j) \in [t] \times [t]$, there are at most $24 \cdot 5 = 120$ vertices in $f^{-1}(i, j) \cap V(P)$ that are adjacent in P to at least one vertex that does not belong to $f^{-1}(i, j)$.

Next, we show that P has no bad edge, which will complete the proof. Targeting a contradiction, suppose that P has some bad edge $\{u, v\}$. By Definition 8, $u \notin \mathsf{Mark}^\star(i, j)$ or $v \notin \mathsf{Mark}^\star(i', j')$ (or both) where $f(u) = (i, j)$ and $f(v) = (i', j')$. Without loss of generality, suppose that $u \notin \mathsf{Mark}^\star(i, j)$. We consider two cases as follows.

Case I. First, suppose that $v \in \mathsf{Mark}_1(i', j')$. Because u is adjacent to v but it is not marked in the second phase, it must hold that $|\mathsf{Mark}_2(v, (i, j))| \geq 121$. By Claim 10, this means that there exists a vertex $\widehat{u} \in \mathsf{Mark}_2(v, (i, j))$ such the vertices incident to it on P – which might be 0 if $\widehat{u}$ does not belong to P, 1 if it is an endpoint of P or 2 if it is an internal vertex of P – also belong to $f^{-1}(i, j)$ (see Fig. 2). In case $\widehat{u} \notin V(P)$, denote $P_1 = P$. Else, by Property 1 in Definition 2, by removing $\widehat{u}$ from P, and if $\widehat{u}$ has two neighbors



(a) The case when $\widehat{u} \notin V(P)$.

(b) Rerouting when $\widehat{u} \notin V(P)$.

(c) The case when $d_P(\widehat{u}) = 1$.

(d) Rerouting when $d_P(\widehat{u}) = 1$.

(e) The case when $d_P(\widehat{u}) = 2$.

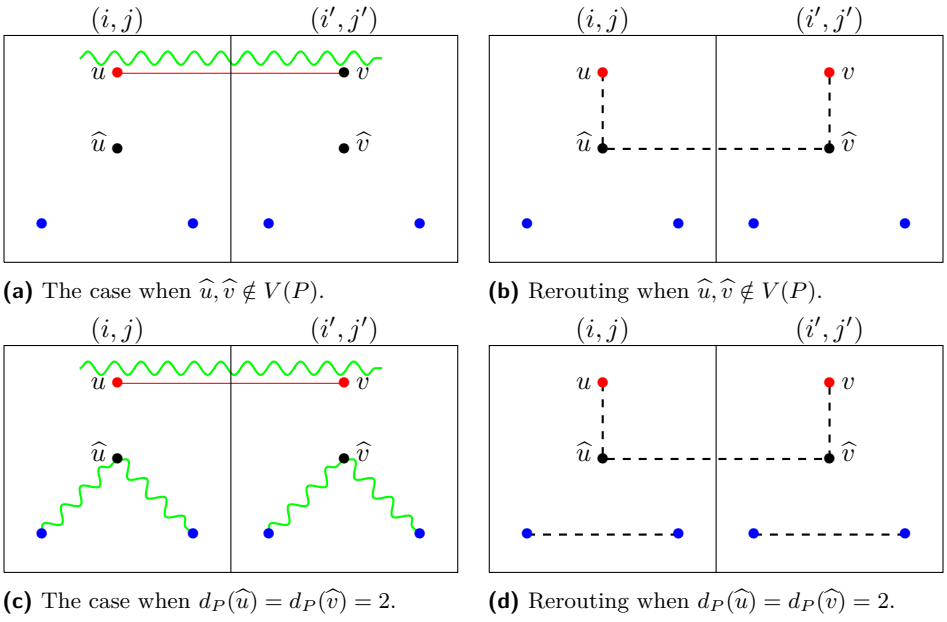
(f) Rerouting when $d_P(\widehat{u}) = 2$.

Figure 2 Case I in the proof of Lemma 9. Vertices colored black and red are marked and unmarked, respectively. Vertices colored blue are either marked or unmarked. Good and bad edges are colored blue and red, respectively. Curves colored green are part of the path P. Dashed lines are part of the path P_2.

on P, then also making these two neighbors adjacent,[1] we still have a path (resp. cycle) in G; we denote this path by P_1. Note that $|V(P_1)| \geq |V(P)| - 1$ and $u \notin V(P_1)$. Now, note that because $\widehat{u} \in \mathsf{Mark}_2(v, (i, j))$, we have that $\widehat{u}$ is adjacent to v in G and also $\widehat{u} \in f^{-1}(i, j)$. Because $u \in f^{-1}(i, j)$, Property 1 in Definition 2 implies that $\widehat{u}$ is also adjacent to u. Thus, by inserting $\widehat{u}$ between u and v in P_1 and making it adjacent to both, we still have a path (resp. cycle) in G, which we denote by P_2 (see Fig. 2). Note that $|V(P_2)| = |V(P_1)| + 1 \geq |V(P)| \geq k$. Moreover, the only edges that appear only in one among P_2 and P are as follows.

1. If $\widehat{u}$ has two neighbors in P, then the edges between $\widehat{u}$ and these two neighbors might belong only to P, and the edge between these two neighbors belongs only to P_2. As $\widehat{u}$ and its neighbors in P belong to the same cell (by the choice of $\widehat{u}$), none of these edges is bad, and also none of these edges crosses different cells.
2. If $\widehat{u}$ has only one neighbor in P, then the edge between $\widehat{u}$ and this neighbor might belong only to P.

[1] If $\widehat{u}$ is an endpoint of P, then only the removal of $\widehat{u}$ is performed.



(a) The case when $\widehat{u}, \widehat{v} \notin V(P)$.

(b) Rerouting when $\widehat{u}, \widehat{v} \notin V(P)$.

(c) The case when $d_P(\widehat{u}) = d_P(\widehat{v}) = 2$.

(d) Rerouting when $d_P(\widehat{u}) = d_P(\widehat{v}) = 2$.

Figure 3 Two subcases of Case II in the proof of Lemma 9. Other subcases are handled similarly to the subcases depicted here. Vertices colored black and red are marked and unmarked, respectively. Vertices colored blue are either marked or unmarked. Good and bad edges are colored blue and red, respectively. Curves colored green are part of the path P. Dashed lines are part of the path P_2.

3. $\{u, v\} \in E(P) \setminus E(P_2)$ is a bad edge that crosses different cells by its initial choice.
4. $\{u, \widehat{u}\}$ might belong only to P_2, and it is a neither a bad edge nor an edge that crosses different cells because u and $\widehat{u}$ belong to the same cell.
5. $\{\widehat{u}, v\} \in E(P_2) \setminus E(P)$ is a not a bad edge because both $\widehat{u}$ and v are marked (since $v \in \mathsf{Mark}_1(i', j')$ and $\widehat{u} \in \mathsf{Mark}_2(v, (i, j))$), but it crosses different cells.

Thus, P_2 has no bad edge that does not belong to P, and P has at least one bad edge that does not belong to P_2 (specifically, $\{u, v\}$), and therefore P_2 has fewer bad edges than P. Moreover, notice that the items above also imply that P_2 has at most one edge that crosses different cells and does not belong to P (specifically, $\{\widehat{u}, v\}$), and P has at least one edge that crosses the *same* cells and does not belong to P_2 (specifically, $\{u, v\}$). Therefore, P_2 also has the property of P that for every two distinct cells $(\widetilde{i}, \widetilde{j})$ and $(\widetilde{i}', \widetilde{j}')$, there exist at most 5 edges $\{\widetilde{u}, \widetilde{v}\} \in E(P_2)$ such that $f(\widetilde{u}) = (\widetilde{i}, \widetilde{j})$ and $f(\widetilde{v}) = (\widetilde{i}', \widetilde{j}')$. Therefore, we have reached a contradiction to the minimality of the number of bad edges in our choice of P.

Case II. Second, suppose that $v \notin \mathsf{Mark}^\star(i', j')$. Then, the addition of $\{u, v\}$ to $\mathsf{Mark}_1(\{(i, j), (i', j')\})$ maintains the property that it is a matching. Therefore, because this edge was not marked in the first phase, it must hold that $|\mathsf{Mark}_1(\{(i, j), (i', j')\})| = 241$. By Claim 10, there are at most 120 vertices in $f^{-1}(i, j) \cap V(P)$ that are adjacent in P to at least one vertex that does not belong to $f^{-1}(i, j)$, and notice that u (which is unmarked) is one of them. Similarly, there are at most 120 vertices in $f^{-1}(i', j') \cap V(P)$ that are adjacent in P to at least one vertex that does not belong to $f^{-1}(i', j')$, and notice that v (which is unmarked) is one of them. Therefore, because $\mathsf{Mark}_1(\{(i, j), (i', j')\})$ is a matching, it must contain at least one edge $\{\widehat{u}, \widehat{v}\}$ such that neither $\widehat{u}$ nor $\widehat{v}$ has a neighbor in P that belongs to a different cell than itself (see Fig. 3) – either because $\widehat{u}$ (and in the same way

$\widehat{v}$) does not belong to P, or it does and all its (one or two) neighbors belong to the same cell as itself. Define P'_1 as follows: if $\widehat{u}$ does not belong to P, then $P'_1 = P$, and otherwise let it be the graph obtained by removing $\widehat{u}$ from P and making its two neighbors (if both exist) adjacent. Because these two neighbors (if they exist) belong to the same cell, Property 1 in Definition 2 implies that P'_1 is a path (resp. cycle) in G. Similarly, let P_1 be the path (resp. cycle) obtained by the same operation with respect to P'_1 and $\widehat{v}$. Now, let P_2 be the graph obtained from P_1 by inserting $\widehat{u}$ and $\widehat{v}$ between u and v with the edges $\{u, \widehat{u}\}, \{\widehat{u}, \widehat{v}\}$ and $\{\widehat{v}, v\}$ (see Fig. 3). Because of Property 1 in Definition 2, and since u and $\widehat{u}$ belong to the same cell, they are adjacent in G. Similarly, v and $\widehat{v}$ are adjacent in G. Moreover, because $\{\widehat{u}, \widehat{v}\} \in \mathsf{Mark}_1(\{(i,j),(i',j')\})$, it is an edge in G. Thus, P_2 is a path (resp. cycle) in G. Additionally, $V(P) \subseteq V(P_2)$, and therefore $|V(P_2)| \geq k$. The only edges that appear only in one among P_2 and P are as follows.

1. If $\widehat{u}$ belongs to P and has two neighbors in P, then the edges between $\widehat{u}$ and these two neighbors might belong only to P, and the edge between these two neighbors belongs only to P_2. As $\widehat{u}$ and its neighbors in P belong to the same cell (by the choice of $\widehat{u}$), none of these edges is bad, and none of them crosses different cells. The same holds for $\widehat{v}$.
2. If $\widehat{u}$ belongs to P and has only one neighbor in P, the edge between $\widehat{u}$ and this neighbor might belong only to P. The same holds for $\widehat{v}$.
3. $\{u, v\} \in E(P) \setminus E(P_2)$ is a bad edge that crosses different cells by its initial choice.
4. $\{u, \widehat{u}\}$ might belong only to P_2, and it is a neither a bad edge nor it crosses different cells because u and $\widehat{u}$ belong to the same cell. The same holds for $\{v, \widehat{v}\}$.
5. $\{\widehat{u}, \widehat{v}\} \in E(P_2) \setminus E(P)$ is a not a bad edge because both $\widehat{u}$ and $\widehat{v}$ are marked (since $\{\widehat{u}, \widehat{v}\} \in \mathsf{Mark}_1(\{(i,j),(i',j')\})$), but it crosses different cells.

Thus, P_2 has no bad edge that does not belong to P, and P has at least one bad edge that does not belong to P_2 (specifically, $\{u, v\}$), and therefore P_2 has fewer bad edges than P. Moreover, notice that the items above also imply that P_2 has at most one edge that crosses different cells and does not belong to P (specifically, $\{\widehat{u}, \widehat{v}\}$), and P has at least one edge that crosses the *same* cells and does not belong to P_2 (specifically, $\{u, v\}$). Therefore, P_2 also has the property of P that for every two distinct cells $(\widetilde{i}, \widetilde{j})$ and $(\widetilde{i'}, \widetilde{j'})$, there exist at most 5 edges $\{\widetilde{u}, \widetilde{v}\} \in E(P_2)$ such that $f(\widetilde{u}) = (\widetilde{i}, \widetilde{j})$ and $f(\widetilde{v}) = (\widetilde{i'}, \widetilde{j'})$. Therefore, we have reached a contradiction to the minimality of the number of bad edges in our choice of P.

In both cases we have reached a contradiction, and therefore the proof is complete. ◀

Next, we further strengthen Lemma 9 with the following definition and Lemma 14. Intuitively, the following definition says that a cell is good with respect to some path if either none of its unmarked vertices is traversed by that path, or all of its unmarked vertices are traversed by that path consecutively and can be "flanked" only by marked vertices (see Fig. 4).

▶ **Definition 11.** *Let G be a clique-grid with representation f. Let P be a path (resp. cycle) in G with endpoints x, y (resp. no endpoints). We say that a cell $(i,j) \in [t] \times [t]$ is* good *if **(i)** $V(P) = f^{-1}(i,j) \setminus \mathsf{Mark}^\star(i,j)$, or **(ii)** $V(P) \cap (f^{-1}(i,j) \setminus \mathsf{Mark}^\star(i,j)) = \emptyset$, or **(iii)** there exist distinct $u, v \in (V(P) \cap \mathsf{Mark}^\star(i,j)) \cup (\{x,y\} \cap f^{-1}(i,j))$ (resp. not necessarily distinct $u, v \in V(P) \cap \mathsf{Mark}^\star(i,j)$) such that the set I of internal vertices of the (resp. a) subpath of P between u and v is precisely $f^{-1}(i,j) \setminus (\mathsf{Mark}^\star(i,j) \cup \{u,v\})$;[2] otherwise, it is* bad.

[2] In other words, $I \subseteq f^{-1}(i,j) \setminus \mathsf{Mark}^\star(i,j)$ and $(f^{-1}(i,j) \setminus \mathsf{Mark}^\star(i,j)) \setminus I$ can only include endpoints of

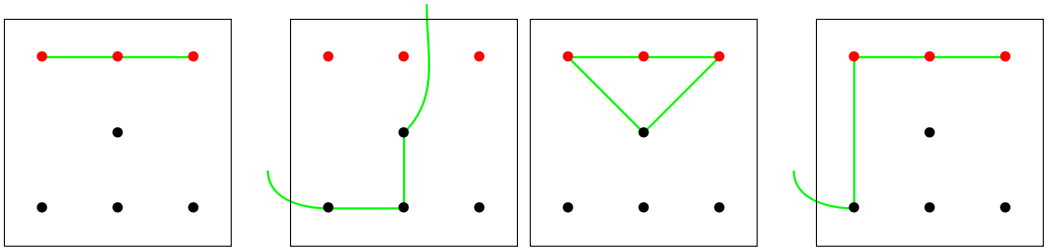


Figure 4 Illustration of good cells. Vertices colored black and red are marked and unmarked vertices, respectively. The green curve represents the path/cycle P.

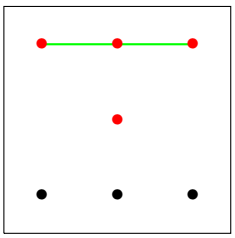


Figure 5 A nice cell which is not good. Vertices colored black and red are marked and unmarked vertices, respectively. The green curve represents the path P.

It will be convenient to have, as an intermediate step, a definition and lemma that are weaker than Definition 11 and Lemma 14. Intuitively, this definition drops the requirement that none or all the unmarked vertices of a cell should be visited by the path at hand, but only requires that those unmarked vertices that are visited, are visited consecutively and can be "flanked" only by marked vertices (see Fig. 5).

▶ **Definition 12.** *Let G be a clique-grid with representation f. Let P be a path (resp. cycle) in G with endpoints x, y (resp. no endpoints). We say that a cell $(i,j) \in [t] \times [t]$ is* nice *if **(i)** $V(P) \subseteq f^{-1}(i,j) \setminus \mathsf{Mark}^\star(i,j)$, or **(ii)** $V(P) \cap (f^{-1}(i,j) \setminus \mathsf{Mark}^\star(i,j)) = \emptyset$, or **(iii)** there exist distinct $u, v \in (V(P) \cap \mathsf{Mark}^\star(i,j)) \cup (\{x,y\} \cap f^{-1}(i,j))$ (resp. not necessarily distinct $u, v \in V(P) \cap \mathsf{Mark}^\star(i,j)$) such that the set of internal vertices of the (resp. a) subpath of P between u and v is precisely $(V(P) \cap f^{-1}(i,j)) \setminus (\mathsf{Mark}^\star(i,j) \cup \{u,v\})$.*

▶ **Lemma 13.** *Let G be a clique-grid with representation f that has a path (resp. cycle) on at least k vertices. Then, G also has a path (resp. cycle) P on at least k vertices with the following property: every cell $(i,j) \in [t] \times [t]$ is nice.*

Proof. Given a path (resp. cycle) P with endpoints x, y (resp. no endpoints) and a cell $(i,j) \in [t] \times [t]$, we say that a subpath of P is (i,j)-*nice* if it has distinct endpoints $u, v \in (V(P) \cap \mathsf{Mark}^\star(i,j)) \cup (\{x,y\} \cap f^{-1}(i,j))$ (resp. $u, v \in V(P) \cap \mathsf{Mark}^\star(i,j)$) and its set of internal vertices is a subset I of $f^{-1}(i,j) \setminus \mathsf{Mark}^\star(i,j)$ such that if this subset I is empty, then the subpath has an endpoint in $f^{-1}(i,j) \setminus \mathsf{Mark}^\star(i,j)$ (which implies that P is a path and $\{u,v\} \cap \{x,y\} \cap (f^{-1}(i,j) \setminus \mathsf{Mark}^\star(i,j)) \neq \emptyset$); we further say that a subpath of P is *nice* if it is (i,j)-nice for some (i,j). By Lemma 9, G has a path (resp. cycle) on at least k vertices with the following property: every edge $\{u,v\}$ of that path where $f(u) \neq f(v)$ is good. Among all such paths (resp. cycles), let P be one with minimum number of nice subpaths, and

this subpath, in which case P is a path and any included endpoint is an endpoint of P as well.

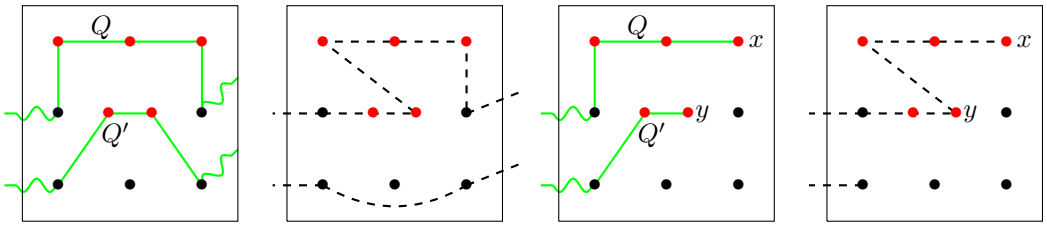


Figure 6 The proof of Lemma 13. Vertices colored black and red are the marked and unmarked vertices in the cell, respectively. In the first figure the union of internal vertices of Q and Q' is the set of unmarked vertices in the cell, and the second figure depicts how to reroute to make the cell nice. The third figure illustrate the case when both the endpoints x and y of the path P are in the cell, and the fourth figure depicts how to reroute to make the cell nice.

let x, y be its endpoints (resp. no endpoints). (Notice that if x is unmarked, then because every edge $\{u, v\}$ of P where $f(u) \neq f(v)$ is good, it must be that x is an endpoint of a nice subpath. The same holds for y.) We next show that for every cell $(i, j) \in [t] \times [t]$, P has at most one nice (i, j)-subpath. Because either $V(P) \subseteq f^{-1}(i, j)$ or every vertex in $(V(P) \cap f^{-1}(i, j)) \setminus (\mathsf{Mark}^\star(i, j) \cup \{x, y\})$ (resp. $(V(P) \cap f^{-1}(i, j)) \setminus \mathsf{Mark}^\star(i, j)$) must be an internal vertex of a nice subpath (since every edge $\{u, v\}$ of P where $f(u) \neq f(v)$ is good), this would imply that every cell $(i, j) \in [t] \times [t]$ is nice, which will complete the proof. Targeting a contradiction, suppose that P yields some cell (i, j) such that there exist two distinct subpaths Q, Q' of P that are (i, j)-nice (see Fig. 6), that is, each of them has both endpoints in $\mathsf{Mark}^\star(i, j) \cup (\{x, y\} \cap f^{-1}(i, j))$ (resp. $\mathsf{Mark}^\star(i, j)$) and the set of its internal vertices is a subset of $f^{-1}(i, j) \setminus \mathsf{Mark}^\star(i, j)$ that is either non-empty or some endpoint belongs to $\{x, y\} \cap (f^{-1}(i, j) \setminus \mathsf{Mark}^\star(i, j))$.

Note that if Q and Q' intersect, then they intersect only at their endpoints. Define $\widehat{P}$ by removing from P all the internal vertices of Q' as well as its endpoint in $f^{-1}(i, j) \setminus \mathsf{Mark}^\star(i, j)$ if such an endpoint exists (in which case P is a path and this endpoint it is also an endpoint of P), and inserting them arbitrarily between the vertices of Q (where multiple vertices can be inserted between two vertices); see Fig. 6. By Property 1 in Definition 2, we have that $\widehat{P}$ is also a path (resp. cycle). Clearly, $|V(\widehat{P})| = |V(P)| \geq k$, and it is also directly implied by the construction that $\widehat{P}$ also has the property that every edge $\{u, v\} \in E(\widehat{P})$ where $f(u) \neq f(v)$ is good (since we did not make any change with respect to the set of edges that cross different cells). Notice that each subpath that is nice with respect to $\widehat{P}$ is either the subpath obtained by merging Q and Q' or a subpath that also exists in P and is therefore also a nice subpath with respect to P. Therefore, $\widehat{P}$ has one fewer nice subpath than P, which contradicts the minimality of P. ◀

We now state the main lemma of this section, whose proof is given in the full version [34].

▶ **Lemma 14.** *Let G be a clique-grid with representation f that has a path (resp. cycle) on at least k vertices. Then, G also has a path (resp. cycle) P on at least k vertices with the following property: every cell $(i, j) \in [t] \times [t]$ is good.*

4 The Algorithm

Our algorithm is based on a reduction of Long Path (resp. Long Cycle) on unit disk graphs to the weighted version of the problem, called Weighted Long Path (resp. Weighted Long Cycle), on unit disk graphs of treewidth $\mathcal{O}(\sqrt{k})$. In Weighted Long Path

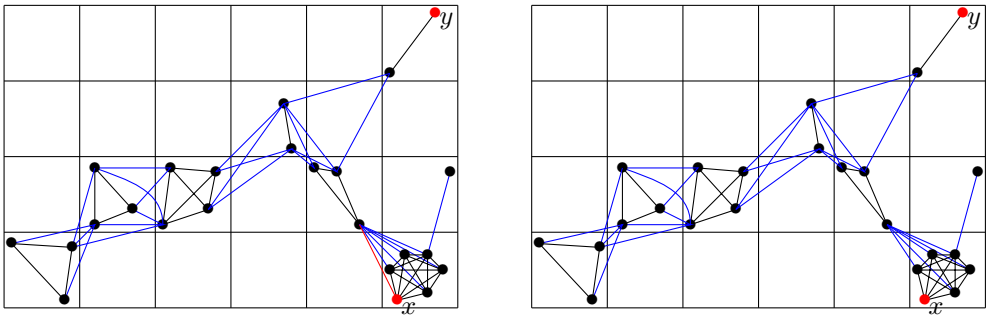


Figure 7 The graphs G' and $G^\star$ constructed from the graph G in Figure 1 are depicted on the left side and right side figures, respectively. Here, $w(x) = 2$, $w(y) = 3$, and for all $z \in V(G') \setminus \{x, y\}$, $w(z) = 1$.

(resp. Weighted Long Cycle), we are given a graph G with a weight function $w : V(G) \to \mathbb{N}$ and an integer $k \in \mathbb{N}$, and the objective is to determine whether G has a path (resp. cycle) whose weight, defined as the sum of the weights of its vertices, is at least k.

The following proposition will be immediately used in our algorithm.

▶ **Proposition 15** ([10, 29]). *Weighted Long Path and Weighted Long Cycle are solvable in time $2^{\mathcal{O}(\mathtt{tw})}n$ where $\mathtt{tw}$ is the treewidth of the input graph.*

Algorithm Specification. We call our algorithm ALG. Given an instance (G, k) of Long Path (resp. Long Cycle) on unit disk graphs, it works as follows.

1. Use Proposition 3 to obtain a representation $f : V(G) \to [t] \times [t]$ of G.
2. Use Observation 6 to compute $\mathsf{Mark}^\star(i, j)$ for every cell $(i, j) \in [t] \times [t]$. Let $\mathsf{Mark}^\star = \bigcup_{(i,j)\in[t]\times[t]} \mathsf{Mark}^\star(i, j)$.
3. Let G' be the graph defined as follows (see Fig. 7). For any cell $(i, j) \in [t] \times [t]$, let $c_{(i,j)}$ denote a vertex in $f^{-1}(i, j) \setminus \mathsf{Mark}^\star(i, j)$ (chosen arbitrarily), where if no such vertex exists, let $c_{(i,j)} = \mathtt{nil}$. Then, $V(G') = \mathsf{Mark}^\star \cup (\{c_{(i,j)} : (i, j) \in [t] \times [t]\} \setminus \{\mathtt{nil}\})$ and $E(G') = E(G[V(G')])$. Because G' is an induced subgraph of G, it is a unit disk graph.
4. Define $w : V(G') \to \mathbb{N}$ as follows. For every $v \in V(G')$, if $v = c_{(i,j)}$ for some $(i, j) \in [t]\times[t]$ then $w(v) = |f^{-1}(i, j) \setminus \mathsf{Mark}^\star(i, j)|$, and otherwise $w(v) = 1$.
5. Let $G^\star$ be the graph defined as follows (see Fig. 7): $V(G^\star) = V(G')$ and $E(G^\star) = E(G') \setminus \{\{c_{(i,j)}, v\} \in E(G') : (i, j) \in [t] \times [t], v \notin f^{-1}(i, j)\}$.
6. Let Δ be the maximum degree of $G^\star$. Use Proposition 4 to decide either $\mathtt{tw}(G^\star) > 100\Delta^3\sqrt{2k}$ or $\mathtt{tw}(G^\star) \leq 500\Delta^3\sqrt{2k}$.
7. If it was decided that $\mathtt{tw}(G^\star) > 100\Delta^3\sqrt{2k}$, then return Yes and terminate.
8. Use Proposition 15 to determine whether $(G^\star, w, k)$ is a Yes-instance of Weighted Long Path (resp. Weighted Long Cycle). If the answer is positive, then return Yes, and otherwise return No.

Analysis. We first analyze the running time of the algorithm.

▶ **Lemma 16.** *The time complexity of ALG is upper bounded by $2^{\mathcal{O}(\sqrt{k})}(n + m)$.*

Proof. By Proposition 3 and Observation 6, Steps 1 and 2 are performed in time $\mathcal{O}(n + m)$. By the definition of G', w and $G^\star$, they can clearly be computed in time $\mathcal{O}(n + m)$ as well (Steps 3, 4 and 5). Moreover, Step 7 is done in time $\mathcal{O}(1)$. By Proposition 4, Step 6 is

performed in time $2^{\mathcal{O}(100\Delta^3\sqrt{2k})}n = 2^{\mathcal{O}(\Delta^3\sqrt{k})}n$. Thus, because we reach Step 8 only if we do not terminate in Step 7, we have that by Proposition 15, Step 8 is performed in time $2^{\mathcal{O}(\mathtt{tw}(G^\star))}n = 2^{\mathcal{O}(500\Delta^3\sqrt{2k})} = 2^{\mathcal{O}(\Delta^3\sqrt{k})}n$.

Thus, to conclude the proof, it remains to show that $\Delta = \mathcal{O}(1)$. Let Δ' be the maximum degree of G'. Since $G^\star$ is a subgraph of G', $\Delta \leq \Delta'$. Thus, to prove $\Delta = \mathcal{O}(1)$, it is enough to prove that $\Delta' = \mathcal{O}(1)$. To this end, let $M = \max_{(i,j)\in[t]\times[t]} |(f^{-1}(i,j) \cap V(G')) \cup (\{c_{(i,j)}\} \setminus \{\mathtt{nil}\})|$. Since G' is a clique-grid, by Property 2 in Definition 2, we have that $\Delta' \leq M^{25}$, hence it suffices to show that $M = \mathcal{O}(1)$. The definition of G' yields that $M \leq \max_{(i,j)\in[t]\times[t]} |\mathsf{Mark}^\star(i,j)| + 1$. By Observation 6, $\max_{(i,j)\in[t]\times[t]} |\mathsf{Mark}^\star(i,j)| = \mathcal{O}(1)$, and therefore indeed $M = \mathcal{O}(1)$. ◀

Finally, we prove that the algorithm is correct.

▶ **Lemma 17.** *ALG solves Long Path and Long Cycle on unit disk graphs correctly.*

Proof. Let (G,k) be an instance of Long Path or Long Cycle on unit disk graphs. By the specification of the algorithm, to prove that it solves (G,k) correctly, it suffices to prove that the two following conditions are satisfied.

1. If $\mathtt{tw}(G^\star) > 100\Delta^3\sqrt{2k}$, then (G,k) is a Yes-instance of Long Path and Long Cycle.
2. (G,k) is a Yes-instance of Long Path (resp. Long Cycle) if and only if $(G^\star, w, k)$ is a Yes-instance of Weighted Long Path (resp. Weighted Long Cycle).

The proof of satisfaction of the first condition is simple and can be found in [34].

Now, we turn to prove the second condition. In one direction, suppose that (G,k) is a Yes-instance of Long Path (resp. Long Cycle). Then, by Lemma 14, G has a path (resp. cycle) P on at least k vertices with the following property: every cell $(i,j) \in [t]\times[t]$ is good. Notice that every maximal subpath Q of P that consists only of unmarked vertices satisfies *(i)* $V(Q) = f^{-1}(i_Q, j_Q) \setminus \mathsf{Mark}^\star(i_Q, j_Q)$ for some cell $(i_Q, j_Q) \in [t]\times[t]$, and *(ii)* the endpoints of Q are adjacent in P to vertices in $f^{-1}(i_Q, j_Q)$ (unless $Q = P$). Obtain $P^\star$ from P as follows: every maximal subpath Q of P that consists only of unmarked vertices is replaced by $c_{(i_Q,j_Q)}$. (Notice that $c_{(i_Q,j_Q)} \neq \mathtt{nil}$ because $V(Q) \neq \emptyset$.) Because of Property *(ii)* above and Property 1 in Definition 2, we immediately have that $P^\star$ is a path (resp. cycle) in $G^\star$. Moreover, by Property *(i)* above and the definition of the weight function w (in Step 4), each subpath Q is replaced by a vertex $c_{(i_Q,j_Q)}$ whose weight equals $|V(Q)|$. Because $|V(P)| \geq k$, we have that $P^\star$ is a path (resp. cycle) of weight at least k in $G^\star$. Thus, $(G^\star, w, k)$ is a Yes-instance of Weighted Long Path (resp. Weighted Long Cycle).

In the other direction, suppose that $(G^\star, w, k)$ is a Yes-instance of Weighted Long Path (resp. Weighted Long Cycle). Then, $G^\star$ has a path (resp. cycle) $P^\star$ of weight at least k. Obtain P from $P^\star$ by replacing each vertex of the form $c_{(i,j)} \in V(P)$ for some $(i,j) \in [t]\times[t]$ by a path Q whose vertex set is $f^{-1}(i,j) \setminus \mathsf{Mark}^\star(i,j)$ (the precise ordering of the vertices on this path is arbitrary). Notice that because all edges in $\{\{c_{(i,j)}, v\} \in E(G') : (i,j) \in [t]\times[t], v \notin f^{-1}(i,j)\}$ were removed from G' to derive $G^\star$, each vertex of the form $c_{(i,j)} \in V(P)$ for some $(i,j) \in [t]\times[t]$ is adjacent in $P^\star$ only to vertices in $\mathsf{Mark}^\star(i,j)$. Therefore, by Property 1 in Definition 2, we have that P is a path (resp. cycle) in G. Moreover, by the definition of the weight function w (in Step 4), each vertex $c_{(i,j)}$ was replaced by $w(c_{(i,j)})$ vertices. Because the weight of $P^\star$ is at least k, we have that P is a path (resp. cycle) on at least k vertices in G. Thus, (G,k) is a Yes-instance of Long Path (resp. Long Cycle). ◀

Thus, Theorem 1 follows from Lemmas 16 and 17.

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 Noga Alon, Phuong Dao, Iman Hajirasouliha, Fereydoun Hormozdiari, and Süleyman Cenk Sahinalp. Biomolecular network motif counting and discovery by color coding. In *Proceedings 16th International Conference on Intelligent Systems for Molecular Biology (ISMB), Toronto, Canada, July 19-23, 2008*, pages 241–249, 2008. doi:10.1093/bioinformatics/btn163.

3 Noga Alon and Shai Gutner. Balanced hashing, color coding and approximate counting. In *Parameterized and Exact Computation, 4th International Workshop, IWPEC 2009, Copenhagen, Denmark, September 10-11, 2009, Revised Selected Papers*, pages 1–16, 2009. doi:10.1007/978-3-642-11269-0_1.

4 Noga Alon and Shai Gutner. Balanced families of perfect hash functions and their applications. *ACM Trans. Algorithms*, 6(3):54:1–54:12, 2010. doi:10.1145/1798596.1798607.

5 Noga Alon, Raphael Yuster, and Uri Zwick. Color-coding. *J. Assoc. Comput. Mach.*, 42(4):844–856, 1995.

6 Vikraman Arvind and Venkatesh Raman. Approximation algorithms for some parameterized counting problems. In *Algorithms and Computation, 13th International Symposium, ISAAC 2002 Vancouver, BC, Canada, November 21-23, 2002, Proceedings*, pages 453–464, 2002. doi:10.1007/3-540-36136-7_40.

7 Ivona Bezáková, Radu Curticapean, Holger Dell, and Fedor V. Fomin. Finding detours is fixed-parameter tractable. In *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, pages 54:1–54:14, 2017.

8 Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Narrow sieves for parameterized paths and packings. *J. Comput. Syst. Sci.*, 87:119–139, 2017.

9 Andreas Björklund, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. Approximate counting of k-paths: Deterministic and in polynomial space. In *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece*, pages 24:1–24:15, 2019.

10 Hans L. Bodlaender, Marek Cygan, Stefan Kratsch, and Jesper Nederlof. Deterministic single exponential time algorithms for connectivity problems parameterized by treewidth. *Inf. Comput.*, 243:86–111, 2015. doi:10.1016/j.ic.2014.12.008.

11 Hans L Bodlaender, Rodney G Downey, Michael R Fellows, and Danny Hermelin. On problems without polynomial kernels. *Journal of Computer and System Sciences*, 75(8):423–434, 2009.

12 Hans L. Bodlaender, Pål Grønås Drange, Markus S. Dregi, Fedor V. Fomin, Daniel Lokshtanov, and Michal Pilipczuk. A $c^k n$ 5-approximation algorithm for treewidth. *SIAM J. Comput.*, 45(2):317–378, 2016.

13 Cornelius Brand, Holger Dell, and Thore Husfeldt. Extensor-coding. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 151–164, 2018. doi:10.1145/3188745.3188902.

14 Heinz Breu and David G Kirkpatrick. Unit disk graph recognition is np-hard. *Computational Geometry*, 9(1-2):3–24, 1998.

15 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.

16 Jianer Chen, Joachim Kneis, Songjian Lu, Daniel Mölle, Stefan Richter, Peter Rossmanith, Sing-Hoi Sze, and Fenghui Zhang. Randomized divide-and-conquer: Improved path, matching, and packing algorithms. *SIAM Journal on Computing*, 38(6):2526–2547, 2009.

17 Brent N. Clark, Charles J. Colbourn, and David S. Johnson. Unit disk graphs. *Discrete Mathematics*, 86(1-3):165–177, 1990.

18 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.

19 Marek Cygan, Jesper Nederlof, Marcin Pilipczuk, Michal Pilipczuk, Johan M. M. van Rooij, and Jakub Onufry Wojtaszczyk. Solving connectivity problems parameterized by treewidth in single exponential time. In *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 150–159, 2011.

20 Mark de Berg, Hans L. Bodlaender, Sándor Kisfaludi-Bak, Dániel Marx, and Tom C. van der Zanden. A framework for eth-tight algorithms and lower bounds in geometric intersection graphs. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 574–586, 2018.

21 Erik D. Demaine, Fedor V. Fomin, Mohammadtaghi Hajiaghayi, and Dimitrios M. Thilikos. Subexponential parameterized algorithms on graphs of bounded genus and H-minor-free graphs. *jacm*, 52(6):866–893, 2005.

22 Erik D. Demaine and MohammadTaghi Hajiaghayi. The bidimensionality theory and its algorithmic applications. *Comput. J.*, 51(3):292–302, 2008. doi:10.1093/comjnl/bxm033.

23 Frederic Dorn, Eelko Penninkx, Hans L. Bodlaender, and Fedor V. Fomin. Efficient exact algorithms on planar graphs: Exploiting sphere cut decompositions. *Algorithmica*, 58(3):790–810, 2010. doi:10.1007/s00453-009-9296-1.

24 Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013.

25 Adrian Dumitrescu and János Pach. Minimum clique partition in unit disk graphs. *Graphs and Combinatorics*, 27(3):399–411, 2011.

26 Jörg Flum and Martin Grohe. The parameterized complexity of counting problems. *SIAM J. Comput.*, 33(4):892–922, 2004.

27 Fedor V. Fomin, Petr A. Golovach, Daniel Lokshtanov, Fahad Panolan, Saket Saurabh, and Meirav Zehavi. Going far from degeneracy. In *27th Annual European Symposium on Algorithms, ESA 2019, September 9-11, 2019, Munich/Garching, Germany*, pages 47:1–47:14, 2019.

28 Fedor V. Fomin, Petr A. Golovach, Daniel Lokshtanov, Fahad Panolan, Saket Saurabh, and Meirav Zehavi. Parameterization above a multiplicative guarantee. In *11th Innovations in Theoretical Computer Science, ITCS 2020 (To Appear)*, 2020.

29 Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, and Saket Saurabh. Efficient computation of representative families with applications in parameterized and exact algorithms. *J. ACM*, 63(4):29:1–29:60, 2016.

30 Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, and Saket Saurabh. Efficient computation of representative families with applications in parameterized and exact algorithms. *jacm*, 63(4):29, 2016. doi:10.1145/2886094.

31 Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, Saket Saurabh, and Meirav Zehavi. Long directed (s, t)-path: FPT algorithm. *Inf. Process. Lett.*, 140:8–12, 2018.

32 Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, Saket Saurabh, and Meirav Zehavi. Decomposition of map graphs with applications. In *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece.*, pages 60:1–60:15, 2019.

33 Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, Saket Saurabh, and Meirav Zehavi. Finding, hitting and packing cycles in subexponential time on unit disk graphs. *Discrete & Computational Geometry*, 62(4):879–911, 2019.

34 Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, Saket Saurabh, and Meirav Zehavi. Eth-tight algorithms for long path and cycle on unit disk graphs. *CoRR*, abs/2003.00938, 2020. arXiv:2003.00938.

35 Fedor V. Fomin, Daniel Lokshtanov, and Saket Saurabh. Bidimensionality and geometric graphs. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 1563–1575, 2012.

36 Harold N Gabow and Shuxin Nie. Finding a long directed cycle. *ACM Transactions on Algorithms (TALG)*, 4(1):7, 2008.

37 William K Hale. Frequency assignment: Theory and applications. *Proceedings of the IEEE*, 68(12):1497–1514, 1980.

38 Danny Hermelin, Stefan Kratsch, Karolina Soltys, Magnus Wahlström, and Xi Wu. A completeness theory for polynomial (turing) kernelization. *Algorithmica*, 71(3):702–730, 2015. doi:10.1007/s00453-014-9910-8.

39 Falk Hüffner, Sebastian Wernicke, and Thomas Zichner. Algorithm engineering for color-coding with applications to signaling pathway detection. *Algorithmica*, 52(2):114–132, 2008.

40 Harry B. Hunt III, Madhav V. Marathe, Venkatesh Radhakrishnan, S. S. Ravi, Daniel J. Rosenkrantz, and Richard Edwin Stearns. NC-approximation schemes for NP- and PSPACE-hard problems for geometric graphs. *J. Algorithms*, 26(2):238–274, 1998.

41 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity. *Journal of Computer and System Sciences*, 63(4):512–530, 2001.

42 Alon Itai, Christos H. Papadimitriou, and Jayme Luiz Szwarcfiter. Hamilton paths in grid graphs. *SIAM J. Comput.*, 11(4):676–686, 1982. doi:10.1137/0211056.

43 Hiro Ito and Masakazu Kadoshita. Tractability and intractability of problems on unit disk graphs parameterized by domain area. In *Proceedings of the 9th International Symposium on Operations Research and Its Applications (ISORA10)*, pages 120–127, 2010.

44 Bart M. P. Jansen. Polynomial kernels for hard problems on disk graphs. In *Proceedings of the 12th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT)*, volume 6139, pages 310–321. Springer, 2010.

45 Bart M. P. Jansen, László Kozma, and Jesper Nederlof. Hamiltonicity below dirac's condition. In *Graph-Theoretic Concepts in Computer Science - 45th International Workshop, WG 2019, Vall de Núria, Spain, June 19-21, 2019, Revised Papers*, pages 27–39, 2019.

46 Bart M. P. Jansen, Marcin Pilipczuk, and Marcin Wrochna. Turing kernelization for finding long paths in graph classes excluding a topological minor. *Algorithmica*, 81(10):3936–3967, 2019.

47 Karl Kammerlander. C 900-an advanced mobile radio telephone system with optimum frequency utilization. *IEEE journal on selected areas in communications*, 2(4):589–597, 1984.

48 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.

49 Ioannis Koutis. Faster algebraic algorithms for path and packing problems. In *Proceedings of the 35th International Colloquium on Automata, Languages and Programming (ICALP 2008)*, volume 5125 of *Lecture Notes in Computer Science*, pages 575–586, 2008.

50 Ioannis Koutis and Ryan Williams. Algebraic fingerprints for faster algorithms. *Commun. ACM*, 59(1):98–105, 2016. doi:10.1145/2742544.

51 Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Slightly superexponential parameterized problems. *SIAM J. Comput.*, 47(3):675–702, 2018.

52 Burkhard Monien. How to find long paths efficiently. In *North-Holland Mathematics Studies*, volume 109, pages 239–254. Elsevier, 1985.

53 George L Nemhauser and Leslie Earl Trotter. Vertex packings: structural properties and algorithms. *Mathematical Programming*, 8(1):232–248, 1975.

54 Fahad Panolan, Saket Saurabh, and Meirav Zehavi. Contraction decomposition in unit disk graphs and algorithmic applications in parameterized complexity. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 1035–1054, 2019.

55 Christos H. Papadimitriou and Mihalis Yannakakis. On limited nondeterminism and the complexity of the V.C dimension (extended abstract). In *Proceedings of the Eigth Annual Structure in Complexity Theory Conference, San Diego, CA, USA, May 18-21, 1993*, pages 12–18, 1993.

56 Hadas Shachnai and Meirav Zehavi. Representative families: A unified tradeoff-based approach. *J. Comput. Syst. Sci.*, 82(3):488–502, 2016.

57 Warren D. Smith and Nicholas C. Wormald. Geometric separator theorems & applications. In *Proceedings of the 39th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 232–243. IEEE Computer Society, 1998.

58 Dekel Tsur. Faster deterministic parameterized algorithm for k-path. *Theor. Comput. Sci.*, 790:96–104, 2019.

59 DW Wang and Yue-Sun Kuo. A study on two geometric location problems. *Information processing letters*, 28(6):281–286, 1988.

60 Ryan Williams. Finding paths of length k in $O^*(2^k)$ time. *Inf. Process. Lett.*, 109(6):315–318, 2009.

61 Yu-Shuan Yeh, J Wilson, and S Schwartz. Outage probability in mobile telephony with directive antennas and macrodiversity. *IEEE journal on selected areas in communications*, 2(4):507–511, 1984.

62 Meirav Zehavi. Mixing color coding-related techniques. In *Algorithms - ESA 2015 - 23rd Annual European Symposium, Patras, Greece, September 14-16, 2015, Proceedings*, pages 1037–1049, 2015. doi:10.1007/978-3-662-48350-3_86.

63 Meirav Zehavi. A randomized algorithm for long directed cycle. *Inf. Process. Lett.*, 116(6):419–422, 2016.

A Near-Linear Time Approximation Scheme for Geometric Transportation with Arbitrary Supplies and Spread

Kyle Fox
Department of Computer Science, The University of Texas at Dallas, TX, USA
kyle.fox@utdallas.edu

Jiashuai Lu
Department of Computer Science, The University of Texas at Dallas, TX, USA
jiashuai.lu@utdallas.edu

Abstract

The geometric transportation problem takes as input a set of points P in d-dimensional Euclidean space and a supply function $\mu : P \to \mathbb{R}$. The goal is to find a transportation map, a non-negative assignment $\tau : P \times P \to \mathbb{R}_{\geq 0}$ to pairs of points, so the total assignment leaving each point is equal to its supply, i.e., $\sum_{r \in P} \tau(q, r) - \sum_{p \in P} \tau(p, q) = \mu(q)$ for all points $q \in P$. The goal is to minimize the weighted sum of Euclidean distances for the pairs, $\sum_{(p,q) \in P \times P} \tau(p, q) \cdot ||q - p||_2$.

We describe the first algorithm for this problem that returns, with high probability, a $(1 + \varepsilon)$-approximation to the optimal transportation map in $O(n \operatorname{poly}(1/\varepsilon) \operatorname{polylog} n)$ time. In contrast to the previous best algorithms for this problem, our near-linear running time bound is independent of the spread of P and the magnitude of its real-valued supplies.

2012 ACM Subject Classification Theory of computation → Computational geometry; Theory of computation → Network flows

Keywords and phrases Transportation map, earth mover's distance, shape matching, approximation algorithms

Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.45

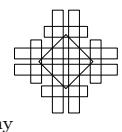
Related Version The full version of the paper is available at `https://arxiv.org/abs/1907.04426`.

Acknowledgements The authors would like to thank Hsien-Chih Chang for some helpful discussions that took place with the first author at Dagstuhl seminar 19181 "Computational Geometry". We would also like to thank the anonymous reviewers for many helpful comments and suggestions.

1 Introduction

We consider the **geometric transportation problem** in d-dimensional Euclidean space for any constant d. In this problem, we are given a set $P \subset \mathbb{R}^d$ of n points. Each point is assigned a real **supply** $\mu : P \to \mathbb{R}$ where $\sum_{p \in P} \mu(p) = 0$. A **transportation map** is a non-negative assignment $\tau : P \times P \to \mathbb{R}_{\geq 0}$ to pairs of points such that for all $q \in P$ we have $\sum_{r \in P} \tau(q, r) - \sum_{p \in P} \tau(p, q) = \mu(q)$. The **cost** of the transportation map is the weighted sum of Euclidean distances across all pairs, i.e. $\sum_{(p,q) \in P \times P} \tau(p, q) \cdot ||q - p||_2$. Our goal is to find a transportation map of minimum cost, and we denote this minimum cost as $\textsc{Cost}(P, \mu)$.

One may imagine the points with positive supply as piles of earth and those with negative supplies as holes in the ground. A transportation map describes how to transfer the earth to the holes without overfilling any hole, and its cost is the total number of "earth-miles" used to do the transfer. Consequently, $\textsc{Cost}(P, \mu)$ is often referred to as the earth mover's distance, although it can also be called the 1-Wasserstein distance between measures over the positively and negatively supplied points. The continuous version of the problem is



sometimes called the *optimal transport* or *Monge-Kantorovich* problem, and it has been studied extensively by various mathematics communities [22]. The discrete version we study here has applications in shape matching, image retrieval, and graphics [5, 7–9, 16, 21].

Computing an optimal transportation map is easily done in polynomial time by reduction to the uncapacitated minimum cost flow problem in a complete bipartite graph between points with positive supply and those with negative supply. The graph has as many as $\Omega(n^2)$ edges, so this approach takes $O(n^3 \operatorname{polylog} n)$ time using a combinatorial minimum cost flow algorithm of Orlin [15]. Assuming integral supplies with absolute values summing to U, we can use an algorithm of Lee and Sidford [14] instead to reduce the running time to $O(n^{2.5} \operatorname{polylog}(n, U))$. Taking advantage of the geometry inherent in the problem, Agarwal et al. [1] describe how to implement Orlin's algorithm for arbitrary supplies to find the optimal transportation map in $O(n^2 \operatorname{polylog} n)$ time, but only for $d = 2$.

We can significantly reduce these running times by accepting a small loss in optimality. Many results along this line focus on estimating just the earth mover's distance without actually computing the associated transportation map. Indyk [11] describes an $O(n \operatorname{polylog} n)$ time algorithm that estimates the earth mover's distance within a constant factor assuming unit supplies. Andoni et al. [3] describe an $O(n^{1+o(1)})$ time algorithm for arbitrary supplies that estimates the cost within a $1 + \varepsilon$ factor (the dependency on ε is hiding in the $o(1)$). As pointed out by Khesin, Nikolov, and Paramonov [12], a $1 + \varepsilon$ factor estimation of the distance is possible in $O(n^{1+o(1)}\varepsilon^{-O(d)})$ time (without the $o(1)$ hiding dependencies on ε) by running an approximation algorithm for minimum cost flow by Sherman [19] on a sparse Euclidean spanner over the input points. However, it is not clear how to extract a nearly optimal transportation map using the spanner's flow.

Finding an actual transportation *map* may be more difficult. Sharathkumar and Agarwal [17] describe a $(1 + \varepsilon)$-approximation algorithm for the integral supply case (i.e., an algorithm returning a map of cost at most $(1 + \varepsilon) \cdot \textsc{Cost}(P, \mu)$) in $O(n\sqrt{U} \operatorname{polylog}(U, \varepsilon, n))$ time. Agarwal et al. [1] describe a randomized algorithm with expected $O(\log^2(1/\varepsilon))$-approximation ratio running in $O(n^{1+\varepsilon})$ expected time for the arbitrary supply case and a deterministic $O(n^{3/2}\varepsilon^{-d} \operatorname{polylog}(U, n))$ time $(1+\varepsilon)$-approximation algorithm for the bounded integral supply case. Lahn et al. [13] describe a $O(n(C\delta)^2 \operatorname{polylog}(U, n))$ ($C = \max_{p \in P} |\mu(p)|$) time algorithm computing a map of cost at most $\textsc{Cost}(P, \mu) + \delta U$. In last year's SoCG proceedings, Khesin et al. [12] described a randomized $(1 + \varepsilon)$-approximation algorithm for the arbitrary supply case running in $O(n\varepsilon^{-O(d)} \log^{O(d)}(\textsc{Sp}(P)) \log(n))$ time, where $\textsc{Sp}(P)$ is the spread of the point set P.[1] The *spread* (also called aspect ratio) of P is the ratio of the diameter of P to the smallest pairwise distance between points in P. As Khesin et al. point out, one can reduce an instance with unbounded spread but bounded integral supplies to the case of bounded spread to get a $(1 + \varepsilon)$-approximation running in $O(n\varepsilon^{-O(d)} \log^{O(d)}(U) \log^2(n))$ time, generalizing a near-linear time $(1 + \varepsilon)$-approximation algorithm by Sharathkumar and Agarwal [18] for the unit supply case. The unit supply case is sometimes referred to as the *geometric bipartite matching problem*. Agarwal and Sharathkumar [2] also describe a *deterministic* $(1/\varepsilon)$-approximation algorithm for geometric bipartite matching that runs in $O(n^{1+\varepsilon} \log n)$ time.

Despite these successes, prior work still does not include a near-linear time $(1 + \varepsilon)$-approximation algorithm for the general case of arbitrary spread and real valued supplies. Often, an algorithm designed for bounded spread cases can be extended to work with cases of

[1] Khesin et al. [12] and Agarwal et al. [1] present geometric transportation with integer supplies, but their unbounded supply algorithms work without modification when presented with real valued supplies.

arbitrary spread. For example, one might substitute in *compressed quadtrees* [10, Chapter 2] in places where the bounded spread algorithm uses standard (uncompressed) quadtrees. This straightforward approach does not appear to work for the geometric transportation problem, however. As detailed below, Khesin et al. [12] use a quadtree to build a sparse graph as part of a reduction to the minimum cost flow problem. Both their running time and approximation analysis rely heavily on the tree having low depth when the spread is bounded. Unfortunately, a compressed quadtree is only guaranteed to have small *size*; the depth can still be linear in the number of leaves. One may also try the strategy of separating out groups of points P' that are much closer to each other than to the rest of the point set P, routing as much supply as possible within P', and then satisfying what remains of the supplies in P' by treating P' as a single point. In fact, the result described below does employ a variant of this strategy (see Section 2.3). However, the simplified instances of the problem one gets using this strategy still yield compressed quadtrees of very high depth.

1.1 Our results and approach

We describe a randomized $(1+\varepsilon)$-approximation algorithm for the geometric transportation problem that runs in near-linear time irrespective of the spread of P or the supplies of its points. Our specific result is spelled out in the following theorem. We say an event occurs with *high probability* if it occurs with probability at least $1 - 1/n^c$ for some constant c.

▶ **Theorem 1.1.** *There exists a randomized algorithm that, given a set of n points $P \in \mathbb{R}^d$ and a supply function $\mu : P \to \mathbb{R}$, runs in time $O(n\varepsilon^{-O(d)} \log^{O(d)} n)$ and with high probability returns a transportation map with cost at most $(1+\varepsilon) \cdot \textsc{Cost}(P, \mu)$.*

At a high level, our algorithm follows the approach laid out by Khesin et al. [12] for the bounded spread case. However, removing the running time's dependency on the spread introduces fundamental and technical issues to nearly every step in their approach.

Let ε_0 be a function of ε and P to be specified later. Taking a cue from prior work on geometric transportation and its specializations [3, 18], Khesin et al.'s algorithm begins by building a random sparse graph over $O(n\varepsilon_0^{-O(d)} \log \textsc{Sp}(P))$ vertices including the points in P. In expectation, the shortest path distance between any pair of points in P is maintained up to an $O(\varepsilon_0 \log \textsc{Sp}(P))$ factor, so computing a transportation map is done by setting ε_0 to $O(\varepsilon/(\log \textsc{Sp}(P))$ and running a minimum cost flow algorithm on the sparse graph.

The graph is constructed by first building a randomly shifted quadtree over P. The quadtree is constructed by surrounding P with an axis-aligned box called a cell, partitioning it into 2^d equal sized child cells, and recursively building a quadtree in each child cell; the whole tree has depth $\log \textsc{Sp}(P)$. After building the quadtree, they add ε_0^d Steiner vertices within each cell along with a carefully selected set of edges. While other methods are known for constructing such a sparse graph even without Steiner vertices [6], the hierarchical structure of Khesin et al.'s construction is necessary for extracting the transportation map after a minimum cost flow is computed. Observe that not only is the quadtree's size dependent on $\textsc{Sp}(P)$, but so is the number of Steiner vertices added to each cell.

As suggested earlier, the natural approach for reducing the quadtree's size is to remove subtrees containing no members of P and to *compress* the tree by replacing each maximal path of cells with exactly one non-empty child each with a single link to the lowest cell in the path. This approach does result in a quadtree of size $O(n)$, but its depth could also be as large as $\Omega(n)$. This large depth introduces many issues, the worst of which is that we can only claim shortest path distances to be maintained up to an $O(\varepsilon_0 n)$ factor. We cannot afford to set ε_0 to ε/n, because the sparse graph would have $O(n^d)$ vertices!

The solution to avoiding such a large increase in expected distances is to use the idea of *moats* around the points as done in the almost-linear time constant factor approximation algorithm of Agarwal et al. [1]. In short, we modify the quadtree construction so that, with high probability, all points are sufficiently far away from the boundary of every quadtree cell they appear in. Assuming this condition holds, there are only a limited number of quadtree "levels" at which a pair of points can be separated, and we use this fact to show distances increase by only an $O(\varepsilon_0 \log n)$ factor in expectation. It turns out modifying the quadtree construction correctly is a surprisingly subtle task. Guaranteeing the moats are avoided potentially requires us to perform independent random shifts at several places throughout the quadtree. However, we need to be selective with where the independent shifts occur so that we can successfully analyze the expected distances between points in the sparse graph.

The second stage of Khesin et al.'s [12] algorithm solves the minimum cost flow problem in the sparse graph using a framework of Sherman [19]. First, they encode the minimum cost flow problem as finding a flow vector f of minimum cost subject to linear constraints $Af = b$ where A is the vertex-edge incidence matrix and b is a supply vector (not necessarily equal to μ). Sherman's framework involves repeatedly finding flows f of approximately optimal cost that approximately satisfy such constraints. Each iteration of this algorithm requires an application of A and A^T to a pair of vectors, and the number of iterations needed in this approach is polynomial in the *condition number* of A. Unfortunately, A may not be well-conditioned, so Khesin et al. describe a *preconditioner* matrix B such that BA has low condition number and is still sparse. They proceed to use Sherman's framework under the equivalent constraints $BAf = Bb$.

One interpretation of Khesin et al.'s [12] preconditioner is that it describes a way to charge each Steiner vertex an amount based on the supply of "descendent" vertices below it so that the sum of charges bound the cost of an optimal flow from below. Consequently, both the number of non-zero entries in each column of B and the condition number of B are proportional to the quadtree's depth.

The high depth of our quadtree again appears to cause issues. However, our use of moats implies additional structure to the sparse graph that we can take advantage of. Our preconditioner B is based on essentially the same charging scheme as Khesin et al., but thanks to the moats, we prove the condition number remains proportional to $O(\varepsilon_0^{-1} \log(n/\varepsilon_0))$ instead of the quadtree depth. This charging scheme still results in a precondition B that *is not* sparse, so a naive implementation of Sherman's [19] framework may take quadratic time per iteration. To address this issue, we describe a pair of algorithms based on the hierarchical structure of the graph that let us apply both BA and its transpose in only linear time.

The final stage of the algorithm is the extraction of an approximately minimum cost transportation map from an approximately minimum cost flow in the sparse graph. Khesin et al.'s [12]'s original procedure modifies the graph's flow by iteratively reassigning flow to travel directly from input points to each of their many ancestor Steiner vertices or vice versa. We use binary search tree based data structures in a novel way to do flow reassignments in bulk, allowing us to extract the transportation map in time near-linear in the graph size.

Our result relies on a computation model where powers of 2, base 2 logarithms, floors, and the first non-zero bit of arbitrary real numbers can be computed in constant (or at least polylogarithmic) time. These are standard operations when working with quadtrees (see Bern et al. [4] and Har-Peled [10, Chapter 2]) and are only used so we may quickly compute the location of points within arbitrary grids. In particular, we perform only additions and multiplications when working with values derived from distances and supplies. Our results (and those of Khesin et al. [12]) can be extended to work with any L_p metric instead of just

Euclidean distance. The rest of the paper proceeds as follows. We describe our sparse graph construction and describe the reduction to minimum cost flow in Section 2. We describe our preconditioner and its use Section 3. Finally, we describe how to extract the approximately optimal transportation map from a flow on the sparse graph in Section 4.

2 Reduction to minimum cost flow in a sparse graph

In this section, we present a way to build a sparse graph $G^* = (V^*, E^*)$ based on P and reduce the transportation problem to finding a minimum cost flow in this sparse graph. Similar to the one presented by Khesin et al. [12], our sparse graph G^* is based on a randomly shifted quadtree whose cells have been subdivided into smaller *subcells*. However, the quadtree we use is compressed under certain conditions to guarantee the number of nodes in it is nearly linear in n. Also, we independently shift certain subtrees to guarantee a low expected distortion for point-to-point distances.

2.1 Construction of the sparse graph

Given a point set $P \in \mathbb{R}^d$ of size n, we say two disjoint subsets A and B of P are *s-well separated* for some $s > 0$ if A and B can be enclosed within two Euclidean balls of radius r such that the distance between these two balls are at least sr. For any constant s, we can compute a collection of $O(n)$ distinct pairs of subsets of P called an *s-well separated pair decomposition*(s-WSPD) of P such that, every pair of subsets in this collection is s-well separated and every pair of points in $P \times P$ is separated in some unique pair of subsets in this s-WSPD [6]. The time to compute the s-WSPD is $O(n \log n)$.

Our sparse graph construction begins by computing a $2-$WSPD for P containing $\ell = O(n)$ s-well separated pairs. Let $Z = \langle z_1, z_2, \ldots, z_\ell \rangle$ be a sequence of distances sorted in decreasing order so that the ith well separated pair (A, B) contains two points $p \in A, q \in B$ such that $z_i = ||q - p||_2$. By definition, the distance between any pair of points separated by the ith pair (A, B) is in $[\frac{z_i}{3}, 3z_i]$. To avoid having to handle boundary conditions later, we append $z_{\ell+1} = 0$ to the end of this sequence. Also, we compute a sub-sequence Z' of sufficiently far apart distances where Z' includes all $z_i, 1 \leq i \leq \ell$ such that $z_i > \frac{18\sqrt{d}n^4}{\varepsilon_0} z_{i+1}$.

We now build a variant of the compressed quadtree on P we call a **conditionally-compressed quadtree**. Let T^* denote this tree. Let $\square_P$ be the minimum bounding square of P. We fix an $\varepsilon_0 = O(\varepsilon / \log n)$ such that $1/\varepsilon_0$ is a power of 2. Suppose the side length of $\square_P$ is Δ^*. Let $\square$ be a square of side length $3\Delta^*$ such that $\square_P$ and $\square$ are concentric. We shift $\square$ by a vector chosen uniformly at random from $[0, \Delta^*)^d$. See Figure 1, left.

Each node of T^* is a square cell in $\mathbb{R}^d$. Set $\square$ to be the root of T^*, and let z be the first element in Z'. We recursively process each cell C as follows. Suppose C has side length Δ and the subset of P in C is P'. Let $\Delta_{P'}$ be the side length of the minimum bounding square $\square_{P'}$ of P'.

1) If $|P'| = 1$, then C is a leaf node.
2) If $|P'| > 1$ and $\Delta_{P'} < \frac{z\varepsilon_0}{3\sqrt{d}}$, we find the minimum bounding square $\square_{P'}$ of P'. Let $\Delta_{P'}$ be the side length of $\square_{P'}$. We recursively build a conditionally-compressed quadtree over P' with an independently shifted root square $\square'$ with side length $3\Delta_{P'}$ that is concentric to $\square_{P'}$ before the shift. We connect the root of this sub-quadtree to T^* as a child of C. We update the value of z for this recursive construction to largest $z' \in Z'$ such that $z' \leq 3\sqrt{d}\Delta_{P'}$. This value can be found via binary search over Z'.

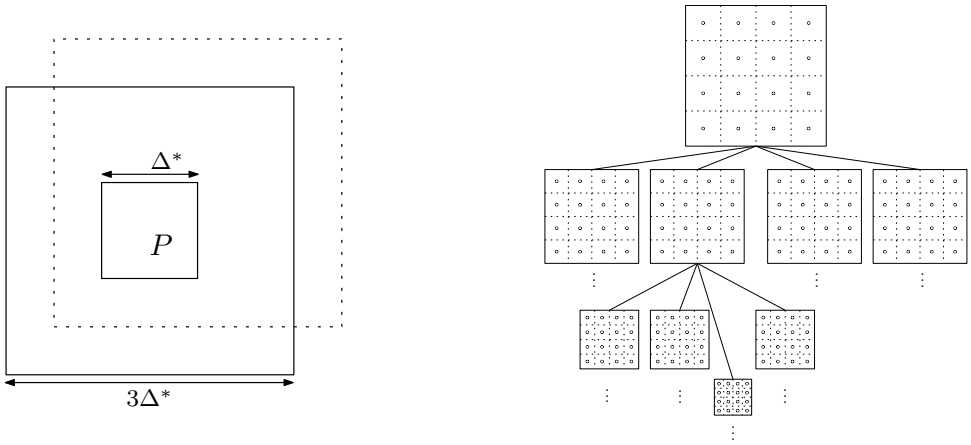


Figure 1 Left: Randomly shifting a box around P. Right: The quadtree cells form a hierarchy. Each cell is partitioned into ε_0^{-d} sub cells, and each subcell has a single net point at its center.

3) If $|P'| > 1$ and $\Delta_{P'} \geq \frac{z\varepsilon_0}{3\sqrt{d}}$, we do the following. Let x be the largest integer such that the grid with cell side length $\Delta \cdot 2^{-x}$ aligned with C contains P' within a single cell C'. Let Δ' be the side length of C'.
   a) If $\Delta' < \frac{\Delta\varepsilon_0}{n^2}$, we connect C' as the sole child of C.
   b) Otherwise, we evenly divide C into 2^d squares in $\mathbb{R}^d$ each of side length $\frac{\Delta}{2}$, and make each square that contains at least one point of P' a child cell of C.

Conditionally-compressed quadtree T^* can be constructed efficiently using standard techniques. See Appendix C for details.

▶ **Lemma 2.1.** *Let m be an upper bound on the number of nodes in T^*. Conditionally-compressed quadtree T^* can be constructed in $O(m + n \log n)$ time.*

We define two types of sub-quadtrees of T^*. A **singly-shifted sub-quadtree** is a sub-quadtree consisting of a cell C that either is the root of T^* or is randomly shifted independently of its parent along with a maximal set of descendent cells of C that were *not* shifted independently of C (i.e., Rule 2 was never applied to create descendent cells of C in the sub-quadtree). A **simple sub-quadtree** is a sub-quadtree consisting of a cell C that either is the root of T^*, is randomly shifted independently of its parent, or is added as the sole child of its parent via Rule 3a along with a maximal set of descendent cells of C created by neither Rule 2 nor Rule 3a. Observe every singly-shifted sub-quadtree consists of one or more complete simple sub-quadtrees.

For every cell C in T^*, we perform a secondary subdivision on C. Let Δ_C denote the side length of C. We divide C into ε_0^{-d} square sub-regions with equal side length $\varepsilon_0\Delta_C$. If a sub-region of C contains a point $p \in P$, we say it is a subcell $\tilde{C}$ of C and we use C^+ to denote the set of subcells of C. Again, see Figure 1.

Utilizing an idea of Agarwal et al. [1], we define the **moat of size h** around a point p as an axis-parallel square of side length h around p. Consider a randomly shifted grid with cells of side length Δ. The probability of any of the grid lines hitting a moat of size $\frac{2\Delta}{n^4}$ around any point $p \in P$ is at most $\frac{2\Delta}{n^4} \cdot n \cdot \frac{d}{\Delta} = O(\frac{1}{n^3})$.

▶ **Lemma 2.2.** *With probability at least* $1 - O((1/n)\log(n/\varepsilon_0))$*, the conditionally-compressed quadtree* T^* *has the following properties:*

1. *The total number of cells is* $O(n \log(n/\varepsilon_0))$.
2. *Suppose cell* C *with side length* Δ_C *contains* $p \in P$ *and let* $\tilde{C}$ *be the subcell of* C *that contains* p*. Then,* p *is at least* $\frac{\Delta_C}{n^4}$ *distance away from any side of* C *and is at least* $\frac{\varepsilon_0 \Delta_C}{n^4}$ *distance away from any side of* $\tilde{C}$*. In other words, the moats of* p *with respect to the uniform grids containing* C *and* $\tilde{C}$ *as cells do not touch the grid lines.*
3. *Let* T' *be any singly-shifted sub-quadtree of* T^* *constructed with a distance parameter* z*. Every leaf cell of* T' *contains at most one point from any pair* $p, q \in P$ *where* $||q-p||_2 \geq \frac{z}{3}$*, and no leaf cell of* T' *contains exactly one point from any pair* $p, q \in P$ *where* $||q-p||_2 < \frac{z}{3}$.
4. *Let* T' *be any simple sub-quadtree of* T^**, and let* C' *be a child cell of some leaf* C *of* T'*. Cell* C' *lies entirely within a* subcell *of* C.

In the proof of Lemma 2.2, we observe from Rule 2 in T^*'s construction that every cell of the root singly-shifted sub-quadtree T_0 comes from a set of $O(n \log(n/\varepsilon_0))$ grids. Therefore, we can do a union bound over the probability that any one of these grids causes a violation of Property 2. The remaining properties concerning cells of T_0 and the simple sub-quadtrees hanging immediately from its leaves either follow immediately from construction or as a consequence of Property 2. The sub-quadtrees making up the remainder of T^* use their own independent random shifts, so we can proceed with the proof inductively and take a union bound over the failure probabilities of the sub-quadtrees. See Appendix C for details.

We assume from here on that the properties described above do hold, but T^* is still randomly constructed conditional on those properties. We now build the sparse graph G^* based on the decomposition.

For every cell C, we add a **net point** ν at the center of every subcell of C, and use $N_{\tilde{C}}$ to denote the net point of a subcell $\tilde{C}$. We add $O(\varepsilon_0^{-2d})$ edges to build a clique among net points of subcells in C^+. Furthermore, if C has a parent cell C^p, for each $\tilde{C} \in C^+$, there exists a $\tilde{C}^p \in C^{p^+}$ such that $\tilde{C}$ is totally contained in $\tilde{C}^p$, because $1/\varepsilon_0$ is power of 2. We add an edge connecting $N_{\tilde{C}^p}$ with $N_{\tilde{C}}$. We say $\tilde{C}^p$ is the **parent subcell** of $\tilde{C}$ and $N_{\tilde{C}^p}$ is the **parent net point of** $N_{\tilde{C}}$. **Children subcells** and **children net points** are defined analogously. Edges are weighted by the Euclidean distance of their endpoints. Let $\tilde{C}(p)$ denote the smallest subcell containing p. As a last step, for every point $p \in P$, we add an edge connecting p to $N_{\tilde{C}(p)}$.

Let V^* be the union of P and the set of all net points we just added, and let E^* be the set of edges we added above. In short, $V^* = \cup_{C \in T}\{N_{\tilde{C}} : \tilde{C} \in C^+\} \cup P$ and $E^* = \cup_{C \in T^*}\{\{uv : u, v \in \{N_{\tilde{C}} : \tilde{C} \in C^+\}, u \neq v\} \cup \{N_{\tilde{C}} N_{\tilde{C}^p}, \tilde{C} \in C^+\}\} \cup \left\{pN_{\tilde{C}(p)}, p \in P\right\}$. The sparse graph upon which we solve minimum cost flow is denoted $G^* = (V^*, E^*)$.

▶ **Lemma 2.3.** *The expected distance between any pair* $p, q \in P$ *in* G^* *is at most* $(1 + O(\varepsilon_0 \log n))||p - q||_2$.

Proof. Let $\text{dist}_{G^*}(p, q)$ be the distance between p and q in G^*. Points p and q must be connected through the net points of some cell containing both of them. Let $C(p, q)$ be the lowest common ancestor cell of p and q. Let $N_{C(p,q)}(p)$ and $N_{C(p,q)}(q)$ be the net points of subcells of $C(p, q)$ that contains p and q, respectively. Then $\text{dist}_{G^*}(p, q) = \text{dist}_{G^*}(p, N_{C(p,q)}(p)) + \text{dist}_{G^*}(N_{C(p,q)}(p), N_{C(p,q)}(q)) + \text{dist}_{G^*}(q, N_{C(p,q)}(q))$. Value $\text{dist}_{G^*}(p, N_{C(p,q)}(p))$ is the distance from $N_{C(p,q)}(p)$ to p through its descendant net points. The upper bound of it is $\sum_{i \geq 1} 2^{-i}\sqrt{d}\varepsilon_0 \Delta_{C(p,q)} \leq \sqrt{d}\varepsilon_0 \Delta_{C(p,q)}$, because subcell side lengths at least halve every level down in T^*. Similarly,$\text{dist}_{G^*}(q, N_{C(p,q)}(q)) \leq \sqrt{d}\varepsilon_0 \Delta_{C(p,q)}$. By the triangle inequality, $\text{dist}_{G^*}(N_{C(p,q)}(p), N_{C(p,q)}(q)) \leq ||p - q||_2 + ||p - N_{C(p,q)}(p)||_2 + ||q - N_{C(p,q)}(q)||_2 \leq ||p - q||_2 + \sqrt{d}\varepsilon_0 \Delta_{C(p,q)}$. Then we have $\text{dist}_{G^*}(p, q) \leq ||p - q||_2 + 3\sqrt{d}\varepsilon_0 \Delta_{C(p,q)}$.

We define the *extra cost* to be $\Phi_{p,q} = \text{dist}_{G^*}(p,q) - ||p-q||_2$. Then $\Phi_{p,q} \leq 3\sqrt{d}\varepsilon_0\Delta_{C(p,q)}$, and the expectation of the extra cost $\mathbb{E}(\Phi_{p,q}) \leq \mathbb{E}(3\sqrt{d}\varepsilon_0\Delta_{C(p,q)}) \leq 3\sqrt{d}\varepsilon_0\mathbb{E}(\Delta_{C(p,q)})$.

Assuming the properties from Lemma 2.2, we may infer that the subset of P defining the singly-shifted sub-quadtree containing $C(p,q)$ is determined only by P itself. In particular, the set of possible shifts of the sub-quadtree's root that don't result in clipping any moats by its cells are all equally likely. Let T be this singly-shifted sub-quadtree. Let Δ^* be the side length of the root cell of T and let $\lambda = ||p-q||_2$. From Property 2 of Lemma 2.2, $\Delta_{C(p,q)} \leq n^4\lambda$, because the grid of side length $> \frac{n^4\lambda}{2}$ cannot separate p and q without clipping a moat. Also, $\Delta_{C(p,q)} \geq \frac{\lambda}{\sqrt{d}}$ so that p and q can fit in the same cell. Let $x = \text{argmax}_i\{2^{-i}\Delta^* : 2^{-i}\Delta^* \leq n^4\lambda, i \in \mathbb{N}\}$ and $y = \text{argmin}_i\{2^{-i}\Delta^* : 2^{-i}\Delta^* \geq \frac{\lambda}{\sqrt{d}}, i \in \mathbb{N}\}$. Possible values of $\Delta_{C(p,q)}$ are in $\{2^{-i}\Delta^* : x \leq i \leq y, i \in \mathbb{N}\}$. We see p and q are separated by a grid with side length Δ containing cells of T with probability at most

$$d \cdot \frac{\Delta^*}{\Delta} \cdot \lambda \cdot \frac{1}{(1 - O((1/n)\log(n/\varepsilon_0)))\Delta^*} = O\left(\frac{\lambda}{\Delta}\right).$$

Let e_i be the event that p and q are separated by the grid of size $2^{-i}\Delta^*$, we have

$$\begin{aligned}
\mathbb{E}(\Delta_{C(p,q)}) &= \sum_{x \leq i \leq y, i \in \mathbb{N}} \mathbb{P}[\bar{e_i} \cap e_{i+1}] \cdot 2^{-i}\Delta^* \\
&\leq \sum_{x \leq i \leq y, i \in \mathbb{N}} \mathbb{P}[e_{i+1}] \cdot 2^{-i}\Delta^* \\
&\leq \sum_{x \leq i \leq y, i \in \mathbb{N}} O\left(\frac{\lambda}{2^{-i-1}\Delta^*} \cdot 2^{-i}\Delta^*\right) \\
&\leq O(\log n) \cdot \lambda
\end{aligned}$$

We conclude

$$\begin{aligned}
\mathbb{E}(\text{dist}_{G^*}(p,q)) &= ||p-q||_2 + \mathbb{E}(\Phi_{p,q}) \\
&\leq ||p-q||_2 + 3\sqrt{d}\varepsilon_0\mathbb{E}(\Delta_{C(p,q)}) \\
&\leq (1 + O(\varepsilon_0 \log n)) \cdot ||p-q||_2.
\end{aligned}$$

◀

2.2 Reduction to minimum cost flow

Having built our sparse graph, we now reduce to a minimum cost flow problem in G^*. We model the minimum cost flow problem as follows to simplify later discussions.

Let $G = (V, E)$ be an arbitrary undirected graph with $V \in \mathbb{R}^d$. Let $\vec{E}$ be the set of edges in E oriented arbitrarily. We call $f \in \mathbb{R}^{\vec{E}}$ a **flow vector** or more simple, a **flow**. Let A be a $|V| \times |\vec{E}|$ **vertex-edge incidence matrix** where $\forall (u,(v,w)) \in V \times \vec{E}$, $A_{u,(v,w)} = 1$ if $u = v$, $A_{u,(v,w)} = -1$ if $u = w$, and $A_{u,(v,w)} = 0$ otherwise. Given f, we define the **divergence** of a vertex v as $(Af)_v = \sum_{(v,w)} f_{(v,w)} - \sum_{(u,v)} f_{(u,v)}$. For simplicity of exposition, we may sometimes refer to $f_{(v,u)}$ even though $(u,v) \in \vec{E}$. In such cases, it is assumed $f_{(v,u)} = -f_{(u,v)}$.

Let $||\cdot||_{\vec{E}}$ be a norm on $\mathbb{R}^{\vec{E}}$ such that $||f||_{\vec{E}} = \sum_{(u,v)\in\vec{E}} |f_{(u,v)}| \cdot ||v-u||_2$. Let $b \in \mathbb{R}^V$ denote a set of divergences for all $v \in V$. We define an instance of **uncapacitated minimum cost flow** as the pair (G, b). We seek a flow vector f minimizing $||f||_{\vec{E}}$ subject to $Af = b$.

In particular, set $b^* \in \mathbb{R}^V$ such that $b^*_p = \mu(p), \forall p \in P$ and $b^*_v = 0, \forall v \in V \setminus P$. Ultimately, we will find an approximate solution to the instance (G^*, b^*). Let $\textsc{Cost}(G^*, b^*) := ||f^*||_{\vec{E}}$ for some optimal solution f^* of this instance. From construction of G^* and Lemma 2.3, $\textsc{Cost}(P,\mu) \leq \textsc{Cost}(G^*, b^*)$ and $\mathbb{E}(\textsc{Cost}(G^*, b^*)) \leq (1 + O(\varepsilon_0 \log n))\textsc{Cost}(P,\mu)$. In particular, $\mathbb{E}(\textsc{Cost}(G^*, b^*) - \textsc{Cost}(P,\mu)) \leq O(\varepsilon_0 \log n)\textsc{Cost}(P,\mu)$. We can guarantee that bound holds with high probability by doubling the constant in the big-Oh and taking the best result from $O(\log n)$ runs of our algorithm.

2.3 Decomposition into simpler subproblems

In the sequel, we apply Sherman's generalized preconditioning framework [12, 19] to find an approximate solution to the minimum cost flow instance (G^*, b^*). For technical reasons, however, we cannot afford to run the framework on the entire sparse graph G^* at once. In Appendix A, we describe a reduction from minimum cost flow instance (G^*, b^*) to several simpler minimum cost flow instances each on the induced subgraph of the net points of one simple sub-quadtree. The reduction is based on the observation that each simple sub-quadtree subgraph has very small diameter compared to the cost of moving one unit of flow to its one parent net point and back down again to its cousin sub-quadtrees. Therefore, any reasonable method of moving the sub-quadtree's net divergence to the parent net point is sufficient for an approximately optimal solution. We must emphasize that simple sub-quadtrees may still have linear depth, so we still need to apply our own techniques to make Sherman's framework run within the desired time bounds.

3 Approximating the minimum cost flow

Let $G = (V, E)$ be an induced subgraph of sparse graph G^* where V is the subset of net points for one simple sub-quadtree T as defined above. Let $m = |E|$, and let A be the vertex-edge incidence matrix for G. We now describe the ingredients we need to provide to efficiently approximate the minimum cost flow problem in G using Sherman's generalized preconditioning framework [12, 19]. We then provide those ingredients one-by-one to achieve a near-linear time $(1 + O(\varepsilon))$-approximate solution for the minimum cost flow instance.

3.1 The preconditioning framework

Consider an instance of the minimum cost flow problem in G with an arbitrary divergence vector $\tilde{b} \in \mathbb{R}^V$, and let $f^*_{\tilde{b}} := \text{argmin}_{f \in \mathbb{R}^{\vec{E}}, Af = \tilde{b}} ||f||_{\vec{E}}$. A flow vector $f \in \mathbb{R}^{\vec{E}}$ is an **$(\boldsymbol{\alpha}, \boldsymbol{\beta})$ solution** to the problem if

$$\begin{aligned} ||f||_{\vec{E}} &\leq \alpha ||f^*_{\tilde{b}}||_{\vec{E}} \\ ||Af - \tilde{b}||_1 &\leq \beta ||A||\, ||f^*_{\tilde{b}}||_{\vec{E}} \end{aligned}$$

where $||A||$ is the norm of the linear map represented by A. An algorithm yielding an (α, β)-solution is called an **$(\boldsymbol{\alpha}, \boldsymbol{\beta})$-solver**.

By arguments in [12], we seek a preconditioner $B \in \mathbb{R}^{V \times V}$ of full column rank such that, for any $\tilde{b} \in \mathbb{R}^V$ with $\sum_{v \in V} \tilde{b}_v = 0$, it satisfies

$$||B\tilde{b}||_1 \leq \min\{||f||_{\vec{E}} : f \in \mathbb{R}^{\vec{E}}, Af = \tilde{b}\} \leq \kappa ||B\tilde{b}||_1 \tag{1}$$

for some sufficiently small function κ of n, ε, and d.

Let M be the time it takes to multiply BA and $(BA)^T$ by a vector. Then there exists a $(1 + \varepsilon, \beta)$-solver for any $\varepsilon, \beta > 0$ for this problem with running time bounded by $O(\kappa^2(|V| + |\vec{E}| + M) \log |\vec{E}|(\varepsilon^{-2} + \log \beta^{-1})$ [19]. Moreover, if a feasible flow $f \in \mathbb{R}^{\vec{E}}$ with cost $||f||_{\vec{E}} \leq \kappa B\tilde{b}$ can be found in time K, there is a $(\kappa, 0)$-solver with running time K. By setting $\beta = \varepsilon\kappa^{-2}$ [12], the composition of these two solvers is a $(1 + 2\varepsilon, 0)$-solver with running time bounded by

$$O(\kappa^2(|V| + |\vec{E}| + M) \log |\vec{E}|(\varepsilon^{-2} + \log \kappa) + K).$$

3.2 Preconditioning the minimum cost flow

We present a way to construct such a preconditioner B similar to the one of Khesin et al. [12] that guarantees κ in (1) is sufficiently small for our performance objective. Our algorithm *does not* compute B directly, because B is not sparse. However, the time for individual applications of BA or $(BA)^T$ is $O(|V| + |\vec{E}|)$.

Let $\tilde{\mathbb{C}}$ denote the set of all subcells defining the net points of G. For any subcell $\tilde{C} \in \tilde{\mathbb{C}}$, let $N_{\tilde{C}}$ denote its net point and let $\Delta_{\tilde{C}}$ denote its side length.

Let B be a matrix indexed by $(u, v) \in V \times V$ such that, for every net point ν in V where ν is the net point of some subcell $\tilde{C}$, we set $B_{\nu,v} = \frac{\Delta_{\tilde{C}}}{\Lambda}$ for all descendent net points v of ν, where $\Lambda = 22 \lg(\frac{n}{\varepsilon_0})$.[2] $B_{\nu,v} = 0$ for all other v. Matrix B has full column rank, because each column specifies exactly which ancestor net points each vertex has in G.

Now, fix any $\tilde{b} \in \mathbb{R}^V$ such that $\sum_{v \in V} \tilde{b}_v = 0$. Observe,

$$||B\tilde{b}||_1 = \sum_{\tilde{C} \in \tilde{\mathbb{C}}} \frac{\Delta_{\tilde{C}}}{\Lambda} |\sum_{v \in \tilde{C}} \tilde{b}_v|. \tag{2}$$

▶ **Lemma 3.1.** *We have* $||B\tilde{b}||_1 \leq \min\{||f||_{\vec{E}} : f \in \mathbb{R}^{\vec{E}}, Af = \tilde{b}\}$.

Lemma 3.1 is analogous to Claim 14 of Khesin et al. [12]. Their proof can be interpreted as charging each of the summands in $\Lambda \cdot ||B\tilde{b}||_1$ to the cost of the optimal flow where they overcharge by a factor equal to the depth of their tree. For our proof, we consider a path decomposition of the flow and charge to the cost of the flow one path at a time. Cell sides do not intersect moats of points in P, so only $O(\log(n/\varepsilon_0))$ charges made to a single path flow are comparable to its cost. The remaining charges are negligible. See Appendix C for details.

▶ **Lemma 3.2.** *We have* $\min\{||f||_{\vec{E}} : f \in \mathbb{R}^{\vec{E}}, Af = \tilde{b}\} \leq \kappa ||B\tilde{b}||_1$ *for some* $\kappa = O(\varepsilon_0^{-1} \log(n/\varepsilon_0))$. *Moreover, a flow vector* f *satisfying* $Af = \tilde{b}$ *of cost at most* $\kappa ||B\tilde{b}||_1$ *can be computed in* $O(m)$ *time.*

The proof of Lemma 3.2 describes a similar greedy algorithm as the one used to prove Claim 15 of Khesin et al. [12]. See Appendix C for details.

▶ **Lemma 3.3.** *Applications of* BA *and* $(BA)^T$ *to arbitrary vectors* $f \in \mathbb{R}^{\vec{E}}$ *and* $\tilde{b} \in \mathbb{R}^V$, *respectively, can be done in* $O(m)$ *time.*

Proof. Both applications can be performed using dynamic programming algorithms.

Computing BAf

Let $A' = Af$. Recall, $\forall v \in V$, A'_v is the divergence of v given flow f. Matrix A has m non-zero entries, so A' can be computed in $O(m)$ time.

We compute BAf by computing BA'. Let ν be any net point of G, and let $\tilde{C}$ be its subcell. From the definition of B, we have $(BA')_\nu = \frac{\Delta_{\tilde{C}}}{\Lambda} \sum_{v \in \tilde{C}} A'_v$. Now, let $\tilde{C}^+$ be the (possibly empty) set of all child subcells of $\tilde{C}$ with net points in G. We have $\sum_{v \in \tilde{C}} A'_v = A'_\nu + \sum_{\tilde{C}' \in \tilde{C}^+} \sum_{v \in \tilde{C}'} A'_v$. Thus, we can use dynamic programming to compute BA' in $O(m)$ time. Each entry is filled in during a postorder traversal of the quadtree cells.

[2] We use lg to denote the logarithm with base 2.

Computing $(BA)^T\tilde{b}$

Recall, $(BA)^T = A^T B^T$. Let $b' = B^T\tilde{b}$. We begin by computing b'. Let $\tilde{C}$ be any subcell with a net point in G, and let $\nu = N_{\tilde{C}}$. Let $\tilde{C}^-$ be the set of all ancestor subcells of $\tilde{C}$ with net points in G including $\tilde{C}$. We have $b'_\nu = \sum_{\tilde{C}' \in \tilde{C}^-} \frac{\Delta_{\tilde{C}'}}{\Delta} \tilde{b}_{N_{\tilde{C}'}} = \frac{\Delta_{\tilde{C}}}{\Delta}\tilde{b}_\nu + b'_{N_{\tilde{C}p}}$. Therefore, we can use dynamic programming to compute b' in $O(m)$ time. Each entry is filled in during a *pre*order traversal of the quadtree cells. Finally, A^T has m non-zero entries, so $A^T B^T \tilde{b} = A^T b'$ can be computed in $O(m)$ time as well. ◀

We have shown there exists a $(1+2\varepsilon, 0)$-solver for the minimum cost flow problem on G. Plugging in all the pieces, we get a running time bounded by

$$O(m\varepsilon_0^{-2} \log^3(n/\varepsilon_0)(\varepsilon^{-2} + \log(n/\varepsilon_0))).$$

Recall, $\varepsilon_0 = O(\varepsilon/\log n)$. We run the preconditioning framework algorithm in each graph G induced by a simple sub-quadtree's net points as described in Section 2.3. The final running time to compute a flow in G^* of cost at most $(1+\varepsilon)\textsc{Cost}(P, \mu)$ is

$$O(n\varepsilon^{-O(d)} \log^{O(d)} n).$$

4 Recovering a transportation map from the minimum cost flow

We now describe how to recover a transportation map of P using the approximately minimum cost flow $\hat{f} \in \vec{E}$ we computed for G^*. Unlike $\hat{f}$, the transportation map τ contains only weighted pairs of points in P. We will *implicitly* maintain a flow f of cost at most $||\hat{f}||_{\vec{E}^*}$ that will eventually describe our transportation map. Abusing notation, we extend the definition of $f_{(u,v)}$ to include any pair of vertices in G^*. Value $f_{(u,v)}$ is initially 0 for all $uv \notin E^*$. We follow the strategy of Khesin et al. [12] of iteratively rerouting flow going through each net point ν to instead go directly between vertices receiving from or sending flow to ν, eventually resulting in no flow going through any net point. Nearly every pair containing a point $p \in P$ and an ancestor net point may at some moment carry flow during this procedure. Because quadtree T^* has such high depth, we must take additional care.

To quickly maintain these flow assignments with points in P, we store two data structures $pt(\nu)$ and $nt(\nu)$ for each net point $\nu \in V^* \setminus P$. We call these data structures the **prefix split trees** of ν. The prefix split tree is stored as an ordered binary tree data structure where each node has a weight. We let $w(x)$ denote the weight of node x in a tree S and $w(S)$ denote the total weight of all nodes in S. These trees support the standard operations of insertion and deletion. They support changing the weight of a single node. They support the $\textsc{Merge}(S, S')$ operation which takes two trees S and S' and combines them into one tree with all members of S appearing in order before S'. Finally, they support the $\textsc{PrefixSplit}(S, t)$ operation defined as follows. Given a target value t and a prefix split tree S, PrefixSplit finds a maximal prefix of S's nodes in order where the sum of node weights in the subset is less than or equal to t. If the sum is less than t, it splits the next node x into two nodes x_1 and x_2 where $w(x_1) + w(x_2) = w(x)$. The split makes sure adding x_1 to the maximal prefix subset makes the sum weight of the subset exactly equal to t. The operation then splits off all members of this subset, including x_1 if a node x was split, into their own tree S' and returns it, leaving S with only the remaining nodes. We emphasize that the order of nodes within the data structure is important for defining PrefixSplit, but the nodes are not "sorted" in any meaningful sense; in particular, any two trees can be merged as defined above. All those operations can be done in amortized $O(\log m)$ time, where m is the number of nodes in the tree, by applying simple modifications to the splay tree data structure of Sleator and Tarjan [20]. We provide details on how to implement a prefix split tree in Appendix B.

In our setting, every node in $pt(\nu)$ and $nt(\nu)$ represents a point $p \in P$. Thanks to our use of the PrefixSplit procedure, some points may be represented multiple times in a single tree. We use $pt(\nu)[p]$ to denote the set nodes representing p in $pt(\nu)$, and define $nt(\nu)[p]$ similarly. Our algorithm implicitly maintains the invariant that for all net points ν and points $p \in P$, $\sum_{x \in pt(\nu)[p]} w(x) - \sum_{x \in nt(\nu)[p]} w(x) = f_{(\nu,p)}$. We proceed with Algorithm 1.

Algorithm 1 Recovering a transportation map from an approximately minimum cost flow in G^*.

⟨⟨***Initialize data structures.***⟩⟩
For all net points $v \in V^* \setminus P$ and $p \in P$ where $f_{(p,v)} > 0$
  Insert a node of weight $f_{(p,v)}$ into $nt(v)$ representing p
For all net points $v \in V^* \setminus P$ and $p \in P$ where $f_{(v,p)} > 0$
  Insert a node of weight $f_{(v,p)}$ into $pt(v)$ representing p
Let $\mathbb{C}$ be the set of all cells
For $C \in \mathbb{C}$ in postorder
  Let $N_C = \{N_{\tilde{C}} : \tilde{C} \in C\}$, $N'_C = \{\text{parent of } v : v \in N_C\}$
  For each $v \in N_C$
    ⟨⟨***Cancel flow to/from other net points.***⟩⟩
    While $\exists u, w \in N_C \cup N'_C : f_{(v,w)} > 0 > f_{(v,u)}$
      $\delta \leftarrow \min\{f_{(u,v)}, f_{(v,w)}\}$
      $f_{(u,w)} \leftarrow f_{(u,w)} + \delta$
      $f_{(u,v)} \leftarrow f_{(u,v)} - \delta$
      $f_{(v,w)} \leftarrow f_{(v,w)} - \delta$
    ⟨⟨***Now, either all other net points send flow to v or all get flow from v.***⟩⟩
    While $\exists u \in N_C \cup N'_C : f_{(u,v)} > 0$
      ⟨⟨***Implicitly reduce $f(v,p)$ and increase $f(u,p)$ for several $p \in P$***⟩⟩
      $pt' \leftarrow \textsc{PrefixSplit}(pt(v), f_{(u,v)})$
      $\textsc{Merge}(pt', pt(u))$
    While $\exists w \in N_C \cup N'_C : f_{(v,w)} > 0$
      ⟨⟨***Implicitly reduce $f(p,v)$ and increase $f(p,w)$ for several $p \in P$***⟩⟩
      $nt' \leftarrow \textsc{PrefixSplit}(nt(v), f_{(v,w)})$
      $\textsc{Merge}(nt', nt(w))$
    ⟨⟨***Now, all flow to/from v involves points $p \in P$.***⟩⟩
    While $pt(v)$ and $nt(v)$ are not empty
      Let $x \in nt(v)[p], y \in pt(v)[q]$ for some $p, q \in P$
      $\delta \leftarrow \min\{w(x), w(y)\}$
      $f_{(p,q)} \leftarrow f_{(p,q)} + \delta$
      $w(x) \leftarrow w(x) - \delta$; if $w(x) = 0$, delete x from $nt(v)$
      $w(y) \leftarrow w(y) - \delta$; if $w(y) = 0$, delete y from $pt(v)$
For all $(p, q) \in P \times P$ where $f_{(p,q)} > 0$
  $\tau(p, q) \leftarrow f_{(p,q)}$

▶ **Lemma 4.1.** *Our algorithm results in a transportation map of cost at most $||\hat{f}||_{\vec{E}^*}$, and it can be implemented to run in $O(n\epsilon_0^{-2d} \log^2(n/\varepsilon_0))$ time.*

As in Khesin et al. [12], we prove Lemma 4.1 by arguing that we remove all flow passing through each net point encountered during our postorder traversal of T^*. Each change in the flow reduces its cost, and the number of changes involving two or more net points is nearly bounded by the size of T^*. To account for the time spend moving flow to or from points $p \in P$ in the forth while loop, we charge such operations to the moving of flow directly between net points. See Appendix C for details.

References

1. Pankaj K. Agarwal, Kyle Fox, Debmalya Panigrahi, Kasturi R. Varadarajan, and Allen Xiao. Faster algorithms for the geometric transportation problem. In *Proc. 33rd Intern. Symp. Comput. Geom.*, pages 7:1–7:16, 2017. doi:10.4230/LIPIcs.SoCG.2017.7.
2. Pankaj K. Agarwal and R. Sharathkumar. Approximation algorithms for bipartite matching with metric and geometric costs. In *Proc. 46th Symp. Theory Comput.*, pages 555–564. ACM, 2014. doi:10.1145/2591796.2591844.
3. Alexandr Andoni, Aleksandar Nikolov, Krzysztof Onak, and Grigory Yaroslavtsev. Parallel algorithms for geometric graph problems. In *Proc. 46th Symp. Theory Comput.*, pages 574–583, 2014. doi:10.1145/2591796.2591805.
4. Marshall W. Bern, David Eppstein, and Shang-Hua Teng. Parallel construction of quadtrees and quality triangulations. *Int. J. Comput. Geometry Appl.*, 9(6):517–532, 1999. doi:10.1142/S0218195999000303.
5. Nicolas Bonneel, Michiel van de Panne, Sylvain Paris, and Wolfgang Heidrich. Displacement interpolation using Lagrangian mass transport. *ACM Trans. Graph.*, 30(6):158, 2011. doi:10.1145/2070781.2024192.
6. Paul B. Callahan and S. Rao Kosaraju. Faster algorithms for some geometric graph problems in higher dimensions. In *Proc. 4th Ann. ACM/SIGACT-SIAM Symp. Discrete Algorithms*, pages 291–300, 1993. URL: http://dl.acm.org/citation.cfm?id=313559.313777.
7. Marco Cuturi and Arnaud Doucet. Fast computation of Wasserstein barycenters. In *Proc. 31st Intern. Conf. Machine Learning*, pages 685–693, 2014. URL: http://proceedings.mlr.press/v32/cuturi14.html.
8. Panos Giannopoulos and Remco C. Veltkamp. A pseudo-metric for weighted point sets. In *Proc. 7th Europ. Conf. Comput. Vision*, pages 715–730, 2002. doi:10.1007/3-540-47977-5_47.
9. Kristen Grauman and Trevor Darrell. Fast contour matching using approximate earth mover's distance. In *Proc. 24th IEEE Conf. Comput. Vision and Pattern Recog.*, pages I:220–I:227, 2004. doi:10.1109/CVPR.2004.104.
10. Sariel Har-Peled. *Geometric approximation algorithms*, volume 173. American Mathematical Soc., 2011.
11. Piotr Indyk. A near linear time constant factor approximation for Euclidean bichromatic matching (cost). In *Proc. 18th Ann. ACM-SIAM Symp. Discrete Algorithms*, pages 39–42, 2007. URL: http://dl.acm.org/citation.cfm?id=1283383.1283388.
12. Andrey Boris Khesin, Aleksandar Nikolov, and Dmitry Paramonov. Preconditioning for the geometric transportation problem. In *Proc. 35th Intern. Symp. Comput. Geom.*, pages 15:1–15:14, 2019. doi:10.4230/LIPIcs.SoCG.2019.15.
13. Nathaniel Lahn, Deepika Mulchandani, and Sharath Raghvendra. A graph theoretic additive approximation of optimal transport. In *Proc 32nd Adv. Neur. Info. Proces. Sys.*, pages 13813–13823, 2019. URL: http://papers.nips.cc/paper/9533-a-graph-theoretic-additive-approximation-of-optimal-transport.
14. Yin Tat Lee and Aaron Sidford. Path finding methods for linear programming: Solving linear programs in õ(vrank) iterations and faster algorithms for maximum flow. In *Proc. 55th IEEE Ann. Symp. Found. Comput. Sci.*, pages 424–433, 2014. doi:10.1109/FOCS.2014.52.
15. James B. Orlin. A faster strongly polynomial minimum cost flow algorithm. *Operations Research*, 41(2):338–350, 1993. doi:10.1287/opre.41.2.338.
16. Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas. The earth mover's distance as a metric for image retrieval. *Intern. J. Comput. Vision*, 40(2):99–121, 2000. doi:10.1023/A:1026543900054.
17. R. Sharathkumar and Pankaj K. Agarwal. Algorithms for the transportation problem in geometric settings. In *Proc. 23rd Ann. ACM-SIAM Symp. Discrete Algorithms*, pages 306–317, 2012. doi:10.1137/1.9781611973099.29.

18 R. Sharathkumar and Pankaj K. Agarwal. A near-linear time ϵ-approximation algorithm for geometric bipartite matching. In *Proc. 44th Symp. Theory Comput.*, pages 385–394, 2012. doi:10.1145/2213977.2214014.

19 Jonah Sherman. Generalized preconditioning and undirected minimum-cost flow. In *Proc. 28th Ann. ACM-SIAM Symp. Discrete Algorithms*, pages 772–780, 2017. doi:10.1137/1.9781611974782.49.

20 Daniel Dominic Sleator and Robert Endre Tarjan. Self-adjusting binary search trees. *J. ACM*, 32(3):652–686, 1985. doi:10.1145/3828.3835.

21 Justin Solomon, Raif M. Rustamov, Leonidas J. Guibas, and Adrian Butscher. Earth mover's distances on discrete surfaces. *ACM Trans. Graph.*, 33(4):67:1–67:12, 2014. doi:10.1145/2601097.2601175.

22 Cédric Villani. *Optimal Transport: Old and New*. Springer Science & Business Media, 2008.

A Decomposing minimum cost flow into simpler subproblems

Here, we reduce finding an approximately optimal flow for minimum cost flow instance (G^*, b^*) to finding $O(n)$ approximately optimal flows, each within an induced subgraph defined by the net points within a single simple sub-quadtree.

Recall, for each point $p \in P$, $\tilde{C}(p)$ denotes the smallest subcell containing p, and $N_{\tilde{C}}$ denotes the net point of subcell $\tilde{C}$. Let f be the flow such that $f_{(p,N_{\tilde{C}(p)})} = b^*_p$ for all $p \in P$. Let $G' = (V', E')$ and A' be the restriction of G^* and its vertex-edge incidence matrix A after removing all vertices $p \in P$. Let b' be the restriction of $b - Af$ to vertices of G'. Every vertex $p \in P$ of G^* has exactly one incident edge, so an optimal solution to our original minimum cost flow instance consists of f along with an optimal solution to the instance defined on A' and b'. From here one, we focus on finding an approximately minimum cost flow in G'.

Suppose there are multiple simple sub-quadtrees. Let $G_0 = (V_0, E_0)$ be the subgraph induced by the m net point vertices of a single simple sub-quadtree with no descendent sub-quadtrees. Let C be the root cell of the simple sub-quadtree for G_0, let u be a net point for an arbitrary subcell of C, and let v be the parent net point of u in G'. In $O(m)$ time, we compute $B = \sum_{w \in V_0} b'_w$, the total divergence of vertices within G_0. We then let f' be the flow in G' that is 0 everywhere except for $f_{(u,v)} := B$. Finally, let $b'' = b' - A'f'$.

Notice that at least B units of flow in G_0 needs to leave or enter C by edge at least the side length of $\tilde{(C)}$. Given $\Delta(C) \leq O(1/n^2)\Delta_{\tilde{C}}$, we can lazily assign the flow between net points of C and v with increasing the cost by at most $2\sqrt{d}\Delta(C)B \leq O(1/n^2)\Delta_{\tilde{C}}B$. This suggests the following lemma.

▶ **Lemma A.1.** *There exists a flow f'' in G' such that $f''_{(w,x)} = 0$ for all $w \in V_0, x \notin V_0$; $Af'' = b''$; and $||f'' + f'||_{\vec{E}'} \leq (1 + O(1/n^2)) \cdot \textsc{Cost}(G', b')$.*

Proof. Let $\tilde{C}$ be the subcell for which v is a net point. Let $\Delta_{\tilde{C}}$ be the side length of $\tilde{C}$. By construction of G', at least B units of flow must travel to or from vertex v from G_0 at a cost of $\Delta_{\tilde{C}}$. Specifically, G_0 is totally inside $\tilde{C}$, v is the only vertex in $\tilde{C}$ incident to some edge crossing the side of $\tilde{C}$, and the nearest vertex $x \notin V_0$ is at least $\Delta_{\tilde{C}}$ far from v. So $\textsc{Cost}(G', b') \geq \Delta_{\tilde{C}}B$.

Suppose f'^* is a flow in G' with cost $\textsc{Cost}(G', b')$. Let N_C be the set of net points of subcells of C. We may assume there is no pair $y, z \in N_C$ such that $f_{(y,v)} > 0$ and $f_{(v,z)} > 0$, because we could send the flow directly between y and z more cheaply. We create flow f''' as follows starting with $f'' = f'^*$. While there exists some vertex $u' \in N_C \backslash \{u\}$ with $f'''_{(u',v)} \neq 0$,

let $\delta = f'''_{(u',v)}$. We divert flow by setting $f'''_{(u,v)} \leftarrow f'''_{(u,v)} + \delta$, $f'''_{(u',u)} \leftarrow f'''_{(u',u)} + \delta$, and $f'''_{(u',v)} \leftarrow 0$. This increases the cost by at most twice of the length of the diagonal of C per diverted unit of flow. Overall, we divert at most B units. The total cost increase is at most $2\sqrt{d}\Delta_C B \leq O(1/n^2)\textsc{Cost}(G', b')$ where Δ_C is side length of C, because $\Delta_C \leq O(1/n^2)\Delta_{\tilde{C}}$. We have $||f'''||_{\vec{E}} \leq (1 + O(1/n^2)) \cdot \textsc{Cost}(G', b')$. Finally, let $f'' = f''' - f'$. ◀

The above lemma implies we can use the following strategy for approximating a minimum cost flow in G': Let b_0 be the restriction of b'' to V_0. We find a flow in G_0 with divergences b_0 of cost at most $(1+O(\varepsilon)) \cdot \textsc{Cost}(G_0, b_0)$ using the algorithm described in the next section. Then, we recursively apply our algorithm on $G'' = (V'', E'')$, the induced subgraph over $V'' = V' \setminus V_0$. The depth of recursion is $O(n)$, so the total cost from combining our separately computed flows is $(1 + O(\varepsilon))(1 + O(1/n)) \cdot \textsc{Cost}(G', b') = (1 + O(\varepsilon))\textsc{Cost}(G', b')$.

B Prefix split trees

We implement our prefix split trees by modifying the splay tree data structure of Sleator and Tarjan [20]. Let S be a prefix split tree. We store the weight $w(x)$ of each node x directly with the node itself. Moreover, every node x keeps another value $W(x)$ equal to the sum weight of all the descendants of x including x itself.

A **splay** of a node x in S is a sequence of double rotations (possibly followed by a standard single rotation) that move x to the root of S. Only those nodes on the path from the root to x have their children pointers updated by a splay. We can update $W(y)$ for every such node y with only a constant factor overhead in the time to perform a splay. Let $s(x)$ denote the number of descendents of x in its prefix split tree, and let $r(x) = \lfloor \lg s(x) \rfloor$. Let $\Phi(S) = \sum_{x \in S} r(x)$. The **amortized time** for an operation on S can be defined as the real time spent on the operation plus the net change to $\Phi(S)$ after the operation. The amortized time for a splay in an m-node tree is $O(\log m)$ [20].

Recall, the order of nodes within a tree is largely irrelevant outside the definition of the PrefixSplit operation. To insert a node x in S, we add x as the child of an arbitrary leaf of S and splay x to the root. The number of operations in the splay dominates, so the amortized cost of insertion is $O(\log m)$. To delete a node x, we splay x to the root and delete it, resulting in two disconnected subtrees S_1 and S_2. We then perform a $\textsc{Merge}(S_1, S_2)$ in $O(\log m)$ amortized time as described below, so the whole deletion has amortized cost $O(\log m)$. To update the weight of a node x, we splay x to the root and update $w(x)$ and $W(x)$ in constant time each. The splay once again dominates, so the total amortized cost is $O(\log m)$.

The operation $\textsc{Merge}(S_1, S_2)$ is implemented as follows. Let x be the rightmost leaf of S_1. We splay x to the root so it has exactly one child. We then make the root of S_2 the other child of x. Let m be the total number of nodes in S_1 and S_2. Adding S_2 as a child increases $\Phi(S_1) + \Phi(S_2)$ by $O(\log m)$, so the amortized time for the Merge is $O(\log m)$.

Finally, we discuss the implementation of $\textsc{PrefixSplit}(S, t)$. We assume $t > 0$. We use the values $W(\cdot)$ to find the prefix of nodes desired. Let y be the next node in order after the prefix. We splay y to the root of S. Let x be the left child of y (if it exists). Suppose $W(x) < t$. We delete y, creating two trees S_1 and S_2 where S_1 contains the nodes in the prefix. We create a new node y_1 of weight $t - W(y)$ and make the root of S_1 its child so that y_1 is the new root. We create a node y_2 of weight $w(y) - w(y_1)$ and make the root of S_2 its child. Now, suppose instead $W(x) = t$. In this case, we simply remove the edge between x and y to create a subtree S_1 with x as its root. Let S_2 be the remainder of S. Whether or not $W(x) = t$, we return S_1 and set $S = S_2$. The amortized time for the PrefixSplit is the amortized time for a single splay and a constant number of edge changes, implying the PrefixSplit takes $O(\log m)$ amortized time total.

C Omitted Proofs

Proof of Lemma 2.1. Suppose we are processing a cell C containing point subset P'. Following standard practice [6], we assume access to d doubly-linked lists containing the points of P'. The points in each list are sorted by distinct choices of one of their d coordinates.

We now describe how to process C. We determine if $|P'| = 1$ in constant time. If so, we stop processing C and discard its data structures. Otherwise, we use the lists to determine $\Delta_{P'}$ and $\square_{P'}$ in $O(1)$ time. If $\Delta_{P'} < \frac{z\varepsilon_0}{3\sqrt{d}}$, we follow Rule 2 by doing the search for the new value of z in $O(\log n)$ time. We pass along the lists for C to the recursive quadtree construction.

Suppose $\Delta_{P'} \geq \frac{z\varepsilon_0}{3\sqrt{d}}$. We can compute C' and Δ' as defined in Rule 3 in constant time by building a standard compressed quadtree over the $2d$ extreme points of P' in each dimension that respects the grid containing C and examining its root [10, Chapter 2]. If $\Delta' < \frac{\Delta\varepsilon_0}{n^2}$, we simply recurse with the same lists as described above.

Suppose all other tests fail and Rule 3b applies. We compute the point subsets and their lists going into each child cell by splitting P' one dimension at a time. For each dimension, for each subset of points we already know go into different cells, we search the relevant linked list for that dimension from both ends simultaneously, so we know where to split the list in time proportional to the number of points in the less populated side of the split. In time proportional to the number of points going to the less populated side of the split, we also perform individual deletions and insertions to make the $d-1$ new linked lists for the points on the less populated side. We pass along the lists we construct when computing subtrees for each child of C.

We spend $O(\log n)$ time per node in addition to the time spent searching, inserting, and deleting points from lists when applying Rule 3b. However, every time a point moves to a new data structure, the number of points in its cell drops by a factor of at least 2. We spend $O(m + n \log n) = O(m \log n)$ time total implementing Rule 3b. ◀

Proof of Lemma 2.2. The condition to trigger Rule 3a guarantees every path of descendent cells with one child each has length $O(\log(n/\varepsilon_0))$. We immediately get Property 1.

Let T_0 be the singly-shifted sub-quadtree containing the root cell of T^*. By construction of Z', the smallest cells of T_0 lie $O(n \log(n/\varepsilon_0))$ (uncompressed) quadtree levels down. Therefore, at most $O(n \log(n/\varepsilon_0))$ shifted grids in $\mathbb{R}^d$ determine the boundaries of T_0's (sub)cells. We see Property 2 is violated for at least one cell in T_0 with probability at most $c \cdot \frac{n}{n^3} \cdot \log \frac{n}{\varepsilon_0}$ for some constant c.

Assume from here on that Property 2 holds for all cells in T_0. The first part of Property 3 is guaranteed for T_0 by construction. Similarly, Property 4 is guaranteed for any simple sub-quadtree within T_0 by construction. Finally, let $p, q \in P$ be any pair of points where $||q-p||_2 < \frac{z}{3}$. By definition of z, we have $||q-p||_2 \leq 3 \cdot \frac{z\varepsilon_0}{18\sqrt{d}n^4} = \frac{z\varepsilon_0}{6\sqrt{d}n^4}$. Both points are distance at least $\frac{z\varepsilon_0}{6\sqrt{d}n^4}$ from the side of any subcell, so they are not separated by any subcell of T_0, implying the second part of Property 3. Finally, Property 3 holds for all pairs of points, including the ones defining the bounding boxes for simple sub-quadtrees whose roots are children of leaves in T_0. The points are far enough away from the subcell boundaries that even the random shift of these simple sub-quadtrees will keep them inside their subcells. Property 4 holds for simple sub-quadtrees whose roots are the children of leaves in T_0.

Now, let $\{T_1, T_2, \ldots\}$ denote the distinct sub-quadtrees, each consisting of a child of a leaf in T_0 and all of the child's descendants in T^*. For each T_i, let n_i be the number of points over which T_i is built. We have $n_i \leq n-1$ and $\sum_i n_i \leq n$. We may inductively assume Properties

2 through 4 fail to hold for T_i with probability at most $c \cdot \frac{n_i^2}{n^3} \cdot \log \frac{n}{\varepsilon_0} \leq c \cdot \frac{(n-1)n_i}{n^3} \cdot \log \frac{n}{\varepsilon_0}$. Taking a union bound, the probability of Properties 2 through 4 failing to hold for either T_0 or any T_i is at most $c \cdot \frac{n^2}{n^3} \cdot \log \frac{n}{\varepsilon_0} = c \cdot \frac{1}{n} \cdot \log \frac{n}{\varepsilon_0}$. ◀

Proof of Lemma 3.1. Let $f^*_{\tilde{b}} := \operatorname{argmin}_{f \in \mathbb{R}^{\vec{E}}, Af = \tilde{b}} ||f||_{\vec{E}}$. We arbitrarily decompose $f^*_{\tilde{b}}$ into a set of flows $F = \{f^1, f^2, \ldots\}$ with the following properties: 1) each flow follows a simple path between two vertices u and v; 2) for each flow $f^i \in F$ and edge $(u, v) \in \vec{E}$ either $f^i(u, v) = 0$ or its sign is equal to the sign of $f^*_{\tilde{b}}(u, v)$; 3) for each flow $f^i \in F$ and vertex v, either $(Af^i)_v = 0$ or its sign is equal to $\tilde{b}_v$; and 4) for each edge $(u, v) \in \vec{E}$, we have $f^*_{\tilde{b}}(u, v) = \sum_{f^i \in F} f^i(u, v)$. The existence of such a decomposition is a standard part of network flow theory and one can be computed in a simple greedy manner (however, our algorithm does not actually need to compute one). From construction, we have $\sum_{f^i \in F} ||f^i||_{\vec{E}} = ||f^*_{\tilde{b}}||_{\vec{E}}$. We describe a way to charge summands of $\sum_{\tilde{C} \in \tilde{\mathbb{C}}} \Delta_{\tilde{C}} |\sum_{v \in \tilde{C}} \tilde{b}_v|$ to the summands of $\sum_{f^i \in F} ||f^i||_{\vec{E}}$. Our charges will cover each of the former and exceed each of the latter by at most a Λ factor. Consider a subcell $\tilde{C}$. For each vertex $u \in \tilde{C}$, for each flow f^i sending flow to or from u, we charge $\Delta_{\tilde{C}} |(Af^i)_u|$. Clearly, we charge at least $\Delta_{\tilde{C}} |\sum_{v \in \tilde{C}} \tilde{b}_v|$ for each subcell $\tilde{C}$.

It remains to prove we did not overcharge by too large a factor. Consider an arbitrary flow $f^i \in F$ sending flow from some vertex u to some vertex v. Let $C(u, v)$ be the lowest common ancestor cell containing u and v. Let $\Delta_{C(u,v)}$ be its side length, and let $C(\hat{u}, v)$ be the child cell of $C(u, v)$ that includes u. Let Δ be the side length of $C(\hat{u}, v)$.

Suppose there exists a descendant cell C' of $C(\hat{u}, v)$ containing u that is at least $5 \lg n$ levels down from $C(\hat{u}, v)$. Its side length $\Delta_{C'}$ is at most $\frac{\Delta}{n^5}$. Because C' contains at least one point $u' \in P$, and from Property 2 of Lemma 2.2, u is at least $\frac{\Delta}{n^4} - \frac{\Delta}{n^5} \geq \frac{\Delta}{2n^4}$ distance away from any side of $C(\hat{u}, v)$ and therefore v as well. Therefore, we charge at most an $\frac{\varepsilon_0}{2n}$ fraction of $||f^i||_{\vec{E}}$ to cover u's subcell in C'. The amounts charged by similar subcells of smaller side length containing u form a decreasing geometric series evaluating to at most that value, so all these small subcells charge at most an $\frac{\varepsilon_0}{n}$ fraction total.

Now, consider the cells with larger side length. Suppose there exists an ancestor cell C'' of $C(\hat{u}, v)$ at least $\lg \varepsilon_0^{-1} + 1$ levels up from $C(\hat{u}, v)$, and let $\tilde{C}''$ be the subcell of C'' containing u. Then the side length of $\tilde{C}''$ is at least $\Delta_{C(u,v)}$ and all points in $C(u, v)$ will be included in $\tilde{C}''$ also. Therefore, we do not charge to $||f^i||_{\vec{E}}$ for subcell $\tilde{C}''$, and there are at most $5 \lg n + \lg \varepsilon_0^{-1} \leq 5 \lg \frac{n}{\varepsilon_0}$ subcells in addition to those handled above for which we do charge to $||f^i||_{\vec{E}}$. Consider any such subcell $\tilde{C}$. The path carrying f^i leaves $\tilde{C}$ through an edge of length at least $\Delta_{\tilde{C}}/2$, so we charge at most $2 \cdot ||f^i||_{\vec{E}}$ to cover $\tilde{C}$. Summing over all $5 \lg \frac{n}{\varepsilon_0}$ choices of $\tilde{C}$ and accounting for the tiny cells as discussed above, we charge at most $(10 \lg \frac{n}{\varepsilon_0} + \varepsilon_0/n)||f^i||_{\vec{E}} \leq 11 \lg (\frac{n}{\varepsilon_0}) \cdot ||f^i||_{\vec{E}}$ to cover subcells containing u. We also charge to $||f^i||_{\vec{E}}$ to cover subcells containing v, so we overcharge by a factor of at most $22 \lg (\frac{n}{\varepsilon_0}) = \Lambda$. The lemma follows. ◀

Proof of Lemma 3.2. We describe a greedy algorithm based on one by Khesin et al. [12] to iteratively construct a feasible flow f satisfying $Af = \tilde{b}$ with a cost $||f||_{\vec{E}} \leq \kappa B \tilde{b}$ in $O(m)$ time. At any point during f's construction, we say the **surplus** of vertex $u \in V$ is $\pi(u, f) = (Af)_u - \tilde{b}_u$, the difference between the current and desired divergences of u.

1. For every cell C in a postorder traversal of G's simple sub-quadtree, for every subcell $\tilde{C}$ of C, we do the following. Let $\nu = N_{\tilde{C}}$. We choose any two child net points v, w of ν such that $\pi(v, f) > 0 > \pi(w, f)$. We then add $\min\{|\pi(v, f)|, |\pi(w, f)|\}$ to $f_{(w,v)}$. In doing so, we make the surplus of at least one child net point of ν equal to 0, and we decrease the absolute values of surpluses of both v and w. Therefore, after at most a number of steps equal to the number of child net points of ν, either all child net points have non-negative

surplus or all child net points have non-positive surplus. Finally, for each vertex v among child net points with non-zero surplus, we set $f_{(\nu,v)} = \pi(v, f)$. Afterward, every child net point of ν has surplus 0. In other words, the unbalance among those child net points is collected into ν. Each net point ν has at most 2^d child net points. Therefore, the total running time for this step is $O(m)$.

2. After performing step 1), all net points with parents have a surplus of 0. We pick up any two net points u, v of subcells of T's root cell with two different surplus signs as described in step 2 and add $\min\{|\pi(u, f)|, |\pi(v, f)|\}$ to $f_{(v,u)}$. After $O(\varepsilon_0^{-d}) = O(m)$ steps, all points $v \in V$ will have surplus 0, and f is a feasible flow satisfying $Af = \tilde{b}$.

We now analyze $||f||_{\vec{E}}$. Consider a subcell $\tilde{C}$ of some cell C with net point ν. Flow does not leave or enter $\tilde{C}$ until we move flow between ν and either another net point in C or ν's parent net point. Therefore, $\pi(\nu, f) = -\sum_{v\in\tilde{C}} \tilde{b}_v$ immediately after moving flow from ν's children to ν in step 1) above. All subsequent steps moving flow to or from ν involve an edge of length at most $\varepsilon_0^{-1}\sqrt{d}\Delta_{\tilde{C}}$ and only serve to reduce $|\pi(\nu, f)|$.

Summing over all subcells, we get

$$||f||_{\vec{E}} \leq \sum_{\tilde{C}\in\tilde{\mathbb{C}}} \varepsilon_0^{-1}\sqrt{d}\Delta_{\tilde{C}}|\sum_{v\in\tilde{C}} \tilde{b}_v| \leq \varepsilon_0^{-1}\sqrt{d}\Lambda||B\tilde{b}||_1.$$

Therefore, $||f^*_{\tilde{b}}||_{\vec{E}} \leq ||f||_{\vec{E}} \leq \kappa||B\tilde{b}||_1$, where $\kappa = O(\varepsilon_0^{-1}\log(n/\varepsilon_0))$. ◀

Proof of Lemma 4.1. As stated, our algorithm implicitly maintains a flow f such that for all net points ν and points $p \in P$, $\sum_{x\in pt(\nu)[p]} w(x) - \sum_{x\in nt(\nu)[p]} w(x) = f_{(\nu,p)}$. One can easily verify that after every iteration of any of the while loops, the divergences among all vertices in G^* remain the same. Further, after processing any net point v in the inner for loop, there are no other vertices u in V^* such that $f_{(u,v)} \neq 0$. Observe the algorithm never changes the flow coming into or out of a net point v unless $f_{(u,v)} \neq 0$ for some vertex u. Therefore, after v is processed, it *never* has flow going into or out of it again (Khesin et al. [12] refer to this property as v having *uniform flow parity*). Because we eventually process every net point in G^*, we eventually end up with a flow f such that $f_{(p,q)} \neq 0$ only if $p, q \in P$. We immediately see τ is a transportation map.

To analyze the cost of τ, observe that after every iteration of a while loop, we replace some δ units of flow passing through v, possibly between multiple sources and one destination or vice versa, with δ units going directly from the source(s) to the destination(s). By the triangle inequality, this new way to route flow is cheaper, so the final flow f, and subsequently τ has smaller cost than $\hat{f}$.

To implement our algorithm quickly, we only explicitly store new flow values whenever we have a line "$f_{(u,w)} \leftarrow \_$" for some pair of vertices (u, w). Observe that every time we finish processing a cell, every one of its net points is also processed. By the above discussion, flow no longer passes through those net points. Therefore, as we process the net points for a cell C, we never send flow from a net point $v \in N_C$ to a net point outside $N_C \cup N'_C$. Every time we change flow going through another net point while processing a net point v, we decrease the number net points u such that $f_{(u,v)} \neq 0$ by one. There are $O(n\varepsilon_0^{-d}\log(n/\varepsilon_0))$ net points, and $O(\varepsilon_0^{-d})$ other net points in each $N_C \cup N'_C$, so the number of iterations total in the first three while loops is $O(n\varepsilon_0^{-2d}\log(n/\varepsilon_0))$. Finally, observe that we only do PrefixSplit operations during these while loops, implying we create a total of $O(n\varepsilon_0^{-2d}\log(n/\varepsilon_0))$ nodes throughout all prefix split trees. Every iteration of the fourth while loop results in deleting a node from at least one of $nt(v)$ or $pt(v)$, so the number of iterations of this while loop is $O(n\varepsilon_0^{-2d}\log(n/\varepsilon_0))$ as well. Finally, every while loop iteration consists of a constant number of operations in addition to a constant number of prefix split tree operations, each of which can be done in $O(\log(n/\varepsilon_0))$ amortized time. ◀

Bounded VC-Dimension Implies the Schur-Erdős Conjecture

Jacob Fox
Department of Mathematics, Stanford University, Stanford, CA, USA
jacobfox@stanford.edu

János Pach
Alfréd Rényi Institute of Mathematics, Budapest, Hungary
IST, Vienna, Austria
MIPT, Moscow, Russia
pach@cims.nyu.edu

Andrew Suk
Department of Mathematics, University of California San Diego, La Jolla, CA, USA
asuk@ucsd.edu

Abstract

In 1916, Schur introduced the Ramsey number $r(3;m)$, which is the minimum integer $n>1$ such that for any m-coloring of the edges of the complete graph K_n, there is a monochromatic copy of K_3. He showed that $r(3;m)\leq O(m!)$, and a simple construction demonstrates that $r(3;m)\geq 2^{\Omega(m)}$. An old conjecture of Erdős states that $r(3;m)=2^{\Theta(m)}$. In this note, we prove the conjecture for m-colorings with bounded VC-dimension, that is, for m-colorings with the property that the set system induced by the neighborhoods of the vertices with respect to each color class has bounded VC-dimension.

2012 ACM Subject Classification Mathematics of computing → Combinatoric problems

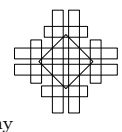
Keywords and phrases Ramsey theory, VC-dimension, Multicolor Ramsey numbers

Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.46

Funding *Jacob Fox*: Supported by a Packard Fellowship and by NSF award DMS-1855635.
János Pach: Supported by Austrian Science Fund (FWF) grant Z 342-N31 and by the Ministry of Education and Science of the Russian Federation in the framework of MegaGrant no 075-15-2019-1926.
Andrew Suk: Supported by NSF CAREER award DMS-1800746 and an Alfred Sloan Fellowship.

1 Introduction

Given n points and n lines in the plane, their incidence graph is a bipartite graph G that contains no $K_{2,2}$ as a subgraph. By a theorem of Erdős [5] and Kővári-Sós-Turán [15], this implies that the number of incidences between the points and the lines is $O(n^{3/2})$. However, a celebrated theorem of Szemerédi and Trotter [20] states that the actual number of incidences is much smaller, only $O(n^{4/3})$, and this bound is tight. There are many similar examples, where extremal graph theory is applicable, but does not yield optimal results. What is behind this curious phenomenon? In the above and in many other examples, the vertices of G are, or can be associated with, points in a Euclidean space, and the fact whether two vertices are connected by an edge can be determined by evaluating a bounded number of polynomials in the coordinates of the corresponding points. In other words, G is a *semi-algebraic graph* of bounded complexity. As was proved by the authors, in collaboration with Sheffer and Zahl [8], for semi-algebraic graphs, one can explore the *geometric* properties of the polynomial surfaces, including separator theorems and the so-called polynomial method [12], to obtain




much stronger results for the extremal graph-theoretic problems in question. In particular, every $K_{2,2}$-free semi-algebraic graph of n vertices with the complexity parameters associated with the point-line incidence problem has $O(n^{4/3})$ edges. This implies the Szemerédi-Trotter theorem.

There is a fast growing body of literature demonstrating that many important results in extremal combinatorics can be substantially improved, and several interesting conjectures proved, if we restrict our attention to semi-algebraic graphs and hypergraphs; see, e.g., [1, 7, 9]. It is a major unsolved problem to decide whether this partly algebraic and partly geometric assumption can be relaxed and replaced by a purely combinatorial condition. A natural candidate is that the graph has *bounded Vapnik-Chervonenkis dimension* (in short, VC-dimension). The *VC-dimension of a set system* (hypergraph) $\mathcal{F}$ on the ground set V is the *largest* integer d for which there exists a d-element set $S \subset V$ such that for every subset $B \subset S$, one can find a member $A \in \mathcal{F}$ with $A \cap S = B$. The *VC-dimension of a graph* $G = (V, E)$ is the VC-dimension of the set system formed by the neighborhoods of the vertices, where the neighborhood of $v \in V$ is $N(v) = \{u \in v : uv \in E\}$. The VC-dimension, introduced by Vapnik and Chervonenkis [21], is one of the most useful combinatorial parameters that measures the complexity of graphs and hypergraphs. It proved to be relevant in many branches of pure and applied mathematics, including statistics, logic, learning theory, and real algebraic geometry. It has completely transformed combinatorial and computational geometry after its introduction to the subject by Haussler and Welzl [14] in 1987. But can it be applied to extremal graph theory problems?

At first glance, this looks rather unlikely. Returning to our initial example, it is easy to verify that the Vapnik-Chervonenkis dimension of every $K_{2,2}$-free graph is at most 2. Therefore, we cannot possibly improve the $O(n^{3/2})$ upper bound on the number of edges of $K_{2,2}$-free graphs by restricting our attention to graphs of bounded VC-dimension. Yet the goal of the present note is to solve the *Schur-Erdős problem*, one of the oldest open questions in Ramsey theory, by placing this restriction.

To describe the problem, we need some notation. For integers $k \geq 3$ and $m \geq 2$, the *Ramsey number* $r(k; m)$ is the smallest integer n such that any m-coloring of the edges of the complete n-vertex graph contains a monochromatic copy of K_k. For the special case when $k = 3$, Issai Schur [19] showed that

$$\Omega(2^m) \leq r(3; m) \leq O(m!).$$

While the form of the upper bound has remained unchanged over the last century, the lower bound was successively improved. The current record is due to Xiaodong *et al.* [22] who showed that $r(3; m) \geq \Omega(3.199^m)$. It is a major open problem in Ramsey theory to close the gap between the lower and upper bounds for $r(3; m)$. Erdős [3] offered cash prizes for solutions to the following problems.

▶ **Conjecture 1** ($100). *We have* $\lim_{m\to\infty} (r(3; m))^{1/m} < \infty$.

It was shown by Chung [4] that $r(3; m)$ is supermultiplicative, so that the above limit exists.

▶ **Problem 2** ($250). *Determine* $\lim_{m\to\infty} (r(3; m))^{1/m}$.

It will be more convenient to work with the *dual VC-dimension*. The *dual* of a set system $\mathcal{F}$ is the set system $\mathcal{F}^*$ obtained by interchanging the roles of V and $\mathcal{F}$. That is, the ground set of $\mathcal{F}^*$ is $\mathcal{F}$, and

$$\mathcal{F}^* = \{\{A \in \mathcal{F} : v \in A\} : v \in V\}.$$

We say that $\mathcal{F}$ has *dual VC-dimension* d if $\mathcal{F}^*$ has VC-dimension d. Notice that $(\mathcal{F}^*)^* = \mathcal{F}$, and it is known that if $\mathcal{F}$ has VC-dimension d, then $\mathcal{F}^*$ has VC-dimension at most $2^{d+1}-1$ (see [16]). In particular, the VC-dimension of $\mathcal{F}$ is bounded if and only if the dual VC-dimension is.

Let χ be an m-coloring of the edges of the complete graph K_n with colors $q_1, \dots, q_m$, and let V be the vertex set of K_n. For $v \in V$ and $i \in [m]$, let $N_{q_i}(v) \subset V$ denote the neighborhood of v with respect to the edges colored with color q_i. We say that χ has *VC-dimension* (or *dual VC-dimension*) d if the set system $\mathcal{F} = \{N_{q_i}(v) : i \in [m], v \in V\}$ has VC-dimension (resp., dual VC-dimension) d.

For $k \geq 3$, $m \geq 2$, and $d \geq 2$, let $r_d(k; m)$ be the smallest integer n such that any m-coloring χ of the edges of K_n with dual VC-dimension at most d contains a monochromatic clique of size k. Even for m-colorings with dual VC-dimension 2, we have $r_2(3; m) = 2^{\Omega(m)}$. Indeed, recursively take two disjoint copies of $K_{2^{m-1}}$, each of which is $(m-1)$-colored with dual VC-dimension at most 2 and no monochromatic copy of K_3. Color all edges between these complete graphs with the mth color, to obtain an m-colored complete graph K_{2^m} with the desired properties. Our main result shows that, apart from a constant factor in the exponent, this construction is tight.

▶ **Theorem 3.** *For every $k \geq 3$ and $d \geq 2$, there is a constant $c = c(k, d)$ such that $r_d(k; m) \leq 2^{cm}$. In other words, for every m-coloring of the edges of a complete graph of 2^{cm} vertices with dual VC-dimension d, there is a monochromatic complete subgraph of k vertices.*

It follows from the Milnor-Thom theorem [17] (proved 15 years earlier by Petrovskiĭ and Oleĭnik [18]) that every m-coloring of the $\binom{n}{2}$ pairs induced by n points in $\mathbb{R}^d$, which is *semi-algebraic* with bounded complexity, has bounded VC-dimension and, hence, bounded dual VC-dimension.

In a recent paper [11], we proved Conjecture 1 for semi-algebraic m-colorings of bounded complexity. Our proof heavily relied on the topology of Euclidean spaces: it was based on the cutting lemma of Chazelle *et al.* [2] and vertical decomposition. These arguments break down in the combinatorial setting, for m-colorings of bounded VC-dimension. In what follows, instead of using "regular" space decompositions with respect to a set of polynomials, our main tool will be a partition result for abstract hypergraphs, which can be easily deduced from the dual of Haussler's packing lemma [13]. The proof of this partition result will be given in Section 2, while Section 3 contains the proof of Theorem 3.

Overview of the proof. We briefly sketch the idea of the proof of the main theorem. Let χ be an m-coloring of the edges of the complete graph K_n with colors $q_1, \dots, q_m$, and let V be the vertex set of K_n. Set $\mathcal{F} = \{N_{q_i}(v) : i \in [m], v \in V\}$. Assuming that $\mathcal{F}$ has bounded VC-dimension, we apply a partition lemma to $\mathcal{F}$ discussed in the next section to obtain a vertex partition $V = S_1 \cup \dots \cup S_r$ such that for any pair of vertices $\{u, v\}$ that lies in the same part, very few sets in $\mathcal{F}$ will cross $\{u, v\}$, that is, contain one vertex but not the other. As a consequence, if any part S_t contains a monochromatic K_{k-1} in color q_i, and there is a vertex $v \in S_t$ with large degree with respect to color q_i, then a vertex $u \in N_{q_i}(v)$ can be added to K_{k-1} to produce a monochromatic K_k in color q_i and we are done. Moreover, if there is a "large" part S_t which does not contain a monochromatic K_{k-1} with respect to many of the colors in $\{q_1, \dots, q_m\}$, then we are also done, by induction. If there is no large part with the above property, then the colors of nearly all edges can be "recovered" by much fewer sets in $\mathcal{F}$. Now we can repeat the argument above on this smaller collection of sets $\mathcal{F}' \subset \mathcal{F}$.

To simplify the presentation, throughout this paper we omit the floor and ceiling signs whenever they are not crucial. All logarithms are in base 2.

2 A partition lemma

Let $\mathcal{F}$ be a set system with dual VC-dimension d and with ground set V. Given two points $u, v \in V$, we say that a set $A \in \mathcal{F}$ *crosses* the pair $\{u, v\}$ if A contains at least one member of $\{u, v\}$, but not both. We say that the set $X \subset V$ is *δ-separated* if for any two points $u, v \in X$, there are at least δ sets in $\mathcal{F}$ that cross the pair $\{u, v\}$. The following *packing lemma* was proved by Haussler in [13].

▶ **Lemma 4.** *Let $\mathcal{F}$ be a set system on a ground set V such that $\mathcal{F}$ has dual VC-dimension d. If $X \subset V$ is δ-separated, then $|X| \leq c_1(|\mathcal{F}|/\delta)^d$ where $c_1 = c_1(d)$.*

As an application of Lemma 4, we obtain the following partition lemma.

▶ **Lemma 5.** *Let $\mathcal{F}$ be a set system on a ground set V such that $|V| = n$ and $\mathcal{F}$ has dual VC-dimension d. Then there is a constant $c_2 = c_2(d)$ such that for any δ satisfying $1 \leq \delta \leq |\mathcal{F}|$, there is a partition $V = S_1 \cup \cdots \cup S_r$ of V into $r \leq c_2(|\mathcal{F}|/\delta)^d$ parts, each of size at most $\frac{2n}{c_1(|\mathcal{F}|/\delta)^d}$, such that any pair of vertices from the same part S_t is crossed by at most 2δ members of $\mathcal{F}$. (Here $c_1 = c_1(d)$ is the same constant as in Lemma 4.*

Proof. Let $X = \{x_1, \ldots, x_{r'}\}$ be a maximal subset of V that is δ-separated with respect to $\mathcal{F}$. By Lemma 4, $|X| = r' \leq c_1(|\mathcal{F}|/\delta)^d$. We define a partition $V = S'_1 \cup \cdots \cup S'_{r'}$ of the vertex set such that $x_i \in S'_i$, and for $v \in V \setminus X$, $v \in S'_i$ if i is the smallest index such that the number of sets from $\mathcal{F}$ that cross the pair $\{v, x_i\}$ is at most δ. Such an i always exists since X is maximal. By the triangle inequality, for any two vertices $u, v \in S'_i$, there are at most 2δ sets in $\mathcal{F}$ that cross the pair $\{u, v\}$.

If a part S'_i has size more than $\frac{2n}{c_1(|\mathcal{F}|/\delta)^d}$, we partition S'_i (arbitrarily) into parts of size $\left\lfloor \frac{2n}{c_1(|\mathcal{F}|/\delta)^d} \right\rfloor$ and possibly one additional part of size less than $\left\lfloor \frac{2n}{c_1(|\mathcal{F}|/\delta)^d} \right\rfloor$. Let $\mathcal{P} : V = S_1 \cup \cdots \cup S_r$ be the resulting partition, where $r \leq c_2(|\mathcal{F}|/\delta)^d$ and $c_2 = c_2(d)$. Then $\mathcal{P}$ satisfies the above properties. ◀

3 Proof of Theorem 3

Let $d, k_1, \ldots, k_m$ be positive integers. We define the Ramsey number $r_d(k_1, \ldots, k_m)$ to be the smallest integer n with the following property. For any m-coloring χ of the edges of K_n with colors $\{q_1, \ldots, q_m\}$ such that χ has dual VC-dimension at most d, there is a monochromatic copy of K_{k_i} in color q_i for some $1 \leq i \leq m$. We now prove the following theorem, from which Theorem 3 immediately follows.

▶ **Theorem 6.** *For fixed integers $d, k \geq 1$, if $k_1, \ldots, k_m \leq k$, then $r_d(k_1, \ldots, k_m) \leq 2^{cm}$ where $c = c(d, k)$.*

Proof. Let $c = c(d, k)$ be a large constant that will be determined later. We will show that $r_d(k_1, \ldots, k_m) \leq 2^{c \sum_{i=1}^m k_i}$ by induction on $s = \sum_{i=1}^m k_i$. The base case $s \leq k2^{16dk}$ follows by setting c to be sufficiently large.

For the inductive step, assume that $s > k2^{16dk}$ and that the statement holds for all $s' < s$. Thus, we have $m \geq 2^{16dk}$. Set $n = 2^{cs}$ and $Q = \{q_1, \ldots, q_m\}$, and let $\chi : E(K_n) \to Q$ be an m-coloring of the edges of K_n with colors $q_1, \ldots, q_m$ such that the set system

$$\mathcal{F} = \{N_{q_i}(v) : v \in V(K_n), q_i \in Q\}$$

has dual VC dimension at most d.

Let $\log^{(j)} m$ denote the j-fold iterated logarithm function, where $\log^{(0)} m = m$ and $\log^{(j)} m = \log(\log^{(j-1)} m)$. For convenience, we will set $\log^{(-1)} m = \infty$. Suppose that there is no color q_i such that χ produces a monochromatic K_{k_i} whose every edge is of color q_i. Otherwise we are done. Then, for every $j \geq 0$ such that $\log^{(j-1)} m > 2^{8dk}$, we recursively construct

1. a set $V_j \subset V$ with $|V_j| \geq n(1 - \frac{1}{\log^{(j-1)} m})$, and
2. an assignment of colors $\chi_j : V_j \to 2^Q$ to each vertex in V_j with the property that the set system $\mathcal{F}_j = \{N_{q_i}(v) \cap V_j : v \in V_j, q_i \in \chi_j(v)\}$ *covers* all but at most $8n^2/\log^{(j-1)} m$ edges of K_n. Here, uv is said to be *covered* by $\mathcal{F}_j$ if $\chi(uv) = q_i$ implies that $q_i \in \chi_j(u) \cap \chi_j(v)$. Moreover, $|\chi_j(v)| \leq \log^{(j)} m$ for all $v \in V_j$.

We start by setting $V_0 = V$, $\chi_0(v) = Q$ for all $v \in V$, and therefore we have $\mathcal{F}_0 = \mathcal{F}$. Suppose we have V_j, χ_j, and $\mathcal{F}_j$ with the properties described above. Before defining the set V_{j+1} and the assignment of colors $\chi_{j+1} : V_{j+1} \to 2^Q$, we introduce $B_j \subset E(K_n)$ to be the set of edges that are not covered by $\mathcal{F}_j$. Hence, $|B_j| \leq \frac{8n^2}{\log^{(j-1)} m}$. We apply Lemma 5 to $\mathcal{F}_j$, whose ground set is V_j, with parameter $\delta = \frac{|\mathcal{F}_j|}{(\log^{(j)} m)^4}$, and obtain a partition $\mathcal{P} : V_j = S_1 \cup \cdots \cup S_r$, where $r \leq c_2(\log^{(j)} m)^{4d}$ and c_2 is defined in Lemma 5, such that $\mathcal{P}$ has the properties described in Lemma 5. For each part $S_t \in \mathcal{P}$, let $Q_t \subset \{q_1, \ldots, q_m\}$ be the set of colors such that $q_i \in Q_t$ if there is a vertex $v \in S_t$ such that

$$|\{u \in V_j : \chi(uv) = q_i, uv \notin B_j\}| \geq \frac{n}{(\log^{(j)} m)^2}.$$

Let $Q'_t \subset \{q_1, \ldots, q_m\}$ be the set of colors such that $q_i \in Q'_t$ if the vertex set S_t contains a monochromatic copy of K_{k_i-1} in color q_i.

▶ **Observation 7.** *If there is a color $q_i \in Q_t \cap Q'_t$, then χ produces a monochromatic copy of K_{k_i} in color q_i.*

Proof. Suppose $q_i \in Q_t \cap Q'_t$ and let $X = \{x_1, \ldots, x_{k_i-1}\} \subset S_t$ be the vertex set of a monochromatic clique of order $k_i - 1$ in color q_i. Fix $v \in S_t$ such that for $U = \{u \in V_j : \chi(uv) = q_i, uv \notin B_j\}$, we have $|U| \geq \frac{n}{(\log^{(j)} m)^2}$. Notice that if $X \not\subset (N_{q_i}(u) \cap V_j)$, where $u \in U$, then the set $(N_{q_i}(u) \cap V_j)$ crosses the pair $\{x, v\}$ for some $x \in X$. Moreover, $(N_{q_i}(u) \cap V_j) \in \mathcal{F}_j$ since $uv \notin B_j$. Since there are at most $2\delta = \frac{2|\mathcal{F}_j|}{(\log^{(j)} m)^4}$ sets in $\mathcal{F}_j$ that cross $\{x, v\}$, there are at most $\frac{2k|\mathcal{F}_j|}{(\log^{(j)} m)^4}$ sets in $\{N_{q_i}(u) \cap V_j : u \in U\} \subset \mathcal{F}_j$ that do not contain X. On the other hand,

$$|U| - k_i \geq \frac{n}{(\log^{(j)} m)^2} - k_i > \frac{2k|\mathcal{F}_j|}{(\log^{(j)} m)^4},$$

where the last inequality follows from the fact that $|\mathcal{F}_j| \leq n \log^{(j)} m$ and $\log^{(j)} m > 2^{8dk}$. Hence, there must be a neighborhood $(N_{q_i}(u) \cap V_j)$ that contains X, which implies that $X \cup \{u\}$ induces a monochromatic copy of K_{k_i} in color q_i. ◀

By the observation above, we can assume that $Q_t \cap Q'_t = \emptyset$ for every t, since otherwise we would be done.

▶ **Observation 8.** *If there is a part $S_t \in \mathcal{P}$ such that $|S_t| \geq n/(\log^{(j)} m)^{6d}$ and $|Q_t| \geq \log^{(j+1)} m$, then S_t contains a monochromatic copy of K_{k_i} in color q_i where $q_i \in Q'_t$.*

Proof. For sake of contradiction, suppose $S_t \in \mathcal{P}$ does not contain a monochromatic copy of K_{k_i} in color $q_i \in Q'_t$. Since $Q_t \cap Q'_t = \emptyset$, S_t also does not contain a monochromatic copy of K_{k_i-1} in color $q_i \in Q_t$. So if $|Q_t| \geq \log^{(j+1)} m$, we have $|Q'_t| \leq m - \log^{(j+1)} m$. By the induction hypothesis, we have

$$\frac{n}{(\log^{(j)} m)^{6d}} \leq |S_t| < 2^{c(s-\log^{(j+1)} m)}.$$

Since $c = c(d,k)$ is sufficiently large, we have $n < 2^{cs}$ which is a contradiction. ◀

Hence, we can assume that for each part $S_t \in \mathcal{P}$ such that $|S_t| \geq n/(\log^{(j)} m)^{6d}$, we have $|Q_t| < \log^{(j+1)} m$.

We now define the set V_{j+1} and the color assignment χ_{j+1} as follows. Let $V_{j+1} \subset V_j$ be the set of vertices $v \in V_j$ such that v does not lie in a part S_t such that $|S_t| < n/(\log^{(j)} m)^{6d}$. Hence,

$$\begin{aligned} |V_{j+1}| &\geq |V_j| - c_2(\log^{(j)} m)^{4d} \tfrac{n}{(\log^{(j)} m)^{6d}} \\ &\geq n - \tfrac{n}{\log^{(j-1)} m} - \tfrac{c_2 n}{(\log^{(j)} m)^{2d}} \\ &\geq n - \tfrac{n}{\log^{(j)} m}. \end{aligned}$$

For each vertex $v \in V_{j+1}$, v lies in a part $S_t \in \mathcal{P}$ with $|S_t| \geq n/(\log^{(j)} m)^{6d}$. We set $\chi_{j+1}(v) = Q_t$, and by the observation above, $|\chi_{j+1}(v)| \leq \log^{(j+1)} m$. Thus, we have the set system $\mathcal{F}_{j+1} = \{N_{q_i}(v) \cap V_{j+1} : v \in V_{j+1}, q_i \in \chi_{j+1}(v)\}$ such that $|\mathcal{F}_{j+1}| \leq n \log^{(j+1)} m$.

Finally, it remains to show that $\mathcal{F}_{j+1}$ covers at least $\binom{n}{2} - \frac{8n^2}{\log^{(j)} m}$ edges of K_n. Let $B_{j+1} \subset E(K_n)$ denote the set of edges that are not covered by $\mathcal{F}_{j+1}$. If $uv \in B_{j+1}$, then either

1. $uv \in B_j$, or
2. u (or v) lies inside a part $S_t \in \mathcal{P}$ such that $|S_t| \leq \frac{n}{(\log^{(j)} m)^{6d}}$, or
3. both u and v lie inside the same part $S_t \in \mathcal{P}$, or
4. uv is covered by $\mathcal{F}_j$, but is not covered by $\mathcal{F}_{j+1}$ since $v \in S_t \in \mathcal{P}$ and $\chi(u,v) \notin Q_t$.

By assumption,

$$|B_j| \leq \frac{8n^2}{\log^{(j-1)} m}. \tag{1}$$

The number of edges of the second type is at most

$$\frac{n^2}{(\log^{(j)} m)^{6d}}. \tag{2}$$

The number of edges of the third type is at most

$$\sum_{i=1}^{r} \binom{|S_t|}{2} \leq c_2(\log^{(j)} m)^{4d} \left(\frac{2n}{c_1(\log^{(j)} m)^{4d}}\right)^2 = \frac{4c_2 n^2}{(c_1)^2(\log^{(j)} m)^{4d}}, \tag{3}$$

where c_1 is defined in Lemma 4. Finally, let us bound the number of edges of the fourth type. Fix $v \in S_t \in \mathcal{P}$ such that $|S_t| > \frac{n}{(\log^{(j)} m)^{6d}}$, and let us consider all edges incident to v that are covered by $\mathcal{F}_j$. Since v contributed at most $\log^{(j)} m$ sets in $\mathcal{F}_j$, there are at most $\log^{(j)} m$ distinct colors among these edges. Fix such a color q_i such that $q_i \notin Q_t$, and consider the set of vertices

$$U = \{u \in V_{j+1} : \chi(uv) = q_i, uv \notin B_j\}.$$

By definition of Q_t, we have $|U| < \frac{n}{(\log^{(j)} m)^2}$. Therefore, the number of edges incident to v of the fourth type is at most

$$\log^{(j)} m \frac{n}{(\log^{(j)} m)^2} = \frac{n}{\log^{(j)} m}.$$

Hence, the total number of edges of the fourth type is at most

$$\frac{n^2}{\log^{(j)} m}. \tag{4}$$

Thus by summing (1), (2), (3), (4), and using the fact that $\log^{(j-1)} m > 2^{8dk}$, we have

$$|B_{j+1}| \leq \frac{8n^2}{\log^{(j-1)} m} + \frac{n^2}{(\log^{(j)} m)^{6d}} + \frac{4c_2 n^2}{(c_1)^2 (\log^{(j)} m)^{4d}} + \frac{n^2}{\log^{(j)} m} < \frac{8n^2}{\log^{(j)} m}.$$

Hence, $\mathcal{F}_{j+1}$ covers at least $\binom{n}{2} - \frac{8n^2}{\log^{(j)} m}$ edges of K_n.

Let w be the minimum integer such that $\log^{(w)} m \leq 2^{8dk}$. Then we have $V_w, \chi_w, \mathcal{F}_w$ with the properties described above. Just as before, let $B_w \subset E(K_n)$ be the set of edges not covered by $\mathcal{F}_w$. This implies $|B_w| \leq n^2/2^{8dk} < n^2/8$ and $|V_w| \geq 7n/8$. Since

$$\binom{7n/8}{2} - \frac{n^2}{8} \geq \frac{n^2}{4},$$

an averaging argument shows that there is a vertex $v \in V_w$ that is incident to at least $n/2$ edges that are covered by $\mathcal{F}_w$. Since v contributes at most $\log^{(w)} m < 2^{8dk}$ sets in $\mathcal{F}_w$, there is a color q_i such that

$$|N_{q_i}(v)| \geq \frac{n}{2 \cdot 2^{8dk}} = \frac{2^{cs}}{2 \cdot 2^{8dk}} \geq 2^{c(s-1)},$$

where the second inequality follows from the fact that $c = c(d,k)$ is sufficiently large. Therefore, by induction, the set $N_{q_i}(v) \subset V$ contains a monochromatic copy of K_{k_ℓ} in color $q_\ell \in \{q_1, \ldots, q_m\} \setminus q_i$, in which case we are done, or contains a monochromatic copy of $K_{k_i - 1}$ in color q_i. In the latter case, we obtain a monochromatic K_{k_i} in color q_i by including vertex v. This completes the proof of Theorem 6. ◀

4 Concluding remarks

We have established tight bounds for multicolor Ramsey numbers for graphs with bounded VC-dimension. It would be interesting to prove other well-known conjectures in extremal graph theory for graphs and hypergraphs with bounded VC-dimension, especially the notorious Erdős-Hajnal conjecture.

An old result of Erdős and Hajnal [6] states that for every hereditary property P which is not satisfied by all graphs, there exists a constant $\varepsilon(P) > 0$ such that every graph of n vertices with property P has a clique or an independent set of size at least $e^{\varepsilon(P)\sqrt{\log n}}$. They conjectured that this bound can be improved to $n^{\varepsilon(P)}$. Thus, every graph G on n vertices with bounded VC-dimension contains a clique or an independent set of size $e^{\Omega(\sqrt{\log n})}$. In [10], the authors improved this bound to $e^{(\log n)^{1-o(1)}}$. However, the following conjecture remains open.

▶ **Conjecture 9.** *For $d \geq 2$, there exists a constant $\varepsilon(d)$ such that every graph on n vertices with VC-dimension at most d contains a clique or an independent set of size $n^{\varepsilon(d)}$.*

References

1 N. Alon, J. Pach, R. Pinchasi, R. Radoičić, and M. Sharir. Crossing patterns of semi-algebraic sets. *J. Comb. Theory, Ser. A*, 111:310–326, 2005. doi:10.1016/j.jcta.2004.12.008.

2 B. Chazelle, H. Edelsbrunner, L. J. Guibas, and M. Sharir. A singly exponential stratification scheme for real semi-algebraic varieties and its applications. *Theor. Comput. Sci.*, 84:77–105, 1991. doi:10.1016/0304-3975(91)90261-Y.

3 F. Chung and R. Graham. *Erdős on Graphs: His Legacy of Unsolved Problems*. A K Peters Series. Taylor & Francis, 1998. URL: https://books.google.com/books?id=Doc_ErUTDcAC.

4 F. R. K. Chung. On the ramsey numbers $n(3,3,...3;2)$. *Discret. Math.*, 5:317–321, 1973. doi:10.1016/0012-365X(73)90125-8.

5 Paul Erdős. On sequences of integers no one of which divides the product of two others and on some related problems. *Inst. Math. Mech. Univ. Tomsk*, 2:74–82, 1938.

6 P. Erdős and A. Hajnal. Ramsey-type theorems. *Discret. Appl. Math.*, 25:37–52, 1989. doi:10.1016/0166-218X(89)90045-0.

7 J. Fox, M. Gromov, V. Lafforgue, A. Naor, and J. Pach. Overlap properties of geometric expanders. *Reine Angew. Math. (Crelle's Journal)*, 2012:49–83, 2010. doi:10.1515/CRELLE.2011.157.

8 J. Fox, J. Pach, A. Sheffer, A. Suk, and J. Zahl. A semi-algebraic version of Zarankiewicz's problem. *J. Eur. Math. Soc. (JEMS)*, 19:1785–1810, 2014. doi:10.4171/JEMS/705.

9 J. Fox, J. Pach, and A. Suk. A polynomial regularity lemma for semialgebraic hypergraphs and its applications in geometry and property testing. *SIAM J. Computing*, 45:2199–2223, 2015. doi:10.1137/15M1007355.

10 J. Fox, J. Pach, and A. Suk. Erdős-Hajnal conjecture for graphs with bounded VC-dimension. *Discret. Comput. Geom.*, 61:809–829, 2019. doi:10.1007/s00454-018-0046-5.

11 J. Fox, J. Pach, and A. Suk. The Schur-Erdős problem for semi-algebraic colorings. *Israel J. Mathematics*, to appear.

12 L. Guth. *Polynomial Methods in Combinatorics*. University Lecture Series. Amer. Math. Soc., 2016.

13 D. Haussler. Sphere packing numbers for subsets of the boolean n-cube with bounded Vapnik-Chervonenkis dimension. *J. Comb. Theory, Ser. A*, 69:217–232, 1995.

14 D. Haussler and E. Welzl. epsilon-nets and simplex range queries. *Discret. Comput. Geom.*, 2:127–151, 1987. doi:10.1007/BF02187876.

15 T. Kóvari, V. Sós, and P. Turán. On a problem of K. Zarankiewicz. *Colloq. Math.*, 3:50–57, 1954. URL: http://eudml.org/doc/210011.

16 J. Matouvsek. *Lectures on discrete geometry*, volume 212 of *Graduate texts in mathematics*. Springer, 2002.

17 J. Milnor. On the betti numbers of real varieties. *Proc. Amer. Math. Soc.*, 15:257–280, 1964.

18 I. G. Petrovskiĭ and O. A. Oleĭnik. On the topology of real algebraic surfaces (russian). *Izv. Akad. Nauk SSSR Ser. Mat.*, 13:389–402, 1949.

19 I. Schur. Über die Kongruenz $x^m + y^m = z^m (mod\ p)$. *Jber. Deutsch. Math. Verein.*, 25:114–116, 1917. URL: http://eudml.org/doc/145475.

20 E. Szemerédi and W. T. Trotter. Extremal problems in discrete geometry. *Combinatorica*, 3:381–392, 1983. doi:10.1007/BF02579194.

21 V. N. Vapnik and A. Ya. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory Probab. Appl.*, 16:264–280, 1971. doi:10.1137/1116025.

22 X. Xu, X. Zheng, G. Exoo, and S. P. Radziszowski. Constructive lower bounds on classical multicolor ramsey numbers. *Electr. J. Comb.*, 11, 2004. URL: http://www.combinatorics.org/Volume_11/Abstracts/v11i1r35.html.

Almost-Monochromatic Sets and the Chromatic Number of the Plane

Nóra Frankl
Department of Mathematics, London School of Economics and Political Science, UK
Laboratory of Combinatorial and Geometric Structures at MIPT, Moscow, Russia
n.frankl@lse.ac.uk

Tamás Hubai
MTA-ELTE Lendület Combinatorial Geometry Research Group, Institute of Mathematics, Eötvös Loránd University (ELTE), Budapest, Hungary
htamas@cs.elte.hu

Dömötör Pálvölgyi
MTA-ELTE Lendület Combinatorial Geometry Research Group, Institute of Mathematics, Eötvös Loránd University (ELTE), Budapest, Hungary
dom@cs.elte.hu

Abstract

In a colouring of $\mathbb{R}^d$ a pair (S, s_0) with $S \subseteq \mathbb{R}^d$ and with $s_0 \in S$ is *almost-monochromatic* if $S \setminus \{s_0\}$ is monochromatic but S is not. We consider questions about finding almost-monochromatic similar copies of pairs (S, s_0) in colourings of $\mathbb{R}^d$, $\mathbb{Z}^d$, and of $\mathbb{Q}$ under some restrictions on the colouring.

Among other results, we characterise those (S, s_0) with $S \subseteq \mathbb{Z}$ for which every finite colouring of $\mathbb{R}$ without an infinite monochromatic arithmetic progression contains an almost-monochromatic similar copy of (S, s_0). We also show that if $S \subseteq \mathbb{Z}^d$ and s_0 is outside of the convex hull of $S \setminus \{s_0\}$, then every finite colouring of $\mathbb{R}^d$ without a monochromatic similar copy of $\mathbb{Z}^d$ contains an almost-monochromatic similar copy of (S, s_0). Further, we propose an approach based on finding almost-monochromatic sets that might lead to a human-verifiable proof of $\chi(\mathbb{R}^2) \geq 5$.

2012 ACM Subject Classification Mathematics of computing → Graph coloring; Theory of computation → Computational geometry

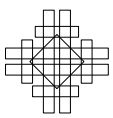
Keywords and phrases discrete geometry, Hadwiger-Nelson problem, Euclidean Ramsey theory

Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.47

Related Version A full version of this paper is available at [6] `https://arxiv.org/abs/1912.02604`.

Funding *Nóra Frankl*: Research was part of project 2018-2.1.1-UK_GYAK-2018-00024 of the summer internship for Hungarian students studying in the UK, supported by the National Research, Development and Innovation Office. She also acknowledges the financial support from the Ministry of Education and Science of the Russian Federation in the framework of MegaGrant no 075-15-2019-1926. The research was also partially supported by the National Research, Development, and Innovation Office, NKFIH Grant K119670.
Tamás Hubai: Research supported by the Lendület program of the Hungarian Academy of Sciences (MTA), under grant number LP2017-19/2017.
Dömötör Pálvölgyi: Research supported by the Lendület program of the Hungarian Academy of Sciences (MTA), under grant number LP2017-19/2017.

Acknowledgements We thank Konrad Swanepoel and the anonymous referees for helpful suggestions on improving the presentation of the paper. We also thank the participants of the Polymath16 project for discussions and consent to publish these related results that were obtained "offline" and separately from the main project.



1 Introduction

A colouring $\varphi : \mathbb{R}^2 \to [k]$ is a (unit-distance-avoiding) *proper k-colouring* of the plane, if $\|p - q\| = 1$ implies $\varphi(p) \neq \varphi(q)$. The *chromatic number $\chi(\mathbb{R}^2)$ of the plane* is the smallest k for which there exists a proper k-colouring of the plane. Determining the exact value of $\chi(\mathbb{R}^2)$, also known as the Hadwiger-Nelson problem, is a difficult problem. In 2018 Aubrey de Grey [1] showed that $\chi(\mathbb{R}^2) \geq 5$, improving the long standing previous lower bound $\chi(\mathbb{R}^2) \geq 4$ which was first noted by Nelson (see [13]). The best known upper bound $\chi(\mathbb{R}^2) \leq 7$ was first observed by Isbell (see [13]), and it is widely conjectured that $\chi(\mathbb{R}^2) = 7$. For history and related results we refer the reader to Soifer's book [13].

A graph $G = (V, E)$ is a *unit-distance graph* in the plane if $V \subseteq \mathbb{R}^2$ such that if $(v, w) \in E$ then $\|v - w\| = 1$. De Grey constructed a unit-distance graph G with 1581 vertices, and checked that $\chi(G) \geq 5$ by a computer program. Following his breakthrough, a polymath project, Polymath16 [2] was launched with the main goal of finding a human-verifiable proof of $\chi(\mathbb{R}^2) \geq 5$. Following ideas proposed in Polymath16 by the third author [10], we present an approach that might lead to a human-verifiable proof of $\chi(\mathbb{R}^2) \geq 5$.

We call a collection of unit circles $C = C_1 \cup \cdots \cup C_n$ having a common point O a *bouquet through O*. For a given colouring of $\mathbb{R}^2$, the bouquet C is *smiling* if there is a colour, say blue, such that every circle C_i has a blue point, but O is not blue.

▶ **Conjecture 1.** *For every bouquet C, every colouring of the plane with finitely many but at least two colours contains a smiling congruent copy of C.*

In Section 4 we show that the statement of Conjecture 1 would provide a human-verifiable proof of $\chi(\mathbb{R}^2) \geq 5$. We prove the conjecture for a specific family of bouquets.

▶ **Theorem 2.** *Let $C = C_1 \cup \cdots \cup C_n$ be a bouquet through O and for every i let O_i be the centre of C_i. If O and $O_1, \ldots, O_n$ are contained in $\mathbb{Q}^2$, further O is an extreme point of $\{O, O_1, \ldots, O_n\}$, then Conjecture 1 is true for C for every proper colouring.*

In Section 4.2 we prove a more general statement which implies Theorem 2. We also prove a statement similar to that of Conjecture 1 for concurrent lines. We call a collection of lines $L = L_1 \cup \cdots \cup L_n$ with a common point O a *pencil through O*. The pencil L is *smiling* if there is a colour, say blue, such that every line L_i has a blue point, but O is not blue.

▶ **Theorem 3.** *For every pencil L, every colouring of the plane with finitely many but at least two colours contains a smiling congruent copy of L.*

1.1 Almost-monochromatic sets

Let $S \subseteq \mathbb{R}^d$ be a finite set with $|S| \geq 3$, and let $s_0 \in S$. In a colouring of $\mathbb{R}^d$ we call S *monochromatic*, if every point of S has the same colour. A pair (S, s_0) is *almost-monochromatic* if $S \setminus \{s_0\}$ is monochromatic but S is not.

We call a colouring a *finite colouring*, if it uses finitely many colours. An *infinite arithmetic progression* in $\mathbb{R}^d$ is a similar copy of $\mathbb{N}$. A colouring is *arithmetic progression-free* if it does not contain a monochromatic infinite arithmetic progression. Motivated by its connections to the chromatic number of the plane,[1] we propose to study the following problem.

[1] The connection is described in details in the the proof of Theorem 24.

▶ **Problem 4.** *Characterise those pairs (S, s_0) with $S \subseteq \mathbb{R}^d$ and with $s_0 \in S$ for which it is true that every arithmetic progression-free finite colouring of $\mathbb{R}^d$ contains an almost-monochromatic similar copy of (S, s_0).*

Note that finding an almost-monochromatic *congruent* copy of a given pair (S, s_0) was studied by Erdős, Graham, Montgomery, Rothschild, Spencer, and Strauss [4]. We solve Problem 4 in the case when $S \subseteq \mathbb{Z}^d$. A point $s_0 \in S$ is called an *extreme point* of S if $s_0 \notin \text{conv}(S \setminus \{s_0\})$. From now on we will use the abbreviations AM for almost-monochromatic and AP for arithmetic progression.

▶ **Theorem 5.** *Let $S \subseteq \mathbb{Z}^d$ and $s_0 \in S$. Then there is an AP-free colouring of $\mathbb{R}^d$ without an AM similar copy of (S, s_0) if and only if $|S| > 3$ and s_0 is not an extreme point of S.*

We prove Theorem 5 in full generality in Section 3.1. The 'only if' direction follows from a stronger statement, Theorem 16. In Section 2 we consider the 1-dimensional case. We prove some statements similar to Theorem 5 for $d = 1$, and illustrate the ideas that are used to prove the theorem in general.

Problem 4 is related to and motivated by Euclidean Ramsey theory, a topic introduced by Erdős, Graham, Montgomery, Rothschild, Spencer, and Strauss [3]. Its central question asks for finding those finite sets $S \subseteq \mathbb{R}^d$ for which the following is true. *For every k if d is sufficiently large, then every colouring of $\mathbb{R}^d$ using at most k colours contains a monochromatic congruent copy of S.* Characterising sets having the property described above is a well-studied difficult question, and is in general wide open. For a comprehensive overview see Graham's survey [7].

The nature of the problem significantly changes if instead of a monochromatic congruent copy we ask for a monochromatic similar copy, or a monochromatic homothetic copy. A *(positive) homothetic copy* (or *(positive) homothet*) of a set $H \subseteq \mathbb{R}^d$ is a set $c + \lambda H = \{c + \lambda h : h \in H\}$ for some $c \in \mathbb{R}^d$ and some (positive) $\lambda \in \mathbb{R} \setminus \{0\}$. Gallai proved that if $S \subseteq \mathbb{R}^d$ is a finite set, then every colouring of $\mathbb{R}^d$ using finitely many colours contains a monochromatic positive homothetic copy of S. This statement first appeared in the mentioned form in the book of Graham, Rothschild, and Spencer [8].

A direct analogue of Gallai's theorem for AM sets is not true: there is no AM similar copy of any (S, s_0) if the whole space is coloured with one colour only. However, there are pairs (S, s_0) for which a direct analogue of Gallai's theorem is true for colourings of $\mathbb{Q}$ with more than one colour. In particular, we prove the following result in the full version of the paper [6]. (This result is not used elsewhere in the paper.)

▶ **Theorem 6.** *Let $S = \{0, 1, 2\}$ and $s_0 = 0$. Then every finite colouring of $\mathbb{Q}$ with more than one colour contains an AM positive homothet of (S, s_0).*

In general, we could ask whether every non-monochromatic colouring of $\mathbb{R}^d$ with finitely many colours contains an AM similar copy of every (S, s_0). This, however, is false, as shown by the following example from [4]. Let $S = \{1, 2, 3\}$ and $s_0 = 2$. If $\mathbb{R}_{>0}$ is coloured red and $\mathbb{R}_{\leq 0}$ is coloured blue, we obtain a colouring of $\mathbb{R}$ without an AM similar copy of (S, s_0). Restricting the colouring to $\mathbb{N}$, using the set of colours $\{0, 1, 2\}$ and colouring every $n \in \mathbb{N}$ with n modulo 3, we obtain a colouring without an AM similar copy of (S, s_0). However, notice that in both examples each colour class contains an infinite monochromatic AP.

Therefore, our reason, apart from its connections to the Hadwiger-Nelson problem, for finding AM similar copies of (S, s_0) in AP-free colourings was to impose a meaningful condition to exclude "trivial" colourings.

2 The line

In this section we prove a statement slightly weaker than Theorem 5 for $d = 1$. The main goal of this section to illustrate some of the ideas that we use to prove Theorem 5, but in a simpler case. Note that in $\mathbb{R}$ the notion of similar copy and homothetic copy is the same.

▶ **Theorem 7.** *Let $S \subseteq \mathbb{Z}$ and $s_0 \in S$. Then there is an AP-free colouring of $\mathbb{N}$ and of $\mathbb{R}$ without an AM positive homothetic copy of (S, s_0) if and only if $|S| > 3$ and s_0 is not an extreme point of S.*

To prove Theorem 7 it is sufficient to prove the "if" direction only for $\mathbb{R}$ and the "only if" direction only for $\mathbb{N}$. Thus it follows from the three lemmas below, that consider cases of Theorem 7 depending on the cardinality of S and on the position of s_0.

▶ **Lemma 8.** *If s_0 is an extreme point of S, then every finite AP-free colouring of $\mathbb{N}$ contains an AM positive homothetic copy of (S, s_0).*

▶ **Lemma 9.** *If $|S| = 3$, then every AP-free finite colouring of $\mathbb{N}$ contains an AM positive homothetic copy of (S, s_0).*

▶ **Lemma 10.** *If $S \subseteq \mathbb{R}$, $|S| > 3$ and s_0 is not an extreme point of S, then there is an AP-free finite colouring of $\mathbb{R}$ without an AM positive homothetic copy of (S, s_0).*

Before turning to the proofs, recall Van der Waerden's theorem [14] and a corollary of it. A colouring is a *k-colouring* if it uses at most k colours.

▶ **Theorem 11** (Van der Waerden [14])**.** *For every $k, l \in \mathbb{N}$ there is an $N(k, l) \in \mathbb{N}$ such that every k-colouring of $\{1, \dots, N(k, l)\}$ contains an l-term monochromatic AP.*

▶ **Corollary 12** (Van der Waerden [14])**.** *For every $k, l \in \mathbb{N}$ and for every k-colouring of $\mathbb{N}$ there is a $t \leq N(k, l)$ such that there are infinitely many monochromatic l-term AP of the same colour with difference t.*

Proof of Lemma 8. Let $S = \{p_1, \dots, p_n\}$ with $1 < p_1 < \dots < p_n$ and φ be an AP-free colouring of $\mathbb{N}$. If s_0 is an extreme point of S, then either $s_0 = p_1$ or $s_0 = p_n$.

Case 1: $s_0 = p_n$. By Theorem 11 φ contains a monochromatic positive homothet $M + \lambda([1, p_n) \cap \mathbb{N})$ of $[1, p_n) \cap \mathbb{N}$ of colour, say, blue. Observe that since φ is AP-free there is a $q \in M + \lambda([p_n, \infty) \cap \mathbb{N})$ which is not blue. Let $M + q\lambda$ be the smallest non-blue element in $M + \lambda([p_n, \infty) \cap \mathbb{N})$. Then $(\lambda(q - p_n) + M + \lambda S, M + \lambda q)$ is an AM homothet of (S, s_0).

Case 2: $s_0 = p_1$. By Corollary 12 there is a $\lambda \in \mathbb{N}$ such that φ contains infinitely many monochromatic congruent copies of $\lambda((1, p_n] \cap \mathbb{N})$, say of colour blue. Without loss of generality, we may assume that infinitely many of these monochromatic copies are contained in $\lambda\mathbb{N}$. Since φ is AP-free, $\lambda\mathbb{N}$ is not monochromatic, and thus there is an i such that $i\lambda$ and $(i + 1)\lambda$ are of different colours. Consider a blue interval $M + \lambda((1, p_n] \cap \mathbb{N})$ such that $M + \lambda > i\lambda$, and let q be the largest non-blue element of $[1, M + \lambda) \cap \lambda\mathbb{N}$. This largest element exists since λi and $\lambda(i + 1)$ are of different colour. Then $(q - \lambda p_1 + \lambda S, q)$ is an AM homothet of (S, s_0). ◀

Proof of Lemma 9. Let $S = \{p_1, p_2, p_3\}$ with $1 < p_1 < p_2 < p_3$ and φ be an AP-free colouring of $\mathbb{N}$. We may assume that $s_0 = p_2$, otherwise we are done by Lemma 8.

There is an $r \in \mathbb{Q}_{>0}$ such that $\{q_1, q_2, q_3\}$ is a positive homothet of S if and only if $q_2 = rq_1 + (1 - r)q_3$. Fix an $M \in \mathbb{N}$ for which $Mr \in \mathbb{N}$. We say that I is an interval of $c + \lambda\mathbb{N}$ of length ℓ if there is an interval $J \subseteq \mathbb{R}$ such that $I = J \cap (c + \lambda\mathbb{N})$ and $|I| = \ell$.

▶ **Proposition 13.** *Let I_1 and I_3 be intervals of $\lambda\mathbb{N}$ of length $2M$ and M respectively such that $\max I_1 < \min I_3$. Then there is an interval $I_2 \subseteq \lambda\mathbb{N}$ of length M such that $\max I_1 < \max I_2 < \max I_3$, and for every $q_2 \in I_2$ there are $q_1 \in I_1$ and $q_3 \in I_3$ such that $\{q_1, q_2, q_3\}$ is a positive homothetic copy of S.*

Proof. Without loss of generality we may assume that $\lambda = 1$. Let I_1^L be the set of the M smallest elements of I_1. By the choice of M for any $q_3 \in \mathbb{N}$ the interval $rI_1^L + (1-r)q_3$ contains at least one natural number. Let q_3 be the smallest element of I_3 and $q_1 \in I_1^L$ such that $rq_1 + (1-r)q_3 \in \mathbb{N}$. Then $I_2 = \{r(q_1+i) + (1-r)(q_3+i) : 0 \le i < M\}$ is an interval of $\mathbb{N}$ of length M satisfying the requirements, since $q_1 + i \in I_1$ and $q_3 + i \in I_3$. ◀

We now return to the proof of Lemma 9. Let I be an interval of $\mathbb{N}$ of length $2M$. By Theorem 12 there is a $\lambda \in \mathbb{N}$ such that φ contains infinitely many monochromatic copies of λI of the same colour, say of blue. Moreover, by the pigeonhole principle there is a $c \in \mathbb{N}$ such that infinitely many of these blue copies are contained in $c + \lambda\mathbb{N}$, and without loss of generality we may assume that $c = 0$.

Consider a blue interval $[a\lambda, a\lambda + 2M\lambda - \lambda]$ of $\lambda\mathbb{N}$ of length $2M$. Since φ is AP-free, $[a\lambda + 2M\lambda, \infty) \cap \lambda\mathbb{N}$ is not completely blue. Let $q\lambda$ be its smallest element which is not blue and let $I_1 = [q\lambda - 2M\lambda, (q-1)\lambda] \cap \lambda\mathbb{N}$. Let I_3 be the blue interval of length M in $\lambda\mathbb{N}$ with the smallest possible $\min I_3$ for which $\max I_1 < \min I_3$. Then Proposition 13 provides an AM positive homothet of (S, s_0).

Indeed, consider the interval I_2 given by the proposition. There exists a $q_2 \in I_2$ which is not blue, otherwise every point of I_2 is blue, contradicting the minimality of $\min I_3$. But then there are $q_1 \in I_1$, $q_3 \in I_3$ such that $(\{q_1, q_2, q_3\}, q_2)$ is an AM homothet copy of (S, s_0). ◀

Proof of Lemma 10. S contains a set S' of 4 points with $s_0 \in S'$ such that s_0 is not an extreme point of S'. Thus we may assume that $S = \{p_1, p_2, p_3, p_4\}$ with $p_1 < p_2 < p_3 < p_4$ and that $s_0 = p_2$ or $s_0 = p_3$. We construct the colouring for these two cases separately. First we construct a colouring φ_1 of $\mathbb{R}_{>0}$ for the case of $s_0 = p_3$, and a colouring φ_2 of $\mathbb{R}_{\ge 0}$ for the case of $s_0 = p_2$. Then we extend the colouring in both cases to $\mathbb{R}$.

Construction of φ_1 ($s_0 = p_3$): Fix K such that $K > \frac{p_4 - p_2}{p_2 - p_1} + 1$ and let $\{0, 1, 2\}$ be the set of colours. We define φ_1 as follows. Colour $(0, 1)$ with colour 2, and for every $i \in \mathbb{N} \cup \{0\}$ colour $[K^i, K^{i+1})$ with i modulo 2. It is not hard to check that φ_1 defined this way is AP-free. Thus we only have to show that it does not contain an AM positive homothet of (S, s_0).

Consider a positive homothet $c + \lambda S = \{r_1, r_2, r_3, r_4\}$ of S with $r_1 < r_2 < r_3 < r_3$. If $\{r_1, r_2, r_3, r_4\} \cap [0, 1) \neq \emptyset$, then $\{r_1, r_2, r_3, r_4\}$ cannot be AM. Thus we may assume that $\{r_1, r_2, r_3, r_4\} \cap [0, 1) = \emptyset$.

Note that by the choice of K we have

$$Kr_2 > r_2 + \frac{p_4 - p_2}{p_2 - p_1} r_2 = r_2 + \frac{p_4 - p_2}{p_2 - p_1} \left(\lambda(p_2 - p_1) + r_1\right) \ge r_2 + \lambda(p_4 - p_2) = r_4.$$

Hence $\{r_2, r_3, r_4\}$ is contained in the union of two consecutive intervals of the form $[K^i, K^{i+1})$. This means that $(\{r_1, \dots, r_4\}, r_3)$ cannot be AM since either $\{r_2, r_3, r_4\}$ is monochromatic, or r_2 and r_4 have different colours.

Construction of φ_2 ($s_0 = p_2$): Fix K such that $K > \frac{p_4 - p_2}{p_2 - p_1} + 1$, let $L = K \cdot \left\lceil \frac{p_3 - p_1}{p_4 - p_3} \right\rceil$ and let $\{0, \dots, 2L\}$ be the set of colours. We define φ_2 as follows. For each odd $i \in \mathbb{N} \cup \{0\}$, divide the interval $[L \cdot K^i, L \cdot K^{i+1})$ into L equal half-closed intervals, and colour the j-th of them with colour j. For even $i \in \mathbb{N} \cup \{0\}$ divide the interval $[L \cdot K^i, L \cdot K^{i+1})$ into L equal

half-closed intervals, and colour the j-th of them with colour $L+j$. That is, for $j=1,\dots,L$ we colour $[L\cdot K^i+(j-1)(K^{i+1}-K^i), L\cdot K^i+j(K^{i+1}-K^i))$ with colour j if i is odd, and with colour $j+L$ if i is even. Finally, colour the points in $[0,L)$ with colour 0.

It is not hard to check that φ_2 defined this way is AP-free, thus we only have to show it does not contain an AM positive homothetic copy of (S,s_0).

Consider a positive homothet $c+\lambda S=\{r_1,r_2,r_3,r_4\}$ of S with $r_1<r_2<r_3<r_4$. If $\{r_1,r_2,r_3,r_4\}\cap[0,L)\neq\emptyset$, then $(\{r_1,r_2,r_3,r_4\},r_2)$ cannot be AM, thus we may assume that $\{r_1,r_2,r_3,r_4\}\cap[0,L)=\emptyset$. Note that by the choice of K we again have

$$Kr_2>r_2+\frac{p_4-p_2}{p_2-p_1}r_2=r_2+\frac{p_4-p_2}{p_2-p_1}(\lambda(p_2-p_1)+r_1)\geq r_2+\lambda(p_4-p_2)=r_4.$$

This means that $\{r_2,r_3,r_4\}$ is contained in the union of two consecutive intervals of the form $[L\cdot K^i, L\cdot K^{i+1})$, which implies that if $(\{r_1,r_2,r_3,r_4\},r_2)$ is AM, then $\{r_3,r_4\}$ is contained in an interval $[L\cdot K^i+(j-1)(K^{i+1}-K^i), L\cdot K^i+j(K^{i+1}-K^i))$ for some $1\leq j\leq L$. However, then by the choice of L we have that r_1 is either contained in the interval $[L\cdot K^i, L\cdot K^{i+1})$ or in the interval $[L\cdot K^{i-1}, L\cdot K^i)$. Indeed,

$$\begin{aligned}r_3-r_1\leq\left\lceil\frac{r_3-r_1}{r_4-r_3}\right\rceil(r_4-r_3)&\leq\left\lceil\frac{r_3-r_1}{r_4-r_3}\right\rceil(K^{i+1}-K^i)\\&=\left\lceil\frac{p_3-p_1}{p_4-p_3}\right\rceil(K^{i+1}-K^i)=L(K^i-K^{i-1}).\end{aligned}$$

Thus, if r_1 has the same colour as r_3 and r_4, then r_1 is also contained in the interval $[L\cdot K^i+(j-1)(K^{i+1}-K^i), L\cdot K^i+j(K^{i+1}-K^i))$, implying that $(\{r_1,r_2,r_3,r_4\},r_2)$ is monochromatic.

We now extend the colouring to $\mathbb{R}$ in the case of $s_0=p_3$. Let φ_2' be a colouring of $\mathbb{R}_{\geq 0}$ isometric to the reflection of φ_2 over 0. Then φ_2' contains no AM positive homothet of (S,s_0). If further we assume that φ_1 and φ_2' use disjoint sets of colours, it is not hard to check that the union of φ_1 and φ_2' is an AP-free colouring of $\mathbb{R}$ containing no AM positive homothet of (S,s_0). We can extend the colouring similarly in the case of $s_0=p_2$. ◀

3 Higher dimensions

In this section we prove Theorem 5.

3.1 Proof of "if" direction of Theorem 5

Let $S\subseteq\mathbb{R}^d$ such that $|S|>3$ and s_0 is not an extreme point of S. To prove the 'if' direction of Theorem 5, we prove that there is an AP-free colouring of $\mathbb{R}^d$ without an AM similar copy of (S,s_0). (Note that for the proof of Theorem 5, it would be sufficient to prove this for $S\subseteq\mathbb{Z}^d$.)

Recall that $C\subseteq\mathbb{R}^d$ is a *convex cone* if for every $x,y\in C$ and $\alpha,\beta\geq 0$, the vector $\alpha x+\beta y$ is also in C. The *angle* of C is $\sup_{x,y\in C\setminus\{o\}}\angle(x,y)$.

We partition $\mathbb{R}^d$ into finitely many convex cones $C_1\cup\dots\cup C_m$, each of angle at most $\alpha=\alpha(d,S)$, where $\alpha(d,S)$ will be set later. We colour the cones with pairwise disjoint sets of colours as follows. First, we describe a colouring φ of the closed circular cone $C=C(\alpha)$ of angle α around the line $x_1=\dots=x_d$. Then for each i we define a colouring φ_i of C_i using pairwise disjoint sets of colours in a similar way. More precisely, let f_i be an isometry with $f_i(C_i)\subseteq C$, and define φ_i such that it is isometric to φ on $f_i(C_i)$.

It is not hard to see that it is sufficient to find an AP-free colouring φ of C without an AM similar copy of (S, s_0). Indeed, since the cones C_i are coloured with pairwise disjoint sets of colours, any AP or AM similar copy of (S, s_0) is contained in one single C_i.

We now turn to describing the colouring φ of C. Note that by choosing α sufficiently small we may assume that $C \subseteq \mathbb{R}^d_{\geq 0}$. For $x \in \mathbb{R}^d$ let $\|x\|_1 = |x_1| + \cdots + |x_d|$. Then for any $x \in \mathbb{R}^d$ we have

$$\|x\| \leq \|x\|_1 \leq \sqrt{d}\|x\|. \tag{1}$$

Let $S = \{p_1, \ldots, p_n\}$ and fix K such that

$$K > 1 + 2\sqrt{d} \max_{p_i, p_j, p_l, p_\ell \in S, p_i \neq p_j} \frac{\|p_k - p_\ell\|}{\|p_i - p_j\|}.$$

For a sufficiently large L, to be specified later, we define $\varphi : C \to \{0, 1, \ldots, 2L\}$ as

$$\varphi(x) = \begin{cases} 0 & \text{if } \|x\|_1 < L \\ j & \text{if for some even } i \in \mathbb{N} \text{ and } j \in [L] \text{ we have} \\ & \quad \|x\|_1 \in [L \cdot K^i + (j-1)(K^{i+1} - K^i), L \cdot K^i + j(K^{i+1} - K^i)) \\ L + j & \text{if for some odd } i \in \mathbb{N} \text{ and } j \in [L] \text{ we have} \\ & \quad \|x\|_1 \in [L \cdot K^i + (j-1)(K^{i+1} - K^i), L \cdot K^i + j(K^{i+1} - K^i)). \end{cases}$$

It is not hard to check that φ is AP-free. Thus we only have to show that it does not contain an AM similar copy of (S, s_0). Let $(\{r_1, \ldots, r_n\}, q_0)$ be a similar copy of (S, s_0), with

$$\|r_1\|_1 \leq \|r_2\|_1 \leq \cdots \leq \|r_n\|_1. \tag{2}$$

▷ Claim 14. $\{\|r_2\|_1, \ldots, \|r_n\|_1\}$ is contained in the union of two consecutive intervals of the form $[L \cdot K^j, L \cdot K^{j+1})$.

Proof. For any r_i with $i \geq 2$ we have

$$\begin{aligned} \|r_i\|_1 &\leq \|r_2\|_1 + \|r_i - r_2\|_1 \\ &\leq \|r_2\|_1 + \sqrt{d}\|r_i - r_2\| && \text{by (1)} \\ &= \|r_2\|_1 + \sqrt{d}\frac{\|r_i - r_2\|}{\|r_2 - r_1\|}\|r_2 - r_1\| \\ &< \|r_2\|_1 + \frac{K-1}{2} \cdot \|r_2 - r_1\| && \text{by the definition of } K \\ &\leq \|r_2\|_1 + \frac{K-1}{2}(\|r_2\| + \|r_1\|) && \text{by the triangle inequality} \\ &\leq \|r_2\|_1 + \frac{K-1}{2}(\|r_2\|_1 + \|r_1\|_1) && \text{by (1)} \\ &\leq K\|r_2\|_1 && \text{by (2).} \end{aligned}$$

◁

Assume now that $(\{r_1, \ldots, r_n\}, q_0)$ is AM. Note that $\|x\|_1$ is $\sqrt{d}$ times the length of the projection of x on the $x_1 = \cdots = x_d$ line for $x \in \mathbb{R}^d_{\geq 0}$. Thus for any similar copy $\psi(S)$ of S we have $\|\psi(s_0)\|_1 \in \operatorname{conv}\{\|p\|_1 : p \in \psi(S \setminus \{s_0\})\}$, and we may assume that $q_0 \neq r_1, r_n$. This means that $\varphi(r_1) = \varphi(r_n)$, and there is exactly one $i \in \{2, \ldots, n-1\}$ with $\varphi(r_i) \neq \varphi(r_1)$. This, by Claim 14 and by the definition of φ, is only possible if $i = 2$ and there are $i \in \mathbb{N}$ and $j \in [L]$ such that

$$\|r_3\|_1, \ldots, \|r_{n-1}\|_1 \in [L \cdot K^i + (j-1)(K^{i+1} - K^i), L \cdot K^i + j(K^{i+1} - K^i)).$$

The following claim finishes the proof.

▷ Claim 15. If L is sufficiently large and α is sufficiently small, then $\|r_1\|_1$ is contained in $[L \cdot K^{i-1}, L \cdot K^i) \cup [L \cdot K^i, L \cdot K^{i+1})$.

The claim indeed finishes the proof. By the definition of φ then $\varphi(r_1) = \varphi(r_n)$ implies

$$\|r_1\|_1 \in [L \cdot K^i + (j-1)(K^{i+1} - K^i), L \cdot K^i + j(K^{i+1} - K^i)).$$

But then we have

$$\|r_2\|_1 \in [L \cdot K^i + (j-1)(K^{i+1} - K^i), L \cdot K^i + j(K^{i+1} - K^i))$$

as well, contradicting $\varphi(r_2) \neq \varphi(r_1)$.

Proof of Claim 15. It is sufficient to show that $\|r_{n-1}\|_1 - \|r_1\|_1 < LK^i - LK^{i-1}$. We have

$$\|r_{n-1}\|_1 - \|r_1\|_1 \leq \sqrt{d}\|r_{n-1} - r_1\| = \sqrt{d}\|r_n - r_{n-1}\| \frac{\|r_{n-1} - r_n\|}{\|r_n - r_{n-1}\|} < \|r_{n-1} - r_n\| \frac{K-1}{2},$$

by (1) and by the definition of K. Let H_1 and H_2 be the hyperplanes orthogonal to the line $x_1 = \cdots = x_d$ at distance $\frac{1}{\sqrt{d}}(L \cdot K^i + (j-1)(K^{i+1} - K^i))$ and $\frac{1}{\sqrt{d}}(L \cdot K^i + j(K^{i+1} - K^i))$ from the origin respectively. Since $\|r_n\|_1, \|r_{n-1}\|_1 \in [L \cdot K^i + (j-1)(K^{i+1} - K^i), L \cdot K^i + j(K^{i+1} - K^i))$ we have that r_n and r_{n-1} are contained in the intersection T of $C(\alpha)$ and the slab bounded by the hyperplanes H_1 and H_2.

Thus $\|r_{n-1} - r_n\|$ is bounded by the length of the diagonal of the trapezoid which is obtained as the intersection of T and the 2-plane through r_n, r_{n-1} and the origin. Scaled by $\sqrt{d}$, this is shown in Figure 1.

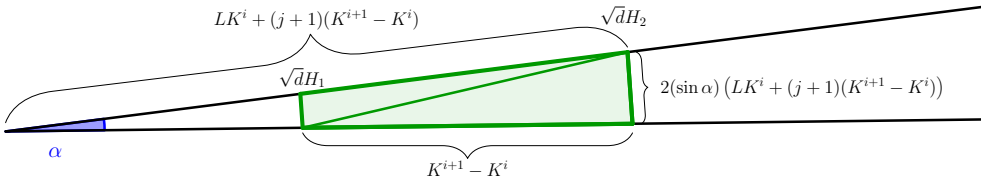


Figure 1 $T \cap C(\alpha)$.

From this, by the triangle inequality we obtain

$$\begin{aligned}\|r_{n-1} - r_n\| &\leq \frac{1}{\sqrt{d}}\left(K^{i+1} - K^i + 2(\sin\alpha)\left(LK^i + (j+1)(K^{i+1} - K^i)\right)\right) \\ &\leq \frac{1}{\sqrt{d}}\left(K^{i+1} - K^i + 2\sin\alpha \cdot LK^{i+1}\right) \leq \frac{2}{\sqrt{d}}\left(K^{i+1} - K^i\right),\end{aligned}$$

where the last inequality holds if α is sufficiently small. Combining these inequalities and choosing $L = \frac{K^2}{\sqrt{d}}$ we obtain the desired bound $\|r_{n-1}\|_1 - \|r_n\|_1 < LK^i - LK^{i-1}$, finishing the proof of the claim. ◁

3.2 Proof of "only if" direction of Theorem 5

The "only if" direction follows from Theorem 7 in the case of $d = 1$, and from the following stronger statement for $d \geq 2$ (since in this case s_0 is an extreme point of S).

▶ **Theorem 16.** *Let $S \subseteq \mathbb{Z}^d$ and $s_0 \in S$ be an extreme point of S. Then for every k there is a constant $\Lambda = \Lambda(d, S, k)$ such that the following is true. Every k-colouring of $\mathbb{Z}^d$ contains either an AM similar copy of (S, s_0) or a monochromatic similar copy of $\mathbb{Z}^d$ with an integer scaling ratio $1 \leq \lambda \leq \Lambda$.*

Before the proof we need some preparation.

▶ **Lemma 17.** *There is an $R > 0$ such that for any ball D of radius at least R the following is true. For every $p \in \mathbb{Z}^d$ at distance at most 1 from D there is a similar copy (S', s'_0) of (S, s_0) in $\mathbb{Z}^d$ such that $s'_0 = p$ and $S' \setminus \{s'_0\} \subset D$.*

Let $\mathbb{Q}_N = \left\{ \frac{a}{b} : a, b \in \mathbb{Z}, b \leq N \right\} \subseteq \mathbb{Q}$. Lemma 17 follows from the next lemma.

▶ **Lemma 18.** *There is an $\varepsilon > 0$ and an $N \in \mathbb{N}$ such that for any ball D of radius 1 the following is true. For all $p \in \mathbb{Q}_N^d$ at distance at most ε from D there is a similar copy (S', s'_0) of (S, s_0) in $\mathbb{Q}_N^d$ such that $s'_0 = p$ and $S' \setminus \{s'_0\} \subset D$.*

Proof. Let D be a ball of radius 1. Since s_0 is an extreme point of S, there is a hyperplane that separates s_0 from $S \setminus \{s_0\}$. With this it is not hard to see that there are $\delta, \varepsilon > 0$ with the following property. If p is ε close to D, then there is a congruent copy (S'', s''_0) of $\delta(S, s_0)$ with $s''_0 = p$ and such that every point of $S'' \setminus \{s''_0\}$ is contained in D at distance at least ε from the boundary of D. Now we use the fact that $O(\mathbb{R}^n) \cap \mathbb{Q}^{n \times n}$, the set of rational rotations, is dense in $O(\mathbb{R}^n)$ (see for example [12]). By this and the compactness of $O(\mathbb{R})^n$, we can find an $N = N(\delta, \varepsilon) \in \mathbb{N}$ and (S', s'_0) in $\mathbb{Q}_N^d$ which is a rotation of (S'', s''_0) around p, ε-close to (S'', s''_0). With this $S' \setminus \{s'_0\}$ is contained in D. Therefore N and ε satisfy the requirements. ◀

The proof of the following variant of Gallai's theorem can be found in the Appendix of the full version of the paper [6].

▶ **Theorem 19** (Gallai). *Let $S \subseteq \mathbb{Z}^d$ be finite. Then there is a $\lambda(d, S, k) \in \mathbb{Z}$ such that every k-colouring of $\mathbb{Z}^d$ contains a monochromatic positive homothet of S with an integer scaling ratio bounded by $\lambda(d, S, k)$.*

Proof of Theorem 16. Let R be as in Lemma 17 and let H be the set of points of $\mathbb{Z}^d$ contained in a ball of radius R. By Theorem 19 there is a monochromatic, say blue, homothetic copy $H_0 = c + \lambda H$ of H for some integer $\lambda \leq \lambda(d, H, k)$. Without loss of generality we may assume that $H_0 = B(O, \lambda R) \cap \lambda \mathbb{Z}^d$ for some $O \in \mathbb{Z}^d$, where $B(O, \lambda R)$ is the ball of radius λR centred at O.

Consider a point $p \in \lambda \mathbb{Z}^d \setminus H$ being at distance at most λ from H_0. If p is not blue then using Lemma 17 we can find an AM similar copy of (S, s_0). Thus we may assume that any point $p \in \lambda \mathbb{Z}^d \setminus H_0$ which is λ close to H_0 is blue as well.

By repeating a similar procedure, we obtain that there is either an AM similar copy of (S, s_0), or every point of $H_i = B(O, \lambda R + i\lambda) \cap \lambda \mathbb{Z}^d$ is blue for every $i \in \mathbb{N}$. But the latter means $\lambda \mathbb{Z}^d$ is monochromatic, which finishes the proof. ◀

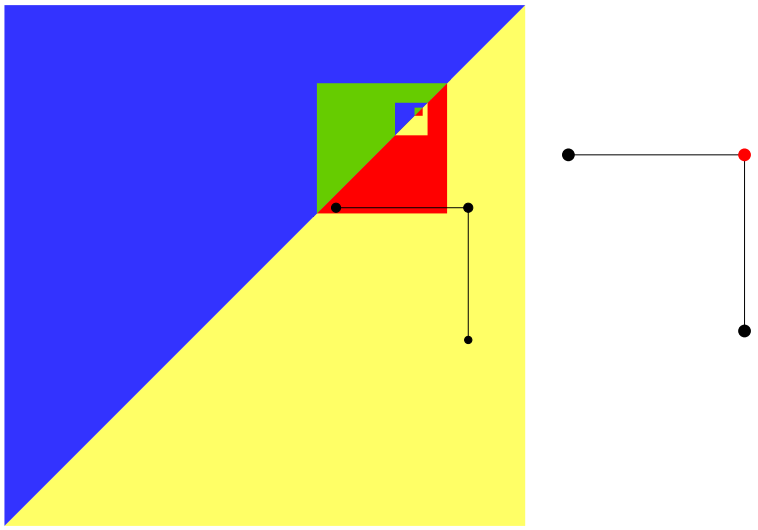


Figure 2 A 4-colouring avoiding AM homothets of (S, s_0).

3.3 Finding an AM positive homothet

The following statement shows that it is not possible to replace an AM similar copy of (S, s_0) with a positive homothet of (S, s_0) in the "only if" direction of Theorem 5.

▶ **Proposition 20.** *Let $S \subseteq \mathbb{Z}^d$ such that S is not contained in a line and $s_0 \in S$. Then there is an AP-free colouring of $\mathbb{R}^d$ without an AM positive homothet of (S, s_0).*

Proof. We may assume that $|S| = 3$ and thus $S \subseteq \mathbb{R}^2$. Since the problem is affine invariant, we may further assume that $S = \{(0,1),(1,1),(1,0)\}$ with $s_0 = (1,1)$, $s_1 = (0,1)$ and $s_2 = (1,0)$. First we describe a colouring of $\mathbb{R}^2$ and then we extend it to $\mathbb{R}^d$.

For every $i \in \mathbb{N}$ let Q_i be the square $[-4^i, 4^{i-1}] \times [-4^i, 4^{i-1}]$, and $Q_0 = \emptyset$. Further let H_+ be the open half plane $x < y$ and H_- be the closed half plane $x \geq y$. We colour $\mathbb{R}^2$ using four colours, green, blue, red and yellow as follows (see also Figure 2).

- **Green:** For every odd $i \in \mathbb{N}$ colour $(Q_i \setminus Q_{i-1}) \cap H_+$ with red.
- **Blue:** For every even $i \in N$ colour $(Q_i \setminus Q_{i-1}) \cap H_+$ with yellow.
- **Red:** For every odd $i \in \mathbb{N}$ colour $(Q_i \setminus Q_{i-1}) \cap H_-$ with green.
- **Yellow:** For every even $i \in \mathbb{N}$ colour $(Q_i \setminus Q_{i-1}) \cap H_-$ with blue.

It is not hard to see that this colouring φ_1 is AP-free. Thus we only have to check that it contains no AM positive homothet of (S, s_0). Let S' be a positive homothet of S. First note that we may assume that S' is contained in one of the half planes bounded by the $x = y$ line, otherwise it is easy to see that it cannot be AM. Thus by symmetry we may assume that $s'_0 \in Q_i \setminus Q_{i-1} \cap H_+$ for some $i \in \mathbb{N}$.

If the y-coordinate of s'_0 is smaller than -4^{i-1}, then $s'_1 \in Q_i \setminus Q_{i-1}$, and hence S' cannot be AM. On the other hand, if the y-coordinate of s'_0 is at least -4^{i-1}, then $\|s'_0 - s'_1\| \leq 2 \cdot 4^{i-1}$. This means that the y-coordinate of s'_2 is at least $-(4^{i-1} + 2 \cdot 4^{i-1}) > -4^i$. Thus, in this case s'_2 is contained in $s'_1 \in Q_i \setminus Q_{i-1}$, and hence S' cannot be monochromatic.

To finish the proof, we extend the colouring to $\mathbb{R}^d$. Let $T \cong \mathbb{R}^{d-2}$ be the orthogonal complement of $\mathbb{R}^2$. Fix an AP-free colouring φ of T using the colour set $\{1, 2\}$. Further let φ_2 be a colouring of $\mathbb{R}^2$ isometric to φ_1, but using a disjoint set of colours. For every $t \in T$ colour $\mathbb{R}^2 + t$ by translating φ_i if $\varphi(t) = i$. It is not hard to check that this colouring is AP-free and does not contain any AM positive homothet of (S, s_0). ◀

4 Smiling bouquets and the chromatic number of the plane

For a graph $G = (V, E)$ with a given origin (distinguished vertex) $v_0 \in V$ a colouring φ with $\varphi : V \setminus \{v_0\} \to \binom{[k]}{1}$ and $\varphi(v_0) \in \binom{[k]}{2}$ is a proper k-colouring with *bichromatic origin* v_0, if $(v, w) \in E$ implies $\varphi(v) \cap \varphi(w) = \emptyset$. There are unit-distance graphs with not too many vertices that do not have a 4-colouring with a certain bichromatic origin. Figure 3 shows such an example, the 34-vertex graph G_{34}, posted by the second author [9] in Polymath16. It is the first example found whose chromatic number can be verified quickly without relying on a computer. Finding such graphs has been motivated by an approach to find a human-verifiable proof of $\chi(\mathbb{R}^5) \geq 5$, proposed by the third author [10] in Polymath16.

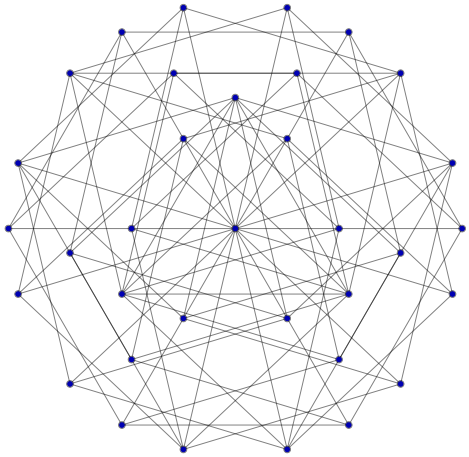


Figure 3 A 34 vertex graph without a 4-colouring if the origin is bichromatic.

Theorem 21 with $G = G_{34}$ shows that a human-verifiable proof of Conjecture 1 for $k = 4$ would provide a human-verifiable proof of $\chi(\mathbb{R}^2) \geq 5$. Note that G_{34} was found by a computer search, and for finding other similar graphs one might rely on a computer program. Thus, the approach we propose, is human-verifiable, however it might be computer-assisted.

For a graph G with origin v_0 let $\{C_1, \ldots, C_n\}$ be the set of unit circles whose centres are the neighbours of v_0, and let $C(G, v_0) = C_1 \cup \cdots \cup C_n$ be the bouquet through v_0.

▶ **Theorem 21.** *If there is a unit-distance graph $G = (V, E)$ with $v_0 \in V$ which does not have a proper k-colouring with bichromatic origin v_0, and Conjecture 1 is true for $C(G, v_0)$, then $\chi(\mathbb{R}^2) \geq k + 1$.*

Proof of Theorem 21. Assume for a contradiction that there is a proper k-colouring φ of the plane. Using φ we construct a proper k-colouring of G with bichromatic origin $v_0 \in V$.

Let $v_1, \ldots, v_n$ be the neighbours of the origin v_0, and C_j be the unit circle centred at v_j. Then $C = C_1 \cup \cdots \cup C_n$ is a bouquet through v_0. If Conjecture 1 is true for φ, then there is a smiling congruent copy $C' = C'_1 \cup \ldots C'_n$ of C through v'_0. That is, there are points $p_1 \in C'_1, \ldots, p_n \in C'_n$ with $\ell = \varphi(p_1) = \cdots = \varphi(p_n) \neq \varphi(v'_0)$.

For $i \in [n]$ let v'_i be the centre of C'_i. We define a colouring φ' of G as $\varphi'(v_0) = \{\varphi(v'_0), \ell\}$ and $\varphi'(v_i) = \varphi(v'_i)$ for $v \in V \setminus \{v_0\}$. We claim that φ' is a proper k-colouring of G with a bichromatic origin v_0, contradicting our assumption.

Indeed, if $v_i \neq v_0 \neq v_j$ then for $(v_i, v_j) \in E$ we have $\varphi'(v_i) \neq \varphi'(v_j)$ because $\varphi(v'_i) \neq \varphi(v'_j)$. For $(v_0, v_i) \in E$, we have $\varphi'(v_i) \neq \varphi(v_0)$ because $\varphi(v'_i) \neq \varphi(v'_0)$, and $\varphi'(v_i) \neq \ell$ because $\varphi(v'_i) \neq \ell$ since $\|v'_i - p_i\| = 1$. This finishes the proof of Theorem 21. ◀

4.1 Smiling pencils

In this section we prove Theorem 3. We start with the following simple claim.

▷ Claim 22. For every pencil L through O there is an $\varepsilon > 0$ for which the following is true. For any circle C or radius R if a point p is at distance at most εR from C, then there is a congruent copy L' of L through p such that every line of L' intersects C.

Proof. It is sufficient to prove the following. If C is a unit circle and p is sufficiently close to C, then there is a congruent copy L' of L through p such that every line of L' intersects C.

Note that if p is contained in the disc bounded by C, clearly every line of every congruent copy L' of L through p intersects C. Thus we may assume that p is outside the disc.

Let $0 < \alpha < \pi$ be the largest angle spanned by lines in L. If p is sufficiently close to C, then the angle spanned by the tangent lines of C through p is larger than α. Thus, any congruent copy L' of L through p can be rotated around p so that every line of the pencil intersects C. ◁

Proof of Theorem 3. Assume for contradiction that φ is a colouring using at least two colours, but there is a pencil L such that there is no congruent smiling copy of L.

First we obtain a contradiction assuming that there is a monochromatic, say red, circle C of radius r. We claim that then every point p inside the disc bounded by C is red. Indeed, translating L to a copy L' through p, each line L'_i will intersect C, and so have a red point. Thus p must be red.

A similar argument together with Claim 22 shows that if there is a non-red point at distance at most εr from C, we would find a congruent smiling copy of L through p. Thus there is a circle C' of radius $(1 + \varepsilon)r$ concentric with C, such that every point of the disc bounded by C' is red. Repeating this argument, we obtain that every point of $\mathbb{R}^2$ is red contradicting the assumption that φ uses at least 2 colours.

To obtain a contradiction, we prove that there exists a monochromatic circle. For $1 \leq i \leq n$ let α_i be the angle of L_i and L_{i+1}. Fix a circle C, and let $a_1, \ldots, a_n \in C$ be points such that if $c \in C \setminus \{a_1, \ldots, a_n\}$, then the angle of the lines connecting c with a_i and c with a_{i+1} is α_i. By Gallai's theorem there is a monochromatic (say red) set $\{a'_1, \ldots, a'_n\}$ similar to $\{a_1, \ldots, a_n\}$. Let C' be the circle that contains $\{a'_1, \ldots, a'_n\}$. Then C' is monochromatic. Indeed, if there is a point p on C' for which $\varphi(p)$ is not red, then by choosing L'_j to be the line connecting p with a'_j we obtain $L' = L'_1 \cup \cdots \cup L'_n$, a smiling congruent copy of L. ◀

4.2 Conjecture 1 for lattice-like bouquets

Using the ideas from the proof of Theorem 16, we prove Conjecture 1 for a broader family of bouquets.

4.2.1 Lattices

A *lattice* $\mathcal{L}$ generated by linearly independent vectors v_1 and v_2 is the set $\mathcal{L} = \mathcal{L}(v_1, v_2) = \{n_1v_1 + n_2v_2 : n_1, n_2 \in \mathbb{Z}\}$. We call a lattice $\mathcal{L}$ *rotatable* if for every $0 \leq \alpha_1 < \alpha_2 \leq \pi$ there is an angle $\alpha_1 < \alpha < \alpha_2$ and scaling factor $\lambda = \lambda(\alpha_2, \alpha_1)$ such that $\lambda\alpha(\mathcal{L}) \subset \mathcal{L}$, where $\alpha(L)$ is the rotated image of $\mathcal{L}$ by angle α around the origin. For example, $\mathbb{Z}^2$, the triangular grid, and $\left\{n_1(1,0) + n_2(0,\sqrt{2}) : n_1, n_2 \in \mathbb{Z}\right\}$ are rotatable, but $\mathcal{L} = \{n_1(1,0) + n_2(0,\pi) : n_1, n_2 \in \mathbb{Z}\}$ is not.[2]

The rotatability of $\mathcal{L}$ allows us to extend Lemma 17 from $\mathbb{Z}^2$ to $\mathcal{L}$. This leads to an extension of Theorem 16 to rotatable lattices.

▶ **Theorem 23.** *Let $\mathcal{L}$ be a rotatable lattice, $S \subseteq \mathcal{L}$ be finite and s_0 be an extreme point of S. Then for every $k \in N$ there exists a constant $\Lambda = \Lambda(\mathcal{L}, S, k)$ such that the following is true. In every k-colouring of $\mathcal{L}$ there is either an AM similar copy of (S, s_0) with a positive scaling factor bounded by Λ, or a monochromatic positive homothetic copy of $\mathcal{L}$ with an integer scaling factor $1 \leq \lambda \leq \Lambda$.*

The proof of extending Lemma 17 to rotatable lattices is analogous to the original one, so is the proof of Theorem 23 to the proof of Theorem 16. Therefore, we omit the details.

4.2.2 Lattice-like bouquets

Let $C = C_1 \cup \cdots \cup C_n$ be a bouquet through O, and for $i \in [n]$ let O_i be the centre of C_i. We call C *lattice-like* if O is an extreme point of $\{O, O_1, \ldots, O_n\}$ and there is a rotatable lattice $\mathcal{L}$ such that $\{O, O_1, \ldots, O_n\} \subseteq \mathcal{L}$. Similarly, we call a unit-distance graph $G = (V, E)$ with an origin $v_0 \in V$ *lattice-like* if there is a rotatable lattice $\mathcal{L}$ such that v_0 and its neighbours are contained in $\mathcal{L}$, and v_0 is not in the convex hull of its neighbours.

Since $\mathbb{Z}^2$ is a rotatable lattice, Theorem 2 is a direct corollary of the result below.

▶ **Theorem 24.** *If C is a lattice-like bouquet, then every proper k-colouring of $\mathbb{R}^2$ contains a smiling congruent copy of C.*

This implies the following, similarly as Conjecture 1 implied Theorem 21.

▶ **Theorem 25.** *If there exists a lattice-like unit-distance graph $G = (V, E)$ with an origin v_0 that does not admit a proper k-colouring with bichromatic origin v_0, then $\chi(\mathbb{R}^2) \geq k + 1$.*

In the proof of Theorem 24, we need a simple geometric statement.

▶ **Proposition 26.** *Let $C = C_1 \cup \cdots \cup C_n$ be a bouquet through O, and let $\mathcal{O} = \{O_1, \ldots, O_n\}$, where O_j is the center of C_j. Then for every $0 < \lambda \leq 2$ there are n points $P_1, \ldots, P_n$ such that $P_j \in C_j$ and $\{P_1, \ldots, P_n\}$ is congruent to $\lambda\mathcal{O}$.*

Proof. For $\lambda = 2$ let P_j be the image of O reflected in O_j. Then $P_j \in C_j$, and $\{P_1, \ldots, P_n\}$ can be obtained by enlarging $\mathcal{O}$ from O with a factor of 2. For $\lambda < 2$, scale $\{P_1, \ldots, P_n\}$ by $\frac{\lambda}{2}$ from O obtaining $\{P'_1, \ldots, P'_n\}$. Then there is an angle α such that rotating $\{P'_1, \ldots, P'_n\}$ around O by α, the rotated image of each P'_j is on C_j. ◀

Proof of Theorem 24. Let $C = C_1 \cup \cdots \cup C_n$ be the lattice-like bouquet through O, O_i be the centre of C_i for $i \in [k]$, and $\mathcal{L}$ be the rotatable lattice containing $S = \{O, O_1, \ldots, O_n\}$. Consider a proper k-colouring φ of $\mathbb{R}^2$ and let $\delta \in \mathbb{Q}$ to be chosen later.

[2] For another characterization of rotatable lattices, see `https://mathoverflow.net/a/319030/955`.

By Theorem 23, the colouring φ either contains an AM similar copy of $\delta(S, s_0)$ with a positive scaling factor bounded by $\lambda(\mathcal{L}, S, k)$, or a monochromatic similar copy of $\delta\mathcal{L}$ with an integer scaling factor bounded by $\lambda(\mathcal{L}, S, k)$.

If the first case holds and δ is chosen so that $\delta\lambda(\mathcal{L}, S, k) \leq 2$, Proposition 26 provides a smiling congruent copy of C. Now assume for contradiction that the first case does not hold. Then there is a monochromatic similar copy $\mathcal{L}'$ of $\delta\mathcal{L}$ with an integer scaling factor λ bounded by $\lambda(\mathcal{L}, S, k)$. However, if we choose $\delta = \frac{1}{\lambda(\mathcal{L},S,k)!}$, then for any $1 \leq \lambda \leq \lambda(\mathcal{L}, S, k)$ we have $\delta\lambda = \frac{1}{N_\lambda}$ for some $N_\lambda \in \mathbb{N}$. But this would imply that there are two points in the infinite lattice $\lambda\delta\mathcal{L}$ at distance 1, contradicting that φ is a proper colouring $\mathbb{R}^2$. ◀

5 Further problems and concluding remarks

Problems in the main focus of this paper are about finding AM sets *similar* to a given one. However, it is also interesting to find AM sets *congruent* to a given one. In this direction, Erdős, Graham, Montgomery, Rothschild, Spencer and Straus made the following conjecture.

▶ **Conjecture 27** (Erdős et al. [4]). *Let $s_0 \in S \subset \mathbb{R}^2$, $|S| = 3$. There is a non-monochromatic colouring of $\mathbb{R}^2$ that contains no AM congruent copy of (S, s_0) if and only if S is collinear and s_0 is not an extreme point of S.*

As noted in [4], the "if" part is easy; colour $(x, y) \in \mathbb{R}^2$ red if $y > 0$ and blue if $y \leq 0$. In fact, this colouring also avoids AM similar copies of such S.

Conjecture 27 was proved in [4] for the vertex set S of a triangle with angles $120°, 30°$, and $30°$ with any $s_0 \in S$. It was also proved for any isosceles triangle in the case when s_0 is one of the vertices on the base, and for an infinite family of right-angled triangles.

Much later, the same question was asked independently in a more general form by the third author [11]. In a comment to this question on the MathOverflow site, a counterexample (to both the MathOverflow question and Conjecture 27) was pointed out by user "fedja" [5], which we sketch in the full version of the paper [6].

Straightforward generalisations of our arguments from Section 4 would also imply lower bounds for the chromatic number of other spaces. For example, if C is a lattice-like bouquet of spheres, then every proper k-colouring of $\mathbb{R}^d$ contains a smiling congruent copy of C. This implies that if one can find a lattice-like unit-distance graph with an origin v_0 that does not admit a proper k-colouring with bichromatic origin v_0, then $\chi(\mathbb{R}^d) \geq k + 1$. Possibly one can even strengthen this further; in $\mathbb{R}^d$ it could be even true that there is a *d-smiling* congruent copy of any bouquet C, meaning that there are d colours that appear on each sphere of C. This would imply $\chi(\mathbb{R}^d) \geq k + d - 1$ if we could find a lattice-like unit-distance graph with an origin v_0 that does not admit a proper k-colouring with d-chromatic origin v_0.

On of our main questions is about characterising those pairs (S, s_0) for which in every colouring of $\mathbb{R}^d$ we either find an AM similar copy of (S, s_0) or an *infinite* monochromatic AP. However, regarding applications to the Hadwiger-Nelson problem the following, weaker version would also be interesting to consider: Determine those (S, s_0) with $S \subseteq \mathbb{R}^d$ and $s_0 \in S$ for which there is a $D = D(k, S)$ such that the following is true. For every n in every k-colouring of $\mathbb{R}^D$ there is an AM similar copy of (S, s_0) or an n-term monochromatic AP with difference $t \in \mathbb{N}$ bounded by D. Note that there are pairs for which the property above does not hold when colouring $\mathbb{Z}$. For example let $S = \{-2, -1, 0, 1, 2\}$, $s_0 = 0$, and colour $i \in \mathbb{Z}$ red if $\lfloor i/D \rfloor \equiv 0 \bmod 2$ and blue if $\lfloor i/D \rfloor \equiv 1 \bmod 2$.

References

1. A. D.N.J. de Grey. The chromatic number of the plane is at least 5. *Geombinatorics*, 28:18–31, 2018.
2. A. D.N.J. de Grey. Polymath proposal: finding simpler unit distance graphs of chromatic number 5. https://polymathprojects.org/2018/04/10/polymath-proposal-finding-simpler-unit-distance-graphs-of-chromatic-number-5/, 2018.
3. P. Erdős, R. L. Graham, P. Montgomery, B. L. Rothschild, J. Spencer, and E. G. Straus. Euclidean Ramsey theorems. I. *Journal of Combinatorial Theory, Series A*, 14(3):341–363, 1973. doi:10.1016/0097-3165(73)90011-3.
4. P. Erdős, R. L. Graham, P. Montgomery, B. L. Rothschild, J. Spencer, and E. G. Straus. Euclidean Ramsey theorems. III. In *Infinite and finite sets (Colloq., Keszthely, 1973; dedicated to P. Erdős on his 60th birthday), Vol. I*, pages 559–583. North-Holland, Amsterdam, 1975.
5. [fedja]. Comment on to the problem "Almost monochromatic point set" on MathOverflow. https://mathoverflow.net/q/300604, 2018.
6. N. Frankl, T Hubai, and D. Pálvölgyi. Almost-monochromatic sets and the chromatic number of the plane, 2019. arXiv:1912.02604.
7. R. L. Graham. Euclidean Ramsey theory. In Joseph O'Rourke Jacob E. Goodman and Csaba D. Tóth, editors, *Handbook of Discrete and Computational Geometry, Third Edition*, pages 281–297. Chapman and Hall/CRC, 2017. doi:10.1201/9781420035315.ch11.
8. R. L. Graham, B. L. Rothschild, and J.H. Spencer. *Ramsey theory*. Wiley-Interscience Series in Discrete Mathematics and Optimization. John Wiley & Sons, Inc., New York, 1990.
9. T. Hubai. Comment in "polymath16, fourth thread: Applying the probabilistic method". https://dustingmixon.wordpress.com/2018/05/05/polymath16-fourth-thread-applying-the-probabilistic-method/#comment-4391, 2018.
10. D. Pálvölgyi [domotorp]. Comment in "polymath16, fourth thread: Applying the probabilistic method". https://dustingmixon.wordpress.com/2018/05/05/polymath16-fourth-thread-applying-the-probabilistic-method/#comment-4306, 2018.
11. D. Pálvölgyi [domotorp]. Proposing the problem "Almost monochromatic point sets" on MathOverflow. https://mathoverflow.net/q/300604, 2018.
12. I. Rivin. Answer to the question "Existence of rational orthogonal matrices" on MathOverflow. https://mathoverflow.net/q/90070, 2012.
13. A. Soifer. *The Mathematical Coloring Book*. Springer-Verlag New York, 2009. doi:10.1007/978-0-387-74642-5.
14. B. L. van der Waerden. Beweis einer baudetschen Vermutung. *Nieuw Arch. Wiskunde*, 15:212–216, 1927. URL: https://ci.nii.ac.jp/naid/10004588776/en/.

Almost Sharp Bounds on the Number of Discrete Chains in the Plane

Nóra Frankl
Department of Mathematics, London School of Economics and Political Science, UK
Laboratory of Combinatorial and Geometric Structures at MIPT, Moscow, Russia
n.frankl@lse.ac.uk

Andrey Kupavskii
Moscow Institute of Physics and Technology, Moscow, Russia
Institute for Advanced Study, Princeton, NJ, US
G-SCOP, CNRS, Grenoble, France
kupavskii@yandex.ru

Abstract

The following generalisation of the Erdős unit distance problem was recently suggested by Palsson, Senger and Sheffer. For a sequence $\boldsymbol{\delta} = (\delta_1, \dots, \delta_k)$ of k distances, a $(k+1)$-tuple $(p_1, \dots, p_{k+1})$ of distinct points in $\mathbb{R}^d$ is called a $(k, \boldsymbol{\delta})$-*chain* if $\|p_j - p_{j+1}\| = \delta_j$ for every $1 \le j \le k$. What is the maximum number $C_k^d(n)$ of $(k, \boldsymbol{\delta})$-chains in a set of n points in $\mathbb{R}^d$, where the maximum is taken over all $\boldsymbol{\delta}$? Improving the results of Palsson, Senger and Sheffer, we essentially determine this maximum for all k in the planar case. It is only for $k \equiv 1 \pmod 3$ that the answer depends on the maximum number of unit distances in a set of n points. We also obtain almost sharp results for even k in dimension 3.

2012 ACM Subject Classification Mathematics of computing → Combinatoric problems

Keywords and phrases unit distance problem, unit distance graphs, discrete chains

Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.48

Related Version A related version of this paper is available at `https://arxiv.org/abs/1912.00224`.

Funding The authors acknowledge the financial support from the Ministry of Education and Science of the Russian Federation in the framework of MegaGrant no. 075-15-2019-1926.
Nóra Frankl: Research was partially supported by the National Research, Development, and Innovation Office, NKFIH Grant K119670.
Andrey Kupavskii: Research supported by the Russian Foundation for Basic Research (grant no. 18-01-00355) and the Council for the Support of Leading Scientific Schools of the President of the Russian Federation (grant no. HW-6760.2018.1). Research was directly supported by the IAS Fund for Math, the Director's Fund and indirectly supported by the National Science Foundation Grant No. CCF-1900460. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

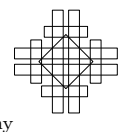
Acknowledgements We thank Konrad Swanepoel and the referees for helpful comments on the manuscript.

1 Introduction

Determining the maximum possible number of pairs $u_d(n)$ at distance 1 apart in a set of n points in $\mathbb{R}^d$ for $d = 2, 3$ is one of the central questions in combinatorial geometry. The planar version, determining $u_2(n)$ is also known as the Erdős unit distances problem. The question dates back to 1946, and despite much effort, the best known upper and lower bounds are still very far apart. For some constants $C, c > 0$, we have

$$n^{1+c/\log\log n} \le u_2(n) \le Cn^{4/3},$$

where the lower bound is due to Erdős [3] and the upper bound is due to Spencer, Szemerédi and Trotter [9].




As in the planar case, the best known upper and lower bounds in the 3-dimensional case are also far apart. For some $c, C > 0$, we have

$$cn^{4/3} \log\log n \leq u_3(n) \leq Cn^{295/137+\varepsilon}, \tag{1}$$

where the lower bound is due to Erdős [4], and the upper bound is due to Zahl [10]. The latter is a recent improvement upon the upper bound $O(n^{3/2})$ by Kaplan, Matoušek, Safernová, and Sharir [5], and Zahl [11]. In contrast, for $d \geq 4$ we have $u_d(n) = \Theta(n^2)$.

Palsson, Senger and Sheffer [8] suggested the following generalisation of the unit distance problem. Let $\boldsymbol{\delta} = (\delta_1, \ldots, \delta_k)$ be a sequence of k positive reals. A $(k+1)$-tuple $(p_1, \ldots, p_{k+1})$ of distinct points in $\mathbb{R}^d$ is called a $(k, \boldsymbol{\delta})$-*chain* if $\|p_i - p_{i+1}\| = \delta_i$ for all $i = 1, \ldots, k$. For every fixed k determine $C_k^d(n)$, the maximum number of $(k, \boldsymbol{\delta})$-chains that can be spanned by a set of n points in $\mathbb{R}^d$, where the maximum is taken over all $\boldsymbol{\delta}$. In the planar case, the following upper bounds were found in [8] in terms of the maximum number of unit distances.

▶ **Proposition 1** (Palsson, Senger, and Sheffer [8])**.**

$$C_k^2(n) = \begin{cases} O\left(n \cdot u_2(n)^{k/3}\right) & \text{if } k \equiv 0 \pmod 3, \\ O\left(u_2(n)^{(k+2)/3}\right) & \text{if } k \equiv 1 \pmod 3, \\ O\left(n^2 \cdot u_2(n)^{(k-2)/3}\right) & \text{if } k \equiv 2 \pmod 3. \end{cases}$$

If $u_2(n) = O(n^{1+\varepsilon})$ for any $\varepsilon > 0$, which is conjectured to hold, then the upper bounds in the proposition above almost match the lower bounds given in Theorem 2. However, as we have already mentioned, determining the order of magnitude of $u_2(n)$ is very far from being done, and in general it proved to be a very hard problem. Thus, it is interesting to obtain "unconditional" bounds, that depend on the value of $u_2(n)$ as little as possible. In [8], the following "unconditional" upper bounds were proved in the planar case.

▶ **Theorem 2** (Palsson, Senger, and Sheffer [8])**.** *$C_2^2(n) = \Theta(n^2)$, and for every $k \geq 3$ we have*

$$C_k^2(n) = \Omega\left(n^{\lfloor (k+1)/3 \rfloor + 1}\right)$$

and

$$C_k^2(n) = O\left(n^{2k/5+1+\gamma(k)}\right),$$

where $\gamma_k \leq \frac{1}{12}$, and $\gamma_k \to \frac{4}{75}$ as $k \to \infty$.

In our main result, in two-third of the cases we almost determine the value of $C_k^2(n)$, no matter what the value of $u_2(n)$ is, by matching the lower bounds given in Theorem 2. Further, we show that in the remaining cases determining $C_k^2(n)$ essentially reduces to determining the maximum number of unit distances.

▶ **Theorem 3.** *For every integer $k \geq 1$ we have*

$$C_k^2(n) = \tilde{\Theta}\left(n^{\lfloor (k+1)/3 \rfloor + 1}\right) \text{ if } k \equiv 0, 2 \pmod 3,$$

and for any $\varepsilon > 0$ we have

$$C_k^2(n) = \Omega\left(n^{(k-1)/3} u_2(n)\right) \text{ and } C_k^2(n) = O\left(n^{(k-1)/3+\varepsilon} u_2(n)\right) \text{ if } k \equiv 1 \pmod 3.$$

Here and in what follows $f(n) = \tilde{O}(g(n))$ means that there exist positive constants c, C such that $f(n)/g(n) \leq C \log^c n$ for every n. We write $f(n) = \tilde{\Omega}(g(n))$ if $g(n) = \tilde{O}(f(n))$, and $f(n) = \tilde{\Theta}(g(n))$ if $f(n) = \tilde{O}(g(n))$ and $g(n) = \tilde{O}(f(n))$.

Let us turn our attention to the 3-dimensional case. The following was proved in [8].

▶ **Theorem 4** (Palsson, Senger, and Sheffer [8]). *For any integer $k \geq 2$, we have*

$$C_k^3(n) = \Omega\left(n^{\lfloor k/2 \rfloor + 1}\right),$$

and

$$C_k^3(n) = \begin{cases} O\left(n^{2k/3+1}\right) & \text{if } k \equiv 0 \pmod 3, \\ O\left(n^{2k/3+23/33+\varepsilon}\right) & \text{if } k \equiv 1 \pmod 3, \\ O\left(n^{2k/3+2/3}\right) & \text{if } k \equiv 2 \pmod 3. \end{cases}$$

We improve their upper bound and essentially settle the problem for even k.

▶ **Theorem 5.** *For any integer $k \geq 2$ we have*

$$C_k^3(n) = \tilde{O}\left(n^{k/2+1}\right).$$

In particular, for even k we have

$$C_k^3(n) = \tilde{\Theta}\left(n^{k/2+1}\right).$$

We also improve the lower bound from Theorem 4 for odd k. Let $us_3(n)$ be the maximum number of pairs at unit distance apart between a set of n points in $\mathbb{R}^3$ and a set of n points on a sphere in $\mathbb{R}^3$.

▶ **Proposition 6.** *Let $k \geq 3$ odd. Then we have*

$$C_k^3(n) = \Omega\left(\max\left\{\frac{u_3(n)^k}{n^{k-1}}, us_3(n) n^{(k-1)/2}\right\}\right).$$

Note that $us_3(n)$ equals the maximum number of incidences between a set of n points and a set of n circles (not necessarily of the same radii) in the plane. Thus we have

$$cn^{4/3} \leq us_3(n) = \tilde{O}\left(n^{15/11}\right)$$

(see [1, 2, 6, 7]). Therefore, in general we cannot tell which of the two bounds in Proposition 6 is better. However, for large k the second term is larger than the first due to (1).

Finally, we note that for $d \geq 4$ we have $C_k^d(n) = \Theta(n^{k+1})$. Indeed, we clearly have $C_k^d(n) = O(n^{k+1})$. To see that $C_k^d(n) = \Omega(n^{k+1})$, take two orthogonal circles of radius $1/\sqrt{2}$ centred at the origin and choose $n/2$ points on each of them.

2 Preliminaries

We denote by $u_d(m,n)$ the maximum number of incidences between a set of m points and n spheres[1] of fixed radius in $\mathbb{R}^d$. In other words, $u_d(m,n)$ is the maximum number of red-blue pairs spanning a given distance in a set of m red and n blue points in $\mathbb{R}^d$. By the result of Spencer, Szemerédi and Trotter [9], we have

$$u_2(m,n) = O\left(m^{\frac{2}{3}}n^{\frac{2}{3}} + m + n\right). \tag{2}$$

We say that a point p is n^α-*rich* with respect to a set $P \subseteq \mathbb{R}^d$ and to a distance δ, if the sphere of radius δ around p contains at least n^α points of P. If $P \subseteq \mathbb{R}^2$ and $|P| = n^x$, then (2) implies that the number of points that are n^α-rich with respect to P and to a given distance δ is

$$O\left(n^{2x-3\alpha} + n^{x-\alpha}\right). \tag{3}$$

The bound

$$u_3(m,n) = O\left(m^{\frac{3}{4}}n^{\frac{3}{4}} + m + n\right) \tag{4}$$

is due to Zahl [10] and Kaplan, Matoušek, Safernová, and Sharir [5]. It implies that for $P \subseteq \mathbb{R}^3$ with $|P| = n^x$ the number of points that are n^α-rich with respect to P and to a given distance δ is

$$O\left(n^{3x-4\alpha} + n^{x-\alpha}\right). \tag{5}$$

3 Bounds in $\mathbb{R}^2$

For $\boldsymbol{\delta} = (\delta_1, \ldots, \delta_k)$ and $P_1 \ldots, P_{k+1} \subseteq \mathbb{R}^2$ we denote by $\mathcal{C}_k^{\boldsymbol{\delta}}(P_1, \ldots, P_k)$ the family of $(k+1)$-tuples $(p_1, \ldots, p_{k+1})$ with $p_i \in P_i$ for all $i \in [k+1]$, $\|p_i - p_{i+1}\| = \delta_i$ for all $i \in [k]$ and with $p_i \neq p_j$ for $i \neq j$. Let $C_k^{\boldsymbol{\delta}}(P_1, \ldots, P_{k+1}) = |\mathcal{C}_k^{\boldsymbol{\delta}}(P_1, \ldots, P_{k+1})|$ and

$$C_k(n_1, \ldots, n_{k+1}) = \max C_k^{\boldsymbol{\delta}}(P_1, \ldots, P_{k+1}),$$

where the maximum is taken over all choices of $\boldsymbol{\delta}$ and sets $P_1, \ldots, P_{k+1}$ subject to $|P_i| \leq n_i$ for all $i \in [k+1]$.

It is easy to see that $C_k^2(n) \leq C_k(n, \ldots, n) \leq C_k^2((k+1)n)$. Since we are only interested in the order of magnitude of $C_k^2(n)$ for fixed k, we are going to bound $C_k(n, \ldots, n)$ instead of $C_k^2(n)$.

In Section 3.1, we are going to prove the lower bounds from Theorem 3. In Section 3.2, we are going to prove an upper bound on $C_k(n, \ldots, n)$, which is almost tight for $k \equiv 0, 2 \pmod 3$. The case $k \equiv 1 \pmod 3$ is significantly more complicated. We will the case $k = 4$ case separately in Section 3.3, and then the general case in Section 3.4.

3.1 Lower bounds

For completeness, we present constructions for all congruence classes modulo 3. For $k \equiv 0, 2$ they were described in [8].

[1] circles, if $d = 2$

First, note that $C_0(n) = n$ and $C_1(n,n) = u_2(n,n) = \Theta(u_2(n))$. For $k = 2$, let $P_2 = \{x\}$ for some point x, and let P_1, P_3 be disjoint sets of n points on the unit circle around x. It is not hard to see that $C_2^{\boldsymbol{\delta}}(P_1, P_2, P_3) = n^2$ with $\boldsymbol{\delta} = (1,1)$, implying the lower bound $C_2(n,n,n) = \Omega(n^2)$. To obtain lower bounds in Theorem 3, it is thus sufficient to show that

$$C_{k+3}(n,\dots,n) \geq nC_k(n,\dots,n).$$

To see this take, a construction with $k+1$ parts $P_1,\dots,P_{k+1}$ of size n that contains $C_k(n,\dots,n)$ $(k,\boldsymbol{\delta})$-chains for some $\boldsymbol{\delta} = (\delta_1,\dots,\delta_k)$. Next, fix an arbitrary point x on the plane and choose distances $\delta_{k+1}, \delta_{k+2}$ to be sufficiently large so that x can be connected to each of the points in P_{k+1} by a 2-chain with distances δ_{k+2} and δ_{k+1}. Set $P_{k+3} = \{x\}$ and let P_{k+2} be the set of intermediate points of the 2-chains described above. Finally, let $\delta_{k+3} = 1$, and P_{k+4} be a set of n points (disjoint from P_{k+2}) on the unit circle around x. It is easy to see that the number of $(k+3,\boldsymbol{\delta})$-chains with $\boldsymbol{\delta} = (\delta_1,\dots,\delta_{k+3})$ in $P_1 \times \dots \times P_{k+4}$ is at least $nC_k(n)$.

Note that it is not hard to modify this construction to show that for any given $\boldsymbol{\delta}$ there is a set of n points with $\Omega(n^{k/3+1})$ many $(k,\boldsymbol{\delta})$-chains if $k \equiv 0 \pmod 3$ and with $\Omega(n^{(k+4)/3})$ many $(k,\boldsymbol{\delta})$-chains if $k \equiv 2 \pmod 3$. However, for $k \equiv 1 \pmod 3$, our construction to find sets of n points with $\Omega(n^{(k-1)/3}u_2(n))$ many $(k,\boldsymbol{\delta})$-chains only works if δ_1 is much smaller than δ_2 and δ_3.

3.2 Upper bound for $k \equiv 0, 2 \pmod 3$

We fix $\boldsymbol{\delta} = (\delta_1,\dots,\delta_k)$ throughout the remainder of Section 3 and leave $\boldsymbol{\delta}$ out of the notation. All logs are base 2.

▶ **Theorem 7.** *For any fixed integer $k \geq 0$ and $x, y \in [0,1]$, we have*

$$C_k(n^x, n, \dots, n, n^y) = \tilde{O}\left(n^{\frac{f(k)+x+y}{3}}\right),$$

where $f(k) = k+2$ if $k \equiv 2 \pmod 3$ and $f(k) = k+1$ otherwise.

Theorem 7 implies the upper bounds in Theorem 3 for $k \equiv 0, 2 \pmod 3$ by taking $x = y = 1$. It is easier, however, to prove this more general statement than the upper bounds in Theorem 3 directly. Having varied sizes of the first and the last groups of points allows for a seamless use of induction.

Proof of Theorem 7. The proof is by induction on k. Let us first verify the statement for $k \leq 2$. (Note that, for $k = 0$, we should have $x = y$.) We have

$$C_0(n^x) \leq n^x = O\left(n^{\frac{1+x+y}{3}}\right),$$

$$C_1(n^x, n^y) \leq u_2(n^x, n^y) = O\left(n^{\frac{2}{3}(x+y)} + n^x + n^y\right) = O\left(n^{\frac{2+x+y}{3}}\right), \tag{6}$$

$$C_2(n^x, n, n^y) \leq n^x n^y = O\left(n^{\frac{4+x+y}{3}}\right), \tag{7}$$

where (6) follows from (2) and (7) follows from the fact that each pair (p_1, p_3) can be extended to a 2-chain (p_1, p_2, p_3) in at most 2 different ways.

Next, let $k \geq 3$. Take $P_1,\dots,P_{k+1} \subseteq \mathbb{R}^2$ with $|P_1| = n^x$, $|P_{k+1}| = n^y$, and $|P_i| = n$ for $2 \leq i \leq k$. Denote by $P_2^\alpha \subseteq P_2$ the set of those points in P_2 that are at least n^α-rich but at most $2n^\alpha$-rich with respect to P_1 and δ_1. Similarly, we denote by $P_k^\beta \subseteq P_k$ the set of those points in P_k that are at least n^β-rich but at most $2n^\beta$-rich with respect to P_{k+1} and δ_k.

It is not hard to see that

$$\mathcal{C}_k(P_1, P_2 \ldots, P_k, P_{k+1}) \subseteq \bigcup_{\alpha,\beta} \mathcal{C}_k(P_1, P_2^\alpha, P_3, \ldots, P_{k-1}, P_k^\beta, P_{k+1}),$$

where the union is taken over all $\alpha, \beta \in \{\frac{i}{\log n} : i = 0, \ldots, \lceil \log n \rceil\}$. Since the cardinality of the latter set is at most $\log n + 2$, it is sufficient to prove that for every α and β we have

$$C_k(P_1, P_2^\alpha, P_3, \ldots, P_{k-1}, P_k^\beta, P_{k+1}) = \tilde{O}\left(n^{\frac{f(k)+x+y}{3}}\right). \tag{8}$$

To prove this, we consider three cases.

Case 1: $\alpha \geq \frac{x}{2}$. By (3) we have $|P_2^\alpha| = O(n^{x-\alpha})$. Therefore the number of pairs $(p_1, p_2) \in P_1 \times P_2^\alpha$ with $\|p_1 - p_2\| = \delta_1$ is at most $O(n^x)$. Since every pair $(p_1, p_2) \in P_1 \times P_2^\alpha$ and every $(k-3)$-chain $(p_4, \ldots, p_{k+1}) \in P_4 \times \cdots \times P_k^\beta \times P_{k+1}$ can be extended to a k-chain $(p_1, \ldots, p_{k+1}) \in P_1 \times \cdots \times P_{k+1}$ in at most two different ways, we obtain

$$C_k(P_1, P_2^\alpha, \ldots, P_k^\beta, P_{k+1}) \leq 4O(n^x) C_{k-3}(P_4, \ldots, P_k^\beta, P_{k+1}).$$

By induction we have

$$C_{k-3}(P_4, \ldots, P_k^\beta, P_{k+1}) = \tilde{O}\left(n^{\frac{f(k-3)+1+y}{3}}\right).$$

These two displayed formulas and the fact that $f(k-3) = f(k) - 3$ imply (8).

Case 2: $\beta \geq \frac{y}{2}$. By symmetry, this case can be treated in the same way as Case 1.

Case 3: $\alpha \leq \frac{x}{2}$ and $\beta \leq \frac{y}{2}$. By (3) we have $|P_2^\alpha| = O\left(n^{2x-3\alpha}\right)$ and $|P_k^\beta| = O\left(n^{2y-3\beta}\right)$. The number of $(k-2)$-chains in $P_2^\alpha \times P_3 \times \cdots \times P_{k-1} \times P_k^\beta$ is $C_{k-2}(P_2^\alpha, P_3, \ldots, P_{k-1}, P_k^\beta)$, and every $(k-2)$-chain $(p_2, \ldots, p_k) \in P_2^\alpha \times P_3 \times \cdots \times P_{k-1} \times P_k^\beta$ can be extended at most $4n^{\alpha+\beta}$ ways to a k-chain in $P_1 \times P_2^\alpha \times \cdots \times P_k^\beta \times P_{k+1}$. Thus

$$C_k(P_1, P_2^\alpha, \ldots, P_k^\beta, P_{k+1}) \leq 4n^{\alpha+\beta} C_{k-2}(P_2^\alpha, \ldots, P_k^\beta).$$

By induction we have

$$C_{k-2}(P_2^\alpha, \ldots, P_k^\beta) = \tilde{O}\left(n^{\frac{f(k-2)+2x-3\alpha+2y-3\beta}{3}}\right).$$

For $k \equiv 0, 2 \pmod 3$ we have $f(k) \geq f(k-2) + 2$, and thus

$$\begin{aligned} C_k(P_1, P_2^\alpha, \ldots, P_k^\beta, P_{k+1}) &= \tilde{O}\left(n^{\alpha+\beta} n^{\frac{f(k-2)+2x-3\alpha+2y-3\beta}{3}}\right) \\ &= \tilde{O}\left(n^{\frac{f(k)-2+2x+2y}{3}}\right) = \tilde{O}\left(n^{\frac{f(k)+x+y}{3}}\right). \end{aligned}$$

If $k \equiv 1 \pmod 3$ then $f(k) < f(k-2) + 2$, and thus the argument above does not work. However, we then have $f(k) = f(k-1) + 1$, and we can use the bound

$$C_k(P_1, P_2^\alpha, \ldots, P_k^\beta, P_{k+1}) \leq 2n^\alpha C_{k-1}(P_2^\alpha, P_3, \ldots, P_{k+1}),$$

obtained in an analogous way. This gives

$$C_k(P_1, P_2^\alpha, P_3, \ldots, P_{k+1}) = \tilde{O}\left(n^\alpha n^{\frac{f(k-1)+2x-3\alpha+y}{3}}\right) = \tilde{O}\left(n^{\frac{f(k)-1+2x+y}{3}}\right) = \tilde{O}\left(n^{\frac{f(k)+x+y}{3}}\right).$$

◀

▶ Remark 8. The proof above is not sufficient to obtain an almost sharp bound in the $k \equiv 1 \pmod 3$ case for two reasons. First, for these k any analogue of Theorem 7 would involve taking maximums of two expressions, where one contains $u_2(n^x, n)$ and the other contains $u_2(n^y, n)$. However, due to our lack of good understanding of how $u_2(n^x, n)$ changes as x is increasing, this is difficult to work with.

Second, on a more technical side, while Case 1 and Case 2 in the above proof would go through with any reasonable inductive statement, Case 3 would fail. The main reason for this is that C_k as a function of k makes jumps at every third value of k, and remains essentially the same, or changes by $u(n,n)/n$ for the other values of k. Thus one would need to remove three vertices from the path to make the induction work. However, the path has only two ends, and removing vertices other than the endpoints turns out to be intractable.

3.3 Upper bound for $k = 4$

In this section we prove the upper bound in Theorem 3 for $k = 4$. Let $P_1, \dots, P_5$ be five sets of n points. We will show that $C_4(P_1, \dots, P_5) = \tilde{O}(u_2(n)n)$, which is slightly stronger than what is stated in Theorem 3.

Instead of (3) we need the following more general bound on the number of rich points.

▶ **Observation 9** (Richness bound)**.** *Let n^y be the maximum possible number of points that are n^α-rich with respect to a set of n^x points and some distance δ. Then we have*

$$n^{y+\alpha} \le u_2(n^x, n^y), \tag{9}$$

or, equivalently

$$n^\alpha \le \frac{u_2(n^x, n^y)}{n^y}.$$

The proof of (9) follows immediately from the definition of n^α richness and $u_2(n^x, n^y)$.

Let $\Lambda := \left\{\frac{i}{\log n} : i = 0, \dots, \lceil \log n \rceil\right\}^4$. For any $\boldsymbol{\alpha} = (\alpha_2, \alpha_3, \alpha_4, \alpha_5) \in \Lambda$ let $Q_1^{\boldsymbol{\alpha}} = P_1$ and for $i = 2, \dots, 5$ define recursively $Q_i^{\boldsymbol{\alpha}}$ to be the set of those points in P_i that are at least n^{α_i}-rich but at most $2n^{\alpha_i}$-rich with respect to Q_{i-1} and δ_i.

It is not difficult to see that

$$\mathcal{C}_4(P_1, \dots, P_5) = \bigcup_{\boldsymbol{\alpha} \in \Lambda} \mathcal{C}_4\left(Q_1^{\boldsymbol{\alpha}}, \dots, Q_5^{\boldsymbol{\alpha}}\right).$$

We have $|\Lambda| = \tilde{O}(1)$ and thus, in order to prove the theorem, it is sufficient to show that for every $\boldsymbol{\alpha} \in \Lambda$ we have

$$C_4\left(Q_1^{\boldsymbol{\alpha}}, \dots, Q_5^{\boldsymbol{\alpha}}\right) = O\left(n \cdot u_2(n,n)\right).$$

From now on, fix $\boldsymbol{\alpha} = (\alpha_2, \dots, \alpha_5)$, and denote $Q_i = Q_i^{\boldsymbol{\alpha}}$. Choose $x_i \in [0,1]$ so that $|Q_i| = n^{x_i}$. Then we have

$$C_4(Q_1, \dots, Q_5) = O\left(n^{x_5+\alpha_5+\alpha_4+\alpha_3+\alpha_2}\right). \tag{10}$$

Indeed, each chain $(p_1, \dots, p_5)$ with $p_i \in Q_i$ can be obtained in the following five steps.

- **Step 1:** Pick $p_5 \in Q_5$.
- **Step i ($2 \le i \le 5$):** Pick a point $p_{6-i} \in Q_{6-i}$ at distance δ_{6-i} from p_{7-i}.

In the first step we have n^{x_5} choices, and for $i \geq 2$ in the i-th step we have at most $2n^{\alpha_{6-i}}$ choices. Further, by Observation 9, for each $i \geq 2$ we have

$$n^{\alpha_i} \leq \frac{u_2(n^{x_{i-1}}, n^{x_i})}{n^{x_i}}. \tag{11}$$

Combining (10) and (11), we obtain

$$C_4(Q_1, \ldots, Q_5) = O\left(u_2(n^{x_4}, n^{x_5}) \frac{u_2(n^{x_3}, n^{x_4})}{n^{x_4}} \frac{u_2(n^{x_2}, n^{x_3})}{n^{x_3}} \frac{u_2(n^{x_1}, n^{x_2})}{n^{x_2}}\right). \tag{12}$$

By (2) we have

$$u_2(n^{x_{i-1}}, n^{x_i}) = O\left(\max\left\{n^{\frac{2}{3}(x_i + x_{i-1})}, n^{x_i}, n^{x_{i-1}}\right\}\right).$$

Note that the maximum is attained on the second (third) term iff $x_{i-1} \leq \frac{x_i}{2}$ ($x_i \leq \frac{x_{i-1}}{2}$). To bound $C_4(Q_1, \ldots, Q_5)$ we consider several cases depending on which of these three terms the maximum above is attained on for different i.

Case 1: For all $2 \leq i \leq 5$ we have $u_2(n^{x_{i-1}}, n^{x_i}) = O\left(n^{\frac{2}{3}(x_i + x_{i-1})}\right)$. Then

$$\frac{u_2(n^{x_4}, n^{x_5}) u_2(n^{x_3}, n^{x_4}) u_2(n^{x_2}, n^{x_3})}{n^{x_2 + x_3 + x_4}} = O\left(n^{\frac{2}{3}x_5 + \frac{1}{3}x_4 + \frac{1}{3}x_3 - \frac{1}{3}x_2}\right)$$

and

$$\frac{u_2(n^{x_3}, n^{x_4}) u_2(n^{x_2}, n^{x_3}) u_2(n^{x_1}, n^{x_2})}{n^{x_2 + x_3 + x_4}} = O\left(n^{-\frac{1}{3}x_4 + \frac{1}{3}x_3 + \frac{1}{3}x_2 + \frac{2}{3}x_1}\right).$$

Substituting each of these two displayed formulas into (12) and taking their product, we obtain

$$C_4(Q_1, \ldots, Q_5)^2 = O\left(u_2(n^{x_1}, n^{x_2}) u_2(n^{x_4}, n^{x_5}) \cdot n^{\frac{2}{3}x_1 + \frac{2}{3}x_3 + \frac{2}{3}x_5}\right) = O\left(u_2(n, n)^2 \cdot n^2\right),$$

which concludes the proof in this case.

Case 2: There is an $2 \leq i \leq 5$ such that

$$\min\{x_{i-1}, x_i\} \leq \frac{1}{2} \max\{x_{i-1}, x_i\} \quad \text{and thus} \quad u_2(n^{x_{i-1}}, n^{x_i}) = O\left(\max\{n^{x_{i-1}}, n^{x_i}\}\right). \tag{13}$$

We distinguish three cases based on for which i holds.

Case 2.1: (13) holds for $i = 2$ or 5. In particular, this implies that $u_2(n^{x_1}, n^{x_2}) = O(n)$ or $u_2(n^{x_4}, n^{x_5}) = O(n)$. The following lemma finishes the proof in this case.

▶ **Lemma 10.** *Let $R_1, \ldots, R_5 \subseteq \mathbb{R}^2$ such that $|R_i| \leq n$ for every $i \in [5]$. If $u_2(R_1, R_2) = O(n)$ or $u_2(R_4, R_5) = O(n)$ holds, then $C_4(R_1, \ldots, R_5) = O\left(n \cdot u_2(n, n)\right)$.*

Proof. We have

$$C_4(R_1, \ldots, R_5) \leq 2u_2(R_1, R_2) u_2(R_4, R_5) = O\left(n \cdot u_2(n, n)\right).$$

Indeed, every 4-tuple (r_1, r_2, r_4, r_5) with $r_i \in R_i$ can be extended in at most two different ways to a 4-chain $(r_1, \ldots, r_5) \in R_1 \times \cdots \times R_5$. At the same time, the number of 4-tuples with $\|r_1 - r_2\| = \delta_1$, $\|r_4 - r_5\| = \delta_4$ is at most $u_2(R_1, R_2) u_2(R_4, R_5)$. ◀

Case 2.2: (13) holds for $i = 4$. Note that if $x_4 \le \frac{x_3}{2} \le \frac{1}{2}$, then $u_2(n^{x_5}, n^{x_4}) = O(n)$, and we can apply Lemma 10 to conclude the proof in this case. Thus we may assume that $x_3 \le \frac{x_4}{2}$, and hence $u_2(n^{x_4}, n^{x_3}) = O(n^{x_4})$. This means that $n^{\alpha_4} = O(1)$ by Observation 9. Thus to finish the proof of this case, it is sufficient to prove the following claim.

▶ **Claim 11.** *Let $R_1, \dots, R_5 \subseteq \mathbb{R}^2$ such that $|R_i| \le n$ for all $i \in [5]$ and every point of R_4 is $O(1)$ rich with respect to R_3 and δ_3. Then $C_4(R_1, \dots, R_5) = O\left(n \cdot u_2(n, n)\right)$.*

Proof. Every 4-chain $(r_1, \dots, r_5)$ can be obtained in the following steps.

- Pick a pair $(r_4, r_5) \in R_4 \times R_5$ with $\|r_4 - r_5\| = \delta_4$.
- Choose $r_3 \in R_3$ at distance δ_3 from r_4.
- Pick a point $r_1 \in R_1$.
- Extend (r_1, r_3, r_4, r_5) to a 4-chain.

In the first step, we have at most $u_2(n, n)$ choices, in the third at most n choices, and in the other two steps at most $O(1)$. ◀

Case 2.3: (13) holds for $i = 3$ *only*. Arguing as in Case 2.2, we may assume that $u_2(n^{x_3}, n^{x_2}) = O(n^{x_2})$. Then we have

$$C_4(Q_1, \dots, Q_5) = O\left(u_2(n^{x_4}, n^{x_5}) \frac{u_2(n^{x_3}, n^{x_4})}{n^{x_4}} \frac{u_2(n^{x_2}, n^{x_3})}{n^{x_3}} \frac{u_2(n^{x_1}, n^{x_2})}{n^{x_2}}\right)$$

$$= O\left(u_2(n^{x_1}, n^{x_2}) \cdot n^{\frac{2}{3}(x_4+x_5)+\frac{2}{3}(x_3+x_4)-x_4-x_3}\right) = O\left(u_2(n, n) \cdot n\right),$$

which finishes the proof.

3.4 Upper bound for $k \equiv 1 \pmod 3$

We will prove the upper bound in Theorem 3 for $k \equiv 1$ by induction. The $k = 1$ case follows from the definition of $u_2(n, n)$, thus we may assume that $k \ge 4$. For the rest of the section fix $\varepsilon' > 0$, and sets $P_1, \dots, P_{k+1} \subseteq \mathbb{R}^2$ of size n, further let $\varepsilon = \frac{\varepsilon'}{4k}$. We are going to show that $C_k(P_1, \dots, P_{k+1}) = O(n^{(k-1)/3+\varepsilon'} u_2(n))$.

The first step of the proof is to find a certain covering of $P_1 \times \dots \times P_{k+1}$, which resembles the one used for the $k = 4$ case, although is more elaborate.[2] (The goal of this covering is to make the corresponding graph between each of the two consecutive parts "regular in both directions" in a certain sense.)

Let

$$\Lambda = \left\{i\varepsilon : i = 0, \dots, \left\lfloor \frac{1}{\varepsilon} \right\rfloor\right\}^{k+1}.$$

We cover the product $\mathbf{P} = P_1 \times \dots \times P_{k+1}$ by fine-grained classes $P_1^{\boldsymbol{\gamma}} \times \dots \times P_{k+1}^{\boldsymbol{\gamma}}$ encoded by the sequence $\boldsymbol{\gamma} = (\boldsymbol{\gamma^1}, \boldsymbol{\gamma^2}, \dots)$ of length at most $(k+1)\varepsilon^{-1} + 1$ with $\boldsymbol{\gamma^j} \in \Lambda$ for each $j = 1, 2, \dots$. One property that we shall have is

$$P_1 \times \dots \times P_{k+1} = \bigcup_{\boldsymbol{\gamma}} P_1^{\boldsymbol{\gamma}} \times \dots \times P_{k+1}^{\boldsymbol{\gamma}}.$$

To find the covering, first we define a function D that receives a parity digit $j \in \{0, 1\}$, a product set $\mathbf{R} := R_1 \times \dots \times R_{k+1}$ and an $\boldsymbol{\alpha} \in \Lambda$, and outputs a product set $D(j, \boldsymbol{R}, \boldsymbol{\alpha}) = \mathbf{R}(\boldsymbol{\alpha}) = R_1(\boldsymbol{\alpha}) \times \dots \times R_{k+1}(\boldsymbol{\alpha})$.

[2] This covering brings in the ε-error term in the exponent, that we could avoid in the $k = 4$ case.

Definition of D.

- If $j = 1$ then let $R_1(\boldsymbol{\alpha}) := R_1$ and for $i = 2, \dots, k+1$ define $R_i(\boldsymbol{\alpha})$ iteratively to be the set of points in R_i that are at least n^{α_i}, but at most $n^{\alpha_i+\varepsilon}$-rich with respect to $R_{i-1}(\boldsymbol{\alpha})$ and δ_{i-1}.
- If $j = 0$ then apply the same procedure, but in reverse order. That is, let $R_{k+1}(\boldsymbol{\alpha}) = R_{k+1}$ and for $i = k, k-1, \dots, 1$ define $R_i(\boldsymbol{\alpha})$ iteratively to be the set of points in R_i that are at least n^{α_i} but at most $n^{\alpha_i+\varepsilon}$-rich with respect to $R_{i+1}(\boldsymbol{\alpha})$ and δ_i.

Note that

$$\mathbf{R} = \bigcup_{\boldsymbol{\alpha}\in\Lambda} \mathbf{R}(\boldsymbol{\alpha}). \tag{14}$$

For a sequence $\boldsymbol{\gamma} = (\boldsymbol{\gamma^1}, \boldsymbol{\gamma^2}, \dots)$ with $\boldsymbol{\gamma^j} \in \Lambda$, we define $\mathbf{P}^{\boldsymbol{\gamma}}$ recursively as follows. Let $\mathbf{P}^{\emptyset} := \mathbf{P}$, and for each $j \geq 1$ let

$$\mathbf{P}^{(\boldsymbol{\gamma^1},\dots,\boldsymbol{\gamma^j})} = D(j \pmod 2, \mathbf{P}^{(\boldsymbol{\gamma^1},\dots,\boldsymbol{\gamma^{j-1}})}, \boldsymbol{\gamma^j}).$$

We say that a sequence $\boldsymbol{\gamma}$ is *stable at* j if

$$\left|\mathbf{P}^{(\boldsymbol{\gamma^1},\dots,\boldsymbol{\gamma^j})}\right| \geq \left|\mathbf{P}^{(\boldsymbol{\gamma^1},\dots,\boldsymbol{\gamma^{j-1}})}\right| \cdot n^{-\varepsilon}.$$

Otherwise $\boldsymbol{\gamma}$ is *unstable at* j.

▶ **Definition 12.** *Let Υ be the set of those sequences $\boldsymbol{\gamma}$ that are stable at their last coordinate, but are not stable for any previous coordinate, and for which $\boldsymbol{P}^{\boldsymbol{\gamma}}$ is non-empty.*

The set Υ has several useful properties, some of which are summarised in the following lemma.

▶ **Lemma 13.**

1. *Any $\boldsymbol{\gamma} \in \Upsilon$ has length at most $(k+1)\varepsilon^{-1} + 1$.*
2. $|\Upsilon| = O_\varepsilon(1)$.
3. $\boldsymbol{P} = \bigcup_{\boldsymbol{\gamma}\in\Upsilon} \boldsymbol{P}^{\boldsymbol{\gamma}}$.

Proof.

1. If $\boldsymbol{\gamma}$ is unstable at j then

   $$|\mathbf{P}^{(\boldsymbol{\gamma}^1,\dots,\boldsymbol{\gamma}^j)}| \leq |\mathbf{P}^{(\boldsymbol{\gamma}^1,\dots,\boldsymbol{\gamma}^{j-1})}| \cdot n^{-\varepsilon}.$$

   Since $|\mathbf{P}| = n^{k+1}$ and $|\mathbf{P}^{\boldsymbol{\gamma}}| \geq 1$, we conclude that $\boldsymbol{\gamma}$ is unstable at at most $(k+1)\varepsilon^{-1}$ indices j.
2. It follows from part 1 by counting all possible sequences of length at most $(k+1)\varepsilon^{-1}+1$ of elements from the set Λ. (Note that $|\Lambda| = O_\varepsilon(1)$.)
3. For a nonnegative integer j let $\Lambda^{\leq j}$ be the set of all sequences of length at most j of elements from Λ. Let

   $$\Upsilon_j := \left(\Upsilon \cap \Lambda^{\leq j}\right) \cup \Psi_j, \text{ where } \Psi_j := \left\{\boldsymbol{\gamma} \in \Lambda^j : \boldsymbol{\gamma} \text{ is not stable for any } \ell \leq j\right\}.$$

   By part 1 of the lemma, $\Upsilon_j = \Upsilon$ for $j > (k+1)\varepsilon^{-1}$. We prove by induction on j that $\mathbf{P} = \bigcup_{\boldsymbol{\gamma}\in\Upsilon_j} \mathbf{P}^{\boldsymbol{\gamma}}$.
   Υ_0 consists of an empty sequence, thus the statement is clear for $j = 0$. Next, assume that the statement holds for j. We have

   $$\mathbf{P} = \bigcup_{\boldsymbol{\gamma}\in\Upsilon_j} \mathbf{P}^{\boldsymbol{\gamma}} = \bigcup_{\boldsymbol{\gamma}\in\Lambda^{\leq j}} \mathbf{P}^{\boldsymbol{\gamma}} \cup \bigcup_{\boldsymbol{\gamma}\in\Psi_j} \mathbf{P}^{\boldsymbol{\gamma}}.$$

By (14) we have that $\mathbf{P}^{\boldsymbol{\gamma}} = \bigcup_{\boldsymbol{\gamma}'} \mathbf{P}^{\boldsymbol{\gamma}'}$ holds for any $\boldsymbol{\gamma} \in \Psi_j$, where the union is taken over the sequences from Λ^{j+1} that coincide with $\boldsymbol{\gamma}$ on the first j entries. This, together with $\boldsymbol{\gamma}' \in \left(\Upsilon \cap \Lambda^{j+1}\right) \cup \Psi_{j+1}$ when $\mathbf{P}^{\boldsymbol{\gamma}'}$ is nonempty finishes the proof. ◀

Parts 2 and 3 of Lemma 13 imply that in order to complete the proof of the $k \equiv 1 \pmod 3$ case, it is sufficient to show that for any $\boldsymbol{\gamma} \in \Upsilon$ we have

$$C_k(P_1^{\boldsymbol{\gamma}}, \dots, P_{k+1}^{\boldsymbol{\gamma}}) = O\left(u_2(n) \cdot n^{\frac{k-1}{3}+4k\varepsilon}\right). \tag{15}$$

From now on fix $\boldsymbol{\gamma} \in \Upsilon$. For each $i = 1, \dots, k+1$ let $R_i := P_i^{\boldsymbol{\gamma}}$ and $Q_i := P_i^{\boldsymbol{\gamma}'}$, where $\boldsymbol{\gamma}'$ is obtained from $\boldsymbol{\gamma}$ by removing the last element of the sequence. Without loss of generality, assume that the length ℓ of $\boldsymbol{\gamma}$ is even. For each $i = 1, \dots, k+1$, choose x_i, y_i such that

$$|Q_i| = n^{x_i}, \quad |R_i| = n^{y_i}.$$

Let $\alpha_i := \boldsymbol{\gamma}_i^{\ell-1}$ and $\beta_i := \boldsymbol{\gamma}_i^{\ell}$. By the definition of $\mathbf{P}^{\boldsymbol{\gamma}}$ we have that each point in Q_i is at least n^{α_i}-rich but at most $n^{\alpha_i+\varepsilon}$-rich with respect to Q_{i-1} and δ_{i-1}, and each point in R_i is at least n^{β_i}-rich but at most $n^{\beta_i+\varepsilon}$-rich with respect to R_{i+1} and δ_i.

By Observation 9, we have

$$n^{\alpha_i} \le \frac{u_2(n^{x_{i-1}}, n^{x_i})}{n^{x_i}} \quad \text{and} \quad n^{\beta_i} \le \frac{u_2(n^{y_i}, n^{y_{i+1}})}{n^{y_i}} \le \frac{u_2(n^{x_i}, n^{x_{i+1}})}{n^{x_i-\varepsilon}}. \tag{16}$$

The last inequality follows from two facts: first $u_2(n^{y_i}, n^{y_{i+1}}) \le u_2(n^{x_i}, n^{x_{i+1}})$ and, second, since $\boldsymbol{\gamma}$ is stable at its last coordinate[3], we have $n^{y_i} = |R_i| \ge |Q_i| \cdot n^{-\varepsilon} = n^{x_i-\varepsilon}$.

In the same fashion as in the beginning of Section 3.3, we can show that

$$C_k(R_1, \dots, R_{k+1}) \le n^{y_1} n^{\beta_1+\dots+\beta_k+k\varepsilon}, \text{ and}$$

$$C_k(R_1, \dots, R_{k+1}) \le C_k(Q_1, \dots, Q_{k+1}) \le n^{x_{k+1}} n^{\alpha_{k+1}+\alpha_k+\dots+\alpha_2+k\varepsilon}.$$

Combining the first of these displayed inequalities with (16), we have

$$C_k(R_1, \dots, R_{k+1}) \le u_2(n^{x_1}, n^{x_2}) \prod_{2 \le i \le k} \frac{u_2\left(n^{x_i}, n^{x_{i+1}}\right)}{n^{x_i}} n^{2k\varepsilon}.$$

Recall that

$$u_2(n^{x_i}, n^{x_{i+1}}) = O\left(\max\{n^{\frac{2}{3}(x_i+x_{i+1})}, n^{x_i}, n^{x_{i+1}}\}\right). \tag{17}$$

To bound $C_k(R_1, \dots, R_{k+1})$, we consider several cases based on which of these three terms can be used to bound $u_2(n^{x_i}, n^{x_{i+1}})$ for different values of i.

Case 1: Either $u_2(n^{x_1}, n^{x_2}) = O(n)$ or $u_2(n^{x_k}, n^{x_{k+1}}) = O(n)$ holds. As in the proof of Lemma 10, we have

$$C_k(R_1, \dots, R_{k+1}) \le \min\left\{2u_2(n^{y_1}, n^{y_2}) C_{k-3}(R_4, \dots, R_{k+1}), 2u_2(n^{y_k}, n^{y_{k+1}}) C_{k-3}(R_1, \dots, R_{k-2})\right\}.$$

By induction we obtain $C_{k-3}(R_4, \dots, R_{k+1}), C_{k-3}(R_1, \dots, R_{k-2}) = O\left(n^{\frac{k-4}{3}+\varepsilon} \cdot u_2(n)\right)$. Together with the assumption of Case 1, and the fact that $u_2(n^{y_1}, n^{y_2}) \le u_2(n^{x_1}, n^{x_2})$ and $u_2(n^{y_k}, n^{y_{k+1}}) \le u_2(n^{x_k}, n^{x_{k+1}})$, this implies (15) and finishes the proof.

[3] This is essentially the only place where we use the stability of $\boldsymbol{\gamma}$.

Case 2: For some $i = 1, \dots, (k-1)/3$, one of the following holds:
- $u_2(n^{x_{3i+1}}, n^{x_{3i+2}}) = O(\max\{n^{x_{3i+1}}, n^{x_{3i+2}}\})$;
- $u_2(n^{x_{3i-1}}, n^{x_{3i}}) = O(n^{x_{3i-1}})$;
- $u_2(n^{x_{3i}}, n^{x_{3i+1}}) = O(n^{x_{3i+1}})$.

We will show how to conclude in the first case. The other cases are very similar and we omit the details of their proofs. If $u_2(n^{x_{3i+1}}, n^{x_{3i+2}}) = O(n^{x_{3i+2}})$ then $n^{\alpha_{3i+2}} = O(1)$ by (16). Every chain $(r_1, \dots, r_{k+1}) \in \mathcal{C}_k(Q_1, \dots, Q_{k+1})$ can be obtained as follows.
1. Pick a $(3i-2)$-chain $(r_1, \dots, r_{3i-1})$ with $r_j \in Q_j$ for every j.
2. Pick a $(k-3i-1)$-chain $(r_{3i+2}, r_{3i+3}, \dots, r_{k+1})$ with $r_j \in Q_j$ for every j.
3. Extend $(r_{3i+2}, r_{3i+3}, \dots, r_{k+1})$ to a $(k-3i-2)$ chain $(r_{3i+1}, r_{3i+2}, \dots, r_{k+1})$.
4. Connect $(r_1, \dots, r_{3i-1})$ and $(r_{3i+1}, r_{3i+2}, \dots, r_{k+1})$ to obtain a k-chain.

In the first step, we have $O\left(n^{\frac{3i-3}{3}+\varepsilon} \cdot u_2(n)\right)$ choices by induction on k. In the second step, we have $\tilde{O}\left(n^{\frac{k-3i+2}{3}}\right)$ choices by the $k \equiv 0 \pmod 3$ case of Theorem 3. In the third step, we have at most $n^{\alpha_{3i+2}+\varepsilon} = O(n^{\varepsilon})$ choices. Finally, in the fourth step we have at most 2 choices. Thus the number of k-chains is at most

$$O\left(n^{\frac{3i-3}{3}+\varepsilon} \cdot u_2(n)\right) \cdot \tilde{O}\left(n^{\frac{k-3i+2}{3}}\right) \cdot O\left(n^{\varepsilon}\right) \cdot 2 = O\left(n^{\frac{k-1}{3}+3\varepsilon} \cdot u_2(n)\right),$$

finishing the proof of the first case.

If $u_2(n^{x_{3i+1}}, n^{x_{3i+2}}) = O(n^{x_{3i+1}})$ then $n^{\beta_{3i+1}} = O(n^{\varepsilon})$ by (16).[4] We proceed similarly in this case, but we count the k-chains now in $R_1 \times \ldots \times R_{k+1}$ instead in $Q_1 \times \ldots \times Q_{k+1}$ (and get an extra factor of n^{ε} in the bound). In all cases, we obtain (15).

Case 3: Neither the assumptions of Case 1 nor that of Case 2 hold. We define four sets S', S'_+, S'_{++}, and S'_- of indices in $\{2, \dots, k\}$ as follows. Let

$$S' := \left\{ i : u_2(n^{x_i}, n^{x_{i-1}}) = O(n^{\frac{2}{3}(x_i+x_{i-1})}) \text{ and } u_2(n^{x_{i+1}}, n^{x_i}) = O(n^{\frac{2}{3}(x_{i+1}+x_i)}) \right\},$$

$$S'_+ := \Big\{ i : u_2(n^{x_i}, n^{x_{i-1}}) = O(n^{\frac{2}{3}(x_i+x_{i-1})}) \text{ and } u_2(n^{x_{i+1}}, n^{x_i}) = O(n^{x_i}), \text{ or}$$
$$u_2(n^{x_i}, n^{x_{i-1}}) = O(n^{x_i}) \text{ and } u_2(n^{x_{i+1}}, n^{x_i}) = O(n^{\frac{2}{3}(x_{i+1}+x_i)}) \Big\},$$

$$S'_{++} := \left\{ i : u_2(n^{x_i}, n^{x_{i-1}}) = O(n^{x_i}) \text{ and } u_2(n^{x_{i+1}}, n^{x_i}) = O(n^{x_i}) \right\}, \text{ and}$$

$$S'_- := \Big\{ i : u_2(n^{x_i}, n^{x_{i-1}}) = O(n^{\frac{2}{3}(x_i+x_{i-1})}) \text{ and } u_2(n^{x_{i+1}}, n^{x_i}) = O(n^{x_{i+1}}), \text{ or}$$
$$u_2(n^{x_i}, n^{x_{i-1}}) = O(n^{x_{i-1}}) \text{ and } u_2(n^{x_{i+1}}, n^{x_i}) = O(n^{\frac{2}{3}(x_{i+1}+x_i)}) \Big\}.$$

Since the conditions of Case 2 are not satisfied, we have

$$\{2, \dots, k\} \subseteq S' \cup S'_+ \cup S'_{++} \cup S'_-.$$

Indeed, for each $i \in \{2, \dots, k\}$, there are 9 possible pairs of maxima in (17) with $i, i+1$. The four sets above encompass 6 possibilities. In total, there are 4 possible pairs of maxima with

[4] This is the key application of (16), and the reason why we needed a decomposition with regularity in both directions between the consecutive parts.

only the two last terms from (17) used. For $i \equiv 1,2 \pmod 3$, any of those 4 are excluded due to the first condition in Case 2 (in fact, then $i \in S' \cup S'_-$). If $i \equiv 0 \pmod 3$, then the second and the third condition in Case 2 rule out all possibilities but the one defining S'_{++}.

From these, it is also easy to see that if $i \in S'_{++}$, then $i-1, i+1 \in S'_-$, while if $i \in S'_+$ then one of $i-1, i+1$ is in S'_-. (Recall that $i \in S'_+ \cup S'_{++}$ only if $i \equiv 0 \pmod 3$.) These together imply

$$|S'_+| + 2|S'_{++}| \le |S'_-|. \tag{18}$$

We partition $\{2,\dots,k\}$ using these sets as follows: let $S_- = S'_-, S = S' \setminus S'_-, S_+ = S'_+ \setminus (S'_- \cup S')$ and $S_{++} = \{2,\dots,k\} \setminus S'_- \cup S' \cup S'_+$. Note that the analogue of (18) holds for the new sets. That is, we have

$$|S_+| + 2|S_{++}| \le |S_-|.$$

Recall that

$$C_k(R_1,\dots,R_{k+1}) \le u_2(n^{x_1}, n^{x_2}) \prod_{2\le i\le k} \frac{u_2\left(n^{x_i}, n^{x_{i+1}}\right)}{n^{x_i}} n^{2k\varepsilon}. \tag{19}$$

Since the assumptions of Case 1 and 2 do not hold, we have $2,k \in S$. Indeed, $2,k \neq 0 \pmod 3$ and thus $2,k \notin S_+, S_{++}$. Further, if say $k \in S_- = S'_-$ then by the definition of S'_- we either have $u_2(n^{x_{k+1}}, n^{x_k}) = O(n)$, or $u_2(n^{x_k}, n^{x_{k-1}}) = O(n^{x_{k-1}})$. The first case cannot hold since the assumption of Case 1 does not hold. Further, the second case cannot hold either, since it would imply $x_k \le \frac{x_{k-1}}{2} \le \frac{1}{2}$, meaning $u_2(n^{x_{k+1}}, n^{x_k}) = O(n)$. Using $2,k \in S$ and expanding (19), we obtain

$$C_k(R_1,\dots,R_{k+1}) \le n^{2k\varepsilon} u_2(n^{x_1}, n^{x_2}) n^{-\frac{1}{3}x_2} n^{\frac{2}{3}x_{k+1}} \prod_{\substack{i\in S,\\ i\neq 2}} n^{\frac{1}{3}x_i} \prod_{i\in S_+} n^{\frac{2}{3}x_i} \prod_{i\in S_{++}} n^{x_i} \prod_{i\in S_-} n^{-\frac{1}{3}x_i}, \tag{20}$$

and

$$C_k(R_1,\dots,R_{k+1}) \le n^{2k\varepsilon} u_2(n^{x_k}, n^{x_{k+1}}) n^{-\frac{1}{3}x_k} n^{\frac{2}{3}x_1} \prod_{\substack{i\in S,\\ i\neq k}} n^{\frac{1}{3}x_i} \prod_{i\in S_+} n^{\frac{2}{3}x_i} \prod_{i\in S_{++}} n^{x_i} \prod_{i\in S_-} n^{-\frac{1}{3}x_i}. \tag{21}$$

Taking the product of (20) and (21) we obtain

$$C_k(R_1,\dots,R_{k+1})^2 \le$$

$$n^{4k\varepsilon} \cdot u_2(n^{x_1}, n^{x_2}) u_2(n^{x_k}, n^{x_{k+1}}) n^{\frac{2}{3}(x_1+x_{k+1})} \left(\prod_{\substack{i\in S,\\ i\neq 2,k}} n^{\frac{1}{3}x_i} \prod_{i\in S_+} n^{\frac{2}{3}x_i} \prod_{i\in S_{++}} n^{x_i} \prod_{i\in S_-} n^{-\frac{1}{3}x_i} \right)^2$$

$$\le n^{4k\varepsilon} \cdot u_2(n,n)^2 \cdot n^{2\left(\frac{2}{3}+\frac{1}{3}|S\setminus\{2,k\}|+\frac{2}{3}|S_+|+|S_{++}|\right)} = u_2(n,n)^2 \cdot n^{\frac{2(k-1)}{3}+4k\varepsilon}.$$

The last equality follows from $|S_+| + 2|S_{++}| \le |S_-|$, which is equivalent to $\frac{2}{3}|S_+| + |S_{++}| \le \frac{1}{3}(|S_+| + |S_{++}| + |S_-|)$, and from the fact that S, S_+,S_{++}, and S_- partition $\{2,\dots,k\}$. This finishes the proof.

4 Bounds in $\mathbb{R}^3$

Similarly as in the planar case, for $\boldsymbol{\delta} = (\delta_1, \dots, \delta_k)$ and $P_1 \dots, P_{k+1} \subseteq \mathbb{R}^3$ we denote by $\mathcal{C}_k^{3,\boldsymbol{\delta}}(P_1, \dots, P_k)$ the family of $(k+1)$-tuples $(p_1, \dots, p_{k+1})$ with $p_i \in P_i$ for all $i \in [k+1]$ and with $\|p_i - p_{i+1}\| = \delta_i$ for all $i \in [k]$. Let $C_k^{3,\boldsymbol{\delta}}(P_1, \dots, P_{k+1}) = |\mathcal{C}_k^{3,\boldsymbol{\delta}}(P_1, \dots, P_{k+1})|$ and

$$C_k^3(n_1, \dots, n_{k+1}) = \max C_k^{3,\boldsymbol{\delta}}(P_1, \dots, P_{k+1}),$$

where the maximum is taken over all choices of $\boldsymbol{\delta}$ and sets $P_1, \dots, P_{k+1}$ subject to $|P_i| \leq n_i$ for all $i \in [k+1]$.

It is easy to see that $C_k^3(n) \leq C_k^3(n, \dots, n) \leq C_k^3((k+1)n)$. Since we are only interested in the order of magnitude of $C_k^3(n)$ for fixed k, sometimes we are going to work with $C_k^3(n, \dots, n)$ instead of $C_k^3(n)$.

4.1 Lower bounds

For completeness, we recall the constructions from [8] for even $k \geq 2$. For every even $2 \leq i \leq k$, let $P_i = \{p_i\}$ be a single point such that the unit spheres centred at p_i and p_{i+2} intersect in a circle. Further, let P_1 and P_{k+1} be a set of n points contained in the unit sphere centred at p_2 and p_k respectively. Finally, for every odd $3 \leq i \leq k-1$, let P_i be a set of n points contained in the intersection of the unit spheres centred at p_{i-1} and p_{i+1}. Then it is not hard to see that $P_1 \times \dots \times P_{k+1}$ contains $n^{\frac{k}{2}+1}$ many $(k, \boldsymbol{\delta})$-chains for $\boldsymbol{\delta} = (1, \dots, 1)$.

Next, we prove the lower bounds for odd $k \geq 3$ given in Proposition 6.

Proof of Proposition 6. First we show that $C_k^3(n) = \Omega\left(\frac{u_3(n)^k}{n^{k-1}}\right)$. Take a set $P' \subset \mathbb{R}^3$ of size n that contains $u_3(n)$ point pairs at unit distance apart. It is a standard exercise in graph theory to show that there is $P \subset P'$ such that $\frac{n}{2} \leq |P| \leq n$ and for every $p \in P$ there are at least $\frac{u_3(n)}{4n}$ points $p' \in P$ at distance 1 from p. Then P contains $\Omega\left(\frac{u_3(n)^k}{n^{k-1}}\right)$ many $(k, \boldsymbol{\delta})$-chains with $\boldsymbol{\delta} = (1, \dots, 1)$.

To prove $C_k^3(n) = \Omega\left(us_3(n)n^{k-2}\right)$, we modify and extend the construction used for $k-1$ as follows. Let $P_1, \dots, P_{k-1}$ be as in the construction for $(k-1)$-chains (from the even case). Further, let P_k be a set of n points on the unit sphere around p_{k-1}, and P_{k+1} be a set of n points such that $u_3(P_k, P_{k+1}) = us_3(n)$. It is not hard to see that $P_1 \times \dots \times P_{k+1}$ contains $\Omega\left(us_3(n)n^{k-2}\right)$ many $(k, \boldsymbol{\delta})$-chains with $\boldsymbol{\delta} = (1, \dots, 1)$. ◀

4.2 Upper bound

We again fix $\boldsymbol{\delta} = (\delta_1, \dots, \delta_k)$ throughout the section and, omit it from the notation. The following result with $x = 1$ implies the upper bound in Theorem 5.

▶ **Theorem 14.** *For any fixed integer $k \geq 0$ and $x \in [0, 1]$, we have*

$$C_k^3(n^x, n, \dots, n) = \tilde{O}\left(n^{\frac{k+1+x}{2}}\right).$$

Proof. The proof is by induction on k. For $k = 0$ the bound is trivial, and for $k = 1$ it follows from (4).

For $k \geq 2$ let $P_1, \dots, P_{k+1} \subseteq \mathbb{R}^3$ be sets of points satisfying $|P_1| = n^x$, and $|P_i| = n$ for $2 \leq n \leq k+1$. Denote by $P_2^\alpha \subseteq P_2$ the set of those points in P_2 that are at least n^α-rich but at most $2n^\alpha$-rich with respect to P_1 and δ_1.

It is not hard to see that

$$\mathcal{C}_k^3(P_1, P_2 \dots, P_{k+1}) \subseteq \bigcup_{\alpha \in \Lambda} \mathcal{C}_k^3(P_1, P_2^\alpha, P_3, \dots, P_{k+1}),$$

where $\Lambda := \{\frac{i}{\log n} : i = 0, 1, \dots, \lfloor \log n \rfloor\}$. Since $|\Lambda| = \tilde{O}(1)$, it is sufficient to prove that, for every $\alpha \in \Lambda$, we have

$$C_k^3(P_1, P_2^\alpha, P_3, \dots, P_{k+1}) = \tilde{O}\left(n^{\frac{k+1+x}{2}}\right).$$

Assume that $|P_2^\alpha| = n^y$. The number of $(k-1)$-chains in $P_2^\alpha \times P_3 \times \dots \times P_{k+1}$ is at most $C_{k-1}^3(n^y, n, \dots, n)$, and each of them may be extended in $2n^\alpha$ ways. By induction, we get

$$C_k^3(P_1, P_2^\alpha, P_3, \dots, P_{k+1}) = \tilde{O}\left(n^\alpha \cdot n^{\frac{k+y}{2}}\right),$$

and we are done as long as

$$2\alpha + k + y \le k + 1 + x. \tag{22}$$

To show this, we need to consider several cases depending on the value of α. Note that $\alpha \le x$.

- If $\alpha \ge \frac{2x}{3}$, then by (5) we have $y \le x - \alpha$, and the LHS of (22) is at most $\alpha + k + x \le 1 + k + x$.
- If $\frac{x}{2} \le \alpha \le \frac{2x}{3}$ then by (5) we have $y \le 3x - 4\alpha$. The LHS of (22) is at most $k + 3x - 2\alpha \le k + 2x \le k + 1 + x$.
- If $\alpha \le \frac{x}{2}$ then we use a trivial bound $y \le 1$. The LHS of (22) is at most $2\alpha + k + 1 \le x + k + 1$. ◀

References

1. P. K. Agarwal, E. Nevo, J. Pach, R. Pinchasi, M. Sharir, and S. Smorodinsky. Lenses in arrangements of pseudo-circles and their applications. *J. ACM*, 51(2):139–186, 2004.
2. B. Aronov and M. Sharir. Cutting circles into pseudo-segments and improved bounds for incidences. *Discrete Comput. Geom.*, 28(4):475–490, 2002.
3. P. Erdős. On sets of distances of n points. *Amer. Math. Monthly*, 53(5):248–250, 1946.
4. P. Erdős. On sets of distances of n points in euclidean space. *Magyar Tud. Akad. Mat. Kutato Int. Közl.*, 5:165–169, 1960.
5. H. Kaplan, J. Matoušek, Z. Safernová, and M. Sharir. Unit distances in three dimensions. *Comb. Probab. Comput.*, 21(4):597–610, 2012.
6. A. Marcus and G. Tardos. Intersection reverse sequences and geometric applications. *J. Combin. Theory Ser. A*, 113(4):675–691, 2006.
7. J. Pach and M. Sharir. Geometric incidences. In *Towards a theory of geometric graphs*, volume 342 of *Contemp. Math.*, pages 185–223. Amer. Math. Soc., Providence, RI, 2004.
8. E. A. Palsson, S. Senger, and A. Sheffer. On the number of discrete chains, 2019. `arXiv:1902.08259`.
9. J. Spencer, E. Szemerédi, and W. T Trotter. Unit distances in the Euclidean plane. In *Graph theory and combinatorics*, pages 294–304. Academic Press, 1984.
10. J. Zahl. An improved bound on the number of point-surface incidences in three dimensions. *Contrib. Discrete Math.*, 8(1):100–121, 2013.
11. J. Zahl. Breaking the 3/2 barrier for unit distances in three dimensions. *Int. Math. Res. Notices*, 2019(20):6235–6284, 2019.

Convex Hulls of Random Order Types

Xavier Goaoc
Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France
xavier.goaoc@loria.fr

Emo Welzl
Department of Computer Science, ETH Zürich, Switzerland
emo@inf.ethz.ch

Abstract

We establish the following two main results on order types of points in general position in the plane (realizable simple planar order types, realizable uniform acyclic oriented matroids of rank 3):

(a) The number of extreme points in an n-point order type, chosen uniformly at random from all such order types, is on average $4 + o(1)$. For labeled order types, this number has average $4 - \frac{8}{n^2-n+2}$ and variance at most 3.

(b) The (labeled) order types read off a set of n points sampled independently from the uniform measure on a convex planar domain, smooth or polygonal, or from a Gaussian distribution are concentrated, i.e., such sampling typically encounters only a vanishingly small fraction of all order types of the given size.

Result (a) generalizes to arbitrary dimension d for labeled order types with the average number of extreme points $2d + o(1)$ and constant variance. We also discuss to what extent our methods generalize to the abstract setting of uniform acyclic oriented matroids. Moreover, our methods allow to show the following relative of the Erdős-Szekeres theorem: for any fixed k, as $n \to \infty$, a proportion $1 - O(1/n)$ of the n-point simple order types contain a triangle enclosing a convex k-chain over an edge.

For the unlabeled case in (a), we prove that for any antipodal, finite subset of the 2-dimensional sphere, the group of orientation preserving bijections is cyclic, dihedral or one of A_4, S_4 or A_5 (and each case is possible). These are the finite subgroups of $SO(3)$ and our proof follows the lines of their characterization by Felix Klein.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases order type, oriented matroid, Sylvester's Four-Point Problem, random convex hull, projective plane, excluded pattern, Hadwiger's transversal theorem, hairy ball theorem

Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.49

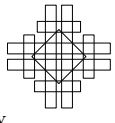
Related Version A full version is available at [17], `https://arxiv.org/abs/2003.08456`.

Funding *Xavier Goaoc*: Supported by grant ANR-17-CE40-0017 of the French National Research Agency (ANR project ASPAG) and Institut Universitaire de France.
Emo Welzl: Supported by the Swiss National Science Foundation within the collaborative DACH project Arrangements and Drawings as SNSF Project 200021E-171681.

Acknowledgements This research started at the Banff Workshop "Helly and Tverberg Type Theorems", October 6-11, 2019, at the Casa Matemática Oaxaca (CMO), Mexico. The authors thank Boris Aronov for helpful discussions, Gernot Stroth for help on the group theoretic aspects of the paper, and Pierre Calka for help on questions involving probabilistic geometry.

1 Introduction

Two finite subsets P and Q of the plane are said to *have the same order type* if there exists a bijection $f : P \to Q$ that preserves orientations: for any three points p, q, r in P, r is to the left (resp. to the right) of the line (pq) oriented from p to q if and only if $f(r)$ is to the left



(resp. to the right) of the line $(f(p)f(q))$ oriented from $f(p)$ to $f(q)$. To have the same order type is an equivalence relation, and an *order type* is an equivalence class for that relation. For each n, properties that depend solely on orientations can be established for the infinitely many n-point sets by proving it for (one representative of) each of the finitely many order types of size n. This even allows proofs by automated case analysis, see for instance [2] for an application to crossing numbers and asymptotic estimates on numbers of triangulations. This notion was studied in discrete and computational geometry as a higher-dimensional analogue of ordering on a line, but also as a geometric relative of oriented matroids, see e.g., [18, 26, 9, 13].

In this paper, we investigate the expected number of *extreme* points in a typical order type. (Since the number of extreme points is the same for all representatives of an order type, we speak of the number of extreme points of the order type; we do the same for every notion independent of the choice of representative, e.g., the size.) Here we consider only simple order types, i.e., with no three points on a line; by "typical" we mean an order type chosen equiprobably among all simple order types of a given size n. As an illustration, for $n = 4$, the only two simple order types are the convex quadrilateral and the triangle with an interior point, so the quantity we are after is $\frac{4+3}{2} = \frac{7}{2}$. For $n = 5$, it is $\frac{5+4+3}{3} = 4$, see Figure 1.

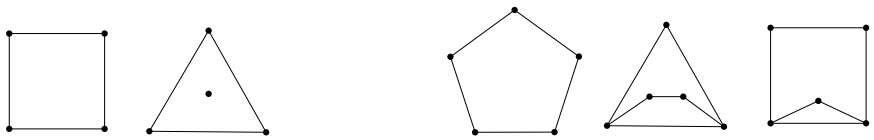


Figure 1 Left: The two simple 4-point order types. Right: The three simple 5-point order types.

1.1 Motivations

Let us say a word on our motivations.

Testing. We are interested in statistics of the uniform distribution on the space of order types. Broadly speaking, this distribution is relevant whenever one wants to *test* a property of finite point sets. Consider the two following examples:

(a) The largest point set in general position with no empty hexagon is known to have size between 29 and 1716 [32, 16], and it is tempting to try and improve the lower bound by testing order types of size 30 or so.

(b) The CGAL library [38] stresses the need, when implementing geometric algorithms, to rely solely on predicates that depend on the input of the algorithm, so as to encapsulate the numerical issues (critical for robustness [25]) into the correct evaluation of signs of polynomials. Thus, the implementation of an algorithm that depends only on orientation predicates can be assessed by running it on a realization of each possible order type.

In both cases, we want to avoid repeating the same order type, as this is redundant computation, and to be able in principle to reach every existing order type without uncontrolled bias. The uniform distribution is natural to consider for that purpose.

Random polytopes. Counting extreme points relates to the study of face vectors of random polytopes, a classical line of research in stochastic geometry initiated by Sylvester in 1865, who asked for "the probability that 4 points in the plane are in convex position". A standard

model of random polytope K_n is the convex hull of n random points chosen uniformly and independently in some fixed convex body K. In this setting, the number of extreme points, i.e., the vertices of K_n, is well understood. Its average is asymptotically proportional to $n^{\frac{d-1}{d+1}} + o\left(n^{\frac{d-1}{d+1}}\right)$ if K is smooth and to $\log^{d-1} n + o\left(\log^{d-1} n\right)$ if K is a polytope [34, 35] (see [33, §2.2.2]), and up to multiplicative constant these are the two extremes [6, Theorems 1–3]. There are also estimates on the variance, concentration inequalities, central limit theorems, and large deviation inequalities. We refer the interested reader to the survey of Reitzner [33].

This model of random polytope naturally generalizes to arbitrary probability measures μ, or even to the convex hull of random non-independent point sets such as determinantal point processes. Much less is known in this direction, aside from the occasional extensively-studied model such as Gaussian polytopes (see [33, §2.3]). In a sense, what we investigate is the average number of extreme points in a random polytope for a *combinatorially defined* probability distribution on point sets.

Exploration of order types. The space of order types is generally not well understood. Already, its size is not known precisely, not even asymptotically. The most precise bounds are given for *labeled* order types, which declare two point *sequences* $P = (p_1, p_2, \ldots)$ and $Q = (q_1, q_2, \ldots)$ equivalent if the monotone map $p_i \mapsto q_i$ preserves orientations: there are $n^{4n}\phi(n)$ labeled order types, where $2^{-cn} \leq \phi(n) \leq 2^{c'n}$ for some positive constants c, c' [19, 3]; factoring out the labelling is not immediate as the number of labeled order types corresponding to a given unlabeled one depends on the symmetries of the latter. We show that in the plane, every unlabeled order type corresponds to at least $(n-1)!$ (and clearly at most $n!$) different labeled ones. Order types have been tabulated up to size 11 [1, 2], for which they are already counted in billions.

Random sampling of order types is also quite unsatisfactory. First, the standard methods in discrete random generation such as Boltzmann samplers are unlikely to work here, as they require structural results (such as recursive decompositions) that usually make counting a routine task. It is of course easy to produce a random order type by merely reading off the order type of n random points; standard models include points chosen independently from the uniform distribution in a square or a disk, from a Gaussian distribution, as well as points obtained as a random 2-dimensional projection of a n-dimensional simplex[1]. There are no results, however, on how well or badly distributed the order types of such random point sets are. More generally, no random generation method is known to be both efficient (say, taking polynomial time per sample) and with controlled bias. This sad state of affairs can perhaps be explained by two fundamental issues: when working with order types symbolically (say as orientation maps to $\{-1, 0, 1\}$, see Section 1.4 below), one has to work around the NP-hardness (actually, $\exists\mathbb{R}$-completeness) of membership testing [37, 29, 36]. When working with explicit point sets, one has to account for the exponential growth of the worst-case number of coordinate bits required to realize an order type of size n [20]. It turns out that our bounds on the expected number of extreme points in an order type imply that several standard models of random point sets typically explore only a vanishingly small fraction of the space of order types (Theorem 3).

Order types with forbidden patterns. Given two order types ω and τ, we say that ω *contains* τ if any point set that realizes ω contains a subset that realizes τ. (Of course this needs only be checked for a single realization of ω.) By the Erdös-Szekeres theorem [14],

[1] This is called the Goodman-Pollack model and is statistically equivalent to points chosen independently from a Gaussian distribution [7, Theorem 1].

almost all order types contain the order type of k points in convex position. Similarly, Carathéodory's theorem implies that almost all order types contain the order type of a triangle with one interior point. Could it be that for any *fixed* order type τ, the number of order types of size n that *do not* contain τ is vanishingly small as $n \to \infty$? This question may seem quite bold given the limited number of observations, but it is also motivated by an analogous phenomenon for permutations: the Marcus-Tardos theorem [27] asserts that for every fixed permutation π, the number of size-n permutations that *do not* contain π is at most exponential in n (see [27] for the definition of containment). We are not aware of any result on such a Marcus-Tardos phenomenon for order types besides the two simple cases mentioned above. It turns out that along the way, we prove some new results in this direction as well (Theorem 4).

1.2 Results

Our first result is on labeled order types. Two affine point sequences $(p_1, p_2, \ldots, p_n)$ and $(q_1, q_2, \ldots, q_n)$ are defined to be of the same *labeled order type* if the map $p_i \mapsto q_i$ preserves orientations: for any indices $1 \leq i, j, k \leq n$, p_k is to the left (resp. to the right) of the line $(p_i p_j)$ oriented from p_i to p_j if and only if q_k is to the left (resp. to the right) of the line $(q_i q_j)$ oriented from q_i to q_j. The labeled order type of a point sequence is *simple* if no three points of that sequence are aligned.

▶ **Theorem 1.** *For $n \geq 3$, the number of extreme points in a random simple labeled order type chosen uniformly among the simple, labeled order types of size n in the plane has average $4 - \frac{8}{n^2-n+2}$ and variance at most 3.*

A set of n points gives rise to $n!$ point sequences, but different sequences may have the same labeled order type. The exact number of labeled order types corresponding to a given order type actually depends on the number of order-preserving bijections, that is *symmetries*, of that order type. We show that the symmetries of a simple affine order type form a (possibly trivial) cyclic group (Theorem 6) and we bound from above the number of simple affine order types with many, but not too many, symmetries. We then prove a non-labeled analogue of Theorem 1:

▶ **Theorem 2.** *For $n \geq 3$, the number of extreme points in a random simple order type chosen uniformly among the simple order types of size n in the plane has average $4 + O\left(n^{-3/4+\varepsilon}\right)$ for any $\varepsilon > 0$.*

Our proof of Theorem 1 extends to arbitrary dimension, but not our proof of Theorem 2. A large part of our methods and results extend to *abstract order types*, that is uniform oriented matroids, where lines are replaced by pseudo-line arrangements. In particular, Theorem 1 holds in the abstract setting with the same bound, also in arbitrary dimension. The proof of Theorem 2 does not completely carry over to the abstract setting, but our methods yield an analogue statement with a bound of $10 + o(1)$.

Theorems 1 and 2 are in sharp contrast with the $\Omega(\log n)$, and possibly polynomially many, extreme points in a uniform random sample of a convex planar domain. Theorems 1 and 2 can actually be used to turn concentration bounds on the number of extreme points in a random point set into concentration results on the distribution of order types produced by these random point sets.

We need some definitions. Let $\mathsf{(L)OT}_n^{\text{aff}}$ denote the set of simple (labeled) affine order types. For $n \geq 3$, let μ_n be a probability measure on $\mathsf{(L)OT}_n^{\text{aff}}$. We say that the family $\{\mu_n\}_{n\geq 3}$ *exhibits concentration* if for every $n \geq 3$ there exists $A_n \subseteq \mathsf{(L)OT}_n^{\text{aff}}$ such that

$\mu_n(A_n) \to 1$ and $|A_n|/|(\mathsf{L})\mathsf{OT}_n^{\text{aff}}| \to 0$. In plain English, families of measures that exhibit concentration typically explore a vanishingly small fraction of the space of simple (labeled) order types. Devillers et al. [12] conjectured that the order types of points sampled uniformly and independently from a unit square exhibit concentration. We prove this conjecture and more:

▶ **Theorem 3.** *Let μ be a probability measure on $\mathbb{R}^2$ given by one of the following: (a) the uniform distribution on a smooth compact convex set, (b) the uniform distribution on a convex compact polygon, (c) a Gaussian distribution. The family of probabilities on $(\mathsf{L})\mathsf{OT}_n^{\text{aff}}$ defined by the (labeled) order type of n random points chosen independently from μ exhibits concentration.*

Since the random projection of the vertices of a regular n-dimensional simplex, the Goodman-Pollack model, is distributed like a set of points sampled independently from a Gaussian distribution [7] (see also [33, §2.3.1]), the distribution on random order types it produces in the plane also exhibits concentration.

As we explain in the next paragraphs, we prove Theorems 1 and 2 by recasting affine order types in a projective setting, where we study so-called projective order types. This relation between affine and projective order types reveals more examples of order types difficult to avoid.

▶ **Theorem 4.** *For any integer $k \geq 2$, the proportion of order types of size n that contain a triangle and k points forming a convex chain over one edge is $1 - O(1/n)$.*

Our final result is a classification of the symmetry groups of simple projective order types: we prove that they are exactly the finite subgroups of $SO(3)$, the group of rotations (Theorem 7).

1.3 Approach

Our proof of Theorems 1 and 2 divides up the simple planar order types into their orbits under the action of projective transforms, and averages the number of extreme points inside each orbit. Let us illustrate this "action" we consider with the two order types of Figure 2. Starting with the left hand-side convex pentagon, any projective transform $\mathbb{R}^2 \to \mathbb{R}^2$ that maps the dashed line to the line at infinity yields the triangle with two interior points on the right. Following up with any other projective transform that sends the dotted line back to infinity will turn the triangle with two interior points back into a convex pentagon. We invite the reader to check that all three simple order types of size 5 (Figure 1) form a single orbit under projective transforms.

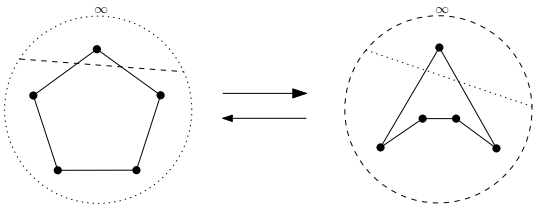


Figure 2 Two projectively equivalent planar order types.

Here is a simple example of how such projective transforms may help:

▶ **Lemma 5.** *Let A be a finite planar point set in general position and $t : \mathbb{R}^2 \to \mathbb{R}^2$ a projective transform with the line sent to infinity disjoint from A, and splitting A. Then there are at most 4 extreme vertices of A whose images are also extreme in $t(A)$.*

Proof. Let ℓ be the line sent by t to infinity. The extreme points of $t(A)$ are exactly the images of the points of A that ℓ can touch by moving continuously without crossing over a point of A. It is the union of two convex chains, on either side of ℓ, and each chain contains at most 2 points extreme in A. ◀

This essentially allows to match order types of size n so that in every pair, the size of the convex hulls add up to at most $n + 4$. Assuming one dealt with issues such as symmetries, this could provide an upper bound of $n/2 + 2$ on the average number of extreme points in a typical order type. We do not formalize this matching idea further, but recast it into a projective that makes it easier to analyze the action of projective transforms on order types.

1.4 Setting and terminology

We take all our points on the origin-centered unit sphere $\mathbb{S}^2$ in $\mathbb{R}^3$, except for occasional mentions of the origin $\mathbf{0}$. Two points p and q on the sphere are called *antipodal*, if $q = -p$. A *great circle* is the intersection of the sphere with a plane containing $\mathbf{0}$, an *open hemisphere* is a connected component of the sphere in the complement of a great circle, and a *closed hemisphere* is the closure of an open one. A finite subset P of the sphere is a *projective set* if $p \in P \Leftrightarrow -p \in P$. We call a finite set of points on the sphere an *affine set* if it is contained in an open hemisphere. An affine set is in *general position* if no three points are coplanar with $\mathbf{0}$; a projective set P is in *general position* if whenever three points in P are coplanar with $\mathbf{0}$, two of them are antipodal.

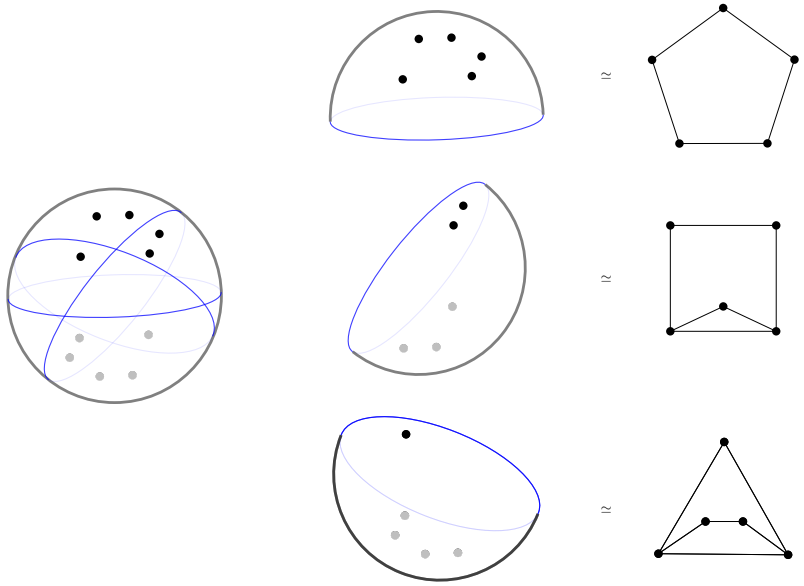


Figure 3 A projective set of size 10 (left) containing the three simple affine order types of size 5.

The *sign*, $\chi(p, q, r)$, of a triple (p, q, r) of points on the sphere is the sign, -1, 0, or 1, of the determinant of the matrix $(p, q, r) \in \mathbb{R}^{3\times 3}$. A bijection $f : S \to S'$ between finite subsets of the sphere is *orientation preserving* if $\chi(f(p), f(q), f(r)) = \chi(p, q, r)$ for every triple of points in S. Two affine (resp. projective) sets have *the same affine* (resp. *projective*) *order type* if there exists an orientation preserving bijection between them. An *affine* (resp. *projective*) *order type* is the equivalence class of all affine (resp. projective) sets that have the same affine (resp. projective) order type. The definitions of *labeled* affine and projective

are similar: the labeling determines the bijection that is required to preserve orientations. It will sometimes be convenient to write a point sequence as $A_{[\lambda]}$, where A is the point set and $\lambda : A \to [n]$, $n = |A|$, the bijection specifying the ordering.

The plane $\mathbb{R}^2$ together with its orientation function can be mapped to any open hemisphere of $\mathbb{S}^2$ together with χ. For example, for the open hemisphere $\mathbb{S}^2 \cap \{z > 0\}$ this can be done by the map

$$\begin{pmatrix} x \\ y \end{pmatrix} \mapsto \frac{1}{x^2+y^2+1} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}.$$

Hence, the planar order types discussed so far coincide with the affine order types and we, in fact, prove Theorems 1 and 2 for (labeled) affine order types. We study affine order types as subsets of projective point sets as shown in Figure 3; this inclusion requires some care and is formalized in Section 3.

Let S be a finite subset of the sphere. A *permutation* of S is a bijection $S \to S$ and a *symmetry* of S is an orientation preserving permutation of S. The symmetries of S form a group, which we call the *symmetry group* of S. This group determines the relations between labeled and non-labeled affine order types: two orderings $A_{[\lambda]}$ and $A_{[\mu]}$ of a point set A determine the same labeled order type if and only if $\mu^{-1} \circ \lambda$ is a symmetry of A. A crucial ingredient in our proof of Theorem 2 is a classification of the symmetry groups of the affine and projective sets. Here it is for affine sets. (The definitions of convex layers and lonely point are given in Section 2.3.)

▶ **Theorem 6.** *The symmetry group of any affine set A in general position is isomorphic to the cyclic group $\mathbb{Z}_k$ for some $k \in \mathbb{N}$ that divides the size of every layer of A other than its lonely point (if A has one). In particular, k divides $|A|$ (if A has no lonely point) or $|A| - 1$ (if A has a lonely point, which can happen for k odd only).*

For all values of k and n satisfying the conditions of Theorem 6, with the exception of $(k, n) = (2, 4)$, there exists an affine order type of size n with $\mathbb{Z}_k$ as symmetry group (see Figure 4).

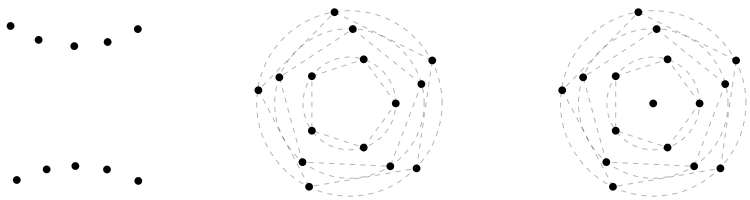


Figure 4 Left: For any even $n \geq 6$, there exists an affine set of n points with symmetry group $\mathbb{Z}_2$: take two sufficiently flat convex chains of $n/2$ points each, facing each other (so-called double chain, [31]). Center and Right: For any $3 \leq k \leq n$ where k divides n or for any odd k where k divides $n - 1$, there exists an affine set of n points with symmetry group $\mathbb{Z}_k$: just pile up regular polygons inscribed in concentric circles.

We also prove that the symmetry groups of projective sets are finite subgroups of $SO(3)$.

▶ **Theorem 7.** *The symmetry group of any projective set of $2n$ points in general position is a finite subgroup of $SO(3)$. In particular, it is one of the following groups: $\mathbb{Z}_1$ (trivial group), $\mathbb{Z}_m$ (cyclic group), D_m (dihedral, with $m \mid n$ or $m \mid n - 1$), S_4 (octahedral = cubical), A_4 (tetrahedral), and A_5 (icosahedral).*

We give examples of projective point sets with symmetry groups of each of the types identified in Theorem 7.

Notation. Let us introduce or recall some notation. For $n \geq 3$ we write $\mathsf{LOT}_n^{\text{aff}}$ for the set of simple labeled affine order types of size n, $\mathsf{OT}_n^{\text{aff}}$ for the set of simple affine order types of size n, and $\mathsf{OT}_n^{\text{proj}}$ for the set of simple projective order types of size $2n$. For an affine point set A with affine order type ω, we write $\mathsf{LOT}_A^{\text{aff}} = \mathsf{LOT}_\omega^{\text{aff}}$ for the set of the labeled affine order types of the orderings of A.

1.5 Related work

Studying planar order types through their projective analogues is not a new idea, and appears for instance in the tabulation of planar order types of size 11 [2]. We are not aware, however, of an earlier analysis of how this relation is affected by symmetries.

Perhaps our most direct predecessor is the work of Miyata [28] on the classification of symmetry groups of oriented matroids. These structures coincide with abstract order types, and the affine order types we consider are special ("realizable") cases. Miyata classifies the symmetries of abstract order types in dimension 1 and 2. Our proof of Theorem 6 extends to the abstract setting and offers a more direct alternative to Miyata's proof [28, §6]. Also related is the $O(n^d)$ time algorithm of Aloupis et al. [4, Theorem 4.1] for computing the automorphisms of an order type (what we will call the symmetry group of orientation preserving permutations) for a set of n points in $\mathbb{R}^d$.

Several recent works have studied order types of random point sets [10, 12, 15, 21, 39], but they do not address the *equiprobable* distribution on n-point order types. The recent work of Chiu et al. [11] comes closer, as they have looked at the average size of the jth level in a random planar arrangement of n lines, chosen by fixing a projective line arrangement of size n and equiprobably choosing a random cell to contain the south-pole. This is similar to what we do, but let us stress that they do not take symmetries into account, so the actual distribution on planar arrangements they consider is not equiprobable (not even among those contained in the projective arrangement).

Order types with forbidden patterns were previously investigated in two directions. On the one hand, the Erdős-Szekeres theorem was strengthened for order types with certain forbidden patterns [30, 23, 24]. On the other hand, Han et al. [21] studied the patterns contained in random samples. We are not aware of previous results on the number of order types with a forbidden pattern.

Finally, let us point out that the study of random polytopes raises other questions close to classical questions in discrete and computational geometry. The analysis through floating bodies [6] of f-vectors of random polytopes obtained from convex bodies is close to the ϵ-net theory for halfspaces (see also [22] and [5, §3.2]). In another direction, Blaschke proved that the probability that 4 points chosen uniformly in a convex domain are in convex position is minimized when the domain is a triangle; for arbitrary planar probability measures, this merely asks for the limit as $n \to \infty$ of the rectilinear crossing number of the complete graph K_n.

1.6 Paper organization

Due to space limitation, we had to make some choices as to what to keep here. We decided to present a self-contained proof of Theorem 1 as it already gives a taste of our methods. This is essentially a prefix of the full version [17].

From here, Section 2 recalls some background material. Then, Section 3 clarifies the relation between affine and projective order types, between their symmetry groups, and between the affine subsets of a projective set and the cells of its dual arrangement. Section 4

then proves Theorem 1 by relating the number of extreme points in a random affine order type to the number of edges in a random cell of an arrangement of great circles, and by analyzing such arrangements via double counting and the zone theorem.

2 Background

We recall here some notions in finite group theory and in discrete geometry on $\mathbb{S}^2$ (duality, arrangements, convexity).

2.1 Groups

The elements of group theory we use deal with a subgroup G of the group of permutations of a finite set X. The identity map, the neutral element in G, is denoted by id or id_X. We will study such a group G through its action on X or some set of subsets of X. The *orbit* $G(x)$ of $x \in X$ is the image of x under G, *ie.* $G(x) \stackrel{\text{def}}{=} \{g(x) \mid g \in G\}$. Any two elements have disjoint or equal orbits, so the orbits partition X. The *stabilizer* of an element $x \in X$ is the set of permutations in G having x as a fixed point, *ie.* $G_x \stackrel{\text{def}}{=} \{g \in G \mid g(x) = x\}$. By the *orbit-stabilizer theorem*, $|G| = |G(x)| \cdot |G_x|$ for any $x \in X$. We write $\simeq$ for group isomorphism.

2.2 Duality and arrangements on $\mathbb{S}^2$

On the sphere, the dual of a point p is the great circle p^* contained in the plane through $\mathbf{0}$ and orthogonal to the line $\mathbf{0}p$. For any finite subset S of the sphere, we write S^* for the arrangement of the family of great circles $\{p^* \mid p \in S\}$.

Let P be a projective set of $2n$ points. Since antipodal points have the same dual great circle, P^* is an arrangement of n great circles. Observe that P is in general position if and only if no three great circles in P^* have a point in common. Any two great circles intersect in two points, so P^* has $2\binom{n}{2}$ vertices. Every vertex is incident to four edges; the total number of edges is therefore $4\binom{n}{2}$. By Euler's formula, P^* has $2\binom{n}{2}+2$ faces of dimension 2, which we call *cells*.

Let us recall that many combinatorial quantities on arrangements of great circles on $\mathbb{S}^2$ are essentially twice their analogues for arrangements of lines in $\mathbb{R}^2$. Indeed, starting with an arrangement P^* of n great circles in general position, we can add another great circle C_∞, chosen so that $P^* \cup \{C_\infty\}$ is also in general position, and consider the two open hemispheres bounded by C_∞. Each open hemisphere can be mapped to $\mathbb{R}^2$ so that the half-circles of P^* are turned into lines, and the two line arrangements are combinatorially equivalent by antipodality. In this way, we can for instance obtain the following version of the zone theorem from the bound given in [8] for the zone of a line in an arrangement of lines[2]:

▶ **Theorem 8** (Zone Theorem). *Let P^* be an arrangement of n great circles on $\mathbb{S}^2$ and let $p^* \in P^*$. Let $Z(p^*)$ denote the* zone *of p^*, i.e., the set of cells of the arrangement incident to p^*. For a cell c, let $|c|$ denote the number of edges incident to c. Then $\sum_{c \in Z(p^*)} |c| \leq 19(n-1) - 10$.*

[2] [8] shows that the cells in the zone of a line h_0 in an arrangement of $n+1$ lines in the plane has edge-complexity at most $\lfloor 19n/2 \rfloor - 1$. For translating this bound to the zone of a great circle in an arrangement of n great circles on $\mathbb{S}^2$, (i) we replace n by $n-1$, (ii) we double for the two sides of C_∞, and (iii) we subtract 8 for the edges that get merged along C_∞ (note that the infinite edges on h_0 get merged and contribute 1 on each of their sides).

2.3 Convexity on the sphere

A point $p \in A$ is *extreme* in an affine set A if there exists a great circle C that strictly separates p from $A \setminus \{p\}$; that is, p and $A \setminus \{p\}$ lie on two different connected components of $\mathbb{S}^2 \setminus C$. An ordered pair $(p, q) \in A^2$ is a *positive extreme edge* of A if for any $r \in A \setminus \{p, q\}$ we have $\chi(p, q, r) = +1$. Assuming general position, a point $p \in A$ is extreme in A if and only if there exists $q \in A$ such that (p, q) is a positive extreme edge; in that case, the point q is unique.

A *CCW order* of the extreme points of A is an order $(p_0, p_1, \ldots, p_{h-1})$ of its extreme points such that for all $i = 0, 1, \ldots, h-1$, (p_i, p_{i+1}) is a positive extreme edge (indices mod h). The *convex hull* of A is

$$\mathsf{conv}(A) \stackrel{\text{def}}{=} \{r \in \mathbb{S}^2 \mid \forall \text{ positive extreme edges } (p, q), \chi(p, q, r) \geq 0\}.$$

An affine set A is *in convex position* if every point is extreme in A. The (onion) *layer sequence* of A is a sequence $(A_0, A_1, \ldots, A_\ell)$ of subsets of A, partitioning A, where A_0 is the set of extreme points in A, and $(A_1, A_2, \ldots, A_\ell)$ is the layer sequence of $A \setminus A_0$. The A_i's are called the *layers* of A. If the innermost layer A_ℓ consists of a sole point, then that point is called *lonely* (there is one or no lonely point).

3 Hemisets: relating affine and projective order types

Any affine set A naturally defines a projective set $A \cup -A$, which we call its *projective completion*. Going in the other direction, consider a projective set P. Any affine set whose projective completion is P must be the intersection of P with some open hemisphere. Remark, however, that the converse is not always true: the set $P = \{(\pm 1, 0, 0), (0, \pm 1, 0), (0, 0, \pm 1)\}$, the vertices of the cross polytope, intersects some open hemispheres in a single point. This reveals that for an open hemisphere to cut out an affine set that completes to P, it must be bounded by a great circle that avoids P. We therefore define a *hemiset* of P as the intersection of P with a *closed* hemisphere, and call a hemiset of P an *affine hemiset* if it is contained in an open hemisphere. With these definitions, we have:

▷ Claim 9. A projective set P is the completion of an affine set A if and only if A is an affine hemiset of P.

Notation. For a projective point set P with projective order type π, we write $\mathsf{OT}_P^{\text{proj}} = \mathsf{OT}_\pi^{\text{proj}}$ for the set of affine order types of the affine hemisets of P.

To understand how affine order types relate to projective order types, an important idea is that the symmetries of a projective point set P act on the (affine) hemisets of P:

▶ **Proposition 10.** *Let $g : P \to P'$ be an orientation preserving bijection between two projective sets in general position. If $|P| = |P'| \geq 6$, then g maps hemisets of P to hemisets of P' and affine hemisets of P to affine hemisets of P'.*

The proof of Proposition 10 starts by a simple observation of independent interest.

▶ **Lemma 11.** *Let $g : S \to S'$ be an orientation preserving bijection between two subsets of the sphere. If S contains two antipodal points $\{p, -p\}$ such that $g(-p) \neq -g(p)$, then S is contained in a great circle.*

Proof. If $g(-p)$ and $g(p)$ are not antipodal, then they are on a unique great circle, which must contain S, as for every $r \in S$ we have $0 = \chi(p, -p, g^{-1}(r)) = \chi\left(g(p), g(-p), r\right)$. ◀

Proof of Proposition 10. Let B be a hemiset of P and $E = B \cap -B$. By general position, $|B|$ is 4, 2 or it is 0 (in which case B is an affine hemiset). Since $|P| \geq 6$ and P is in general position, P is not contained in a great circle and g therefore preserves antipodality by Lemma 11. In particular, if g preserves hemisets, it also preserves affine hemisets.

If $|E| = 4$, then there are two points $p, q \in E$ such that $B = \{r \in P \mid \chi(p,q,r) \geq 0\}$. Since g preserves orientations and is bijective, it comes that

$$g(B) = \{r \in P \mid \chi(g(p), g(q), r) \geq 0\} = P \cap \{s \in \mathbb{S}^2 \mid \chi(g(p), g(q), s) \geq 0\},$$

and $g(B)$ is also a hemiset.

So assume that $|E| \leq 2$ and fix some closed hemisphere Σ such that $B = \Sigma \cap P$ and B intersects the boundary of Σ into E. We extend B into a set B'' with $|B'' \cap -B''| = 4$ as follows:

- If $|E| = 2$, then we set $B' \stackrel{\text{def}}{=} B$, $\Sigma' \stackrel{\text{def}}{=} \Sigma$ and $\{q, -q\} \stackrel{\text{def}}{=} E$. Otherwise, we fix a point p on the boundary of Σ, rotate Σ about $\mathbf{0}p$ until we first touch a point $q \in P \setminus B$ (at the same moment, $-q \in B$ moves from the interior to the boundary of the rotating hemisphere); we let Σ' denote the resulting hemisphere and put $B' \stackrel{\text{def}}{=} B \cup \{q\}$ (note $B' \cap -B' = \{q, -q\}$).
- We now rotate Σ' about $\mathbf{0}q$ until we first touch a point $r \in P \setminus B'$; we put $B'' \stackrel{\text{def}}{=} B' \cup \{r\}$.

Now, $E'' \stackrel{\text{def}}{=} B'' \cap -B'' = \{q, -q, r, -r\}$ and there exists a closed hemisphere Σ^* such that $g(B'') = P \cap \Sigma^*$ (by our previous analysis above for case $|E| = 4$). The boundary of Σ^* intersects P in precisely $g(E'') = \{g(-q), g(q), g(-r), g(r)\}$, and two adequate rotations kick only $g(r)$, then $g(q)$ out, witnessing that $g(B)$ is also a hemiset. ◀

Given a projective set P with symmetry group G and a hemiset B of P, we write G_B for the stabilizer of B in the action of G on hemisets of P. We also write $\mathsf{G}(B)$ for the orbit of B in that action.

▶ **Lemma 12.** *Let P be a projective set of $2n$ points, $n \geq 3$, in general position and A an affine hemiset of P.*

(a) *The symmetry group of A, as an affine set, is isomorphic to G_A.*

(b) *An affine hemiset of P has the same affine order type as A if and only if it is in $\mathsf{G}(A)$.*

Proof. Let F denote the symmetry group of A as an affine set. Since $P = A \cup -A$, we can extend any $f \in \mathsf{F}$ into a permutation $\hat{f}$ of P by setting $\hat{f}(p) \stackrel{\text{def}}{=} f(p)$ for $p \in A$ and $\hat{f}(p) \stackrel{\text{def}}{=} -f(-p)$ for $p \notin A$. Let $\hat{\mathsf{F}} \stackrel{\text{def}}{=} \{\hat{f} \colon f \in \mathsf{F}\}$. Remark that $\hat{\mathsf{F}}$ is isomorphic to F since for any two symmetries f_1, f_2 of A, we have $\widehat{f_1 \circ f_2} = \hat{f}_1 \circ \hat{f}_2$. Moreover, any element $g \in \hat{\mathsf{F}}$ fixes A and, conversely, any symmetry $g : P \to P$ that fixes A writes $g = \widehat{g_{|A}}$. Then, $\hat{\mathsf{F}} = \mathsf{G}_A$ and statement (a) follows.

For statement (b), consider an affine hemiset A' of P with the same affine order type as A. There exists an orientation preserving bijection $f : A \to A'$. The extension $\hat{f}$ of f to P also preserves orientations, and is therefore in G. It follows that $A' \in g(A)$. The reverse inclusion follows from the fact that every symmetry of G preserves orientations. ◀

With Lemma 12, the orbit-stabilizer theorem readily implies:

▶ **Corollary 13.** *Let P be a projective set of $2n$ points, $n \geq 3$, in general position and A an affine hemiset of P. Let F and G denote the symmetry groups of A and P, respectively. There are $|\mathsf{G}|/|\mathsf{F}|$ affine hemisets of P with same affine order type as A.*

4 Analysis of labeled affine order types

Perhaps surprisingly, Corollary 13 is all we need to prove Theorem 1.

4.1 The two roles of affine symmetries

The number of symmetries of an affine order type determines both its number of labelings, and how often it occurs among the hemisets of a projective completion of one of its realizations. These two roles happen to balance each other out nicely:

▶ **Proposition 14.** *Let P be a projective set of $2n$ points, $n \geq 3$, in general position. Let R be a random affine hemiset chosen uniformly among all affine hemisets of P. Let λ be a random permutation $R \to [n]$ chosen uniformly among all such permutations. The labeled affine order type of $R_{[\lambda]}$ is uniformly distributed in $\bigcup_{\omega \in OT_P^{\text{aff}}} LOT_\omega^{\text{aff}}$.*

Proof. Let N denote the number of affine hemisets of P. Let $\omega_1, \omega_2, \ldots, \omega_k$ denote the order types of the affine hemisets of P, without repetition (that is, the ω_i are pairwise distinct). Let G denote the symmetry group of P and let F_i, $1 \leq i \leq k$, denote the symmetry group of ω_i. Let ρ denote the affine order type of R. By Corollary 13, we have

$$\mathbb{P}[\rho = \omega_i] = \frac{|\mathsf{G}|/|\mathsf{F}_i|}{N}.$$

Next, the number of distinct labelings of the order type of an affine set A is $n!/|\mathsf{F}|$, since two labelings $A_{[\lambda]}$ and $A_{[\mu]}$ of A have the same labeled order type if and only if $\mu^{-1} \circ \lambda$ is a symmetry of A. Let $\overline{\rho}$ denote the labeled affine order type of $R_{[\lambda]}$. For any $\overline{\sigma} \in \mathsf{LOT}_{\omega_i}^{\text{aff}}$, we have

$$\mathbb{P}[\overline{\rho} = \overline{\sigma} \mid \rho = \omega_i] = \frac{|\mathsf{F}_i|}{n!}.$$

Altogether, for any $\overline{\sigma} \in \bigcup_{i=1}^{k} \mathsf{LOT}_{\omega_i}^{\text{aff}}$, we have $\mathbb{P}[\overline{\rho} = \overline{\sigma}] = \frac{|G|}{Nn!}$ and the distribution is uniform as we claimed. ◀

4.2 Hemisets and duality

The following dualization will make counting easy.

▶ **Lemma 15.** *There is a bijection ϕ between the affine hemisets of a projective point set P and the cells of the dual arrangement P^*, such that a point p is extreme in an affine hemiset A if and only if the great circle p^* supports an edge of $\phi(A)$.*

Proof. For any point p we write p^+ for the hemisphere centered in p, that is the closed hemisphere containing p and bounded by p^*. For any closed hemisphere H we write H^+ for its center, that is the point q with $H = q^+$. Now, a point p is in a closed hemisphere H if and only if the scalar product $\langle p, H^+ \rangle$ is nonnegative. Thus, p lies in H if and only if H^+ lies in p^+. It follows that two hemispheres H_0 and H_1 intersect P in the same hemiset if and only if H_0^+ and H_1^+ lie in the same cell of P^*. Moreover, as H^+ moves in the cell the hemisphere H also moves while enclosing the same set of points; the boundary of H touches a point p if and only if H^+ touches p^*. ◀

For example, we now see that a projective set of $2n$ points, $n \geq 3$, in general position has $2\binom{n}{2}+2$ distinct affine hemisets. Also, it should be clear from the final computations of the proof of Proposition 14 that if that projective point set has symmetry group G, then it supports $\left(2\binom{n}{2}+2\right)\frac{n!}{|G|}$ distinct labeled affine order types.

4.3 Counting extreme points: expectation and variance

We can now prove Theorem 1 on the expectation and variance of the number of extreme points in a random labeled affine order type.

▶ **Lemma 16.** *Let P be a projective set of $2n$ points, $n \geq 3$, in general position. If X_P denotes the number of extreme points in a labeled affine order type chosen uniformly among those supported by P, then*

$$\mathbb{E}[X_P] = \frac{4n(n-1)}{n(n-1)+2} = 4 - \tfrac{8}{n^2-n+2} \qquad \text{and} \qquad \mathbb{E}\left[{X_P}^2\right] \leq \frac{19n(n-1)-10n}{n(n-1)+2} \leq 19.$$

Proof. By Lemma 15, X_P has the same distribution as the number of edges in a cell chosen uniformly at random in P^*. The arrangement P^* has $2\binom{n}{2}+2$ cells and $4\binom{n}{2}$ edges. Since every edge bounds exactly two cells, it comes that

$$\mathbb{E}[X_P] = \frac{8\binom{n}{2}}{2\binom{n}{2}+2} = \frac{4n(n-1)}{n(n-1)+2} = 4 - \tfrac{8}{n^2-n+2}.$$

Moreover, the random variable ${X_P}^2$ has the same distribution as the square of the number of edges in a random cell chosen uniformly in P^*. Let $F_2(P^*)$ denote the set of cells of P^* and for $c \in F_2(P^*)$ let $|c|$ denote its number of edges. We thus have

$$\left(2\binom{n}{2}+2\right)\mathbb{E}\left[{X_P}^2\right] = \sum_{c \in F_2(P^*)} |c|^2.$$

In the right-hand term, every edge e of P^* is counted $|c_1|+|c_2|$ times, where c_1 and c_2 are its two adjacent cells. For any point $p \in P$, the contribution of the edges supported by p^* to that sum equals $\sum_{c \in Z(p^*)} |c| \leq 19(n-1)-10$ (following notation and bound in Theorem 8). Altogether,

$$\left(2\binom{n}{2}+2\right)\mathbb{E}\left[{X_P}^2\right] \leq n(19(n-1)-10)$$

and $\mathbb{E}\left[{X_P}^2\right] \leq \dfrac{19n(n-1)-10n}{n(n-1)+2} \leq 19.$ ◀

Here comes the announced proof.

Proof of Theorem 1. Let $\overline{\rho}$ be a simple labeled order type chosen uniformly at random in $\mathsf{LOT}_n^{\text{aff}}$. Let X_n denote the number of extreme points in ρ, where ρ denotes the unlabeling of $\overline{\rho}$ and let π be the projective completion of ρ. By Lemma 16, we have for any $\pi' \in \mathsf{OT}_n^{\text{proj}}$

$$\mathbb{E}[X_n \mid \pi = \pi'] = \frac{4n(n-1)}{n(n-1)+2} \quad \text{and} \quad \mathbb{E}\left[{X_n}^2 \mid \pi = \pi'\right] \leq \frac{19n(n-1)-10n}{n(n-1)+2}.$$

The formula of total probability therefore yields

$$\mathbb{E}[X_n] = \frac{4n(n-1)}{n(n-1)+2} \quad \text{and} \quad \mathbb{E}\left[{X_n}^2\right] \leq \frac{19n(n-1)-10n}{n(n-1)+2}.$$

From there, $\mathrm{Var}[X_n] = \mathbb{E}\left[{X_n}^2\right] - \mathbb{E}[X_n]^2 \leq 3$. (A bound of $3+o(1)$ is readily seen from $\mathbb{E}[X_n] = 4+o(1)$ and $\mathbb{E}\left[{X_n}^2\right] = 19+o(1)$; the bound of 3 holds exploiting $n \geq 3$.) ◀

As a consequence, we obtain for instance the following estimates.

▶ **Corollary 17.** *The proportion of simple labeled affine n-point order types with $h \geq 6$ vertices on the convex hull is at most $3/(h-4)^2$.*

Proof. By the Bienaymé-Chebyshev inequality, for any real $t > 0$ and any random variable X with finite expected value and non-zero variance, we have

$$\mathbb{P}\left[|X - \mathbb{E}[X]| \geq t\sqrt{\mathrm{Var}[X]}\right] \leq \frac{1}{t^2}.$$

Together with Theorem 1, this implies the statement. ◀

References

1 Oswin Aichholzer, Franz Aurenhammer, and Hannes Krasser. Enumerating order types for small point sets with applications. *Order*, 19(3):265–281, 2002. doi:10.1023/A:1021231927255.

2 Oswin Aichholzer and Hannes Krasser. Abstract order type extension and new results on the rectilinear crossing number. *Computational Geometry*, 36(1):2–15, 2007. Special Issue on the 21st European Workshop on Computational Geometry. doi:10.1016/j.comgeo.2005.07.005.

3 Noga Alon. The number of polytopes, configurations and real matroids. *Mathematika*, 33(1):62–71, 1986.

4 Greg Aloupis, John Iacono, Stefan Langerman, Özgür Özkan, and Stefanie Wuhrer. The complexity of order type isomorphism. In Chandra Chekuri, editor, *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 405–415. SIAM, 2014. doi:10.1137/1.9781611973402.30.

5 Imre Bárány, Matthieu Fradelizi, Xavier Goaoc, Alfredo Hubard, and Günter Rote. Random polytopes and the wet part for arbitrary probability distributions. *arXiv preprint*, 2019. arXiv:1902.06519.

6 Imre Bárány and David G Larman. Convex bodies, economic cap coverings, random polytopes. *Mathematika*, 35(2):274–291, 1988.

7 Yuliy M Baryshnikov and Richard A Vitale. Regular simplices and Gaussian samples. *Discrete & Computational Geometry*, 11(2):141–147, 1994.

8 Marshall W. Bern, David Eppstein, Paul E. Plassmann, and F. Frances Yao. Horizon theorems for lines and polygons. In Jacob E. Goodman, Richard Pollack, and William Steiger, editors, *Discrete and Computational Geometry: Papers from the DIMACS Special Year*, volume 6 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 45–66. DIMACS/AMS, 1990. doi:10.1090/dimacs/006/03.

9 Anders Bjorner, Anders Björner, Michel Las Vergnas, Bernd Sturmfels, Neil White, and Gunter M Ziegler. *Oriented matroids*. Number 46 in Encyclopedia of Mathematics and its Applications. Cambridge University Press, 1999 .

10 Jean Cardinal, Ruy Fabila-Monroy, and Carlos Hidalgo-Toscano. Chirotopes of random points in space are realizable on a small integer grid. *arXiv preprint*, 2020. arXiv:2001.08062.

11 Man-Kwun Chiu, Stefan Felsner, Manfred Scheucher, Patrick Schnider, Raphael Steiner, and Pavel Valtr. On the average complexity of the k-level. *CoRR*, abs/1911.02408, 2019. arXiv:1911.02408.

12 Olivier Devillers, Philippe Duchon, Marc Glisse, and Xavier Goaoc. On order types of random point sets, 2018. arXiv:1812.08525v2.

13 David Eppstein. *Forbidden configurations in discrete geometry*. Cambridge University Press, 2018.

14 Paul Erdős and George Szekeres. A combinatorial problem in geometry. *Compositio mathematica*, 2:463–470, 1935.

15 Ruy Fabila-Monroy and Clemens Huemer. Order types of random point sets can be realized with small integer coordinates. In *XVII Spanish Meeting on Computational Geometry: book of abstracts, Alicante, June 26-28*, pages 73–76, 2017.

16 Tobias Gerken. Empty convex hexagons in planar point sets. *Discrete & Computational Geometry*, 39(1-3):239–272, 2008.

17 Xavier Goaoc and Emo Welzl. Convex hulls of random order types. *CoRR*, abs/2003.08456, 2020. `arXiv:2arXiv:2003.08456`.

18 Jacob E Goodman and Richard Pollack. Multidimensional sorting. *SIAM Journal on Computing*, 12(3):484–507, 1983.

19 Jacob E Goodman and Richard Pollack. Upper bounds for configurations and polytopes inr d. *Discrete & Computational Geometry*, 1(3):219–227, 1986.

20 Jacob E Goodman, Richard Pollack, and Bernd Sturmfels. The intrinsic spread of a configuration in $\mathbb{R}^d$. *Journal of the American Mathematical Society*, pages 639–651, 1990.

21 Jie Han, Yoshiharu Kohayakawa, Marcelo T Sales, and Henrique Stagni. Extremal and probabilistic results for order types. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 426–435. SIAM, 2019.

22 Sariel Har-Peled. On the expected complexity of random convex hulls. *CoRR*, abs/1111.5340, 2011. `arXiv:1111.5340`.

23 Gyula Károlyi and Jozsef Solymosi. Erdős–Szekeres theorem with forbidden order types. *Journal of Combinatorial Theory, Series A*, 113(3):455–465, 2006.

24 Gyula Károlyi and Géza Tóth. Erdős–Szekeres theorem for point sets with forbidden subconfigurations. *Discrete & Computational Geometry*, 48(2):441–452, 2012.

25 Lutz Kettner, Kurt Mehlhorn, Sylvain Pion, Stefan Schirra, and Chee Yap. Classroom examples of robustness problems in geometric computations. *Computational Geometry*, 40(1):61–78, 2008.

26 Donald Ervin Knuth. *Axioms and hulls*, volume 606. Springer, 1992.

27 Adam Marcus and Gábor Tardos. Excluded permutation matrices and the Stanley–Wilf conjecture. *Journal of Combinatorial Theory, Series A*, 107(1):153–160, 2004.

28 Hiroyuki Miyata. On symmetry groups of oriented matroids, 2013. `arXiv:1301.6451`.

29 Nicolai E. Mnëv. The universality theorem on the oriented matroid stratification of the space of real matrices. In Jacob E. Goodman, Richard Pollack, and William Steiger, editors, *Discrete and Computational Geometry: Papers from the DIMACS Special Year*, volume 6 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 237–244. DIMACS/AMS, 1990. `doi:10.1090/dimacs/006/16`.

30 Jaroslav Nešetřil and Pavel Valtr. A Ramsey property of order types. *Journal of Combinatorial Theory, Series A*, 81(1):88–107, 1998.

31 Alfredo García Olaverri, Marc Noy, and Javier Tejel. Lower bounds on the number of crossing-free subgraphs of K_N. *Comput. Geom.*, 16(4):211–221, 2000. `doi:10.1016/S0925-7721(00)00010-9`.

32 Mark Overmars. Finding sets of points without empty convex 6-gons. *Discrete & Computational Geometry*, 29(1):153–158, 2002.

33 M. Reitzner. Random polytopes. In Wilfrid S. Kendall and Ilya Molchanov, editors, *New Perspectives in Stochastic Geometry*, chapter 2, pages 45–75. Oxford University Press, 2009.

34 Alfréd Rényi and Rolf Sulanke. Über die konvexe Hülle von n zufällig gewählten Punkten. *Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete*, 2(1):75–84, 1963.

35 Alfréd Rényi and Rolf Sulanke. Über die konvexe Hülle von n zufällig gewählten Punkten. II. *Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete*, 3(2):138–147, 1964.

36 Marcus Schaefer. Complexity of some geometric and topological problems. In David Eppstein and Emden R. Gansner, editors, *Graph Drawing, 17th International Symposium, GD 2009, Chicago, IL, USA, September 22-25, 2009. Revised Papers*, volume 5849 of *Lecture Notes in Computer Science*, pages 334–344. Springer, 2009. `doi:10.1007/978-3-642-11805-0_32`.

37 Peter Shor. Stretchability of pseudolines is NP-hard. *Applied Geometry and Discrete Mathematics-The Victor Klee Festschrift*, 1991.

38 The CGAL Project. *CGAL User and Reference Manual*. CGAL Editorial Board, 4.14 edition, 2019. URL: `https://doc.cgal.org/4.14/Manual/packages.html`.

39 Ivor van der Hoog, Tillmann Miltzow, and Martijn van Schaik. Smoothed analysis of order types. *arXiv preprint*, 2019. `arXiv:1907.04645`.

Fast Algorithms for Geometric Consensuses

Sariel Har-Peled
Dept. of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL, USA
sariel@illinois.edu

Mitchell Jones
Dept. of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL, USA
mfjones2@illinois.edu

Abstract

Let P be a set of n points in $\mathbb{R}^d$ in general position. A median hyperplane (roughly) splits the point set P in half. The *yolk* of P is the ball of smallest radius intersecting all median hyperplanes of P. The *egg* of P is the ball of smallest radius intersecting all hyperplanes which contain exactly d points of P.

We present exact algorithms for computing the yolk and the egg of a point set, both running in expected time $O(n^{d-1} \log n)$. The running time of the new algorithm is a polynomial time improvement over existing algorithms. We also present algorithms for several related problems, such as computing the Tukey and center balls of a point set, among others.

2012 ACM Subject Classification Theory of computation → Computational geometry

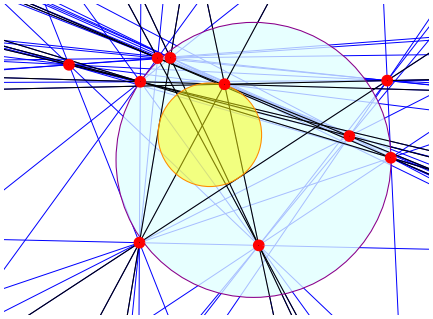
Keywords and phrases Geometric optimization, centerpoint, voting games

Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.50

Related Version A full version of this paper is available at `https://arxiv.org/abs/1912.01639`.

Funding *Sariel Har-Peled*: Supported in part by NSF AF award CCF-1907400.
Mitchell Jones: Supported in part by NSF AF award CCF-1907400.

Acknowledgements The authors thank Joachim Gudmundsson for bringing the problem of computing the yolk to our attention. The second author thanks Sampson Wong for discussions on computing the yolk in higher dimensions. We also thank Timothy Chan for useful comments (in particular, the improved algorithm for the yolk in 3D, see Remark 26).



1 Introduction

Voting games and the yolk. Suppose there is a collection of n voters in $\mathbb{R}^d$, where each dimension represents a specific ideology. In a fixed dimension, each voter maintains a value along this continuum representing their stance on a given ideology. One can interpret $\mathbb{R}^d$ as a *policy space*, and each point in $\mathbb{R}^d$ represents a single policy. In the Euclidean spatial model, a voter $p \in \mathbb{R}^d$ always prefers policies which are closer to p under the Euclidean norm. For two policies $x, y \in \mathbb{R}^d$ and a set of voters $P \subset \mathbb{R}^d$, x *beats* y if more voters in P prefer policy x compared to y. A plurality point is a policy which beats all other policies in $\mathbb{R}^d$. For $d = 1$, the plurality point is the median voter (when n is odd) [3]. However for $d > 1$,

a plurality point is not always guaranteed to exist [18]. It is known that one can test if a plurality point exists (and if so, compute it) in $O(dn \log n)$ time [10]. Note that the plurality point is a point of Tukey depth $\lceil n/2 \rceil$ – in general this is the largest possible Tukey depth any point can have; while the centerpoint is a point that guarantees a "respectable" minority of size at least $n/(d+1)$.

Since plurality points may not always exist, one generalization of a plurality point is the yolk [17]. A hyperplane is a **median hyperplane** if the number of voters lying in each of the two closed halfspaces is at least $\lceil n/2 \rceil$. The **yolk** is the ball of smallest radius intersecting all such median hyperplanes. Note that when a plurality point exists, the yolk has radius zero (equivalently, all median hyperplanes intersect at a common point).

We also consider the following restricted problem. A hyperplane is **extremal** if and only if it passes through d points, under the assumption that the points are in general position. The **extremal yolk** is the ball of smallest radius intersecting all extremal median hyperplanes. Importantly, the yolk and the extremal yolk are different problems – the radius of the yolk and extremal yolk can differ [21].

The egg of a point set. A problem related to computing the yolk is the following: For a set of n points P in $\mathbb{R}^d$, compute the smallest radius ball intersecting all extremal hyperplanes of P (i.e., all hyperplanes passing through d points of P). Such a ball is the **egg** of P. See Figure 1.1 for an illustration of the yolk and egg of a point set.

Linear programs with many implicit constraints. The problem of computing the egg can be written as a linear program (LP) with $\Theta(n^d)$ constraints, defined implicitly by the point set P. One can apply Seidel's algorithm [19] (or any other linear time LP solver in constant dimension) to obtain an $O(n^d)$ expected time algorithm for computing the egg (or the yolk, with a bit more work). However, as each d-tuple of points forms a constraint, it is natural to ask if one can obtain a faster algorithm in this setting. Specifically, we are interested in the following problem: Let I be an instance of a d-dimensional LP specified via a set of n entities P, where each k-tuple of P induces a linear constraint in I, for some (constant) integer k. The problem is to efficiently solve I, assuming access to some additional subroutines.

1.1 Previous work

The yolk. Let P be a set of n points in $\mathbb{R}^d$. Both the yolk and extremal yolk have been studied in the literature. The first polynomial time exact algorithm for computing the yolk in $\mathbb{R}^d$ was by Tovey in $O\big(n^{(d+1)^2}\big)$ time – in the plane, the running time can be improved to $O(n^4)$ [22]. Following Tovey, the majority of results have focused on computing the yolk in the plane. In 2018, de Berg et al. [10] gave an $O(n^{4/3} \log^{1+\varepsilon} n)$ time algorithm (for any fixed $\varepsilon > 0$) for computing the yolk. Obtaining a faster exact algorithm remained an open problem. Gudmundsson and Wong [12, 13] presented a $(1+\varepsilon)$-approximation algorithm with $O(n \log^7 n \log^4 \varepsilon^{-1})$ running time. An unpublished result of de Berg et al. [8] achieves a randomized $(1+\varepsilon)$-approximation algorithm for the extremal yolk running in expected time $O(n\varepsilon^{-3} \log^3 n)$.

The egg. The egg of a point set in $\mathbb{R}^d$ can be computed by solving a linear program with $\Theta(n^d)$ constraints. The egg is a natural extension to computing the yolk, and thus obtaining faster exact algorithms is of interest. The authors are not aware of any previous work on this specific problem. Bhattacharya et al. [2] gave an algorithm which computes the smallest radius ball intersecting a set of m hyperplanes in $O(m)$ time, when $d = O(1)$, by formulating the problem as an LP (see also Lemma 11). However we emphasize that in our problem the set of hyperplanes are implicitly defined by the point set P, and is of size $\Theta(n^d)$ in $\mathbb{R}^d$.

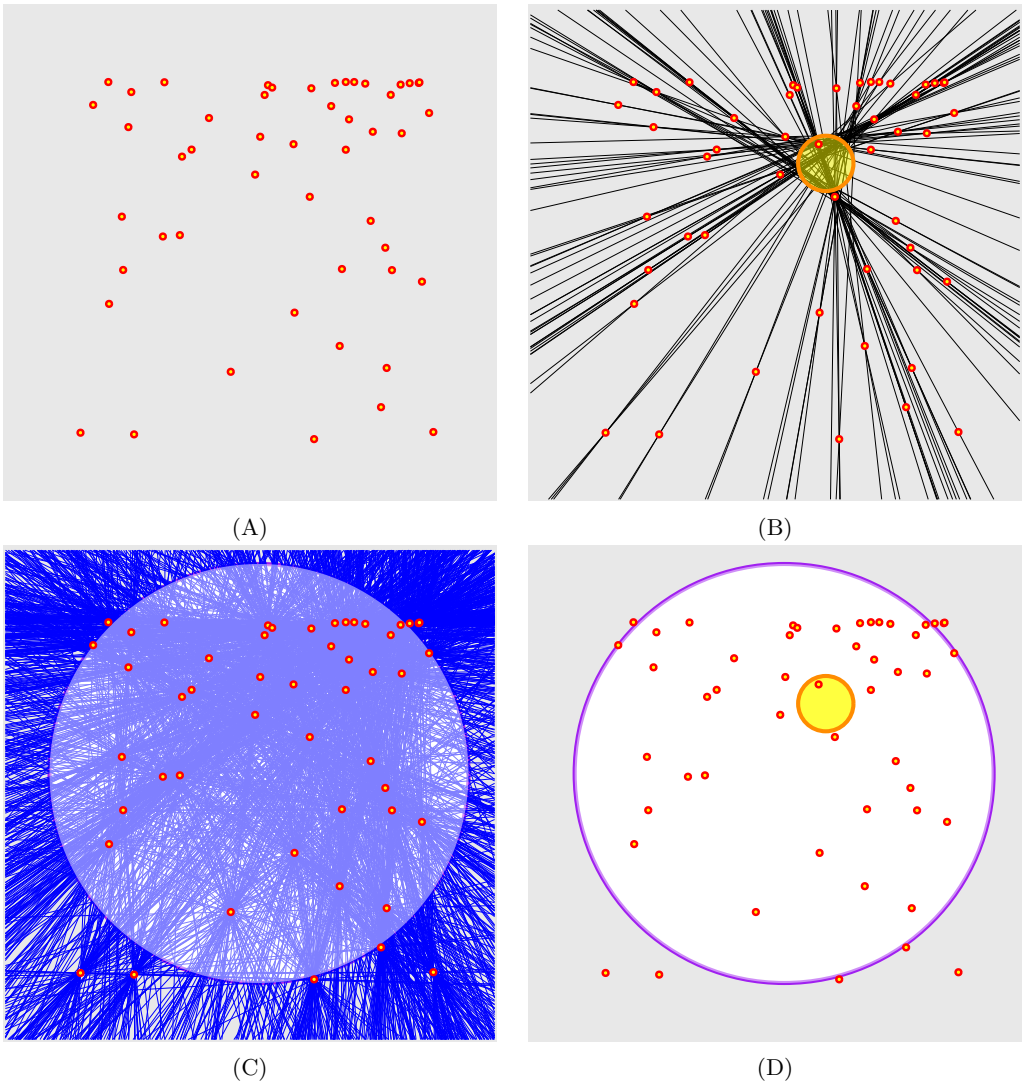


Figure 1.1 (A) Points. (B) Median lines and the extremal yolk. (C) All lines and the egg. (D) Points with the extremal yolk and the egg.

Table 1.1 Some previous work on the yolk and our results. Existing algorithms are deterministic, while the running time of our algorithms holds in expectation.

$d = 2$	$(1+\varepsilon)$-approx	Exact	Our results (Exact)
Extremal yolk	$O(n\varepsilon^{-3}\log^3 n)$ [8]	$O(n^{4/3}\log^{1+\epsilon} n)$ [10]	$O(n\log n)$ Theorem 16
Yolk	$O(n\log^7 n\log^4 \varepsilon^{-1})$ [13]	$O(n^{4/3}\log^{1+\epsilon} n)$ Variant of [10]	$O(n\log n)$ Theorem 25
$d = 3$			
Yolk	?	$O(n^3)$ Known techniques	$O(n^2)$ Remark 26
$d > 3$			
Extremal yolk	?	$O(n^d)$ Known techniques	$O(n^{d-1}\log n)$ Theorem 16
Yolk	?	$O(n^d)$ Known techniques	$O(n^{d-1}\log n)$ Theorem 25

Implicit LPs. In 2004, Chan [4] developed a framework for solving LPs with many implicit constraints (the motivation was to obtain an efficient algorithm for computing the Tukey depth of a point set). Informally, suppose that each input set P of entities maps to a set $\mathcal{H}(P)$ of implicit constraints. For n entities P and a candidate solution, suppose one can decide if the candidate solution violates any constraints of $\mathcal{H}(P)$ in $D(n)$ time. Additionally, assume that from P, one can construct $r = O(1)$ sets $P_1, \ldots, P_r$, each of size at most n/c (for some constant $c > 1$) with $\mathcal{H}(P) = \bigcup_{i=1}^{r} \mathcal{H}(P_i)$. If this partition step can be performed in $D(n)$ time, then both assumptions imply that the resulting LP can be solved in $O(D(n))$ expected time.

1.2 Our results

Note. Due to space limitations, not all results discussed below are presented in the paper. We refer the reader to the full version of the paper on arXiv [14].

In this paper we revisit Chan's algorithm for solving LPs with many implicitly defined constraints [4]. The technique leads to efficient algorithms for the following problems. Throughout, let $P \subset \mathbb{R}^d$ be a set of n points in general position:

(A) The yolk (and extremal yolk) of P can be computed *exactly* in $O(n^{d-1}\log n)$ expected time. Hence in the plane, the yolk can be computed *exactly* in $O(n\log n)$ expected time. This improves all existing algorithms (both exact and approximate) [22, 10, 13, 12, 8] for computing the yolk in the plane, and our algorithm easily generalizes to higher dimensions. See Table 1.1 for a summary of our results and previous work.

(B) By a straight-forward modification of the above algorithm, see Lemma 17, implies that the egg of P can be computed in $O(n^{d-1}\log n)$ expected time. The authors are not aware of any previous work on this specific problem.

(C) Let $H_k(P)$ be the collection of all open halfspaces which contain at least $n-k$ points of P. Consider the convex polygon $\mathcal{T}_k = \cap_{h\in H_k(P)} h$. Observe that $\mathcal{T}_0$ is the convex hull of P, with $\mathcal{T}_0 \supseteq \mathcal{T}_1 \supseteq \cdots$. The centerpoint theorem implies that $\mathcal{T}_{n/(d+1)}$ is non-empty (and contains the centerpoint). The Tukey depth of a point q in the minimal k such that $q \in \mathcal{T}_k \setminus \mathcal{T}_{k+1}$.

When $\mathcal{T}_k$ is non-empty, the **center ball** of P is the ball of largest radius contained inside $\mathcal{T}_k$. For $\mathcal{T}_k$ empty, we define the **Tukey ball** of P as the smallest radius ball intersecting all halfspaces of $H_k(P)$.
In the full version of the paper [14] we show that the Tukey ball and center ball can both be computed in $\widetilde{O}\big(k^{d-1}\big[1+(n/k)^{\lfloor d/2\rfloor}\big]\big)$ expected time. Here, $\widetilde{O}$ hides polylogarithmic factors in n. In particular when k is a (small) constant, a point of Tukey depth k can be computed in time $\widetilde{O}(n^{\lfloor d/2\rfloor})$. This improves Chan's $O(n^{d-1}\log n)$ expected time algorithm for deciding if there is a point of Tukey depth at least k [4].

(D) For a set $Q \subseteq \mathbb{R}^d$, let $\mathsf{conv}(Q)$ denote the convex hull of Q. For a given integer k let $\mathcal{C}(P,k) = \left\{\mathsf{conv}(Q) \;\middle|\; Q \in \binom{P}{k}\right\}$, where $\binom{P}{k}$ is the set of all k-tuples of points of P. We define the ***k*-ball** of P as the smallest radius ball intersecting all convex bodies in $\mathcal{C}(P,k)$.
While one may be tempted to apply the techniques discussed so far for implicit LPs, there is a faster algorithm using $(\leq k)$-sets. When k is constant, in [14] we present an algorithm for computing the k-ball in $O(n^{\lfloor d/2\rfloor}+n\log n)$ expected time. As such, the smallest ball intersecting all triangles induced by triples of a set of n points in $\mathbb{R}^3$ can be computed in $O(n\log n)$ expected time.

In [14], we present another application of Chan's technique for solving implicit LP-type problems.

(E) Given a set L of n lines in the plane, the **crossing distance** between two points $p, q \in \mathbb{R}^2$ is the number of lines of L intersecting the segment pq. Given a point $q \in \mathbb{R}^2$ not lying on any lines of L, the disk of smallest radius containing all vertices of $\mathcal{A}(L)$, within crossing distance at most k from q, can be computed, in $O(n\log n)$ expected time.

2 Preliminaries

▶ Notation. Throughout, the O hides factors which depend (usually exponentially) on the dimension d. Additionally, the $\widetilde{O}$ notation hides factors of the form $\log^c n$, where c may depend on d.

2.1 LP-type problems

An LP-type problem, introduced by Sharir and Welzl [20], is a generalization of a linear program. Let $\mathcal{H}$ be a set of constrains and f be an objective function. For any $\mathcal{B} \subseteq \mathcal{H}$, let $f(\mathcal{B})$ denote the value of the optimal solution for the constraints of $\mathcal{B}$. The goal is to compute $f(\mathcal{H})$. If the problem is infeasible, let $f(\mathcal{H}) = \infty$. Similarly, define $f(\mathcal{H}) = -\infty$ if the problem is unbounded.

▶ **Definition 1.** *Let $\mathcal{H}$ be a set of constraints, and let $f : 2^{\mathcal{H}} \to \mathbb{R} \cup \{\infty, -\infty\}$ be an objective function. The tuple $(\mathcal{H}, f)$ forms an **LP-type problem** if the following properties hold:*

(A) *Monotonicity. For any $\mathcal{B} \subseteq \mathcal{C} \subseteq \mathcal{H}$, we have $f(\mathcal{B}) \leq f(\mathcal{C})$.*

(B) *Locality. For any $\mathcal{B} \subseteq \mathcal{C} \subseteq \mathcal{H}$ with $f(\mathcal{C}) = f(\mathcal{B}) > -\infty$, and for all $s \in \mathcal{H}$, $f(\mathcal{C}) < f(\mathcal{C}+s) \iff f(\mathcal{B}) < f(\mathcal{B}+s)$, where $\mathcal{B}+s = \mathcal{B}\cup\{s\}$.*

A **basis** of $\mathcal{H}$ is an inclusion-wise minimal subset $\mathcal{b} \subseteq \mathcal{H}$ with $f(\mathcal{b}) = f(\mathcal{H})$. The **combinatorial dimension** δ is the maximum size of any feasible basis of any subset of $\mathcal{H}$. Throughout, we consider δ to be constant. For a basis $\mathcal{b} \subseteq \mathcal{H}$, we say that $h \in \mathcal{H}$ **violates** the current solution induced by $\mathcal{b}$ if $f(\mathcal{b}+h) > f(\mathcal{b})$. LP-type problems with n constraints can be solved in randomized time $O(n)$, hiding constants depending (exponentially) on δ [7], where the bound on the running time holds with high probability.

2.2 Implicit LPs using Chan's algorithm

Our algorithms will need the following result of Chan [4] on solving LPs with implicitly defined constraints.

▶ **Lemma 2** ([4]). *Let $(\mathcal{H}, f)$ be an LP-type problem of constant combinatorial dimension δ, and let c_δ be a constant that depends only on δ. Let $\psi, c > 1$ be fixed constants, such that $c_\delta \log^\delta \psi < c$. For an input space Π, suppose that there is a function $g : \Pi \to 2^{\mathcal{H}}$ which maps inputs to constraints. Furthermore, assume that for any input $P \in \Pi$ of size n, we have:*

- **(I)** *When $n = O(1)$, a basis for $g(P)$ can be computed in constant time.*
- **(II)** *For a basis $\mathcal{b}$, one can decide if $\mathcal{b}$ satisfies $g(P)$ in $D(n)$ time.*
- **(III)** *In $D(n)$ time, one can construct sets $P_1, \ldots, P_\psi \in \Pi$, each of size at most n/c, such that $g(P) = \bigcup_{i=1}^{\psi} g(P_i)$.*

Then a basis for $g(P)$ can be computed in $O(D(n))$ expected time, assuming that $D(n/k) = O(D(n)/k)$, for all positive integers $k \leq n$.

2.3 Duality, levels, and zones

2.3.1 Duality

▶ **Definition 3** (Duality). *The **dual hyperplane** of a point $p = (p_1, \ldots, p_d) \in \mathbb{R}^d$ is the hyperplane $p^\star$ defined by the equation $x_d = -p_d + \sum_{i=1}^{d-1} x_i p_i$. The **dual point** of a hyperplane h defined by $x_d = a_d + \sum_{i=1}^{d-1} a_i x_i$ is the point $h^\star = (a_1, a_2, \ldots, a_{d-1}, -a_d)$.*

▶ Fact 4. Let p be a point and let h be a hyperplane. Then p lies above h if and only if the hyperplane $p^\star$ lies below the point $h^\star$.

Given a set of objects T (e.g., points in $\mathbb{R}^d$), we let $T^\star = \{x^\star \mid x \in T\}$ denote the dual set of objects.

2.3.2 k-Levels

▶ **Definition 5** (Levels). *For a collection of hyperplanes H in $\mathbb{R}^d$, the **level** of a point $p \in \mathbb{R}^d$ is the number of hyperplanes of H lying on or below p. The k**-level** of H is the union of points in $\mathbb{R}^d$ which have level equal to k. The $(\leq k)$**-level** of H is the union of points in $\mathbb{R}^d$ which have level at most k.*

By Fact 4, if h is a hyperplane which contains k points of P lying on or above it, then the dual point $h^\star$ is a member of the k-level of $P^\star$.

2.3.3 Zones of surfaces

For a set of hyperplanes H, we let $\mathcal{A}(H)$ denote the arrangement of H and $\mathsf{V}(\mathcal{A}(H))$ denote the vertices of the arrangement of H.

▶ **Definition 6** (Zone of a surface). *For a collection of hyperplanes H in $\mathbb{R}^d$, the complexity of a cell ψ in the arrangement $\mathcal{A}(H)$ is the number of faces (of all dimensions) which are contained in the closure of ψ. For a $(d-1)$-dimensional surface γ, the **zone** $\mathcal{Z}(\gamma, H)$ of γ is the subset of cells of $\mathcal{A}(H)$ which intersect γ. The **complexity of a zone** is the sum of the complexities of the cells in $\mathcal{Z}(\gamma, H)$.*

The complexity of a zone of a hyperplane is known to be $\Theta(n^{d-1})$ [11]; for general algebraic surfaces it is larger by a logarithmic factor. Furthermore, the cells in the zone of a surface can be computed efficiently using lazy randomized incremental construction [9].

▶ **Lemma 7** ([1, 9]). *Let H be a set of n hyperplanes in $\mathbb{R}^d$ and let γ be a $(d-1)$-dimensional algebraic surface of degree δ. The complexity of the zone $\mathcal{Z}(\gamma, H)$ is $O(n^{d-1}\log n)$, where the hidden constants depend on d and δ. The collection of cells in $\mathcal{Z}(\gamma, H)$ can be computed in $O(n^{d-1}\log n)$ expected time.*

3 Computing the extremal yolk

3.1 Background

▶ **Definition 8.** *Let $P \subset \mathbb{R}^d$ be a set of n points in general position. A median hyperplane is a hyperplane such that each of its two closed halfspaces contain at least $\lceil n/2 \rceil$ points of P. A hyperplane is extremal if it passes through d points of P. The **extremal yolk** is the ball of smallest radius interesting all extremal median hyperplanes of P.*

We give an $O(n^{d-1}\log n)$ expected time exact algorithm computing the extremal yolk. To do so, we focus on the more general problem.

▶ Problem 9. Let $E_k(P)$ be the collection of extremal hyperplanes which contain exactly k points of P on or above it. Here, k is not necessarily constant. The goal is to compute the smallest radius ball intersecting all hyperplanes of $E_k(P)$.

We observe that computing the extremal yolk can be reduced to the above problem.

▶ **Lemma 10.** *The problem of computing the extremal yolk can be reduced to Problem 9.*

Proof. Suppose that n is even, and define the set $S_{\text{even}} = \{n/2, n/2+1, \ldots, n/2+d\}$. A case analysis shows that any extremal median hyperplane h must have exactly m points of P above or on h, where $m \in S_{\text{even}}$. Thus, computing the extremal yolk reduces to computing smallest radius ball intersecting all hyperplanes in the set $\bigcup_{m \in S_{\text{even}}} E_m(P)$.

When n is odd, a similar case analysis shows that any extremal median hyperplane must have exactly m points above or on it, where $m \in S_{\text{odd}} = \{\lceil n/2 \rceil, \lceil n/2 \rceil + 1, \ldots, \lceil n/2 \rceil + d - 1\}$. Analogously, computing the extremal yolk with n odd reduces to computing the smallest radius ball intersecting all hyperplanes in the set $\bigcup_{m \in S_{\text{odd}}} E_m(P)$. ◀

To solve Problem 9, we apply Chan's result for solving implicit LP-type problems [4], stated in Lemma 2. We first prove that Problem 9 is an LP-type problem when the constraints are explicitly given (the following Lemma was also observed by Bhattacharya et al. [2]).

▶ **Lemma 11.** *Problem 9 when the constraints (i.e., hyperplanes) are explicitly given, is an LP-type problem and has combinatorial dimension $\delta = d+1$.*

Proof. We prove something stronger, namely that the problem can be written as a linear program, implying it is an LP-type problem. Let $\mathcal{H}$ be the set of n hyperplanes. For each hyperplane $h \in \mathcal{H}$, let $\langle a_h, x \rangle + b_h = 0$ be the equation describing h, where $a_h \in \mathbb{R}^d$, $\|a_h\| = 1$, and $b_h \in \mathbb{R}$. Because of the requirement that $\|a_h\| = 1$, for a given point $p \in \mathbb{R}^d$, the distance from p to a hyperplane h is $|\langle a_h, p \rangle + b_h|$.

The linear program has $d+1$ variables and $2n$ constraints. The $d+1$ variables represent the center $p \in \mathbb{R}^d$ and radius $\nu \geq 0$ of the egg. The resulting LP is

$$\begin{aligned} \min \quad & \nu \\ \text{subject to} \quad & \nu \geq \langle a_h, p \rangle + b_h && \forall h \in \mathcal{H} \\ & \nu \geq -\big(\langle a_h, p \rangle + b_h\big) && \forall h \in \mathcal{H} \\ & p \in \mathbb{R}^d. \end{aligned}$$

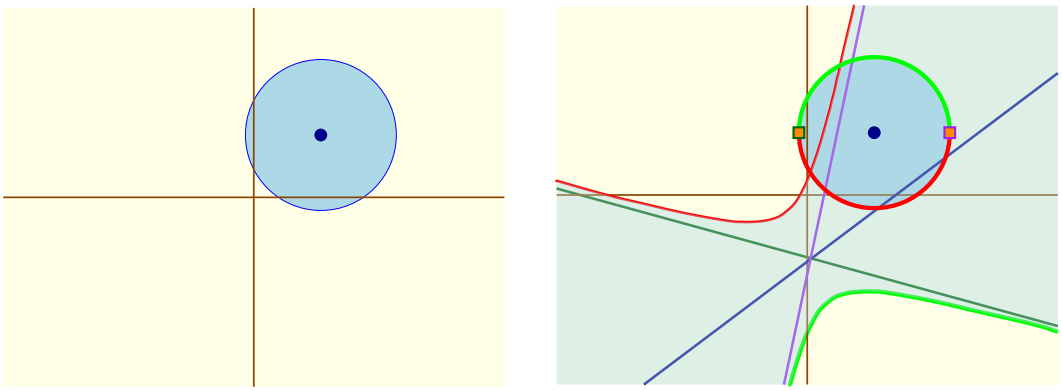


Figure 3.1 A disk and its dual.

As for the combinatorial dimension, observe that any basic feasible solution for the above linear program will be tight for at most $d+1$ of the above $2n$ constraints. Namely, these $d+1$ planes are tangent to the optimal radius ball, and as such form a basis $\mathcal{b} \subseteq \mathcal{H}$. ◀

To apply Lemma 2 we need to:

(i) design an appropriate input space,

(ii) develop a decider, and

(iii) construct a constant number of subproblems which cover the constraint space.

3.2 Building the decider

The algorithm will work in the dual space. In the dual, the interior of a ball b corresponds to a closed region $\mathsf{b}^\star$ which lies between two branches of a hyperboloid, see Figure 3.1.

▶ **Lemma 12.** *The dual of the set of points in a ball is the set of hyperplanes whose union forms the region enclosed between two branches of a hyperboloid.*

Proof. In $\mathbb{R}^d$ the hyperplane h defined by $x_d = \beta + \sum_{i=1}^{d-1} \alpha_i x_i$, or more compactly $\langle x, (-\alpha, 1)\rangle = \beta$, intersects a disk b centered at $p = (p_1, \ldots, p_d)$ with radius $r \iff$ the distance of h from p is at most r. That is, h intersects b if

$$\begin{aligned}
\frac{|\langle p, (-\alpha, 1)\rangle - \beta|}{\|(-\alpha, 1)\|} \leq r &\iff (\langle p, (-\alpha, 1)\rangle - \beta)^2 \leq r^2 \|(-\alpha, 1)\|^2 \\
&\iff \left(p_d - \beta - \sum_{i=1}^{d-1} \alpha_i p_i\right)^2 \leq r^2(\|\alpha\|^2 + 1). \\
&\iff \frac{\left(p_d - \beta - \sum_{i=1}^{d-1} \alpha_i p_i\right)^2}{r^2} - \|\alpha\|^2 \leq 1.
\end{aligned}$$

The boundary of the above inequality is a hyperboloid in the variables $p_d - \beta - \sum_{i=1}^{d-1} \alpha_i p_i$ and $\alpha_1, \ldots, \alpha_{d-1}$. This corresponds to an affine image of a hyperboloid in the dual space $\alpha \times -\beta$. ◀

Throughout, we let $\mathsf{b}^\star$ denote the region between the two branches of the hyperboloid dual to a ball b.

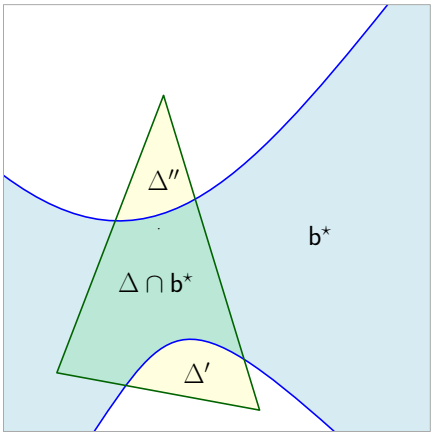


Figure 3.2 The region $\Delta \cap (\mathbb{R}^d \setminus \mathsf{b}^\star)$ consists of (at most) two disjoint convex regions, Δ' and Δ''.

3.2.1 Algorithm

Given a candidate solution (i.e., a ball b in the primal) and a collection of points $Q \subseteq P$. Our goal is to construct a decider which detects if there is a hyperplane of $E_k(P)$, passing through d points of Q, which avoids the interior of the ball b. In the dual setting, the problem is to decide if there is a vertex of $\mathcal{A}(Q^\star)$ which is a member of the k-level, and is inside the region $\mathbb{R}^d \setminus \mathsf{b}^\star$.

The input. The input to the algorithm is a simplex Δ, the set of hyperplanes

$$H = P^\star \cap \Delta = \{h \in P^\star \mid h \cap \Delta \neq \emptyset\}$$

(i.e., all hyperplanes of $P^\star$ that intersect Δ), a candidate solution $\mathsf{b}^\star$, and a parameter u which is the number of hyperplanes of $P^\star$ lying completely below Δ.

The task. Decide if there is a vertex of $\mathcal{A}(P^\star)$ of the k-level in $\Delta \cap (\mathbb{R}^d \setminus \mathsf{b}^\star)$. That is, there is a vertex of level k that is outside $\mathsf{b}^\star$ but inside Δ.

The decision procedure. Consider the set $\Delta \cap (\mathbb{R}^d \setminus \mathsf{b}^\star)$, where Δ is a simplex, and notice that the set is the union of at most two convex regions. Indeed, the set $\mathbb{R}^d \setminus \mathsf{b}^\star$ consists of two disjoint connected components, where each component is a convex body. Intersecting a simplex Δ with each component of $\mathbb{R}^d \setminus \mathsf{b}^\star$ produces two (disjoint) convex bodies Δ' and Δ'' (it is possible that Δ' or Δ'' are empty). See Figure 3.2. Let Δ' be one of these two regions of interest. The algorithm will process Δ'' in exactly the same way.

If Δ' is empty, then no constraints are violated. Otherwise, we need to check for any violated constraints inside Δ'. Let $\partial\Delta'$ denote the boundary of Δ'. Define $H' \subseteq H$ to be the subset of hyperplanes intersecting Δ'. Observe that it suffices to check if there is a vertex v in the arrangement $\mathcal{A}(H')$ such that:

(i) v has level k in $P^\star$,
(ii) v is a member of some cell in the zone $\mathcal{Z}(\partial\Delta', H')$, and
(iii) v is contained in Δ'.

The algorithm computes $\mathcal{Z}(\partial\Delta', H')$. Next, it chooses a vertex v of the arrangement $\mathcal{A}(H')$ which lies inside Δ' and computes its level in H' (adding u to the count). The algorithm then walks around the vertices of the zone *inside* Δ', computing the level of

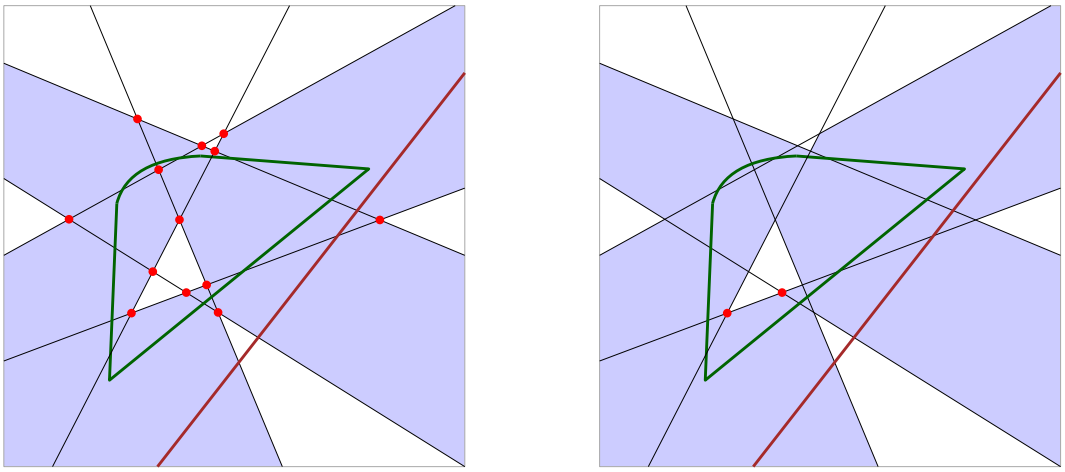


Figure 3.3 Left: A convex region Δ'. Let H' be the set of lines intersecting Δ', with one line lying completely below Δ' ($\mathsf{u} = 1$). The shaded regions are the cells of $\mathcal{A}(H')$ intersecting $\partial\Delta'$. The vertices of the cells in the zone $\mathcal{Z}(\partial\Delta', H')$ are highlighted. Right: The vertices of $\mathcal{Z}(\partial\Delta', H')$ which are part of the 3-level and contained inside Δ'.

each vertex along the walk. Note that the level between any two adjacent vertices in the arrangement differ by at most a constant (depending on d). If at any point we find a vertex of the desired level (such a vertex also lies inside Δ'), we report the corresponding median hyperplane which violates the given ball b. See Figure 3.3 for an illustration.

3.2.2 Analysis

The running time of the algorithm is proportional to the complexity of the zone $\mathcal{Z}(\partial\Delta', H')$. Because the boundary of Δ' is constructed from $d+1$ hyperplanes and the boundary of the hyperboloid, Lemma 7 implies that the zone complexity is no more than $O(|H|^{d-1}\log|H|)$. As such, our decision procedure runs in time $D(n) = O(n^{d-1}\log n)$.

3.3 Constructing subproblems

To decompose a given input into smaller subproblems, we need the notion of cuttings.

▶ **Definition 13** (Cuttings). *Given n hyperplanes in $\mathbb{R}^d$, a $(1/c)$-**cutting** is a collection of interior disjoint simplices covering $\mathbb{R}^d$, such that each simplex intersects at most n/c hyperplanes. A $(1/c)$-cutting of size $O(c^d)$ can be constructed in $O(nc^{d-1})$ time [6].*

Given a simplex Δ and the set of hyperplanes $H = P^\star \cap \Delta$, we compute a $(1/c)$-cutting of H into $O(c^d)$ simplices, and clip this cutting inside Δ. For each cell in this new cutting, we compute the set of hyperplanes which intersect it, and the number of hyperplanes lying completely below the cell naively in $O(|H|)$ time. Repeating this process for the $O(c^d)$ cells implies that this decomposition procedure can be completed in $O(|H|)$ time (ignoring dependencies on d), as $(1/c)$-cuttings can be constructed in deterministic linear time for constant c [6].

The above shows that we can decompose a given input of size n into $\psi = O(c^d)$ subproblems, each of size at most n/c. Furthermore, this decomposition preserves all implicit constraints of interest (vertices of $\mathcal{A}(H)$). Choosing c to be a sufficiently large constant (possibly depending on d), to meet the requirements of Lemma 2, finishes the construction.

3.4 Putting it all together

The above discussions together with Lemma 2 and $D(n) = O(n^{d-1} \log n)$ implies the following.

▶ **Lemma 14.** *Let $P \subset \mathbb{R}^d$ be a set of n points in general position. For a given integer k, one can compute in $O(n^{d-1} \log n)$ expected time the smallest radius ball intersecting all of the hyperplanes of $E_k(P)$*

▶ Notation. For an integer $n > 0$, let $[\![n]\!] = \{1, \ldots, n\}$.

▶ **Corollary 15.** *Let $P \subset \mathbb{R}^d$ be a set of n points in general position, and let $S \subseteq [\![n]\!]$. One can compute in $O(n^{d-1} \log n)$ expected time the smallest radius ball intersecting all of the hyperplanes of $\bigcup_{k \in S} E_k(P)$*

Proof. The algorithm is a slight modification of Lemma 14. During the decision procedure, for each vertex in the zone, we check if it is a member of the k-level for some $k \in S$. If S is of non-constant size, membership in S can be checked in constant time using hashing. ◀

3.5 Computing the extremal yolk and the egg

▶ **Theorem 16.** *Let $P \subset \mathbb{R}^d$ be a set of n points in general position. One can compute the extremal yolk of P in $O(n^{d-1} \log n)$ expected time.*

Proof. The result follows by applying Corollary 15 with the appropriate choice of S. When n is even, Lemma 10 tells us to choose $S = \{n/2, n/2 + 1, \ldots, n/2 + d\}$. When n is odd, we set $S = \{\lceil n/2 \rceil, \lceil n/2 \rceil + 1, \ldots, \lceil n/2 \rceil + d - 1\}$. ◀

▶ **Lemma 17.** *Let $P \subset \mathbb{R}^d$ be a set of n points in general position. One can compute the egg of P in $O(n^{d-1} \log n)$ expected time.*

Proof. Follows by Corollary 15 with $S = [\![n]\!]$. (Alternatively, by directly modifying the decision procedure to check if any vertex of the zone $\mathcal{Z}(\Delta', H')$ lies inside Δ'.) ◀

3.6 An algorithm sensitive to k

Recall that to compute the extremal yolk, we reduced the problem to computing the smallest ball intersecting all hyperplanes which contain a fixed number of points of P above or on them (see Lemma 10). In particular, we developed an algorithm for Problem 9 and applied it when k is proportional to n. It is natural to ask for an algorithm for Problem 9 which is faster when k is small. The algorithm will work for all values of k. However when k is large, the running time deteriorates to the running time of the algorithm of Lemma 14.

To develop an algorithm sensitive to k, we use the result of Lemma 14 as a black-box and introduce the notion of shallow cuttings.

▶ **Definition 18** (Shallow cuttings). *Let H be a set of n hyperplanes in $\mathbb{R}^d$. A **k-shallow cutting** is a collection of simplices such that:*

- **(i)** *the union of the simplices covers the $(\leq k)$-level of H (see Definition 5), and*
- **(ii)** *each simplex intersects at most k hyperplanes of H.*

Matoušek was the first to prove existence of k-shallow cuttings of size $O((n/k)^{\lfloor d/2 \rfloor})$ [15]. When $d = 2, 3$, a k-shallow cutting of size $O(n/k)$ can be constructed in $O(n \log n)$ time [5]. For $d \geq 4$, we sketch a randomized algorithm which computes a k-shallow cutting, based on Matoušek's original proof of existence [15].

▶ **Lemma 19** (Proof sketch in [14]). *Let H be a set of n hyperplanes in $\mathbb{R}^d$. A k-shallow cutting of size $O((n/k)^{\lfloor d/2 \rfloor})$ can be constructed in $O(k(n/k)^{\lfloor d/2 \rfloor} + n \log n)$ expected time. For each simplex Δ in the cutting, the algorithm returns the set of hyperplanes intersecting Δ and the number of hyperplanes lying below Δ.*

Let $P \subset \mathbb{R}^d$ be a set of n points and let $H = P^\star$ be the set of dual hyperplanes. The algorithm itself is a randomized incremental algorithm, mimicking Seidel's algorithm for solving LPs [19]. First, compute a k-shallow cutting for the set of hyperplanes H using Lemma 19. Let $\Delta_1, \ldots, \Delta_\ell$, where $\ell = O((n/k)^{\lfloor d/2 \rfloor})$, be the collection of simplices in the cutting. For each simplex Δ_i, we have the subset $H \cap \Delta_i$ and the number of hyperplanes lying completely below H (which is at most k). For each cell Δ_i, let $g(\Delta_i)$ be the set of vertices of $\mathcal{A}(H)$ which have level k and are contained in Δ_i.

The algorithm. The input to the algorithm is a set of simplices and an initial ball b_0. Such a ball is uniquely defined by a subset of $d+1$ constraints, and this is a basis for the LP-type problem.

Begin by randomly permuting the simplices $\Delta_1, \ldots, \Delta_\ell$. At all times, the algorithm maintains a ball b_i of smallest radius which meets all the constraints defined by $\cup_{j=1}^{i} g(\Delta_i)$. In the ith iteration, the algorithm performs a *violation test*: it decides if any constraint of $g(\Delta_i)$ is violated by b_{i-1}. If so, the algorithm executes a *basis computation*, in which it computes the ball b'_i of smallest radius which obeys the constraints of $g(\Delta_i)$ and the $d+1$ constraints defining b_{i-1}. The algorithm then computes a ball b_i by invoking itself recursively on the subset of cells $\Delta_1, \ldots, \Delta_i$ with b'_i as the initial basis.

▶ **Lemma 20.** *Let $P \subset \mathbb{R}^d$ be a set of n points in general position. For a given integer k, one can compute in $\widetilde{O}\big(k^{d-1}\big(1 + (n/k)^{\lfloor d/2 \rfloor}\big)\big)$ expected time the smallest radius ball intersecting all of the hyperplanes of $E_k(P)$.*

Proof. The algorithm is described above. As for the analysis, it is similar to any randomized incremental algorithm for LP-type problems. The key difference is that we are not adding a single constraint incrementally, but rather a collection of constraints in each iteration. Fortunately, this does not change the analysis of the algorithm (for further details, see the proof of Lemma 2 in [4] or the full version of our paper [14]).

It is well-known that in expectation, the algorithm performs $O((n/k)^{\lfloor d/2 \rfloor})$ violation tests and $O(\log^{d+1}(n/k))$ basis computations [20]. Since each simplex Δ_i intersects $O(k)$ hyperplanes of H, each of these subroutines can be implemented in $O(k^{d-1} \log k)$ time using Lemma 14. Finally, we account for the time needed to construct the shallow cutting – by Lemma 19 this can be done in $O(k(n/k)^{\lfloor d/2 \rfloor} + n \log n)$ expected time. ◀

4 Computing the (continuous) yolk

▶ **Definition 21.** *Let $P \subset \mathbb{R}^d$ be a set of n points in general position. The **continuous yolk** of P is the ball of smallest radius intersecting all median hyperplanes of P.*

In contrast to Definition 8, we emphasize that the (continuous) yolk must intersect all median hyperplanes defined by P (not just extremal median hyperplanes).

As before, the algorithm works in the dual space. For an integer k, let $H_k(P)$ be the collection of halfspaces containing exactly k points of P on or above it. Equivalently, $P^\star$ is the collection of hyperplanes defined by P in the dual space, and $\big(H_k(P)\big)^\star$ is the k-level of $P^\star$. Our problem can be restated in the dual space as follows.

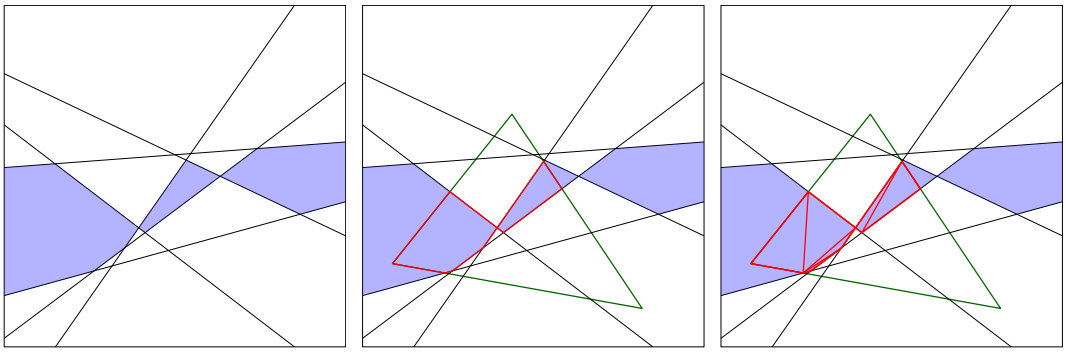


Figure 4.1 Left: A set of lines and the cells of the 3-level. Middle: A simplex Δ, with the portion of the 3-level inside Δ. Right: Triangulating the portion of the 3-level contained inside Δ. All red triangles together with the lower dimensional faces of the 3-level form the set of constraints $g(\Delta)$.

▶ Problem 22. Let P be a set of points in $\mathbb{R}^d$ in general position and let k be a given integer. Compute the ball b of smallest radius so that all points in the k-level of $P^\star$ are contained inside the region $\mathsf{b}^\star$.

Let $L_k(P) = \big(H_k(P)\big)^\star$ denote the set of all points in the k-level of $P^\star$. Note that $L_k(P)$ consists of points which are either contained in the interior of some ℓ-dimensional flat, where $0 \leq \ell \leq d-1$, or in the interior of some d-dimensional cell of $\mathcal{A}(P^\star)$.

We take the same approach as the algorithm of Theorem 16 – building a decider subroutine, and showing that the input space can be decomposed into subproblems efficiently. However the problem is more subtle, as the collection of constraints (i.e., median hyperplanes) is no longer a finite set.

The input space. The input consists of a simplex Δ. The algorithm, in addition to Δ, maintains the set of hyperplanes

$$H = P^\star \cap \Delta = \{h \in P^\star \mid h \cap \Delta \neq \emptyset\},$$

and a parameter u which is equal to the number of hyperplanes of $P^\star$ lying completely below Δ.

The implicit constraint space. Each input Δ maps to a region R which is the portion of the k-level $L_k(P)$ contained inside Δ. For each d-dimensional cell in R, we compute its bottom-vertex triangulation (see, e.g., [16, Section 6.5]), and collect all of these simplices, and all lower dimensional faces of R, into a set $g(\Delta)$, see Figure 4.1.

Let Ξ be the collection of all simplices formed from $d+1$ vertices of the arrangement $\mathcal{A}(P^\star)$. We let $\mathcal{H}$ be the union of the sets $g(\Delta)$ over all simplices $\Delta \in \Xi$. To see why this suffices, each simplex in the input space is a simplex generated by a cutting algorithm. One property of cutting algorithms [6] is that the simplices returned are induced by hyperplanes of $P^\star$. Indeed, each simplex has (at most) $d+1$ vertices, and upon inspection of the cutting algorithm, each vertex is defined by d hyperplanes of $P^\star$. There are a finite number of simplices Δ to consider, and each Δ induces a fixed subset of constraints $g(\Delta) \subseteq \mathcal{H}$.

As such, $\mathcal{H}$ forms our constraint set, where each constraint is of constant size (depending on d). Clearly, a solution satisfies all constraints of $\mathcal{H}$ if and only if the solution intersects all hyperplanes in the set $H_k(P)$. For a given subset $\mathcal{C} \subseteq \mathcal{H}$, the objective function is the minimum radius ball b such that all regions of $\mathcal{C}$ are contained inside the region $\mathsf{b}^\star$. In particular, the problem of computing the minimum radius ball b such that $\mathsf{b}^\star$ contains all points of $L_k(P)$ in its interior is an LP-type problem of constant combinatorial dimension.

Constructing subproblems. For a given input simplex Δ (along with the set $H = P^\star \cap \Delta$ and the number u) a collection of subproblems $\Delta_1, \ldots, \Delta_\psi$ (with the corresponding sets H_i and numbers u_i for $i = 1, \ldots, \psi$) can be constructed as described in Section 3.3, by computing a cutting of the planes H and clipping this cutting inside Δ. In particular, we have that $\bigcup_i g(\Delta_i) = g(\Delta)$. Strictly speaking, we have not decomposed the constraints of $g(\Delta)$ (as required by Lemma 2), but rather have decomposed the region which is the union of the constraints of $g(\Delta)$. This step is valid, as a solution satisfies the constraints of $\bigcup_i g(\Delta_i)$ if and only if it satisfies the constraints of $g(\Delta)$.

The decision procedure. Given a candidate solution $\mathsf{b}^\star$, the problem is to decide if $\mathsf{b}^\star$ contains $g(\Delta)$ in its interior. The decision algorithm itself is similar as in the proof of Theorem 16. Consider the set $\Delta \cap (\mathbb{R}^d \setminus \mathsf{b}^\star)$, where Δ is a simplex, and notice that it is the union of the most two convex regions. Let Δ' be one of these two regions of interest. Observe that it suffices to check if there is a point on the boundary of Δ' which is part of the k-level. Let $H' \subseteq H$ be the subset of hyperplanes intersecting Δ'.

To this end, compute $\mathcal{Z}(\partial\Delta', H')$. For each $(d-1)$-dimensional face f of Δ', the collection of regions $\Xi = \{f \cap s \mid s \in \mathcal{Z}(\partial\Delta', H')\}$ forms a $(d-1)$-dimensional arrangement restricted to f. Furthermore, the complexity of this arrangement lying on f is at most $O(n^{d-1} \log n)$. Notice that the level of all points in the interior of a face of Ξ is constant, and two adjacent faces (sharing a boundary) have their level differ by at most a constant. The algorithm picks a face in Ξ, computes the level of an arbitrary point inside it (adding u to the count). Then, the algorithm walks around the arrangement, exploring all faces, using the level of neighboring faces to compute the level of the current face. If at any step a face has level k, we report that the input (Δ, H, u) violates the candidate solution $\mathsf{b}^\star$.

Analysis of the decision procedure. We claim the running time of the algorithm is proportional to the complexity of the zone $\mathcal{Z}(\partial\Delta', H')$. Indeed, for each $(d-1)$-dimensional face f of Δ' (where f may either be part of a hyperplane or part of the boundary of $\mathsf{b}^\star$), we can compute the set $\{f \cap s \mid s \in \mathcal{Z}(\partial\Delta', H')\}$ in time proportional to the total complexity of $\mathcal{Z}(\partial\Delta', H')$ (assuming we can intersect a hyperplane with a portion of a constant degree surface efficiently). The algorithm then computes the level of an initial face naively in $O(|H'|)$ time, and computing the level of all other faces can be done in $O(|\mathcal{Z}(\partial\Delta', H')|)$ time by performing a graph search on the arrangement.

Because the boundary of Δ' is constructed from $d+1$ hyperplanes and the boundary of the hyperboloid, Lemma 7 implies that the zone complexity is $O(|H|^{d-1} \log |H|)$. As such, our decision procedure runs in time $D(n) = O(n^{d-1} \log n)$.

▶ **Lemma 23.** *Problem 22 can be solved in $O(n^{d-1} \log n)$ expected time, where $n = |P|$.*

Proof. Follows by plugging the above discussion into Lemma 2. ◀

By modifying the decision procedure appropriately, we also obtain a similar result to Corollary 15.

▶ **Corollary 24.** *Let $P \subset \mathbb{R}^d$ be a set of n points in general position, and let $S \subset [\![n]\!]$. The smallest ball intersecting all hyperplanes in $\bigcup_{k \in S} H_k(P)$ can be computed in $O(n^{d-1} \log n)$ expected time.*

▶ **Theorem 25.** *Let $P \subset \mathbb{R}^d$ be a set of n points in general position. One can compute the yolk of P in $O(n^{d-1} \log n)$ expected time.*

Proof. The result follows by applying Corollary 24 with the appropriate choice of S. When n is even, Lemma 10 tells us to choose $S = \{n/2, n/2+1, \ldots, n/2+d\}$. When n is odd, we set $S = \{\lceil n/2 \rceil, \lceil n/2 \rceil + 1, \ldots, \lceil n/2 \rceil + d - 1\}$. ◀

▶ Remark 26. In $\mathbb{R}^3$, one can shave the $O(\log n)$ factor to obtain an $O(n^2)$ expected time algorithm for the yolk. We modify the decision procedure as follows, which avoids computing the zone $\mathcal{Z}(\partial\Delta', H')$. For each 2D face f of Δ', simply compute the arrangement of the set of lines $\{f \cap h \mid h \in H\}$ on f in $O(n^2)$ time. As before, we perform a graph search on this arrangement, computing the level of each face. If any time we discover a point on the boundary of Δ' of the desired level, we report that the given input violates the given candidate solution.

5 Conclusion

The natural open problem is to improve the running times for computing the yolk (and extremal yolk) even further. It seems believable, that for $d > 3$, the log factors in Theorem 16 and Theorem 25 might not be necessary. We leave this as an open problem for further research.

References

1. Boris Aronov, Marco Pellegrini, and Micha Sharir. On the zone of a surface in a hyperplane arrangement. *Discrete Comp. Geom.*, 9:177–186, 1993. doi:10.1007/BF02189317.
2. Binay K. Bhattacharya, Shreesh Jadhav, Asish Mukhopadhyay, and Jean-Marc Robert. Optimal algorithms for some intersection radius problems. *Computing*, 52(3):269–279, 1994. doi:10.1007/BF02246508.
3. Duncan Black. On the rationale of group decision-making. *Journal of Political Economy*, 56(1):23–34, 1948. doi:10.1086/256633.
4. Timothy M. Chan. An optimal randomized algorithm for maximum Tukey depth. In J. Ian Munro, editor, *Proc. 15th ACM-SIAM Sympos. Discrete Algs.* (SODA), pages 430–436. SIAM, 2004. URL: http://dl.acm.org/citation.cfm?id=982792.982853.
5. Timothy M. Chan and Konstantinos Tsakalidis. Optimal deterministic algorithms for 2-d and 3-d shallow cuttings. *Discrete Comp. Geom.*, 56(4):866–881, 2016. doi:10.1007/s00454-016-9784-4.
6. Bernard Chazelle. Cutting hyperplanes for divide-and-conquer. *Discrete Comp. Geom.*, 9:145–158, 1993. doi:10.1007/BF02189314.
7. Kenneth L. Clarkson. Las vegas algorithms for linear and integer programming when the dimension is small. *J. ACM*, 42(2):488–499, 1995. doi:10.1145/201019.201036.
8. Mark de Berg, Jonathan Chung, and Joachim Gudmundsson. Computing the yolk in spatial voting games, 2019.
9. Mark de Berg, Katrin Dobrindt, and Otfried Schwarzkopf. On lazy randomized incremental construction. *Discrete Comp. Geom.*, 14(3):261–286, 1995. doi:10.1007/BF02570705.
10. Mark de Berg, Joachim Gudmundsson, and Mehran Mehr. Faster algorithms for computing plurality points. *ACM Trans. Algorithms*, 14(3):36:1–36:23, 2018. doi:10.1145/3186990.
11. Herbert Edelsbrunner, Raimund Seidel, and Micha Sharir. On the zone theorem for hyperplane arrangements. *SIAM J. Comput.*, 22(2):418–429, 1993. doi:10.1137/0222031.
12. Joachim Gudmundsson and Sampson Wong. Computing the yolk in spatial voting games without computing median lines. In *33th Conf. Artificial Intell.* (AAAI), 2019.
13. Joachim Gudmundsson and Sampson Wong. Computing the yolk in spatial voting games without computing median lines. *CoRR*, abs/1902.04735, 2019. arXiv:1902.04735.
14. Sariel Har-Peled and Mitchell Jones. Fast algorithms for geometric consensuses. *CoRR*, abs/1912.01639, 2019. arXiv:1912.01639.

15 Jiří Matoušek. Reporting points in halfspaces. *Comput. Geom.*, 2:169–186, 1992. doi:10.1016/0925-7721(92)90006-E.

16 Jiří Matoušek. *Lectures on Discrete Geometry*, volume 212 of *Grad. Text in Math.* Springer, 2002. doi:10.1007/978-1-4613-0039-7/.

17 Richard D. McKelvey. Covering, dominance, and institution-free properties of social choice. *American Journal of Political Science*, 30(2):283–314, 1986. doi:10.2307/2111098.

18 Ariel Rubinstein. A note about the "nowhere denseness" of societies having an equilibrium under majority rule. *Econometrica*, 47(2):511–514, 1979. doi:10.2307/1914198.

19 Raimund Seidel. Small-dimensional linear programming and convex hulls made easy. *Discrete Comp. Geom.*, 6:423–434, 1991. doi:10.1007/BF02574699.

20 Micha Sharir and Emo Welzl. A combinatorial bound for linear programming and related problems. In *9th Symp. on Theoretical Aspects of Comput. Sci.* (STACS), pages 569–579, 1992. doi:10.1007/3-540-55210-3_213.

21 Richard E. Stone and Craig A. Tovey. Limiting median lines do not suffice to determine the yolk. *Social Choice and Welfare*, 9(1):33–35, 1992. doi:10.1007/BF00177668.

22 Craig A. Tovey. A polynomial-time algorithm for computing the yolk in fixed dimension. *Math. Program.*, 57:259–277, 1992. doi:10.1007/BF01581084.

Dynamic Approximate Maximum Independent Set of Intervals, Hypercubes and Hyperrectangles

Monika Henzinger
Faculty of Computer Science, University of Vienna, Austria
monika.henzinger@univie.ac.at

Stefan Neumann
Faculty of Computer Science, University of Vienna, Austria
stefan.neumann@univie.ac.at

Andreas Wiese
Department of Industrial Engineering, Universidad de Chile, Santiago, Chile
awiese@dii.uchile.cl

Abstract

Independent set is a fundamental problem in combinatorial optimization. While in general graphs the problem is essentially inapproximable, for many important graph classes there are approximation algorithms known in the offline setting. These graph classes include interval graphs and geometric intersection graphs, where vertices correspond to intervals/geometric objects and an edge indicates that the two corresponding objects intersect.

We present *dynamic* approximation algorithms for independent set of intervals, hypercubes and hyperrectangles in d dimensions. They work in the fully dynamic model where each update inserts or deletes a geometric object. All our algorithms are deterministic and have worst-case update times that are polylogarithmic for constant d and $\varepsilon > 0$, assuming that the coordinates of all input objects are in $[0, N]^d$ and each of their edges has length at least 1. We obtain the following results:

- For weighted intervals, we maintain a $(1+\varepsilon)$-approximate solution.
- For d-dimensional hypercubes we maintain a $(1+\varepsilon)2^d$-approximate solution in the unweighted case and a $O(2^d)$-approximate solution in the weighted case. Also, we show that for maintaining an unweighted $(1+\varepsilon)$-approximate solution one needs polynomial update time for $d \geq 2$ if the ETH holds.
- For weighted d-dimensional hyperrectangles we present a dynamic algorithm with approximation ratio $(1+\varepsilon)\log^{d-1} N$.

2012 ACM Subject Classification Theory of computation → Dynamic graph algorithms; Theory of computation → Computational geometry

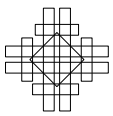
Keywords and phrases Dynamic algorithms, independent set, approximation algorithms, interval graphs, geometric intersection graphs

Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.51

Related Version The full version of this paper is available on arXiv [23] under `http://arxiv.org/abs/2003.02605`.

Funding *Monika Henzinger*: The research leading to these results has received funding from the European Research Council under the European Community's Seventh Framework Programme (FP7/2007-2013) / ERC grant agreement No. 340506.
Stefan Neumann: Part of this work was done while visiting Brown University. Stefan Neumann gratefully acknowledges the financial support from the Doctoral Programme "Vienna Graduate School on Computational Optimization" which is funded by the Austrian Science Fund (FWF, project no. W1260-N35). The research leading to these results has received funding from the European Research Council under the European Community's Seventh Framework Programme (FP7/2007-2013) / ERC grant agreement No. 340506.
Andreas Wiese: Andreas Wiese was supported by the grant Fondecyt Regular 1170223.

Acknowledgements We are grateful to the anonymous reviewers for their helpful comments.



1 Introduction

A fundamental problem in combinatorial optimization is the independent set (IS) problem. Given an undirected graph $G = (V, E)$ with n vertices and m edges, the goal is to select a set of nodes $V' \subseteq V$ of maximum cardinality such that no two vertices $u, v \in V'$ are connected by an edge in E. In general graphs, IS cannot be approximated within a factor of $n^{1-\varepsilon}$ for any $\varepsilon > 0$, unless $\mathsf{P} = \mathsf{NP}$ [31]. However, there are many approximation algorithms known for special cases of IS where much better approximation ratios are possible or the problem is even polynomial-time solvable. These cases include interval graphs and, more generally, geometric intersection graphs.

In interval graphs each vertex corresponds to an interval on the real line and there is an edge between two vertices if their corresponding intervals intersect. Thus, an IS corresponds to a set of non-intersecting intervals on the real line; the optimal solution can be computed in time $O(n+m)$ [20] when the input is presented as an interval graph and in time $O(n \log n)$ [26, Chapter 6.1] when the intervals themselves form the input (but not their corresponding graph). Both algorithms work even in the weighted case where each interval has a weight and the objective is to maximize the total weight of the selected intervals.

When generalizing this problem to higher dimensions, the input consists of axis-parallel d-dimensional hypercubes or hyperrectangles and the goal is to find a set of non-intersecting hypercubes or hyperrectangles of maximum cardinality or weight. This is equivalent to solving IS in the *geometric intersection graph* of these objects which has one (weighted) vertex for each input object and two vertices are adjacent if their corresponding objects intersect. This problem is NP-hard already for unweighted unit squares [19], but if all input objects are weighted hypercubes then it admits a PTAS for any constant dimension d [10, 18]. For hyperrectangles there is a $O((\log n)^{d-2} \log \log n)$-approximation algorithm in the unweighted case [9] and a $O((\log n)^{d-1} / \log \log n)$-approximation algorithm in the weighted case [12, 9]. IS of (hyper-)cubes and (hyper-)rectangles has many applications, e.g., in map labelling [2, 29], chip manufacturing [24], or data mining [25]. Therefore, approximation algorithms for these problems have been extensively studied, e.g., [12, 9, 2, 1, 11, 14].

All previously mentioned algorithms work in the static offline setting. However, it is a natural question to study IS in the *dynamic* setting, i.e., where (hyper-)rectangles appear or disappear, and one seeks to maintain a good IS while spending only little time after each change of the graph. The algorithms above are not suitable for this purpose since they are based on dynamic programs in which $n^{\Omega(1/\varepsilon)}$ many sub-solutions might change after an update or they solve linear programs for the entire input. For general graphs, there are several results for maintaining a *maximal* IS dynamically [4, 5, 22, 15, 28, 13, 7], i.e., a set $V' \subseteq V$ such that $V' \cup \{v\}$ is not an IS for any $v \in V \setminus V'$. However, these algorithms do not imply good approximation ratios for the geometric setting we study: Already in unweighted interval graphs, a maximal IS can be by a factor $\Omega(n)$ smaller than the maximum IS. For dynamic IS of intervals, Gavruskin et al. [21] showed how to maintain an exact maximum IS with polylogarithmic update time in the special case when no interval is fully contained in any another interval.

Our contributions. In this paper, we present dynamic algorithms that maintain an approximate IS in the geometric intersection graph for three different types of geometric objects: intervals, hypercubes and hyperrectangles. We assume throughout the paper that the given objects are axis-parallel and contained in the space $[0, N]^d$, that we are given the value N in advance, and that each edge of an input object has length at least 1 and at most N. We

Table 1 Summary of our dynamic approximation algorithms and lower bounds. All algorithms are deterministic and work in the fully dynamic setting where in each update one interval/hypercube is inserted or deleted. We assume that all input objects are contained in $[0, N]^d$ and have weights in $[1, W]$; we do *not* assume that the input objects have integer-coordinates. Here, we write $O_\varepsilon(1)$ and $O_{d,\varepsilon}(1)$ to hide terms which only depend on d and ε.

	Approximation ratio	Worst-case update time
Unweighted intervals	$1+\varepsilon$	$O_\varepsilon(1)\log^2 n \log^2 N$
	1	$\Omega(\log N/\log\log N)$
Weighted intervals	$1+\varepsilon$	$O_\varepsilon(1)\log^2 n \log^5 N \log W$
Unweighted d-dimensional hypercubes	$(1+\varepsilon)2^d$	$O_{d,\varepsilon}(1)\log^{2d+1} n \log^{2d+1} N$
	$1+\varepsilon$	$n^{(1/\varepsilon)^{\Omega(1)}}$
Weighted d-dimensional hypercubes	$(4+\varepsilon)2^d$	$O_{d,\varepsilon}(1)\log^{2d-1} n \log^{2d+1} N \log W$
Weighted d-dimensional hyperrectangles	$(1+\varepsilon)\log^{d-1} N$	$O_{d,\varepsilon}(1)\log^2 n \log^5 N \log W$

study the fully dynamic setting where in each update an input object is inserted or deleted. Note that this corresponds to inserting and deleting *vertices* of the corresponding intersection graph. In particular, when a vertex is inserted/deleted then potentially $\Omega(n)$ edges might be inserted/deleted, i.e., there might be more edge changes per operation than in the standard dynamic graph model in which each update can only insert or delete a single edge.

(1) For independent set in weighted interval graphs we present a dynamic $(1+\varepsilon)$-approximation algorithm. For weighted d-dimensional hypercubes our dynamic algorithm maintains a $(4+\varepsilon)2^d$-approximate solution; in the case of unweighted d-dimensional hypercubes we obtain an approximation ratio of $(1+\varepsilon)2^d$. Thus, for constant d we achieve a constant approximation ratio. Furthermore, for weighted d-dimensional hyperrectangles we obtain a dynamic algorithm with approximation ratio of $(1+\varepsilon)\log^{d-1} N$.

Our algorithms are deterministic with worst-case update times that are polylogarithmic in n, N, and W, where W is the maximum weight of any interval or hypercube, for constant d and ε; we also show how to obtain faster update times using randomized algorithms that compute good solutions with high probability. In each studied setting our algorithms can return the computed IS I in time $O_{d,\varepsilon}(|I| \cdot \text{poly}(\log n, \log N))$, where $|I|$ denotes the cardinality of I and the $O_{d,\varepsilon}(\cdot)$ notation hides factors which only depend on d and ε. *Up to a $(1+\varepsilon)$-factor our approximation ratios match those of the best known near-linear time offline approximation algorithms for the respective cases* (with ratios of 2^d and $O(2^d)$ via greedy algorithms for unweighted and weighted hypercubes and $\log^{d-1} N$ for hyperrectangles [2]). See Table 1 for a summary of our algorithms.

(2) Apart from the comparison with the static algorithm we show two lower bounds: We prove that one cannot maintain a $(1+\varepsilon)$-approximate IS of unweighted hypercubes in $d \geq 2$ dimensions with update time $n^{O((1/\varepsilon)^{1-\delta})}$ for any $\delta > 0$ (so even with polynomial instead of polylogarithmic update time), unless the Exponential Time Hypothesis fails. Also, we show that maintaining a maximum weight IS in an interval graph requires $\Omega(\log N/\log\log N)$ amortized update time.

Techniques. Our main obstacle is that the maximum IS is a *global* property, i.e., when the input changes slightly, e.g., a single interval is inserted or deleted, then it can cause a change of the optimal IS which propagates through the entire instance (see Figure 1). Even worse, there are instances in which *any* solution *with a non-trivial approximation guarantee* requires $\Omega(n)$ changes after an update (see Figure 2).

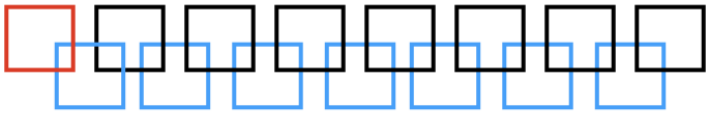


Figure 1 An instance of unweighted IS of intervals. Observe that before inserting the red interval at the left, the solution consisting of the blue intervals in the bottom row is optimal. However, after inserting the red interval, the optimal solution is unique and consists of the red interval together with all black intervals in the top row.

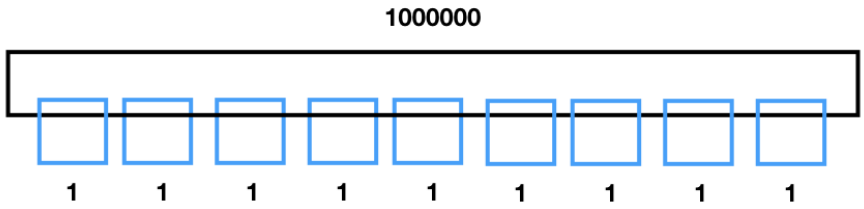


Figure 2 An instance of weighted IS of intervals. Note that when the large black interval with weight 1000000 is present, any solution with non-trivial approximation ratio must contain the black interval. However, when the black interval is not present, the solution must contain many small blue intervals. Thus, inserting or deleting the black interval requires $\Omega(n)$ changes to the solution.

To limit the propagation effects, our algorithms for intervals and hypercubes use a hierarchical grid decomposition. We partition the space $[0, N]^d$ recursively into equally-sized grid cells with $\log N$ levels, halving the edge length in each dimension of each cell when going from one level to the next (similar to the quad-tree decomposition in [3]). Thus, each grid cell Q has 2^d *children* cells which are the cells of the respective next level that are contained in Q. Also, each input object C (i.e., interval or hypercube) is contained in at most $\log N$ grid cells and it is *assigned* to the grid level $\ell(C)$ in which the size of the cells is "comparable" to the size of C. When an object C is inserted or deleted, we recompute the solution for each of the $\log N$ grid cells containing C, in a bottom-up manner. More precisely, for each such cell Q we decide which of the hypercubes assigned to it we add to our solution, based on the solutions of the children of Q. Thus, a change of the input does *not* propagate through our entire solution but only affects $\log N$ grid cells and the hypercubes assigned to them.

Also, we do not store the computed solution explicitly as this might require $\Omega(n)$ changes after each update. Instead, we store it *implicitly*. In particular, in each grid cell Q we store a solution only consisting of objects assigned to Q and pointers to the solutions of children cells. Finally, at query time we output only those objects that are contained in a solution of a cell Q and which do not overlap with an object in the solution of a cell of higher level. In this way, if a long interval with large weight appears or disappears, only the cell corresponding to the interval needs to be updated, the other changes are done implicitly.

Another challenge is to design an algorithm that, given a cell Q and the solutions for the children cells of Q, computes an approximate solution for Q in time $\text{poly}(\log n, \log N)$. In such a small running time, we cannot afford to iterate over all input objects assigned to Q. We now explain in more detail how our algorithm overcome this obstacle.

Weighted hypercubes. Let us first consider our $(4+\varepsilon)2^d$-approximation algorithm for weighted hypercubes. Intuitively, we consider the hypercubes ordered non-decreasingly by size and add a hypercube C to the IS if the weight of C is at least twice the total weight of all hypercubes in the current IS overlapping with C. We then remove all hypercubes in the solution that overlap with C.

To implement this algorithm in polylogarithmic time, we need to make multiple adjustments. First, for each cell Q we maintain a range counting data structure $P(Q)$ which contains the (weighted) vertices of all hypercubes that were previously selected in the IS solutions of children cells $Q' \subseteq Q$. We will use $P(Q)$ to estimate the weight of hypercubes that a considered hypercube C overlaps with. Second, we use $P(Q)$ to construct an *auxiliary grid* within Q. The auxiliary grid is defined such that in each dimension the grid contains $O_{d,\varepsilon}(\text{polylog}\, N)$ grid slices; thus, there are $(\log N)^{O_{d,\varepsilon}(1)}$ subcells of Q induced by the auxiliary grid. Third, we cannot afford to iterate over all hypercubes contained in Q to find the smallest hypercube C that has at least twice the weight w' of the hypercubes in the current solution that overlap with C. Instead, we iterate over all subcells $S \subseteq Q$ which are induced by the auxiliary grid and look for a hypercube C of large weight within S; we show that the total weight of the points in $S \cap P(Q)$ is a sufficiently good approximation of w'. If we find a hypercube C with these properties, we add C to the current solution for Q, add the vertices of C to $P(Q)$ and adjust the auxiliary grid accordingly. In this way, we need to check only $(\log N)^{O_{d,\varepsilon}(1)}$ subcells of Q which we can do in polylogarithmic time, rather than iterating over all hypercubes assigned to Q. We ensure that for each cell Q we need to repeat this process only a polylogarithmic number of times. To show the approximation bound we use a novel charging argument based on the points in $P(Q)$. We show that the total weight of the points stored in $P(Q)$ estimates the weight of the optimal solution for Q up to a constant factor. We use this to show that our computed solution is a $(4+\varepsilon)2^d$-approximation.

Weighted intervals. Next, we sketch our dynamic $(1+\varepsilon)$-approximation algorithm for weighted IS of intervals. A greedy approach would be to build the solution such that the intervals are considered in increasing order of their lengths and then for each interval to decide whether we want to select it and whether we want to remove some previously selected intervals to make space for it. However, this cannot yield a $(1+\varepsilon)$-approximate solution. There are examples in which one can choose only one out of multiple overlapping short intervals and the wrong choice implies that one cannot obtain a $(1+\varepsilon)$-approximation together with the long intervals that are considered later (see Figure 3). However, in these examples the optimal solution (say for a cell Q) consists of only $O_\varepsilon(1)$ intervals. Therefore, we show that in this case we can compute a $(1+\varepsilon)$-approximate solution in time $O_\varepsilon(\log^2 n)$ by guessing the rounded weights of the intervals in the optimal solution, guessing the order of the intervals with these weights, and then selecting the intervals greedily according to this order. On the other hand, if the optimal solution for a cell Q contains $\Omega_\varepsilon(1)$ many intervals with similar weights then we can take the union of the previously computed solutions for the two children cells of Q. This sacrifies at most one interval in the optimal solution for Q that overlaps with both children cells of Q and we can charge this interval to the $\Omega_\varepsilon(1)$ intervals in the solutions for the children cells of Q.

Our algorithm interpolates between these two extreme cases. To this end, we run the previously described $O(1)$-approximation algorithm for hypercubes as a subroutine and use it to split each cell Q into *segments*, guided by the set $P(Q)$ above. Then we use that for each set $S \subseteq Q$ the weight of $S \cap P(Q)$ approximates the weight of the optimal solutions of intervals contained in S within a constant factor. This is crucial for some of our charging

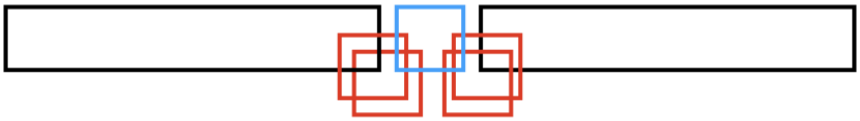


Figure 3 An instance of unweighted IS of intervals. Observe that the optimal solution has size 3 and consists of the small blue interval and the two large black intervals at the top. If an algorithm decides to pick any of the small red intervals, its solution can have size at most 2.

arguments in which we show that the intervals contained in some sets S can be ignored. We show that for each cell Q there is a $(1+\varepsilon)$-approximate solution in which Q is partitioned into segments such that each of them is either *dense* or *sparse*. Each dense segment contains many intervals of the optimal solution and it is contained in one of the children cells of Q. Therefore, we can copy the previously computed solution for the respective child of Q. Each sparse segment only contains $O_\varepsilon(1)$ intervals and hence we can compute its solution directly using guesses as described above. In each level, this incurs an error and we use several involved charging arguments to ensure that this error does not accumulate over the $\log N$ levels, but that instead it stays bounded by $1+\varepsilon$.

Other related work. Emek et al. [17], Cabello and Pérez-Lantero [8] and Bakshi et al. [6] study IS of intervals in the streaming model and obtain algorithms with sublinear space usage. In [17, 8] insertion-only streams of unweighted intervals are studied. They present algorithms which are $(3/2+\varepsilon)$-approximate for unit length intervals and $(2+\varepsilon)$-approximate for arbitrary-length intervals; they also provide matching lower bounds. Bakshi et al. [6] study turnstile streams in which intervals can be inserted and deleted. They obtain algorithms which are $(2+\varepsilon)$-approximate for weighted unit length intervals and a $O(\log n)$-approximate for unweighted arbitrary length intervals; they also prove matching lower bounds.

In two dimensions, Agarwal et al. [2] presented a static algorithm which computes $O(\log n)$ approximation of the maximum IS of n arbitrary axis-parallel rectangles in time $O(n \log n)$. They also show how to compute a $(1+1/k)$-approximation of unit-height rectangles in time $O(n \log n + n^{2k-1})$ for any integer $k \geq 1$.

Problem definition and notation. We assume that we obtain a set $\mathcal{C} = \{C_1, \ldots, C_n\}$ of d-dimensional hyperrectangles in the space $[0, N]^d$ for some global value $N \in \mathbb{R}$. Each hyperrectangle $C_i \in \mathcal{C}$ is characterized by coordinates $x_i^{(1)}, y_i^{(1)}, \ldots, x_i^{(d)}, y_i^{(d)} \in [0, N]$ such that $C_i := (x_i^{(1)}, y_i^{(1)}) \times \cdots \times (x_i^{(d)}, y_i^{(d)})$ and a weight $w_i \in [1, W]$ for some global value W; we do *not* assume that the coordinates of the input objects are integer-valued. We assume that $1 \leq y_i^{(j)} - x_i^{(j)} \leq N$ for each $j \in [d]$. If C_i is a hypercube then we define s_i such that $s_i = y_i^{(j)} - x_i^{(j)}$ for each dimension j. Two hypercubes $C_i, C_{i'} \in \mathcal{C}$ with $i \neq i'$ are *independent* if $C_i \cap C_{i'} = \emptyset$. Note that we defined the hypercubes as open sets and, hence, two dependent hypercubes cannot overlap in only a single point. A set of hyperrectangles $\mathcal{C}' \subseteq \mathcal{C}$ is an *independent set (IS)* if each pair of hypercubes in $\mathcal{C}'$ is independent. The *maximum IS* problem is to find an IS $\mathcal{C}' \subseteq \mathcal{C}$, that maximizes $w(\mathcal{C}') := \sum_{C_i \in \mathcal{C}'} w_i$.

Due to space constraints we present some results and missing proofs in the full version [23].

2 Hierarchical Grid Decomposition

We describe a hierarchical grid decomposition that we use for all our algorithms for hypercubes (for any d), that is similar to [3]. It is based on a hierarchical grid $\mathcal{G}$ over the space $[0,N]^d$ where we assume w.l.o.g. that N is a power of 2 and N is an upper bound on the coordinates of every object in each dimension. The grid $\mathcal{G}$ has $\log N$ *levels*. In each level, the space $[0,N]^d$ is divided into *cells*; the union of the cells from each level spans the whole space. There is one grid cell of level 0 that equals to the whole space $[0,N]^d$. Essentially, each grid cell of a level $\ell < \log N$ contains 2^d grid cells of level $\ell+1$. We *assign* the input hypercubes to the grid cells. In particular, for a grid cell $Q \in \mathcal{G}$ we assign a set $\mathcal{C}'(Q) \subseteq \mathcal{C}$ to Q which are all input hypercubes that are contained in Q and whose side length is a $\Theta(\varepsilon/d)$-fraction of the side length of Q (we will make this formal later). This ensures the helpful property that any IS consisting only of hypercubes in $\mathcal{C}'(Q)$ has size at most $O\left((\frac{d}{\varepsilon})^d\right)$. For each cell Q we define $\mathcal{C}(Q) := \bigcup_{Q':Q'\subseteq Q} \mathcal{C}'(Q)$ which are all hypercubes contained in Q. One subtlety is that there can be input hypercubes that are not assigned to any grid cell, e.g., hypercubes that are very small but overlap more than one very large grid cell. Thus, we shift the grid by some offset $a \in [0,N]$ in each dimension which ensures that those hypercubes are negligible.

Formally, let $\varepsilon > 0$ such that $1/\varepsilon$ is an integer and a power of 2. For each $\ell \in \{0,\dots,\log N\}$ let $\mathcal{G}_\ell$ denote the set of grid cells of level ℓ defined as $Q_{\ell,k} := [0,N]^d \cap \prod_{j=1}^d [a+k^{(j)}\cdot N/2^{\ell-1}, a+(k^{(j)}+1)\cdot N/2^{\ell-1}]$ for each $k = (k^{(1)},\dots,k^{(d)}) \in \mathbb{Z}^d$. Then $\mathcal{G}_0$ consists of only one cell $[0,N]^d =: Q^*$.We define $\mathcal{G} := \bigcup_{\ell=0}^{\log N} \mathcal{G}_\ell$. For a grid cell $Q \in \mathcal{G}$, we let $\ell(Q)$ denote the level of Q in $\mathcal{G}$. Note that for each cell Q of level $\ell(Q) < \log N$, there are at most 2^d grid cells Q_i of level $\ell(Q_i) = \ell(Q)+1$ and that are contained in Q, i.e., such that $Q_i \subseteq Q$. We call the latter cells the *children* of Q and denote them by $\mathrm{ch}(Q)$. Informally, a hypercube C_i has *level* ℓ if s_i is within a $\Theta(\varepsilon/d)$-fraction of the side length of grid cells of level ℓ; formally, C_i has level ℓ if $s_i \in [\varepsilon N/(d2^{\ell-1}), 2\cdot\varepsilon N/(d2^{\ell-1}))$ for $\ell = 1,\dots,\log N$ and $s_i \in [\varepsilon N/d, N]$ for $\ell = 0$. For each $C \in \mathcal{C}$ denote by $\ell(C)$ the level of C. We *assign* a hypercube C to a cell Q if $C_i \subseteq Q$ and $\ell(C) = \ell(Q)$; the set of all these hypercubes for a cell C is defined by $\mathcal{C}'(Q) := \{C_i \in \mathcal{C} | C_i \subseteq Q \wedge \ell(C) = \ell(Q)\}$. For each grid cell Q we define $\mathcal{C}(Q)$ to be the set of all hypercubes contained in Q that are assigned to Q or to grid cells contained in Q, i.e., $\mathcal{C}(Q) := \bigcup_{Q':Q'\subseteq Q} \mathcal{C}'(Q)$.

For each cell Q, we partition the hypercubes in $\mathcal{C}(Q)$ and $\mathcal{C}'(Q)$ based on their weights in powers of $1+\varepsilon$. For each $k \in \mathbb{Z}$ we define $\mathcal{C}_k := \{C_i \in \mathcal{C} : w_i \in [(1+\varepsilon)^k, (1+\varepsilon)^{k+1})\}$ and for each grid cell Q we define $\mathcal{C}_k(Q) := \mathcal{C}_k \cap \mathcal{C}(Q)$ and $\mathcal{C}'_k(Q) := \mathcal{C}_k \cap \mathcal{C}'(Q)$. Note that $\mathcal{C}_k = \emptyset$ if $k < 0$ or $k > \log_{1+\varepsilon} W$.

In the next lemma we prove that there is a value for the offset a such that there is a $(1+\varepsilon)$-approximate solution OPT' that is *grid-aligned*, i.e., for each $C \in \mathrm{OPT}'$ there is a grid cell Q in the resulting grid for a such that $C \in \mathcal{C}'(Q)$.

▶ **Lemma 1.** *In time $(d/\varepsilon)^{O(1/\varepsilon)} \log N$ we can compute a set $\mathrm{off}(\varepsilon)$ with $|\mathrm{off}(\varepsilon)| \le (d/\varepsilon)^{O(1/\varepsilon)}$ that is independent of the input objects $\mathcal{C}$ and that contains an offset $a \in \mathrm{off}(\varepsilon)$ for the grid for which the optimal grid-aligned solution OPT' satisfies that $w(\mathrm{OPT}') \ge (1-O(\varepsilon))w(\mathrm{OPT})$. If we draw the offset a uniformly at random from $\mathrm{off}(\varepsilon)$, then $\mathbb{E}[w(\mathrm{OPT}')] \ge (1-O(\varepsilon))w(\mathrm{OPT})$ and $w(\mathrm{OPT}') \ge (1-O(\varepsilon))w(\mathrm{OPT})$ with constant probability.*

For the deterministic results in this paper we run our algorithms for each choice of $a \in \mathrm{off}(\varepsilon)$ in parallel and at the end we output the solution with maximum weight over all choices of a. For our randomized results we choose $O(\log N)$ offsets $a \in \mathrm{off}(\varepsilon)$ uniformly at random and hence there exists a grid-aligned solution OPT' with $w(\mathrm{OPT}') \ge (1-O(\varepsilon))w(\mathrm{OPT})$ with high probability (i.e., with probability at least $1-(1/N)^{O(1)}$).

▶ **Lemma 2.** *Each grid cell $Q \in \mathcal{G}$ has a volume of $(N/2^{\ell(Q)-1})^d$ and can contain at most $(d/\varepsilon)^d$ independent hypercubes from $\mathcal{C}'(Q)$. Also, each hypercube $C_i \in \mathcal{C}$ is contained in $\mathcal{C}'(Q')$ for at most one grid cell Q' and in $\mathcal{C}(Q')$ for at most $\log N$ cells Q'.*

Data structures. We define a data structure which will allow us to access the hypercubes in the sets $\mathcal{C}(Q)$, $\mathcal{C}'(Q)$, etc. for each cell Q efficiently. Roughly speaking, these data structures let us insert and delete hypercubes and answer queries of the type: "Given a hyperrectangle B, return a hypercube which is contained in B." They are constructed using data structures for range counting/reporting [27, 30, 16].

▶ **Lemma 3.** *Let $d \in \mathbb{N}$. There is a data structure that maintains a set $\mathcal{C}'$ of weighted hypercubes in $\mathbb{R}^d$ and allows the following operations:*

1. *Initialize the data structure, report whether $\mathcal{C}' = \emptyset$, both in worst-case time $O(1)$.*
2. *Insert or delete a hypercube into (from) $\mathcal{C}'$ in worst-case time $O(\log^{2d+1} |\mathcal{C}'|)$.*
3. *For a hyperrectangle $B \subseteq \mathbb{R}^d$, check whether there is a hypercube $C_i \in \mathcal{C}'$ with $C_i \subseteq B$ in time $O(\log^{2d-1} |\mathcal{C}'|)$. If yes, return one such hypercube in time $O(\log^{2d-1} |\mathcal{C}'|)$ and the smallest such hypercube (i.e., with smallest size s_i) in time $O(\log^{2d+1} |\mathcal{C}'|)$.*
4. *If $d = 1$, given a value $t \in \mathbb{R}$, return the element $C_i = (x_i^{(1)}, y_i^{(1)})$ with minimum value $y_i^{(1)}$ among all elements $C_{i'} = (x_{i'}^{(1)}, y_{i'}^{(1)})$ with $t \leq x_{i'}^{(1)}$, in time $O(\log^2 n)$.*

Using Lemma 3 for each cell Q we define data structures $D(Q)$, $D'(Q)$, $D_k(Q)$, and $D'_k(Q)$ for maintaining the sets $\mathcal{C}(Q)$, $\mathcal{C}'(Q)$, $\mathcal{C}_k(Q)$, and $\mathcal{C}'_k(Q)$ for each $k = 1, \ldots, \log_{1+\varepsilon} W$, respectively, where W is an upper bound on the maximum weight of all hypercubes. The grid $\mathcal{G}$ as defined above contains $\Omega(N^d)$ cells in total. However, there are only $O(n \log N)$ cells in $\mathcal{G}$ such that $\mathcal{C}(Q) \neq \emptyset$ (by Lemma 2), denote them by $\mathcal{G}'$. We use a data structure that maintains these cells $\mathcal{G}'$ such that in worst-case time $O(\log |\mathcal{G}'|)$ we can add and remove a cell, get pointers to the data structures $D(Q)$, $D'(Q)$, $D_k(Q)$, $D'_k(Q)$ for a cell Q, and get and set pointers to a solution that we compute for a cell Q. See [23] for details.

Algorithmic framework. Now we sketch the framework for implementing our dynamic algorithms. Due to space constraints we postpone its formal definition to the full version [23].

For each cell Q we maintain a solution $\mathrm{DP}(Q) \subseteq \mathcal{C}(Q)$ that is near-optimal, i.e., with $w(\mathrm{OPT}(Q)) \leq \alpha \cdot w(\mathrm{DP}(Q))$ for the approximation ratio α of the respective setting. We ensure that $\mathrm{DP}(Q)$ depends only on $\mathcal{C}(Q)$ and not on hypercubes C with $C \notin \mathcal{C}(Q)$.

We implement update operations as follows. When a hypercube C is inserted or deleted, we update only the solutions $\mathrm{DP}(Q)$ for the at most $\log N$ cells Q such that $C \in \mathcal{C}(Q)$. We update the solutions $\mathrm{DP}(Q)$ in a bottom-up manner, i.e., we order the cells Q with $C \in \mathcal{C}(Q)$ decreasingly by level and update their respective solutions $\mathrm{DP}(Q)$ in this order. To ensure a total update time of $(\log n + \log N)^{O_{d,\varepsilon}(1)}$, we will define algorithms that update $\mathrm{DP}(Q)$ for a cell Q in time $(\log n + \log N)^{O_{d,\varepsilon}(1)}$, given that we already updated the solutions $\mathrm{DP}(Q')$ for all cells $Q' \subsetneq Q$. In fact, we will essentially re-compute the solution $\mathrm{DP}(Q)$ for a cell Q from scratch, using only the solutions $\{\mathrm{DP}(Q')\}_{Q' \in \mathrm{ch}(Q)}$ computed for the children of Q.

Finally, to implement query operations, i.e., to output an approximate solution for the whole space $[0, N]^d$, we return the solution $\mathrm{DP}(Q^*)$ (recall that Q^* is the grid cell at level 0 which contains the whole space). We will show in the respective sections how we can output the weight of $\mathrm{DP}(Q^*)$ in time $O_{d,\varepsilon}(1)\mathrm{poly}(\log n, \log N)$ $\mathrm{DP}(Q^*)$ and how to output all hypercubes in $\mathrm{DP}(Q^*)$ in time $O_{d,\varepsilon}(|\mathrm{DP}(Q^*)|\mathrm{poly}(\log n, \log N))$.

3 Weighted Hypercubes

We study now the weighted case for which we present a dynamic $(4+\varepsilon)2^d$-approximation algorithm for d-dimensional hypercubes. Our strategy is to mimic a greedy algorithm that sorts the hypercubes by size s_i and adds a hypercube C_i with weight w_i if it does not overlap with any previously selected hypercube or if the total weight of the previously selected hypercube that C_i overlaps with is at most $w_i/2$. Using a charging argument one can show that this yields a 2^{d+2}-approximate solution. The challenge is to implement this approach such that we obtain polylogarithmic update time.

From a high-level point of view, our algorithm works as follows. In each cell Q, we maintain a set of points $P(Q)$ containing the vertices of all hypercubes which have been added to independent sets $\mathrm{DP}(Q')$ for cells $Q' \subset Q$. The weight of each point is the weight of the corresponding hypercube. Based on the points in $P(Q)$, we construct an auxiliary grid inside Q which allows to perform the following operation efficiently: "Given a set of auxiliary grid cells A, find a hypercube $C \in \mathcal{C}'(Q)$ in A whose weight is at least twice the weight of all points in $P(Q) \cap A$." When we try to add a hypercube to Q we do not iterate over all hypercubes contained in Q but instead enumerate a polylogarithmic number of sets A and perform the mentioned query for each of them. Also, we do not maintain the current independent set explicitly (which might change a lot after an update), but we update only the weight of the points in $P(Q) \cap A$, which can be done efficiently. For each cell Q we add only a polylogarithmic number of hypercubes to $\mathrm{DP}(Q)$. If a hypercube $C_i \in \mathrm{DP}(Q)$ overlaps with a hypercube $C_{i'} \in \mathrm{DP}(Q')$ for some cell $Q' \subset Q$ then we exclude $C_{i'}$ from the solution that we output, but do not delete $C_{i'}$ from $\mathrm{DP}(Q')$. In this way, we obtain polylogarithmic update time, even if our computed solution changes a lot.

Before we describe our algorithm in detail, let us first elaborate on how we maintain the points $P(Q)$. In the unweighted settings, for each cell Q we stored in $\mathrm{DP}(Q)$ a set of hypercubes or pointers to such sets. Now, we define each set $\mathrm{DP}(Q)$ to be a pair $(\bar{\mathcal{C}}(Q), P(Q))$. Here, $\bar{\mathcal{C}}(Q) \subseteq \mathcal{C}'(Q)$ is a set of hypercubes from $\mathcal{C}'(Q)$ that we selected for the independent set (recall that $\mathcal{C}'(Q)$ contains the hypercubes $C \subseteq Q$ with $\ell(C_i) = \ell(Q)$); and $P(Q)$ is the data structure for the range counting/reporting problem according to Lemma 4. We will often identify $P(Q)$ with the set of points stored in $P(Q)$.

▶ **Lemma 4** ([27, 30, 16]). *There exists a data structure that maintains a set of weighted points $P \subseteq \mathbb{R}^d$ and allows the following operations:*

- *add or delete a point in P in worst-case time $O(\log^d |P|)$,*
- *report or change the weight of a point in P in worst-case time $O(\log^d |P|)$,*
- *given an open or closed hyperrectangle $B \subseteq \mathbb{R}^d$, report the total weight of the points $B \cap P$, in worst-case time $O(\log^{d-1} |P|)$.*
- *given $d' \in [d]$ and an interval $I = [x, z] \subseteq \mathbb{R}$, in worst-case time $O(\log |P|)$ report a value y such that at most $\Gamma := \left| P \cap \left(\mathbb{R}^{d'-1} \times [x, z] \times \mathbb{R}^{d-d'} \right) \right| /2$ points are contained in $\mathbb{R}^{d'-1} \times (x, y) \times \mathbb{R}^{d-d'}$ and at most Γ points are contained in $\mathbb{R}^{d'-1} \times (y, z) \times \mathbb{R}^{d-d'}$.*

Now we describe our algorithm in detail. Let again $x_Q^{(1)}, \dots, x_Q^{(d)}$ and $y_Q^{(1)}, \dots, y_Q^{(d)}$ be such that $Q = [x_Q^{(1)}, y_Q^{(1)}] \times \cdots \times [x_Q^{(d)}, y_Q^{(d)}]$. We construct a data structure $P(Q)$ according to Lemma 4 such that initially it contains the points $\bigcup_{Q' \in \mathrm{ch}(Q)} P(Q')$; this will ensure that initially the points in $P(Q)$ are the vertices of all hypercubes in $\bar{\mathcal{C}}(Q')$ for each $Q' \subset Q$. Constructing $P(Q)$ might take more than polylogarithmic time since the sets $P(Q')$ with $Q' \in \mathrm{ch}(Q)$ might contain more than polylogarithmically many points. However, we show in the full version [23] how to adjust our hierarchical grid decomposition and the algorithm to obtain polylogarithmic update time.

We want to compute a set $\bar{\mathcal{C}}(Q) \subseteq \mathcal{C}'(Q)$ containing the hypercubes from $\mathcal{C}'(Q)$ that we add to the independent set. At the beginning, we initialize $\bar{\mathcal{C}}(Q) := \emptyset$. We compute an auxiliary grid $(Z^{(1)}, \ldots, Z^{(d)})$ in order to search for hypercubes to insert, similar to the unweighted case. To define this auxiliary grid, we first compute the total weight $\tilde{W} = w(P(Q))$ of all points that are in $P(Q)$ at the beginning of the algorithm in time $O(\log^{d-1} |P(Q)|)$, where we define $w(P) := \sum_{p \in P} w_p$ for any set of weighted points P. Then we define the auxiliary grid within Q such that *in the interior* of each *grid slice* the points in $P(Q)$ have a total weight at most $\varepsilon^{d+2}\tilde{W}/(d^{d+1} \log N)$, where a grid slice is a set of the form $\mathbb{R}^{d'-1} \times (x, y) \times \mathbb{R}^{d-d'}$ for some $d' \in [d]$. We emphasize here that this property only holds for the *interior* of the grid slices and that the sets in the first point of Lemma 5 are open.

▶ **Lemma 5.** *Given a cell Q and the data structure $P(Q)$, in $O\left(\left(\frac{d}{\varepsilon}\right)^{d+2} \cdot \log^d |P(Q)| \cdot \log N\right)$ time we can compute sets $Z^{(1)}, \ldots, Z^{(d)}$ of coordinates with $Z^{(d')} = \{z_1^{(d')}, z_2^{(d')}, \ldots\}$ for each d' such that*

- *the total weight of the points in $\left(\mathbb{R}^{d'-1} \times (z_j^{(d')}, z_{j+1}^{(d')}) \times \mathbb{R}^{d-d'}\right) \cap P(Q)$ is at most $\varepsilon^{d+2}\tilde{W}/(d^{d+1} \log N)$ for each $d' \in [d]$ and each $j \in \{1, \ldots, |Z^{(d')}| - 1\}$,*
- *$Q = \prod_{d'=1}^{d} [z_1^{(d')}, z_{|Z^{(d')}|}^{(d')}]$, and*
- *$|Z^{(d')}| \leq d^{d+1} \log N/\varepsilon^{d+2} + 1$ for each $d' \in [d]$.*

To select hypercubes to add to $\bar{\mathcal{C}}(Q)$ our algorithm runs in iterations, and in each iteration we add one hypercube to $\bar{\mathcal{C}}(Q)$. In each iteration we enumerate all hyperrectangles $A \subseteq Q$ that are aligned with $Z^{(1)}, \ldots, Z^{(d)}$; note that there are only $\left(d^{d+1} \log N/\varepsilon^{d+2} + 1\right)^{2d}$ such hyperrectangles (by the third point of Lemma 5). For each such hyperrectangle A we use the data structures $\{D'_k(Q)\}_{k \in \mathbb{N}}$ to determine whether there is a hypercube $C_i \subseteq A$ contained in $\mathcal{C}'_k(Q)$ for some k such that $(1+\varepsilon)^k \geq 2w(P(Q) \cap A)$ (recall that $D'_k(Q)$ maintains the intervals in $\mathcal{C}'$ which have weights in the range $[(1+\varepsilon)^k, (1+\varepsilon)^{k+1})$ and also recall that $D'_k(Q)$ is contained in the input $\mathcal{D}(Q)$ of the algorithm as discussed in Section 2). We say that such a hypercube C_i is *addible*. If there is no addible hypercube C_i then we stop and return $(\bar{\mathcal{C}}(Q), P(Q))$. Otherwise, we determine the smallest addible hypercube C_i (i.e., with minimum value s_i) and we add C_i to $\bar{\mathcal{C}}(Q)$. We add to $P(Q)$ the 2^d vertices of C_i with weight w_i; if a vertex of C_i has been in $P(Q)$ before then we increase its weight by w_i. We remove from $\bar{\mathcal{C}}(Q)$ all hypercubes that C_i overlaps with. Intuitively, we remove also all other previously selected hypercubes that C_i intersects; however, we do not do this explicitly since this might require $\Omega(n)$ time, but we will ensure this implicitly via the query algorithm that we use to output the solution and that we define below. Finally, we add the coordinates of C_i to the coordinates of the grid $Z^{(1)}, \ldots, Z^{(d)}$, i.e., we make the grid finer; formally, for each $d' \in [d]$ we add to $Z^{(d')}$ the coordinates $\{x_i^{(d')}, y_i^{(d')}\}$. This completes one iteration.

▶ **Lemma 6.** *The algorithm runs for at most $\left(\frac{d}{\varepsilon}\right)^d \log W$ iterations and computes $\bar{\mathcal{C}}(Q)$ in time $O\left(\left(\frac{d}{\varepsilon}\right)^{2d^2+5d+1} \cdot \log W \cdot \log^{2d-1} n \log^{2d} N\right)$.*

After the computation above, we define that our solution $\mathrm{SOL}(Q)$ for Q contains all hypercubes in a set $\bar{\mathcal{C}}(Q')$ for some cell $Q' \subseteq Q$ that are not overlapped by a hypercube in a set $\bar{\mathcal{C}}(Q'')$ for some cell $Q'' \supset Q'$. So if two hypercubes $C_{i'} \in \bar{\mathcal{C}}(Q')$, $C_{i''} \in \bar{\mathcal{C}}(Q'')$ overlap and $\ell(Q') < \ell(Q'')$, then we select $C_{i'}$ but not $C_{i''}$. We can output $\mathrm{SOL}(Q)$ in time $O_{d,\varepsilon}(|\mathrm{SOL}(Q)| \log^{d+1} N)$, see [23]. If we only want return the approximate weight of $\mathrm{SOL}(Q)$, we can return $w(P(Q))$ which is a $O(2^d)$-approximation by Lemma 7 below.

Finally, we bound our approximation ratio. Whenever we add a hypercube C_i to a set $\bar{\mathcal{C}}(Q)$ for some cell Q, then we explicitly or implicitly remove from our solution all hypercubes $C_{i'}$ with $C_i \cap C_{i'} \neq \emptyset$ such that $C_{i'} \in \bar{\mathcal{C}}(Q')$ for a cell $Q' \subseteq Q$. However, the total weight of

these removed hypercubes is bounded by $w_i/2$ since in the iteration in which we selected a hypercube $C_i \in \mathcal{C}_k$ there was a set $A \supseteq C_i$ with $(1+\varepsilon)^k \geq 2w(p(Q) \cap A)$ and by definition of $\mathcal{C}_k$, $w_i \geq (1+\varepsilon)^k$. Thus, we can bound our approximation ratio using a charging argument.

▶ **Lemma 7.** *For each cell $Q \in \mathcal{G}'$, we have that*

$$w(\mathrm{SOL}(Q)) \leq w(\mathrm{OPT}(Q)) \leq (2+O(\varepsilon))w(P(Q)) \leq (4+O(\varepsilon))2^d w(\mathrm{SOL}(Q))).$$

Before we prove Lemma 7, we prove three intermediate results. To this end, recall that $\mathrm{SOL}(Q)$ consists of hypercubes in sets $\bar{\mathcal{C}}(Q')$ for cells $Q' \subseteq Q$. First, we show via a token argument that the total weight of *all* hypercubes of the latter type is at most $2w(\mathrm{SOL}(Q))$, using that when we inserted a new hypercube in our solution then it overlapped with previously selected hypercubes of weight at most $w_i/2$.

▶ **Lemma 8.** *We have that $w(P(Q)) \leq 2^{d+1} w(\mathrm{SOL}(Q))$.*

Proof. We assign to each hypercube $C_i \in \mathrm{SOL}(Q)$ a budget of $2w_i$. We define now an operation that moves these budgets. Assume that a hypercube $C_{i'} \in \bar{\mathcal{C}}(Q')$ for some cell Q' now has a budget of $2w_i$ units. For each hypercube $C_{i''} \in \bar{\mathcal{C}}(Q'')$ for some cell $Q'' \subsetneq Q'$ such that one of the vertices of $C_{i''}$ is overlapped by $C_{i'}$, we move $2w_{i''}$ units of the budget of $C_{i'}$ to the budget of $C_{i''}$. Note that a hypercube $C_{i''} \in \mathcal{C}'(Q'')$ in a cell $Q'' \subsetneq Q'$ overlaps $C_{i'}$ if and only if $C_{i'}$ overlaps a vertex of $C_{i''}$ since $s_{i''} < s_{i'}$. When we selected $C_{i'} \in \mathcal{C}_k(Q')$ then there was a corresponding set $A \subseteq Q'$ such that $w_i \geq (1+\varepsilon)^k \geq 2w(p(Q') \cap A)$. Therefore, when we move the budget of $C_{i'}$ as defined then $C_{i'}$ keeps $w_{i'}$ units of its budget. After this operation, we say that $C_{i'}$ is *processed*. We continue with this operation until each hypercube $C_{i'}$ with a positive budget is processed. At the end, each hypercube $C_{i'}$ such that $C_{i'} \in \bar{\mathcal{C}}(Q')$ for some cell Q' has a budget of w_i. Therefore, $\sum_{Q' \subseteq Q} \sum_{C_i \in \bar{\mathcal{C}}(Q')} w_i \leq 2w(\mathrm{SOL}(Q))$.

Given the previous inequality and since we insert 2^d points for each $C_i \in \bar{C}(Q')$, we obtain that $w(P(Q)) = 2^d \cdot 2 \sum_{Q' \subseteq Q} \sum_{C_i \in \bar{\mathcal{C}}(Q')} w_i \leq 2^{d+1} w(\mathrm{SOL}(Q))$. ◀

We want to argue that $w(\mathrm{OPT}(Q)) \leq (4+O(\varepsilon)) \cdot 2^d w(\mathrm{SOL}(Q))$. To this end, we classify the hypercubes in $\mathrm{OPT}(Q)$. For each $C_i \in \mathrm{OPT}(Q)$ such that $C_i \in \mathcal{C}'(Q')$ for some grid cell $Q' \subseteq Q$ we say that C_i is *light* if $w_i \leq w(P(Q'))\varepsilon^{d+1}/(d^d \log N)$ and *heavy* otherwise (for the set $P(Q')$ when the algorithm finishes).

Next, we show that the total weight of light hypercubes is $\varepsilon w(P(Q))$. We do this by observing that since each cell $Q' \subseteq Q$ contains at most $(d/\varepsilon)^d$ light hypercubes in $\mathrm{OPT}(Q) \cap \mathcal{C}'(Q')$ (by Lemma 2), we can charge their weights to $w(P(Q))$.

▶ **Lemma 9.** *The total weight of light hypercubes is at most $\varepsilon w(P(Q))$.*

Proof. Let $Q' \subseteq Q$. For each light hypercube $C_i \in \mathcal{C}'(Q') \cap \mathrm{OPT}(Q)$ we charge $w_i \leq w(P(Q'))\varepsilon^{d+1}/(d^d \log N)$ to the points $p \in P(Q')$, proportionally to their respective weight w_p. There are at most $(d/\varepsilon)^d$ light hypercubes in $\mathcal{C}'(Q') \cap \mathrm{OPT}(Q)$ (by Lemma 2). Hence, the total charge is at most $w(P(Q')) \cdot \varepsilon / \log N$ and each point $p \in P(Q')$ receives a total charge of at most $w_p \cdot \varepsilon / \log N$ for Q'. Each point p is contained in at most $\log N$ sets in $\{P(Q')\}_{Q' \subseteq Q}$. Thus, the total weight of all light hypercubes in $\mathrm{OPT}(Q)$ is at most $\varepsilon w(P(Q))$. ◀

For the heavy hypercubes, we pretend that we increase the weight of each point in $P(Q)$ by a factor $2+O(\varepsilon)$. We show that then each heavy hypercube $C_i \in \mathrm{OPT}(Q)$ contains points in $P(Q)$ whose total weight is at least w_i. Hence, after increasing the weight, the weight of the points in $P(Q)$ "pays" for all heavy hypercubes in $\mathrm{OPT}(Q)$.

Let $\beta := 1/(\frac{1}{2+2\varepsilon} - 2\varepsilon) = 2 + O(\varepsilon)$. For each cell $Q' \subseteq Q$ and for each hypercube $C_i \in \bar{\mathcal{C}}(Q')$ we place a weight of βw_i essentially on each vertex of C_i. Since each hypercube C_i is an open set, C_i does not contain any of its vertices. Therefore, we place the weight βw_i not exactly on the vertices of C_i, but on the vertices of C_i slightly perturbed towards the center of C_i. Then, the weight we placed for the vertices of C_i contributes towards "paying" for C_i. Formally, for a small value $\delta > 0$ we place a weight of βw_i on each point of the form $(x_i^{(1)} + k^{(1)}s_i + \delta - k^{(1)} \cdot 2\delta, \ldots, x_i^{(d)} + k^{(d)}s_i + \delta - k^{(d)} \cdot 2\delta)$ with $k^{(d')} \in \{0, 1\}$ for each $d' \in [d]$. We choose δ such that any input hypercube $C_{i'}$ overlaps each point $(x_i^{(1)} + k^{(1)}s_i + \delta - k^{(1)} \cdot 2\delta+, \ldots, x_i^{(d)} + k^{(d)}s_i + \delta - k^{(d)} \cdot 2\delta)$ corresponding to C_i if and only if $C_i \cap C_{i'} \neq \emptyset$. We say that these points are the *charge points* of C_i. If on one of these points we already placed some weight then we increase its weight by βw_i. Let $\tilde{P}(Q')$ denote the points on which we placed a weight in the above procedure for Q' or for a cell $Q'' \subseteq Q'$. For each point $p \in \tilde{P}(Q)$ let $\tilde{w}_p$ denote the total weight that we placed on p in this procedure. Since for each point $p_i \in P(Q')$ with weight w_i we introduced a point $\tilde{p}_i \in \tilde{P}(Q')$ with weight $\tilde{w}_i \geq \beta w_i \geq w_i$, we have that $\sum_{p \in P(Q')} w_p \leq \sum_{p \in \tilde{P}(Q')} \tilde{w}_p$.

▶ **Lemma 10.** *The total weight of heavy hypercubes is at most* $(2 + O(\varepsilon))w(P(Q))$.

Proof. Let $C_i \in \mathrm{OPT}(Q)$ be a heavy hypercube. We claim that for each heavy hypercube $C_i \in \mathrm{OPT}(Q)$ it holds that $\sum_{p \in \tilde{P}(Q) \cap C_i} \tilde{w}_p \geq w_i$. This implies the claim since $(2 + O(\varepsilon)) \sum_{p \in P(Q) \cap C_i} w_p \geq \sum_{p \in \tilde{P}(Q) \cap C_i} \tilde{w}_p$.

Let Q' denote the cell such that $C_i \in \mathcal{C}'(Q')$. Let $\tilde{\mathcal{C}}(Q')$ denote the hypercubes that are in the set $\bar{\mathcal{C}}(Q')$ at some point while the algorithm processes the cell Q'. If $C_i \in \tilde{\mathcal{C}}(Q')$ then the claim is true since we placed a weight of βw_i on essentially each of its vertices (slightly perturbed by δ towards the center of C_i). Assume that $C_i \notin \tilde{\mathcal{C}}(Q')$. Let k be such that $C_i \in \mathcal{C}_k(Q')$. Consider the first iteration when we processed Q' such that we added a hypercube $C_{i'}$ with size $s_{i'} > s_i$ or the final iteration if no hypercube with size larger s_i is added. Let $A \subseteq Q'$ denote the smallest set that is aligned with the auxiliary grid $Z^{(1)}, \ldots, Z^{(d)}$ for the cell Q' such that $C_i \subseteq A$. If $(1+\varepsilon)^k \geq 2w(P(Q') \cap A)$ then in this iteration we would have added C_i instead of $C_{i'}$ which is a contradiction. If $(1+\varepsilon)^k < 2w(P(Q') \cap A)$ for the set $P(Q')$ at the beginning of this iteration then $w_i \leq (1+\varepsilon)^{k+1} < (1+\varepsilon)2w(P(Q') \cap A)$ and

$$\begin{aligned}
\sum_{p \in \tilde{P}(Q) \cap C_i} \tilde{w}_p &= \sum_{p \in \tilde{P}(Q) \cap C_i} \beta w_p \\
&\geq \beta \left(w(P(Q') \cap A) - 2d\varepsilon^{d+2}\tilde{W}/(d^{d+1} \log N) \right) \\
&\geq \beta \left(w(P(Q') \cap A) - 2\varepsilon^{d+2} w(P(Q'))/(d^d \log N) \right) \\
&\geq \beta \left(w(P(Q') \cap A) - 2\varepsilon w_i \right) \\
&\geq \beta \left(w_i/(2+2\varepsilon) - 2\varepsilon w_i \right) \\
&= w_i.
\end{aligned}$$

To see that the first inequality holds, note that $w(P(Q') \cap A) \leq w(P(Q') \cap C_i) + Y$, where Y is the weight of the points in the auxiliary grid slices of A which C_i does not fully overlap. Since A is the smallest aligned hyperrectangle containing C_i, in each dimension there are only two slices which are in A and which C_i partially overlaps (and none which are in A and do not overlap with C_i at all). Thus, there are at most $2d$ such slices in total. Using the definition of the auxiliary grid (Lemma 5), we obtain that $Y \leq 2d \cdot \varepsilon^{d+2}\tilde{W}/(d^{d+1} \log N)$, where $\tilde{W} = w(P(Q'))$. This provides the first inequality. The third inequality holds because C_i is heavy, the fourth inequality uses the above condition on w_i and the last equality is simply the definition of β. ◀

Proof of Lemma 7. We can bound the weight of all light and heavy hypercubes by $(2 + O(\varepsilon))w(P(Q))$ by Lemmas 9 and 10. Then applying Lemma 8 yields that

$$(2 + O(\varepsilon)) \sum_{p \in P(Q)} w_p \leq (2 + O(\varepsilon)) \cdot 2^{d+1} w(\mathrm{SOL}(Q))) = (4 + O(\varepsilon))2^d w(\mathrm{SOL}(Q))).$$

Thus, $w(\mathrm{SOL}(Q)) \leq w(\mathrm{OPT}(Q)) \leq (2 + O(\varepsilon))w(P(Q)) \leq (4 + O(\varepsilon))2^d w(\mathrm{SOL}(Q)))$. ◀

In [23] we describe how to adjust the hierarchical grid decomposition and the algorithm slightly such that we obtain polylogarithmic update time overall. We output the solution $\mathrm{SOL} := \mathrm{SOL}(Q^*)$. Using the data structure $P(Q^*)$, in time $O(1)$ we can also output $w(P(Q^*))$ which is an estimate for $w(\mathrm{SOL})$ due to Lemma 7. We obtain the following theorem.

▶ **Theorem 11.** *For the weighted maximum independent set of hypercubes problem with weights in* $[1, W]$ *there are fully dynamic algorithms that maintain* $(4+O(\varepsilon))2^d$*-approximate solutions deterministically with worst-case update time* $(d/\varepsilon)^{O(d^2+1/\varepsilon)} \cdot \log W \cdot \log^{2d-1} n \log^{2d+1} N$ *and with high probability with worst-case update time* $\left(\frac{d}{\varepsilon}\right)^{O(d^2)} \cdot \log W \cdot \log^{2d-1} n \log^{2d+2} N$.

References

1. Anna Adamaszek, Sariel Har-Peled, and Andreas Wiese. Approximation schemes for independent set and sparse subsets of polygons. *J. Assoc. Comput. Mach.*, 66(4):29:1–29:40, June 2019. doi:10.1145/3326122.
2. Pankaj K. Agarwal, Marc J. van Kreveld, and Subhash Suri. Label placement by maximum independent set in rectangles. *Comput. Geom.*, 11(3-4):209–218, 1998.
3. Sanjeev Arora. Polynomial time approximation schemes for euclidean tsp and other geometric problems. In *FOCS*, pages 2–11, 1996.
4. Sepehr Assadi, Krzysztof Onak, Baruch Schieber, and Shay Solomon. Fully dynamic maximal independent set with sublinear update time. In *STOC*, pages 815–826, 2018.
5. Sepehr Assadi, Krzysztof Onak, Baruch Schieber, and Shay Solomon. Fully dynamic maximal independent set with sublinear in n update time. In *SODA*, pages 1919–1936. SIAM, 2019.
6. Ainesh Bakshi, Nadiia Chepurko, and David P. Woodruff. Weighted maximum independent set of geometric objects in turnstile streams. *CoRR*, abs/1902.10328, 2019. arXiv:1902.10328.
7. Soheil Behnezhad, Mahsa Derakhshan, Mohammad Taghi Hajiaghayi, Cliff Stein, and Madhu Sudan. Fully dynamic maximal independent set with polylogarithmic update time. In *FOCS*, 2019.
8. Sergio Cabello and Pablo Pérez-Lantero. Interval selection in the streaming model. *Theor. Comput. Sci.*, 702:77–96, 2017. doi:10.1016/j.tcs.2017.08.015.
9. Parinya Chalermsook and Julia Chuzhoy. Maximum independent set of rectangles. In *SODA*, pages 892–901, 2009.
10. Timothy M. Chan. Polynomial-time approximation schemes for packing and piercing fat objects. *J. Algorithms*, 46(2):178–189, 2003.
11. Timothy M Chan. A note on maximum independent sets in rectangle intersection graphs. *Information Processing Letters*, 89(1):19–23, 2004.
12. Timothy M. Chan and Sariel Har-Peled. Approximation algorithms for maximum independent set of pseudo-disks. *Discrete & Computational Geometry*, 48(2):373–392, September 2012. doi:10.1007/s00454-012-9417-5.
13. Shiri Chechik and Tianyi Zhang. Fully dynamic maximal independent set in expected poly-log update time. In *FOCS*, 2019.
14. Julia Chuzhoy and Alina Ene. On approximating maximum independent set of rectangles. In *FOCS*, pages 820–829, 2016.
15. Yuhao Du and Hengjie Zhang. Improved algorithms for fully dynamic maximal independent set. *CoRR*, abs/1804.08908, 2018. arXiv:1804.08908.

16 Herbert Edelsbrunner. A note on dynamic range searching. *Bull. EATCS*, 15(34-40):120, 1981.

17 Yuval Emek, Magnús M. Halldórsson, and Adi Rosén. Space-constrained interval selection. *ACM Trans. Algorithms*, 12(4):51:1–51:32, 2016.

18 Thomas Erlebach, Klaus Jansen, and Eike Seidel. Polynomial-time approximation schemes for geometric intersection graphs. *SIAM Journal on Computing*, 34(6):1302–1323, 2005.

19 Robert J. Fowler, Michael S. Paterson, and Steven L. Tanimoto. Optimal packing and covering in the plane are np-complete. *Information Processing Letters*, 12(3):133–137, 1981. doi:10.1016/0020-0190(81)90111-3.

20 András Frank. Some polynomial algorithms for certain graphs and hypergraphs. In *Proceedings of the 5th British Combinatorial Conference*. Utilitas Mathematica, 1975.

21 Alexander Gavruskin, Bakhadyr Khoussainov, Mikhail Kokho, and Jiamou Liu. Dynamic algorithms for monotonic interval scheduling problem. *Theor. Comput. Sci.*, 562:227–242, 2015.

22 Manoj Gupta and Shahbaz Khan. Simple dynamic algorithms for maximal independent set and other problems. *CoRR*, abs/1804.01823, 2018. arXiv:1804.01823.

23 Monika Henzinger, Stefan Neumann, and Andreas Wiese. Dynamic approximate maximum independent set of intervals, hypercubes and hyperrectangles. *CoRR*, abs/2003.02605, 2020. arXiv:2003.02605.

24 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.

25 Sanjeev Khanna, S. Muthukrishnan, and Mike Paterson. On approximating rectangle tiling and packing. In *SODA*, pages 384–393, 1998.

26 Jon M. Kleinberg and Éva Tardos. *Algorithm Design*. Addison-Wesley, 2006.

27 D. T. Lee and Franco P. Preparata. Computational geometry - A survey. *IEEE Trans. Computers*, 33(12):1072–1101, 1984.

28 Morteza Monemizadeh. Dynamic maximal independent set. *CoRR*, abs/1906.09595, 2019. arXiv:1906.09595.

29 Bram Verweij and Karen Aardal. An optimisation algorithm for maximum independent set with applications in map labelling. In *ESA*, pages 426–437, 1999.

30 Dan E. Willard and George S. Lueker. Adding range restriction capability to dynamic data structures. *J. ACM*, 32(3):597–617, 1985.

31 D. Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. *Theory of Computing*, 3:103–128, 2007.

How to Find a Point in the Convex Hull Privately

Haim Kaplan
School of Computer Science, Tel Aviv University, Israel
Google, Tel Aviv, Israel
haimk@tau.ac.il

Micha Sharir
School of Computer Science, Tel Aviv University, Israel
michas@tau.ac.il

Uri Stemmer
Department of Computer Science, Ben-Gurion University, Beer Sheva, Israel
Google, Tel Aviv, Israel
u@uri.co.il

Abstract

We study the question of how to compute a point in the convex hull of an input set S of n points in $\mathbb{R}^d$ in a differentially private manner. This question, which is trivial without privacy requirements, turns out to be quite deep when imposing differential privacy. In particular, it is known that the input points must reside on a fixed *finite* subset $G \subseteq \mathbb{R}^d$, and furthermore, the size of S must grow with the size of G. Previous works [1, 2, 3, 4, 5, 11] focused on understanding how n needs to grow with $|G|$, and showed that $n = O\left(d^{2.5} \cdot 8^{\log^* |G|}\right)$ suffices (so n does not have to grow significantly with $|G|$). However, the available constructions exhibit running time at least $|G|^{d^2}$, where typically $|G| = X^d$ for some (large) discretization parameter X, so the running time is in fact $\Omega(X^{d^3})$.

In this paper we give a differentially private algorithm that runs in $O(n^d)$ time, assuming that $n = \Omega(d^4 \log X)$. To get this result we study and exploit some structural properties of the Tukey levels (the regions $D_{\geq k}$ consisting of points whose Tukey depth is at least k, for $k = 0, 1, \ldots$). In particular, we derive lower bounds on their volumes for point sets S in general position, and develop a rather subtle mechanism for handling point sets S in degenerate position (where the deep Tukey regions have zero volume). A naive approach to the construction of the Tukey regions requires $n^{O(d^2)}$ time. To reduce the cost to $O(n^d)$, we use an approximation scheme for estimating the volumes of the Tukey regions (within their affine spans in case of degeneracy), and for sampling a point from such a region, a scheme that is based on the volume estimation framework of Lovász and Vempala [14] and of Cousins and Vempala [7]. Making this framework differentially private raises a set of technical challenges that we address.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms; Theory of computation → Randomness, geometry and discrete structures; Security and privacy → Formal methods and theory of security

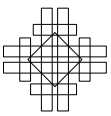
Keywords and phrases Differential privacy, Tukey depth, Convex hull

Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.52

Related Version A full version of the paper is available at `http://arxiv.org/abs/2003.13192`.

Funding *Haim Kaplan*: Partially supported by ISF grant 1595/19 and grant 1367/2016 from the German-Israeli Science Foundation (GIF)
Micha Sharir: Partially supported by ISF Grant 260/18, by grant 1367/2016 from the German-Israeli Science Foundation (GIF), and by Blavatnik Research Fund in Computer Science at Tel Aviv University.
Uri Stemmer: Partially supported by ISF grant 1871/19.

Acknowledgements We thank Santosh Vempala for many helpful discussions.



1 Introduction

We often would like to analyze data while protecting the privacy of the individuals that contributed to it. At first glance, one might hope to ensure privacy by simply deleting all names and ID numbers from the data. However, such anonymization schemes are proven time and again to violate privacy. This gave rise of a theoretically-rigorous line of work that has placed private data analysis on firm foundations, centered around a mathematical definition for privacy known as *differential privacy* [8].

Consider a database S containing personal information of individuals. Informally, an algorithm operating on such a database is said to preserve differential privacy if its outcome distribution is (almost) insensitive to any arbitrary change to the data of one individual in the database. Intuitively, this means that an observer looking at the outcome of the algorithm (almost) cannot distinguish between whether Alice's information is x or y (or whether Alice's information is present in the database at all) because in any case it would have (almost) no effect on the outcome distribution of the algorithm.

▶ **Definition 1.1** (Dwork et al. [8]). *Two databases (multisets) S and S' are called* neighboring *if they differ in a single entry. That is, $S = S_0 \cup \{x\}$ and $S' = S_0 \cup \{y\}$ for some items x and y. A randomized algorithm A is* (ε, δ)-differentially private *if for every two neighboring databases S, S' and for any event T we have*

$$\boldsymbol{Pr}[A(S) \in T] \leq e^{\varepsilon} \cdot \boldsymbol{Pr}[A(S') \in T] + \delta.$$

When $\delta = 0$ this notion is referred to as pure *differential privacy, and when $\delta > 0$ it is referred to as* approximate *differential privacy.*

▶ Remark 1.2. Typically, ε is set to be a small constant, say $\varepsilon = 0.1$, and δ is set to be a small function of the database size $|S|$ (much smaller than $1/|S|$). Note that to satisfy the definition (in any meaningful way) algorithm A must be randomized.

Differential privacy is increasingly accepted as a standard for rigorous treatment of privacy. However, even though the field has witnessed an explosion of research in the recent years, much remains unknown and answers to fundamental questions are still missing. In this work we study one such fundamental question, already studied in [1, 2, 3, 4, 5, 11]: Given a database containing points in $\mathbb{R}^d$ (where every point is assumed to be the information of one individual), how can we *privately* identify a point in the *convex hull* of the input points? This question, which is trivial without privacy requirements, turns out to be quite deep when imposing differential privacy. In particular, Bun et al. [5] showed that in order to be able to solve it, we must assume that the input points reside on a fixed *finite* subset $G \subseteq \mathbb{R}^d$, and furthermore, the number of input points must grow with the size of G.

The Private Interior Point (PIP) Problem.
Let $\beta, \varepsilon, \delta, X$ be positive parameters where $\beta, \varepsilon, \delta$ are small and X is a large integer. Let $G \subseteq [0,1]^d$ be a finite uniform grid with side steps $1/X$ (so $|G| = (X+1)^d$). Design an algorithm A such that for some $n \in \mathbb{N}$ (as small as possible as a function of $\beta, \varepsilon, \delta, X$) we have

1. **Utility:** For every database S containing at least n points from G it holds that $A(S)$ returns a point in the convex hull of S with probability at least $1 - \beta$. (The outcome of A does not have to be in G.)
2. **Privacy:** For every pair of neighboring databases S, S', each containing at least n points from G, and for any event T, we have $\mathbf{Pr}[A(S) \in T] \leq e^{\varepsilon} \cdot \mathbf{Pr}[A(S') \in T] + \delta$.

The parameter n is referred to as *the sample complexity* of the algorithm. It is the smallest number of points on which we are guaranteed to succeed (not to be confused with the actual size of the input). The PIP problem is very natural on its own. Furthermore, as was observed in [2], an algorithm for solving the PIP problem can be used as a building block in other applications with differential privacy, such as learning halfspaces and linear regression. Previous works [1, 2, 3, 4, 5, 11] have focused on the task of minimizing the sample complexity n while ignoring the runtime of the algorithm. In this work we seek an efficient algorithm for the PIP problem, that still keeps the sample complexity n "reasonably small" (where "reasonably small" will be made precise after we introduce some additional notation).

1.1 Previous Work

Several papers studied the PIP problem for $d = 1$. In particular, three different constructions with sample complexity $2^{O(\log^* |G|)}$ were presented in [3, 4, 5] (for $d = 1$). Recently, Kaplan et al. [11] presented a new construction with sample complexity $O((\log^* |G|)^{1.5})$ (again, for $d = 1$). Bun et al. [5] gave a lower bound showing that every differentially private algorithm for this task must have sample complexity $\Omega(\log^* |G|)$. Beimel et al. [2] incorporated a dependency in d to this lower bound, and showed that every differentially private algorithm for the PIP problem must use at least $n = \Omega(d + \log^* |G|)$ input points.

For the case of *pure* differential privacy (i.e., $\delta = 0$), a lower bound of $n = \Omega(\log X)$ on the sample complexity follows from the results of Beimel et al. [1]. This lower bound is tight, as an algorithm with sample complexity $n = O(\log X)$ (for $d = 1$) can be obtained using a generic tool in the literature of differential privacy, called the *exponential mechanism* [16]. We sketch this application of the exponential mechanism here. (For a precise presentation of the exponential mechanism see Section 2.1.) Let $G = \{0, \frac{1}{X}, \frac{2}{X}, \ldots, 1\}$ be our (1-dimensional) grid within the interval $[0, 1]$, and let S be a multiset containing n points from G. The algorithm is as follows.

1. For every $y \in G$ define the *score* $q_S(y) = \min\left\{|\{x \in S \mid x \geq y\}|,\ |\{x \in S \mid x \leq y\}|\right\}$.
2. Output $y \in G$ with probability proportional to $e^{\varepsilon \cdot q_S(y)}$.

Intuitively, this algorithm satisfies differential privacy because changing one element of S changes the score $q_S(y)$ by at most ± 1, and thus changes the probabilities with which we sample elements by roughly an e^{ε} factor. As for the utility analysis, observe that $\exists y \in G$ with $q_S(y) \geq \frac{n}{2}$, and the probability of picking this point is (at least) proportional to $e^{\varepsilon n/2}$. As this probability increases exponentially with n, by setting n to be big enough we can ensure that points y' outside of the convex hull (those with $q_S(y') = 0$) get picked with very low probability.

Beimel et al. [2] observed that this algorithm extends to higher dimensions by replacing $q_S(y)$ with the Tukey depth $\mathsf{td}_S(y)$ of the point y with respect to the input set S (the *Tukey depth* of a point y is the minimal number of points that need to be removed from S to ensure that y is *not* in the convex hull of the remaining input points). However, even though there must exist a point $y \in \mathbb{R}^d$ with high Tukey depth (at least $n/(d+1)$; see [15]), the finite grid $G \subseteq \mathbb{R}^d$ might fail to contain such a point. Hence, Beimel et al. [2] first *refined* the grid G into a grid G' that contains a point with high Tukey depth, and then randomly picked a point y from G' with probability proportional to $e^{\varepsilon \cdot \mathsf{td}_S(y)}$. To compute the probabilities with which grid points are sampled, the algorithm in [2] explicitly computes the Tukey depth of every point in G', which, because of the way in which G' is defined, results in running time of at least $\Omega(|G|^{d^2}) = \Omega(X^{d^3})$ and sample complexity $n = O(d^3 \log |G|) = O(d^4 \log X)$. Beimel et al. then presented an improvement of this algorithm with reduced sample complexity of $n = O(d^{2.5} \cdot 8^{\log^* |G|})$, but the running time remained $\Omega(|G|^{d^2}) = \Omega(X^{d^3})$.

1.2 Our Construction

We give an approximate differentially private algorithm for the private interior point problem that runs in $O(n^d)$ time,[1] and succeeds with high probability when the size of its input is $\Omega(\frac{d^4}{\varepsilon}\log\frac{X}{\delta})$. Our algorithm is obtained by carefully implementing the exponential mechanism and reducing its running time from $\Omega(|G|^{d^2}) = \Omega(X^{d^3})$ to $O(n^d)$. We now give an informal overview of this result.

To avoid the need to extend the grid and to calculate the Tukey depth of each point in the extended grid, we sample our output directly from $[0,1]^d$. To compute the probabilities with which we sample a point from $[0,1]^d$ we compute, for each k in an appropriate range, the *volume* of the Tukey region of depth k, which we denote as D_k. (That is, D_k is the region in $[0,1]^d$ containing all points with Tukey depth exactly k.) We then sample a value $k \in [n]$ with probability proportional to $\mathrm{Vol}(D_k)\cdot e^{\varepsilon k}$, and then sample a random point uniformly from D_k.

Observe that this strategy picks a point with Tukey depth k with probability proportional to $\mathrm{Vol}(D_k)\cdot e^{\varepsilon k}$. Hence, if for a "large enough" value of k (say $k \geq \frac{n}{cd}$ for a suitable absolute constant $c > 1$) we have that $\mathrm{Vol}(D_k)$ is "large enough", then a point with Tukey depth k is picked with high probability. However, if $\mathrm{Vol}(D_k) = 0$ (or too small) then a point with Tukey depth k is picked with probability zero (or with too small a probability). Therefore, to apply this strategy, we derive a lower bound on the volume of every non-degenerate Tukey region, showing that if the volume is non-zero, then it is at least $\Omega\left(1/X^{d^3}\right)$.

There are two issues here. The first issue is that the best bound we know on the complexity of a Tukey region is $O(n^{(d-1)\lfloor d/2\rfloor})$, so we cannot compute these regions explicitly (in the worst-case) in time $O(n^d)$ (which is our target runtime complexity). We avoid the need to compute the Tukey regions explicitly by using an approximation scheme for estimating the volume of each region and for sampling a point from such a region, a scheme that is based on the volume estimation framework of Lovász and Vempala [14] and of Cousins and Vempala [7]. The second issue is that it might be the case that all Tukey regions for large values of k are degenerate, i.e., have volume 0, in which case this strategy might fail to identify a point in the convex hull of the input points.

Handling degeneracies. We show that if the Tukey regions of high depth are degenerate, then many of the input points must lie in a lower-dimensional affine subspace. This can be used to handle degenerate inputs S as follows. We first (privately) check whether there exists an affine proper subspace that contains many points of S. If we find such a subspace f, we recursively continue the procedure within f, with respect to $S \cap f$. Otherwise, if no such subspace exists, then it must be the case that the Tukey regions of high depth are full-dimensional (with respect to the subspace into which we have recursed so far), so we can apply our algorithm for the non-degenerate case and obtain a point that lies, with high probability, in the convex hull of the surviving subset of S, and thus of the full set S.

We remark that it is easy to construct algorithms with running time polynomial in the input size n, when n grows exponentially in d. (In that case one can solve the problem using a reduction to the 1-dimensional case.) In this work we aim to reduce the running time while keeping the sample complexity n at most polynomial in d and in $\log|G|$.

[1] When we use O-notation for time complexity we hide logarithmic factors in X, $1/\varepsilon$, $1/\delta$, and polynomial factors in d. We assume operations on real numbers in $O(1)$ time (the so-called real RAM model).

2 Preliminaries

We assume that our input set S consists of n points that lie in the intersection of a grid G with the cube $[0,1]^d$ in $\mathbb{R}^d$. We assume that G is of side length $1/X$ for a given accuracy integer parameter X, so it partitions the cube into X^d cells.

2.1 The exponential mechanism

Let G^* denote the set of all finite databases over a grid G, and let F be a finite set. Given a database $S \in G^*$, a *quality* (or *scoring*) function $q : G^* \times F \to \mathbb{N}$ assigns a number $q(S,f)$ to each element $(S,f) \in G^* \times F$, identified as the "quality" of f with respect to S. We say that the function q has *sensitivity* Δ if for all neighboring databases S and S' and for all $f \in F$ we have $|q(S,f) - q(S',f)| \leq \Delta$.

The *exponential mechanism* of McSherry and Talwar [16] privately identifies an element $f \in F$ with large quality $q(S,f)$. Specifically, it chooses an element $f \in F$ randomly, with probability proportional to $\exp(\varepsilon \cdot q(S,f)/(2\Delta))$. The privacy and utility of the mechanism are:

▶ **Theorem 2.1** (McSherry and Talwar [16]). *The exponential mechanism is $(\varepsilon, 0)$-differentially private. Let q be a quality function with sensitivity Δ. Fix a database $S \in G^*$ and let $\mathrm{OPT} = \max_{f \in F}\{q(S,f)\}$. For any $\beta \in (0,1)$, with probability at least $(1-\beta)$, the exponential mechanism outputs a solution f with quality $q(S,f) \geq \mathrm{OPT} - \frac{2\Delta}{\varepsilon} \ln \frac{|F|}{\beta}$.*

2.2 Tukey depth

The *Tukey depth* [18] $\mathsf{td}_S(q)$ of a point q with respect to S is the minimum number of points of S we need to remove to make q lie outside the convex hull of the remaining subset. Equivalently, $\mathsf{td}_S(q)$ is the smallest number of points of S that lie in a closed halfspace containing q. We will write $\mathsf{td}(q)$ for $\mathsf{td}_S(q)$ when the set S is clear from the context. It easily follows from Helly's theorem that there is always a point of Tukey depth at least $n/(d+1)$ (see, e.g., [15]). We denote the largest Tukey depth of a point by $\mathsf{td}_{\max}(S)$ (the maximum Tukey depth is always at most $n/2$).

We define the regions $D_{\geq k}(S) = \left\{q \in [0,1]^d \mid \mathsf{td}_S(q \geq k\right\}$ and $D_k(S) = D_{\geq k}(S) \setminus D_{\geq k+1}(S)$ for $k = 0, \ldots, \mathsf{td}_{\max}(S)$. Note that $D_{\geq 1}$ is the convex hull of S and that $D_{\geq 0} = [0,1]^d$. It is easy to show that $D_{\geq k}$ is convex; $D_{\geq k}$ is in fact the intersection of all (closed) halfspaces containing at least $n-k+1$ points of S; see [17]. It is easy to show that all this is true also when S is degenerate. See Figure 1 for an illustration. The following lemma is easy to establish.

▶ **Lemma 2.2.** *If $D_{\geq k}$ is of dimension d (we refer to such a region as non-degenerate) then $C_k = \partial D_{\geq k}(S)$ is a convex polytope, each of whose facets is contained in a simplex σ spanned by d points of S, such that one of the open halfspaces bounded by the hyperplane supporting σ contains exactly $k-1$ points of S.*

3 The case of general position

As already said, we apply the exponential mechanism for privately identifying a point in $[0,1]^d$ with (hopefully) large Tukey depth with respect to the input set S. This satisfies (pure) differential privacy since the sensitivity of the Tukey depth is 1. In this section we show that when the input points are in general position, then this application of the exponential mechanism succeeds (with high probability) in identifying a point that has positive Tukey depth, that is, a point inside the convex hull of S.

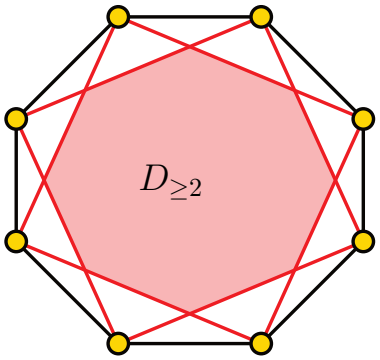


Figure 1 The Tukey layers $D_{\geq 2}$ and $D_{\geq 1}$.

To implement the exponential mechanism (i.e., to sample a point from $[0,1]^d$ appropriately), we need to compute the Tukey regions $D_{\geq k}$ and their volumes. In this section we compute these regions explicitly in $O\left(n^{1+(d-1)\lfloor d/2 \rfloor}\right)$ time. In Section 5 we will show that the cost can be reduced to $O(n^d)$.

Computing $D_{\geq k}$. We pass to the dual space, and construct the arrangement $\mathcal{A}(S^*)$ of the hyperplanes dual to the points of S. A point h^* dual to a hyperplane h supporting $D_{\geq k}$ has at least $n-k+1$ dual hypeplanes passing below h^* or incident to h^*, or, alternatively, passing above h^* or incident to h^*. Furthermore, if we move h^* slightly down in the first case (resp., up in the second case), the number of hypeplanes below (resp., above) it becomes smaller than $n-k+1$.

When h^* is a vertex of $\mathcal{A}(S^*)$ we refer to it as *k-critical*, or simply as *critical*. If $D_{\geq k}$ is non-degenerate then, by Lemma 2.2, each hyperplane h that supports a facet of $D_{\geq k}$ must be spanned by d affinely independent points of S. That is, all these hyperplanes are dual to k-critical vertices of $\mathcal{A}(S^*)$.

We compute all critical dual vertices that have at least $n-k+1$ dual hyperplanes passing below them or incident to them, or those with at least $n-k+1$ dual hyperplanes passing above them or incident to them. The intersection of the appropriate primal halfspaces that are bounded by the hyperplanes corresponding to these dual vertices is $D_{\geq k}$. This gives an algorithm for constructing $D_{\geq k}$ when it is non-degenerate. Otherwise $D_{\geq k}$ is degenerate and its volume is 0, and we need additional techniques, detailed in the next subsection, to handle such situations.

We compute the volume of each non-degenerate $D_{\geq k}$, for $k = 1, \ldots, \mathsf{td}_{\max}(S)$. We do that in brute force, by computing and triangulating $D_{\geq k}$ and adding up the volumes of the simplices in this triangulation. Then we subtract the volume of $D_{\geq k+1}$ from the volume of $D_{\geq k}$ to get the volume of D_k.

The sampling mechanism. We assign to each D_k the weight $e^{\varepsilon k/2}$ and sample a region D_k, for $k = 0, \ldots, \mathsf{td}_{\max}(S)$, with probability

$$\mu_k = \frac{e^{\varepsilon k/2}\mathrm{Vol}(D_k)}{\sum_{j\geq 0} e^{\varepsilon j/2}\mathrm{Vol}(D_j)},$$

where $\mathrm{Vol}(D_k)$ denotes the volume of D_k. Then we sample a point uniformly at random from D_k. We do this in brute force by computing D_k, triangulating it, computing the volume of each simplex, drawing a simplex from this triangulation with probability proportional to its volume, and then drawing a point uniformly at random from the chosen simplex.[2]

This is an instance of the exponential mechanism in which the score (namely the Tukey depth) has sensitivity 1, i.e., $|\mathsf{td}_S(q) - \mathsf{td}_{S'}(q)| \leq 1$ for any point $q \in [0,1]^d$, when S and S' differ by only one element. It thus follows from the properties of the exponential mechanism (Theorem 2.1) that this procedure is (purely) ε-differentially private.

Complexity. Computing the dual arrangement $\mathcal{A}(S^*)$ takes $O(n^d)$ time [10]. Assume that $D_{\geq k}$ is non-degenerate and let M_k denote the number of hyperplanes defining $D_{\geq k}$ (i.e., the hyperplanes supporting its facets). It takes $O(M_k^{\lfloor d/2 \rfloor})$ time to construct $D_{\geq k}$, as the intersection of M_k halfspaces, which is a dual version of constructing the convex hull (see [6]). Within the same asymptotic bound we can triangulate $D_{\geq k}$ and compute its volume. We obviously have $M_k = O(n^d)$, but the number can be somewhat reduced. The following lemma is known.

▶ **Lemma 3.1** (Proposition 3 in [13]). *The number of halfspaces needed to construct $D_{\geq k}$ is $O(n^{d-1})$.*

Proof. Fix a $(d-1)$-tuple σ of points of S, and consider all the relevant (closed) halfspaces, each of which is bounded by a hyperplane that is spanned by σ and another point of S, and contains at least $n-k+1$ points of S. It is easy to check that, as long as the intersection of these halfspaces is full-dimensional, it is equal to the intersection of just two of them. ◀

Summing up, we get that computing the volume of all the non-degenerate regions $D_{\geq k}$, for $k = 1, \ldots, \mathsf{td}_{\max}(S)$, takes $O\left(\sum_{k \geq 1} M_k^{\lfloor d/2 \rfloor}\right) = O\left(n^{1+(d-1)\lfloor d/2 \rfloor}\right)$ time.

Utility. We continue to assume that $D_{\geq k}$ is non-degenerate, and give a lower bound on its volume. By Lemma 2.2, such a $D_{\geq k}$ is the intersection of halfspaces, each bounded by a hyperplane that is spanned by d points of S. Denote by H the set of these hyperplanes. To obtain the lower bound, it suffices to consider the case where $D_{\geq k}$ is a simplex, each of whose vertices is the intersection point of d hyperplanes of H.

The equation of a hyperplane h that passes through d points, $a_1, \ldots, a_d$, of S is

$$\begin{vmatrix} 1 & x_1 & \cdots & x_d \\ 1 & a_{1,1} & \cdots & a_{1,d} \\ & & \vdots & \\ 1 & a_{d,1} & \cdots & a_{d,d} \end{vmatrix} = 0,$$

where $a_i = (a_{i,1}, \ldots, a_{i,d})$, for $i = 1, \ldots, d$. The coefficients of the x_i's in the equation of h are $d \times d$ subdeterminants of this determinant, where each determinant has one column of 1's, and $d-1$ other columns, each of whose entries is some $a_{i,j}$, which is a rational of the form m/X, with $0 \leq m \leq X$ (the same holds for the 1's, with $m = X$). The value of such a determinat (coefficient) is a rational number with denominator X^d. By Hadamard's

[2] A simple way to implement the last step is to draw uniformly and independently d points from $[0,1]$, compute the lengths $\lambda_1, \ldots, \lambda_{d+1}$ of the intervals into which they partition $[0,1]$, and return $\sum_{i=1}^{d+1} \lambda_i v_i$, where $v_1, \ldots, v_{d+1}$ are the vertices of the simplex.

inequality, its absolute value is at most the product of the Euclidean norms of its rows, which is at most $d^{d/2}$. That is, the numerator of the determinant is an integer of absolute value at most $d^{d/2}X^d$. The free term is a similar sub-determinant, but all its entries are the $a_{i,j}$'s, so it too is a rational with denominator X^d, and with numerator of absolute value at most $d^{d/2}X^d$. Multiplying by X^d, all the coefficients become integers of absolute value at most $d^{d/2}X^d$.

Each vertex of any region D_k of Tukey depth k, for any k, is a solution of a linear system of d hyperplane equations of the above form. It is therefore a rational number whose denominator, by Cramer's rule, is the determinant of all non-free coefficients of the d hyperplanes. This is an integer whose absolute value, again by Hadamard's inequality, is at most

$$\left(\sqrt{d}d^{d/2}X^d\right)^d \leq d^{d(d+1)/2}X^{d^2}.$$

Since the free terms are also integers, we conclude that the coordinates of the intersection point are rationals with a common integral denominator of absolute value at most $d^{d(d+1)/2}X^{d^2}$.

We can finally obtain a lower bound for the nonzero volume of a simplex spanned by any $d+1$ linearly independent intersection points $v_1, \ldots, v_{d+1}$. This volume is

$$\frac{1}{d!}\begin{vmatrix} 1 & v_{1,1} & \cdots & v_{1,d} \\ 1 & v_{2,1} & \cdots & v_{2,d} \\ & & \vdots & \\ 1 & v_{d+1,1} & \cdots & v_{d+1,d} \end{vmatrix},$$

where again $v_i = (v_{i,1}, \ldots, v_{i,d})$, for $i = 1, \ldots, d+1$. Note that all the entries in any fixed row have the same denominator. The volume is therefore a rational number whose denominator is $d!$ times the product of these denominators, which is thus an integer with absolute value at most

$$d! \cdot \left(d^{d(d+1)/2}X^{d^2}\right)^d \leq (dX)^{d^3}$$

(for $d \geq 2$). That is, we get the following lemma.

▶ **Lemma 3.2.** *If the volume of $D_{\geq k}$ is not zero then it is at least $1/(dX)^{d^3}$.*

Assume that the volume of $D_{\geq k}$ is not zero for $k = k_0 := n/(4d)$. Since the score of a point outside the convex hull is zero and the volume of $D_{\geq 0}$ is at most 1, we get that the probability to sample a point outside of the convex hull is at most

$$\frac{1}{e^{\varepsilon k_0}\mathrm{Vol}(D_{k_0})} \leq \frac{(dX)^{d^3}}{e^{\varepsilon n/(4d)}}.$$

This inequality leads to the following theorem, which summarizes the utility that our instance of the exponential mechanism provides.

▶ **Theorem 3.3.** *If $n \geq \dfrac{4d^4\log(dX)}{\varepsilon} + \dfrac{4d}{\varepsilon}\log\dfrac{1}{\beta}$ and $D_{\geq n/4d}$ has non-zero volume then the exponential mechanism, implemented as above, returns a point in the convex hull with probability at least $1-\beta$, in $O\left(n^{1+(d-1)\lfloor d/2\rfloor}\right)$ time.*

4 Handling degeneracies

In general we have no guarantee that $D_{\geq n/4d}$ has non-zero volume. In this section we show how to overcome this and compute (with high probability) a point in the convex hull of any input. We rely on the following lemma, which shows that if $D_{\geq k}$ has zero volume then many points of S are in a lower-dimensional affine subspace.

▶ **Lemma 4.1.** *If $D_{\geq k}$ spans an affine subspace f of dimension j then*

$$|S \cap f| \geq n - (d - j + 1)(k - 1).$$

Proof. Recall that $D_{\geq k}$ is the intersection of all closed halfspaces h that contain at least $n-k+1$ points of S. Note that a halfspace that bounds $D_{\geq k}$ and whose bounding hyperplane properly crosses f, defines a proper halfspace within f, and, by assumption, the intersection of these halfspaces has positive relative volume. This means that the intersection of these halfspaces in $\mathbb{R}^d$ has positive volume too, and thus cannot confine $D_{\geq k}$ to f. To get this confinement, there must exist (at least) $d - j + 1$ halfspaces in the above collection, whose intersection is f. Hence the union of their complements is $\mathbb{R}^d \setminus f$. Since this union contains at most $(d - j + 1)(k - 1)$ points of S, the claim follows. ◀

In what follows, to simplify the expressions that we manipulate, we use the weaker lower bound $n-(d-j+1)k$. In order for the lemma to be meaningful, we want k to be significantly smaller than the centerpoint bound $n/(d+1)$, so we set, as above, $k = n/(4d)$.

We use Lemma 4.1 to handle degenerate inputs S, using the following high-level approach. We first (privately) check whether there exists an affine proper subspace that contains many points of S. If we find such a subspace f, we recursively continue the procedure within f, with respect to $S \cap f$. Lemma 4.1 then implies that we do not lose too many points when we recurse within f (that is, $|S \cap f|$ is large), using our choice $k = n/(4d)$. Otherwise, if no such subspace exists, Lemma 4.1 implies that $D_{\geq k}$ is full-dimensional (with respect to the subspace into which we have recursed so far), so we can apply the exponential mechanism, as implemented in Section 3, and obtain a point that lies, with high probability, in the convex hull of the surviving subset of S, and thus of the full set S. We refer to this application of the exponential mechanism in the appropriate affine subspace as the *base case* of the recursion.

The points of $S \cap f$ are not on a standard grid within f. (They lie of course in the standard uniform grid G of side length $1/X$ within the full-dimensional cube, but $G \cap f$ is not a standard grid and in general has a different, coarser resolution.) We overcome this issue by noting that there always exist j coordinates, which, without loss of generality, we assume to be $x_1, \ldots, x_j$, such that f can be expressed in parametric form by these coordinates. We then project f (and $S \cap f$) onto the $x_1 x_2 \cdots x_j$-coordinate subspace f'. We recurse within f', where the projected points of $S \cap f$ do lie in a standard grid (a cross-section of G), and then lift the output point x'_0, which lies, with high probability, in $\mathrm{conv}(S'_0)$, back to a point $x' \in f$. It is straightforward to verify that if x'_0 is in the convex hull of the projected points then x' is in the convex hull of $S \cap f$.

4.1 Finding an affine subspace with many points privately

For every affine subspace f, of dimension $0 \leq j \leq d-1$, spanned by some subset of (at least) $j+1$ points of G, we denote by $c(f)$ the number of points of S that f contains, and refer to it as the *size* of f.

We start by computing $c(f)$ for every subspace f spanned by points of S, as follows. We construct the (potentially degenerate) arrangement $\mathcal{A}(S^*)$ of the set S^* of the hyperplanes dual to the points of S. During this construction, we also compute the multiplicity of each

flat in this arrangement, namely, the number of dual hyperplanes that contain it. Each intersection flat of the hyperplanes is dual to an affine subspace f defined by the corresponding subset of the primal points of S (that it contains), and $c(f)$ is the number of dual hyperplanes containing the flat. In other words, as a standard byproduct of the construction of $\mathcal{A}(S^*)$, we can compute the sizes of all the affine subspaces that are spanned by points of S, in $O(n^d)$ overall time.

We define

$$M_i = M_i(S) = \max\{c(f) \mid \ f \text{ is spanned by a subset of } S \text{ and is of dimension } \textit{at most } i\},$$

and compute $M'_i = M_i + Y_i$, where Y_i is a random variable drawn (independently for each i) from a Laplace distribution with parameter $b := \frac{1}{\varepsilon}$ centered at the origin. (That is, the probability density function of Y_i is $\frac{1}{2b}e^{-|x|/b} = \frac{\varepsilon}{2}e^{-\varepsilon|x|}$.)

Our algorithm now uses a given confidence parameter $\beta \in (0,1)$ and proceeds as follows. If for every $j = 0, \ldots, d-1$

$$M'_j \leq n - (d-j+1)k - \frac{1}{\varepsilon}\log\frac{2}{\beta}, \tag{1}$$

we apply the base case. Otherwise, set j to be the smallest index such that

$$M'_j > n - (d-j+1)k - \frac{1}{\varepsilon}\log\frac{2}{\beta}. \tag{2}$$

Having fixed j, we find (privately) a subspace f of dimension j that contains a large number of points of S. To do so, let Z_j be the collection of all j-dimensional subspaces that are spanned by $j+1$ affinely independent points of the grid G (not necessarily of S). We apply the exponential mechanism to pick an element of Z_j, by assigning a *score* $s(f)$ to each subspace of Z_j, which we set to be

$$s(f) = \max\left\{0, c(f) - M_{j-1}\right\} \ ,$$

if $j \geq 1$, and $s(f) = c(f)$ if $j = 0$. Note that by its definition, $s(f)$ is zero if f is not spanned by points of S. Indeed, in this case the $c(f)$ points contained in f span some subspace of dimension $\ell \leq j-1$ and therefore M_{j-1} must be at least as large as $c(f)$. We will shortly argue that $s(f)$ has sensitivity at most 2 (Lemma 4.4), and thus conclude that this step preserves the differential privacy of the procedure.

We would like to apply the exponential mechanism as stated above in time proportional to the number of subspaces of non-zero score, because this number depends only on n (albeit being exponential in d) and not on (the much larger) X. However, to normalize the scores to probabilities, we need to know the number of elements of Z_j with zero score, or alternatively to obtain the total number of subspaces spanned by $j+1$ points of G (that is, the size of Z_j).

We do not have a simple expression for $|Z_j|$ (although this is a quantity that can be computed, for each j, independently of S, once and for all), but clearly $|Z_j| \leq X^{d(j+1)}$. We thus augment Z_j (only for the purpose of analysis) with $X^{d(j+1)} - |Z_j|$ "dummy" subspaces, and denote the augmented set by Z'_j, whose cardinality is now exactly $X^{d(j+1)}$. We draw a subspace f from Z'_j using the exponential mechanism. To do so we need to compute, for each score $s \geq 0$, the number N_s of elements of Z'_j that have score s, give each such element weight $e^{\varepsilon s/4}$, choose the index s with probability proportional to $N_s e^{\varepsilon s/4}$, and then choose uniformly a subspace from those with score s. It is easy to check that this is indeed an implementation of the exponential mechanism as described in Section 2.1.

If the drawing procedure decides to pick a subspace that is not spanned by points of S, or more precisely decides to pick a subspace of score 0, we stop the whole procedure, with a failure. If the selected score is positive, the subspace to be picked is spanned by $j+1$ points of S, and those subspaces are available (from the dual arrangement construction outlined above). We thus obtain a selected subspace f (by randomly choosing an element from the available list of these subspaces), and apply the algorithm recursively within f, restricting the input to $S \cap f$. (Strictly speaking, as noted above, we apply the algorithm to a projection of f onto a suitable j-dimensional coordinate subspace.)

It follows that we can implement the exponential mechanism on all subspaces in Z'_j in time proportional to the number of subspaces spanned by points of S, which is $O(n^d)$, and therefore the running time of this subspace selection procedure (in each recursive call) is $O(n^d)$.

4.1.1 Privacy analysis

▶ **Lemma 4.2.** *Let $S_1 = S_0 \cup \{x\}$ and $S_2 = S_0 \cup \{y\}$ be two neighboring data sets. Then, for every $i = 0, \ldots, d-1$, we have $|M_i(S_1) - M_i(S_2)| \leq 1$.*

Proof. Let f be a subspace of dimension at most i that is spanned by points of S_1 and contains $M_i(S_1)$ such points. If f does not contain x then f is also a candidate for $M_i(S_2)$, so in this case $M_i(S_2) \geq M_i(S_1)$. If f does contain x then $S_1 \cap f \setminus \{x\} \subseteq S_0$ spans a subspace f' of f which is of dimension at most i, so it is a candidate for $M_i(S_2)$. Since it does not contain x (and may contain y) we have in this case that $M_i(S_2) \geq M_i(S_1) - 1$. Therefore we can conclude that in anycase $M_i(S_2) \geq M_i(S_1) - 1$. A symmetric argument shows that $M_i(S_1) \geq M_i(S_2) - 1$. Combining these two inequalities the lemma follows. ◀

▶ **Lemma 4.3.** *The value of each M'_i, for $i = 0, \ldots, d-1$, is ε-differentially private.*

Proof. This follows from standard arguments in differential privacy (e.g., see [9, 19]), since, by Lemma 4.2, M_i is of *sensitivity* 1 (in the sense shown in the lemma). ◀

Since we choose j by comparing each of the M'_j's to a value which is the same for neighboring data sets S_1 and S_2 (which have the same cardinality n), Lemma 4.3 implies that the choice of the dimension j is differentially private.

The next step is to choose the actual subspace f in which to recurse. The following lemma implies that this step too is differentially private.

▶ **Lemma 4.4.** *Let S_1 and S_2 be as in Lemma 4.2. Then, for each $j = 0, \ldots, d-1$ and for every subspace $f \in Z'_j$, we have $|s_{S_1}(f) - s_{S_2}(f)| \leq 2$.*

Proof. Fix j and $f \in Z'_j$. Clearly, $|c_{S_1}(f) - c_{S_2}(f)| \leq 1$, and, by Lemma 4.2, M_{j-1} is also of *sensitivity* 1, and the claim follows. ◀

▶ **Lemma 4.5.** *The subspace-selection procedure described in this section (with all its recursive calls) is $2d^2\varepsilon$-differentially private.*

Proof. By Lemma 4.3 the computation of each M'_i is ε-differentially private, and by Lemma 4.4 the exponential mechanism on the scores $s(f)$ is also ε-differentially private. Since we compute at most d values M'_i at each step, and we recurse at most d times, the claim follows by composition [9, 19]. ◀

▶ Remark 4.6. We can save a factor of d in Lemma 4.5 by using a framework called *the sparse vector technique*, see e.g., [9].

4.1.2 Utility analysis

The following lemma is the key for our utility analysis.

▶ **Lemma 4.7.** *Let $\beta \in (0,1)$ be a given parameter. For $k = \frac{n}{4d}$ and for every $j = 0, \ldots, d-1$ the following properties hold.*

(i) *If $M_j \geq n - (d-j+1)k$ then, with probability at least $1-\beta$,*

$$M'_j \geq n - (d-j+1)k - \frac{1}{\varepsilon}\log\frac{2}{\beta}.$$

(ii) *On the other hand, if $M_j \leq n - (d-j+1)k - \frac{2}{\varepsilon}\log\frac{2}{\beta}$, then, with probability at least $1-\beta$,*

$$M'_j \leq n - (d-j+1)k - \frac{1}{\varepsilon}\log\frac{2}{\beta}.$$

Proof. (i) follows since the probability of the Laplace noise Y_j to be smaller than $-\frac{1}{\varepsilon}\log\frac{2}{\beta}$ is at most β, and (ii) follows since the probability of Y_j to be larger than $\frac{1}{\varepsilon}\log\frac{2}{\beta}$ is also at most β. ◀

We say that (the present recursive step of) our algorithm *fails* if one of the complements of the events specified in Lemma 4.7 happens, that is, the step fails if for some j, either (i) $M_j \geq n-(d-j+1)k$ and $M'_j < n-(d-j+1)k-\frac{1}{\varepsilon}\log\frac{2}{\beta}$, or (ii) $M_j \leq n-(d-j+1)k-\frac{2}{\varepsilon}\log\frac{2}{\beta}$ and $M'_j > n-(d-j+1)k-\frac{1}{\varepsilon}\log\frac{2}{\beta}$. Otherwise we say that (this step of) our algorithm *succeeds*.[3]

It follows from Lemma 4.1 that if the algorithm succeeds and applies the base case then $D_{\geq k}$ is full dimensional. Furthermore, if the algorithm succeeds and decides to recurse on a subspace of dimension j (according to the rule in (1) and (2)) then, for every $\ell < j$, $M_\ell \leq n-(d-\ell+1)k$ and $M_j \geq n-(d-j+1)k-\frac{2}{\varepsilon}\log\frac{2}{\beta}$. The following lemma is an easy consequence of this reasoning.

▶ **Lemma 4.8.** *If the algorithm succeeds, with dimension j, and applies the exponential mechanism to pick a j-dimensional subspace, then there exists a j-dimensional subspace f with score $s(f) = M_j - M_{j-1} \geq k - \frac{2}{\varepsilon}\log\frac{2}{\beta}$. Furthermore, if $k \geq \frac{8d^2}{\varepsilon}\log X + \frac{8}{\varepsilon}\log\frac{1}{\beta}$ then, with probability at least $1-\beta$, the exponential mechanism picks a subspace f with $s(f) \geq M_j - M_{j-1} - \frac{k}{2} \geq \frac{k}{2} - \frac{2}{\varepsilon}\log\frac{2}{\beta}$.*

Proof. The first part of the lemma follows from the definition of success, as just argued. For the second part notice that, since $|Z'_j| \leq X^{d^2}$, the probability of drawing a subspace $f' \in Z'_j$ of score smaller than $M_j - M_{j-1} - \frac{k}{2}$ is at most

$$X^{d^2} \cdot \frac{e^{\varepsilon(M_j - M_{j-1} - k/2)/4}}{e^{\varepsilon(M_j - M_{j-1})/4}} = X^{d^2} \cdot e^{-\varepsilon k/8} .$$

This expression is at most β if $k \geq \frac{8d^2}{\varepsilon}\log X + \frac{8}{\varepsilon}\log\frac{1}{\beta}$. ◀

[3] Note that there is a "grey zone" of values of M_j between these two bounds, in which the step always succeeds.

If follows that if our algorithm succeeds and recurses in a subspace f of dimension j then, with probability at least $1-\beta$,

$$\begin{aligned} c(f) &\geq M_{j-1} + s(f) \geq M_j - \frac{k}{2} \\ &\geq n - (d-j+1)k - \frac{2}{\varepsilon}\log\frac{1}{\beta} - \frac{k}{2} \geq n - \left(d-j+\frac{3}{2}\right)k - \frac{2}{\varepsilon}\log\frac{1}{\beta}. \end{aligned}$$

That is, when we recurse in f of dimension j we lose at most $\left(d-j+\frac{3}{2}\right)k + \frac{2}{\varepsilon}\log\frac{1}{\beta}$ points. Denote by $d_0 = d, d_1, \ldots, d_t$ the sequence of dimensions into which the procedure recurses (reaching the base case at dimension $d_t \geq 0$). Hence, keeping k fixed throughout the recursion, at the r-th recursive step we lose at most $\left(d_r - d_{r+1} + \frac{3}{2}\right)k + \frac{2}{\varepsilon}\log\frac{1}{\beta}$ points. Summing up these losses over $r = 0, \ldots, t-1$, the total loss is at most

$$(d_0 - d_t)k + \frac{3}{2}kt + \frac{2t}{\varepsilon}\log\frac{1}{\beta} \leq \frac{5d}{2}\cdot k + \frac{2d}{\varepsilon}\log\frac{1}{\beta}.$$

Substituting $k = \frac{n}{4d}$, we get that the total number of points that we loose is at most $\frac{2n}{3}$ if $n = \Omega\left(\frac{d}{\varepsilon}\log\frac{1}{\beta}\right)$, with a sufficiently large constant of proportionality.

Notice that we keep k fixed throughout the recursion and n may decrease. Consequently, if n' is the number of points in some recursive call in some dimension $\ell < d$, then $n' \geq \frac{n}{3}$ and therefore $k = \frac{n}{4d} \leq \frac{3n'}{4d}$ which is still smaller than the centerpoint guarantee of $\frac{n'}{\ell+1}$.

As described, our subspace-selection procedure (with all its recursive calls) is $2d^2\varepsilon$-differentially private. Dividing ε by $2d^2$ we get that our subspace-selection procedure is ε-differentially private, and that the total number of points we lose is much smaller than n if $n = \Omega\left(\frac{d^3}{\varepsilon}\log\frac{1}{\beta}\right)$.

Recall Section 3, where we showed that we need $n = \Omega\left(\frac{d^4 \log dX}{\varepsilon}\right)$ for the (ε-differentially private) base case to work correctly. (Recall also that the base case is actually applied in a suitable projection of the terminal subspace onto some coordinate-frame subspace of the same dimension, and that the above lower bound on n suffices for any such lower-dimensional instance too.)

The following theorem summarizes the result of this section.

▶ **Theorem 4.9.** *If $n = \Omega\left(\frac{d^4 \log dX}{\varepsilon} + \frac{d^3 \log\frac{1}{\beta}}{\varepsilon}\right)$, our algorithm (including all recursive calls and the base case) is ε-differentially private, runs in $O\left(n^{1+(d-1)\lfloor d/2\rfloor}\right)$ time, and finds a point of depth at least $k = \frac{n}{4d}$ with probability at least $1 - 2d^2\beta$.*

Proof. The privacy statement follows by composition, using Lemma 4.5 and the privacy properties of the exponential mechanism. The confidence bound follows since the probability of failure of the recursive call in a subspace of dimension j is at most $(j+1)\beta$. The running time of the algorithm is dominated by the running time of the exponential mechanism that we perform at the bottom (base case) of the recursion. ◀

5 An $O(n^d)$ algorithm via volume estimation

The upper bound on the running time in Theorem 4.9 is dominated by the running time of the base case, in which we compute all the regions $D_{\geq\ell}$ explicitly, which takes $n^{O(d^2)}$ time. To reduce this cost, we use instead a mechanism that (a) estimates the volume of D_k to a sufficiently small relative error, and (b) samples a random point "almost" uniformly from D_k.

We show how to accomplish (a) and (b) using the volume estimation mechanisms of Lovász and Vempala [14] and later of Cousins and Vempala [7]. We also show how to use these procedures to implement approximately the exponential mechanism described in Section 3. These algorithms are Monte Carlo, so they fail with some probability, and when they fail we may lose our ε-differential privacy guarantee. As a result, the modified algorithm will not be purely differentially private, as the one in Section 3, and we will only be able to guarantee that it is (ε, δ)-differentially private, for any prescribed $\delta > 0$. The following theorem is our main result. We prove it in the full version of this paper [12].

▶ **Theorem 5.1.** *Given $n = \Omega\left(\frac{d^4 \log \frac{dX}{\delta}}{\varepsilon}\right)$ points, our algorithm (including all recursive calls and the base case) is (ε, δ)-differentially private, runs in $O\left(n^d\right)$ time, and finds a point of depth at least $k = \frac{n}{4d}$ with probability at least $1 - \delta$.*

6 Conclusions

We gave an $O(n^d)$-time algorithm for privately computing a point in the convex hull of $\Omega(d^4 \log X)$ points with coordinates that are multiples of $1/X$ in $[0, 1]$. Even though this gives a huge improvement of what was previously known and requires some nontrivial technical effort, and sophisticated sampling and volume estimation tools, this running time is still not satisfactory for large values of d. The main hurdle in improving it further is the nonexistence of efficient algorithms for computing Tukey depths and Tukey levels.

The main question that we leave open is whether there exists a differentially private algorithm for this task which is polynomial in n and d? (when the input size, n, is still polynomial in $\log X$ and d).

References

1 Amos Beimel, Shiva Prasad Kasiviswanathan, and Kobbi Nissim. Bounds on the sample complexity for private learning and private data release. In *TCC*, volume 5978 of *LNCS*, pages 437–454. Springer, 2010.

2 Amos Beimel, Shay Moran, Kobbi Nissim, and Uri Stemmer. Private center points and learning of halfspaces. In *Conference on Learning Theory (COLT)*, pages 269–282, 2019.

3 Amos Beimel, Kobbi Nissim, and Uri Stemmer. Private learning and sanitization: Pure vs. approximate differential privacy. In *APPROX-RANDOM*, volume 8096 of *LNCS*, pages 363–378. Springer, 2013.

4 Mark Bun, Cynthia Dwork, Guy N. Rothblum, and Thomas Steinke. Composable and versatile privacy via truncated cdp. In *50th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 74–86, 2018.

5 Mark Bun, Kobbi Nissim, Uri Stemmer, and Salil P. Vadhan. Differentially private release and learning of threshold functions. In *IEEE 56th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 634–649, 2015.

6 Bernard Chazelle. An optimal convex hull algorithm in any fixed dimension. *Discrete Comput. Geom.*, 10:377–409, 1993.

7 Ben Cousins and Santosh S. Vempala. Gaussian cooling and $O^*(n^3)$ algorithms for volume and gaussian volume. *SIAM J. Comput.*, 47(3):1237–1273, 2018.

8 Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, volume 3876 of *LNCS*, pages 265–284. Springer, 2006.

9 Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3-4), 2014.

10 Herbert Edelsbrunner, Raimund Seidel, and Micha Sharir. On the zone theorem for hyperplane arrangements. *SIAM J. Comput.*, 22(2):418–429, 1993.

11 Haim Kaplan, Katrina Ligett, Yishay Mansour, Moni Naor, and Uri Stemmer. Privately learning thresholds: Closing the exponential gap. *CoRR*, 2019. `arXiv:1911.10137`.

12 Haim Kaplan, Micha Sharir, and Uri Stemmer. How to find a point in the convex hull privately, 2020. `arXiv:2003.13192`.

13 Xiaohui Liu, Karl Mosler, and Pavlo Mozharovskyi. Fast computation of Tukey trimmed regions and median in dimension p $>$ 2. *J. of Comput. and Graph. Stat.*, 28(3):682–697, 2019.

14 László Lovász and Santosh S. Vempala. Simulated annealing in convex bodies and an $O^*(n^4)$ volume algorithm. *J. Comput. Syst. Sci.*, 72(2):392–417, 2006.

15 Jiří Matousek. *Lectures on Discrete Geometry*. Springer-Verlag, Berlin, Heidelberg, 2002.

16 Frank McSherry and Kunal Talwar. Mechanism design via differential privacy. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 94–103, 2007.

17 Peter J. Rousseeuw and Ida Ruts. Constructing the bivariate Tukey median. *Statistica Sinica*, 8(3):827–839, 1998.

18 John W. Tukey. Mathematics and the picturing of data. In *Proc. of the International Congress of Mathematicians*, volume 2, page 523–531, 1975.

19 Salil Vadhan. The complexity of differential privacy. In Yehuda Lindell, editor, *Tutorials on the Foundations of Cryptography: Dedicated to Oded Goldreich*, pages 347–450. Springer, 2017.

Efficient Approximation of the Matching Distance for 2-Parameter Persistence

Michael Kerber
Graz University of Technology, Austria
kerber@tugraz.at

Arnur Nigmetov
Graz University of Technology, Austria
nigmetov@tugraz.at

Abstract

In topological data analysis, the matching distance is a computationally tractable metric on multi-filtered simplicial complexes. We design efficient algorithms for approximating the matching distance of two bi-filtered complexes to any desired precision $\varepsilon > 0$. Our approach is based on a quad-tree refinement strategy introduced by Biasotti et al., but we recast their approach entirely in geometric terms. This point of view leads to several novel observations resulting in a practically faster algorithm. We demonstrate this speed-up by experimental comparison and provide our code in a public repository which provides the first efficient publicly available implementation of the matching distance.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases multi-parameter persistence, matching distance, approximation algorithm

Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.53

Related Version A full version of the paper is availabe at [14], `https://arxiv.org/abs/1912.05826`.

Supplementary Material Our code is available as part of the HERA library `https://bitbucket.org/grey_narn/hera/src/master/matching/` and provides an efficient implementation for computing the matching distance for bi-filtrations.

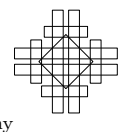
Funding *Michael Kerber*: Supported by Austrian Science Fund (FWF) grant number P 29984-N35.

1 Introduction

Persistent homology [10, 4, 9, 18] is one of the major concepts in the quickly evolving field of topological data analysis. The concept is based on the idea that studying the topological properties of a data set across various scales yields valuable information that is more robust to noise than restricting to a fixed scale.

We distinguish the case of *single-parameter persistence*, where the scale is expressed by a single real parameter, and the case of *multi-parameter persistence*, in which the scale consists of two or more parameters that vary independently. The former case is the predominant one in the literature. The entire homological multi-scale evolution of the data set can be expressed by a multi-set of points in the plane, the so-called *persistence diagram*. Moreover, the *interleaving distance* yields a distance measure between two data sets by measuring the difference in their topological evolution. For a single parameter, this distance can be rephrased as a combinatorial matching problem of the corresponding persistence diagrams (known as the *bottleneck distance*) and computed efficiently [12]. These results are part of a rich theory of single-parameter persistence, with many algorithmic results and applications.

The case of *multi-parameter persistence* received significantly less attention until recently. One reason is the early result that a complete combinatorial structure such as the persistence diagram does not exist for two or more parameters [5]. Moreover, while the interleaving distance can be straight-forwardly generalized to several parameters, its computation becomes




NP-hard already for two parameters, as well as any approximation to a factor less than 3 [3]. On the other hand, data sets with several scale parameters appear naturally in applications, and an efficiently computable distance measure is therefore highly important.

We focus on the *matching distance* [6, 2, 13] as a computationally tractable lower bound on the interleaving distance [15]. It is based on the observation that when restricting the multi-parameter space $\mathbb{R}^d$ to a one-dimensional affine subspace (that is, a line in $\mathbb{R}^d$), we are back in the case of single-parameter persistence. We can compute the bottleneck distance between the two persistence diagrams restricted to the same subspace. The matching distance is then defined as the supremum of all bottleneck distances over all subspaces (see Section 3 for the precise definition). The matching distance has been used in shape analysis [6, 2] (where it is known as the matching distance between size functions), for virtual screening in computational chemistry [11], and a recent algorithm [13] computes the distance exactly in polynomial time (with a large exponent).

Our contribution is an improvement of an approximation algorithm by Biasotti et al. [2] for the 2-parameter case which we summarize next. We parameterize the space of all lines of interest as a bounded rectangle $R \subset \mathbb{R}^2$. To each point p in the rectangle, we assign $f(p)$ as the bottleneck distance between the two persistence diagrams when restricting the data sets to the line parameterized by p. The matching distance is then equal to $\sup_{p \in R} f(p)$. The major ingredient of the algorithm is a *variation bound* which tells how much $f(p)$ varies when p is perturbed by a fixed amount. For any subrectangle $S \subseteq R$ with center c, $f(c)$ and the variation bound yield an upper bound of f within S. We then obtain an ε-approximation with a simple branch-and-bound scheme, subdividing R with a quad-tree in BFS order and stopping the subdivision of a rectangle when its upper bound is sufficiently small.

Our contributions. We aim for a fast implementation useful for practical applications of multi-parameter persistent homology. Towards this goal, we make the following contributions:

1. We rephrase the approximation algorithm by Biasotti et al. entirely in elementary geometric terms. We think that the geometric point of view complements their formulation and makes the structure of the algorithm more accessible. Indeed, we are able to simplify several arguments from [2] (see Appendix E in [14]).
2. We provide a simple yet crucial algorithmic improvement: instead of using the global variation bound for all rectangles of the subdivision, we derive adaptive local variation bounds for each rectangle individually. This results in much smaller upper bounds and avoids many subdivisions in the approximation algorithm.
3. We experimentally compare our version of the global bound with the usage of the adaptive bounds. We show that the speed-up factor of the sharpest adaptive bound is typically between 3 and 8, depending on the input bi-filtrations (for some inputs the speed-up is 15).
4. Our code is available as part of Hera library[1] and provides an efficient implementation for computing the matching distance for bi-filtrations.

Outline. We start with a short introduction to filtrations and persistent homology in Section 2. We define the matching distance in Section 3. The algorithm to approximate it is described in Section 4, and our local variation bounds are derived in Section 5. We do experiments in Section 6 and conclude in Section 7.

[1] `https://bitbucket.org/grey_narn/hera/src/master/matching/`

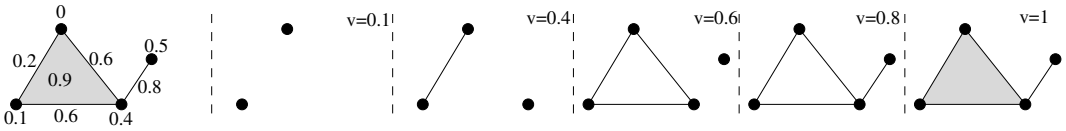


Figure 1 Left: Mono-filtration of a simplicial complex K of dimension 2. The critical value of each simplex is displayed. Right: Examples of the complexes K_v for various values of v.

2 Background

Mono-Filtrations. Fixing a base set V, a *k-simplex* σ is a non-empty subset of V of cardinality $k+1$. A *face* τ of σ is a non-empty subset of σ. A *simplicial complex* K is a collection of simplices such that whenever $\sigma \in K$, every face of σ is in K as well. The dimension of a simplicial complex K is the maximal k such that K contains a k-simplex. As an example, a graph is merely a simplicial complex of dimension 1. Following graph-theoretic notations, we call 0-simplices of K *vertices* and 1-simplices of K *edges*. A *subcomplex* of K is a subset $L \subseteq K$ such that L is again a simplicial complex. We will henceforth assume that the base set V is finite, which implies also that the simplicial complex K is finite.

A *mono-filtration* is a simplicial complex K equipped with a function $\varphi : K \mapsto \mathbb{R}$ such that for any simplex σ and any face τ of σ, it holds that $\varphi(\tau) \leqslant \varphi(\sigma)$. We call $\varphi(\sigma)$ the *critical value of* σ and define for any $v \in \mathbb{R}$

$$K_v := \{\sigma \in K \mid \varphi(\sigma) \leqslant v\}.$$

By the condition on φ from above, K_v is a subcomplex of K for each v. Moreover, whenever $v \leqslant w$, we have that $K_v \subseteq K_w$. Hence, the collection $(K_v)_{v\in\mathbb{R}}$ yields a nested sequence of simplicial complexes, which is entirely determined by the critical values of each simplex in K. See Figure 1 for an illustration.

Persistence diagrams. We are interested in the topological changes of $(K_v)_{v\in\mathbb{R}}$ when v increases continuously. A *persistence diagram* is a multi-set of points in $\mathbb{R} \times (\mathbb{R} \cup \{\infty\})$ with all points strictly above the diagonal $x = y$. The general definition requires a digression into representation theory and homological algebra (e.g., see [18]). Instead, we explain the idea on the problem of tracking connected components of K_v within the filtration, which is a special case of the general theory. Assume for simplicity that no two simplices have the same critical value. Whenever we reach the critical value b of a vertex, a new connected component comes into existence. We call this a *birth*. We say that the component born at b *dies* at value d if there is an edge with critical value d that merges the connected component with another connected component which was born before b. In that case, (b, d) is a point in the persistence diagram, denoting that the corresponding connected component persisted from scale b to scale d. Assuming that K is connected, each component gets assigned a unique death value except the component born at the minimal critical value. We assign the death value ∞ to this component, adding an infinite point to the diagram. The resulting diagram is called the *persistence diagram in (homological) dimension* 0. See Figure 2 (left). Similar diagrams can be defined for detecting tunnels, voids, and higher-dimensional holes in the simplicial complex.

We define a distance on two persistence diagrams D_1 and D_2 next. Fixing a partial matching between D_1 and D_2, we assign to each match of $p \in D_1$ and $q \in D_2$ the cost $\|p-q\|_\infty = \max\{|p_x - q_x|, |p_y - q_y|\}$, with the understanding that $\infty - \infty = 0$. In particular, the cost of matching a finite to an infinite point is ∞. Every unmatched point p gets assigned

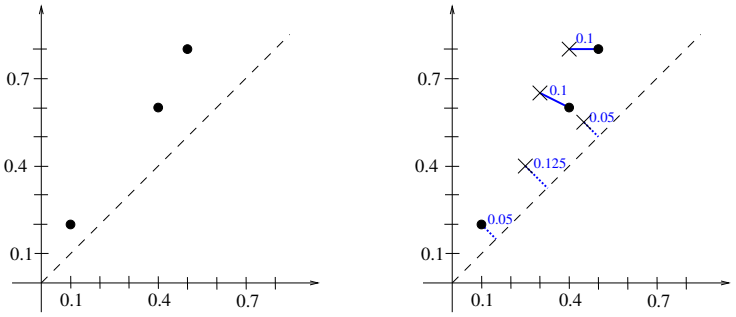


Figure 2 Left: The persistence diagram in dimension 0 of the example from Figure 1 (plus the point $(0, \infty)$ that is not drawn). Note that indeed, connected components are born at 0, 0.1, 0.4 and 0.5, and the latter three components die at 0.2, 0.6 and 0.8, respectively. Right: A partial matching of two diagrams (depicted by circles and x-shapes). The cost of each match and of each unmatched vertex is displayed. The cost of this matching is 0.125 which is in fact the optimal cost in this example, so the bottleneck distance between the diagrams is 0.125.

the cost $\frac{p_y - p_x}{2}$ which corresponds to the L_∞-distance from p to the diagonal. Taking the maximum over all matched and unmatched points in D_1 and D_2 results in the cost of the chosen partial matching. The *bottleneck distance* between D_1 and D_2 is then the minimum cost over all possible partial matchings between D_1 and D_2. See Figure 2 (right) for an example. Since filtrations give rise to persistence diagrams, we also talk about the bottleneck distance between two filtrations and denote it by $d_B(\cdot, \cdot)$ from now on.

We will need the following properties of the bottleneck distance. The proofs of the first three of them follow directly from the definition.

- d_B satisfies the triangle inequality: $d_B(F, H) \leqslant d_B(F, G) + d_B(G, h)$ for three filtrations F, G, H.
- d_B is shift-invariant: let $F = (K, \varphi)$ be a filtration, define F_r be the filtration $(K, \varphi + r)$, where the critical value of each simplex is shifted by r. Then $d_B(F, G) = d_B(F_r, G_r)$.
- d_B is homogeneous: let $F = (K, \varphi)$ be a filtration, λ be a positive number, define λF be the filtration $(K, \lambda\varphi)$.Then $d_B(\lambda F, \lambda G) = \lambda d_B(F, G)$.
- d_B is stable [7]: let $F_1 = (K, \varphi_1)$ and $F_2 = (K, \varphi_2)$ be two filtrations of the same complex such that for each $\sigma \in K$, $|\varphi_1(\sigma) - \varphi_2(\sigma)| \leqslant \varepsilon$. Then, $d_B(F_1, F_2) \leqslant \varepsilon$.

Bi-filtrations. Define the partial order $\leqslant$ on $\mathbb{R}^2$ as $p \leqslant q$ if and only if $p_x \leqslant q_x$ and $p_y \leqslant q_y$. Geometrically, $p \leqslant q$ if and only if q lies in the upper-right quadrant with corner p. A *(1-critical) bi-filtration* is a simplicial complex K together with a function $\varphi : K \to \mathbb{R}^2$ such that for every simplex σ and every face τ of σ, we have that $\varphi(\tau) \leqslant \varphi(\sigma)$. As before, $\varphi(\sigma)$ is called the *critical value* of σ. Our assumption that every simplex has a unique critical value is just for the sake of simpler exposition; our ideas extend to the *k-critical* case where each σ has up to k incomparable critical values (Appendix H, [14]). Fixing $p \in \mathbb{R}^2$, we define

$$K_p := \{\sigma \in K \mid \varphi(\sigma) \leqslant p\}.$$

Similar to the mono-filtration case, K_p is a subcomplex and $K_p \subseteq K_q$, whenever $p \leqslant q$.

It is worth visualizing the construction of K_p geometrically. We can represent the bi-filtration as a multi-set of points in $\mathbb{R}^2$, where each point corresponds to a simplex and is placed at the critical value of the simplex. The complex K_p then consists of all simplices that are placed in the lower-left quadrant with p at its corner. See Figure 3 for an example.

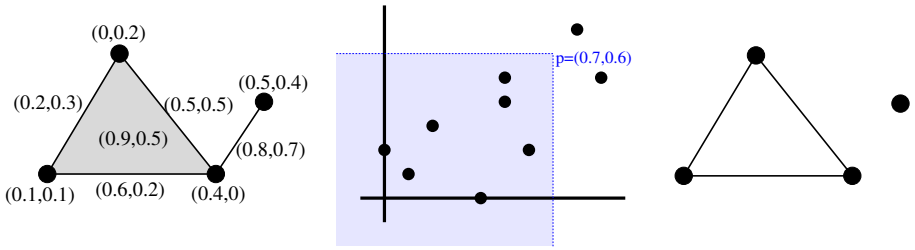


Figure 3 Left: Bi-filtration of a simplicial complex K of dimension 2. Middle: Every point in the plane denotes the critical value of a simplex. The shaded rectangle yields the simplices that belong to K_p. Right: Illustration of K_p as a subcomplex of K.

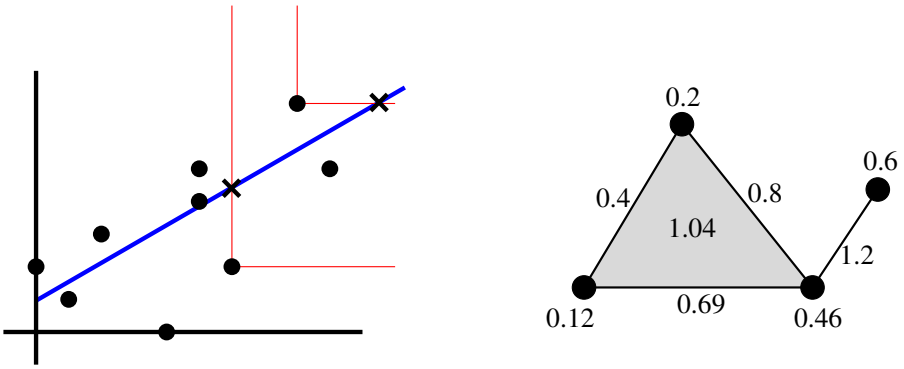


Figure 4 Left: The slice parameterized by $(\frac{\pi}{6}, 0.1)$. For two critical values of the bi-filtration from above, we illustrate the construction of the point q (displayed by a cross shape). The push of the critical value is simply the Euclidean distance to the point $(0, 0.1)$, which is the origin of the slice. Right: The non-weighted restriction on the slice. Each simplex gets its push as critical value.

3 The matching distance

Slices. Bi-filtrations are too wild to admit a simple combinatorial description such as a persistence diagram. But we can obtain a persistence diagram when restricting to a one-dimensional affine subspace. For all concepts in this subparagraph, see Figure 4 for an illustration. We consider a non-vertical line L with positive slope, which we call a *slice*. For every slice, we distinguish a point $\mathcal{O}$, called the *origin* of the slice. We let $\mathcal{L}$ denote the set of all slices. Since the slope is positive, for any two distinct points p, q on L either $p \leqslant q$ or $q \leqslant p$ holds. Hence, $\leqslant$ becomes a total order along L.

Given $p \in \mathbb{R}^2$, let q be the minimal point on L (with respect to $\leqslant$) such that $p \leqslant q$. Geometrically, q is the intersection of L with the boundary of the upper-right quadrant of p, or equivalently, the horizontally-rightwards projection of p to L if p lies above L, or the vertically-upwards projection of p to L if p lies below L. Since q lies on L, q can be written as

$$\mathcal{O} + \lambda_p \begin{pmatrix} \cos\gamma \\ \sin\gamma \end{pmatrix}$$

where γ is the angle between L and x-axis, and $\lambda_p \in \mathbb{R}$. We define λ_p as the *push* of p to L, which can be formally written as a function $\text{push} : \mathbb{R}^2 \times \mathcal{L} \to \mathbb{R}$. Geometrically, the push is simply the (signed) distance of the point q to the origin of the slice. Fixing a bi-filtration $F = (K, \varphi)$, the composition $\text{push}(\cdot, L) \circ \varphi$ yields a function $K \to \mathbb{R}$, and it can be readily checked that this function yields a mono-filtration, which we call the *non-weighted restriction* of F onto L. See Figure 4 (right) for an example.

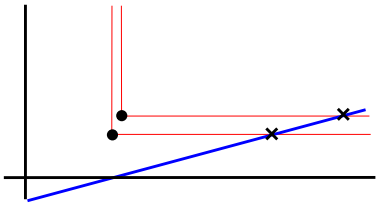


Figure 5 Two points that are close to each other might have pushes far from each other. Note that by making the slice more flat, the distance between the pushes can be made arbitrarily large.

Matching distance. Given two bi-filtrations F^1, F^2, we could try to define a distance between them by taking the supremum of bottleneck distances between their non-weighted restrictions on all slices. However, this does not yield a meaningful result. The reason is that for almost horizontal and almost vertical slices, the pushes of two close-by points can move very far away from each other – see Figure 5 for an example. As a result, the bottleneck distance along such slices becomes arbitrarily large.

Instead, we introduce a weight for each slice. Let γ denote the angle between the slice L and x-axis. We call L *flat* if $\gamma \leqslant \frac{\pi}{4}$ (i.e., if its slope is $\leqslant 1$) and *steep* if $\gamma \geqslant \frac{\pi}{4}$. Then we set

$$w(L) := \begin{cases} \sin\gamma & \text{if } L \text{ is flat} \\ \cos\gamma & \text{if } L \text{ is steep.} \end{cases}$$

We define the matching distance between the bi-filtrations $F^1 = (K^1, \varphi^1)$ and $F^2 = (K^2, \varphi^2)$ as

$$\mathrm{d_M}(F^1, F^2) := \sup_{L \in \mathcal{L}} w(L) \cdot d_B(restr(F^1, L), restr(F^2, L)),$$

where $restr(F^i, L)$ denotes the non-weighted restriction of F^i onto L. Note that while the non-weighted restrictions depend on the choice of the origin, a different choice of origin for a slice only results in a uniform translation of the critical values of both mono-filtrations. Hence, the bottleneck distance does not change because of shift-invariance. This means that the matching distance is independent of the choice of the origins.

Moreover, the shift-invariance of d_B implies that if we alter φ_1 and φ_2 such that every value is translated by the same vector $v \in \mathbb{R}^2$, the matching distance does not change. Recalling that we can visualize bi-filtrations as finite multi-sets of points in $\mathbb{R}^2$, we can hence assume without loss of generality that all these points are in the upper-right quadrant of the plane, that is, $\varphi^i(\sigma) \in [0, \infty) \times [0, \infty)$.

Let us now define the *weighted push* of a point p to a slice L as $\mathrm{wpush}(p, L) = w(L)\,\mathrm{push}(p, L)$, and let F_L denote the mono-filtration induced by $\sigma \mapsto \mathrm{wpush}(\varphi(\sigma), L)$. We call F_L a *weighted restriction* of F onto L. Note that F_L equals $restr(F, L)$ except that all critical values are scaled by the factor $w(L)$. Using homogeneity of d_B, we see that

$$\mathrm{d_M}(F^1, F^2) = \sup_{L \in \mathcal{L}} d_B(F^1_L, F^2_L). \tag{1}$$

We will use this equivalent definition of the matching distance in the remaining part of the paper, and "restriction" will always mean "weighted restriction".

4 The approximation algorithm

The idea of the approximation algorithm for $\mathrm{d_M}$ is to sample the set of slices through a finite sample, and chose the maximal bottleneck distance between the (weighted) restriction encountered as the approximation value. In order to execute this plan, we need to parameterize the space of slices and need to compute the restriction of a parameterized slice efficiently.

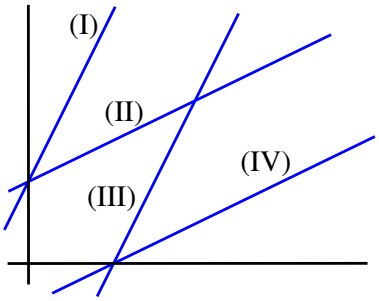


Figure 6 A steep y-slice (I), a flat y-slice (II), a steep x-slice(III) and a flat x-slice. The slopes are 2 for the steep and $\frac{1}{2}$ for the flat slices, and the origin is at $(0, 2)$ for the y-slices and at $(2, 0)$ for the x-slices. Consequently, all four slices are parameterized by $(\frac{1}{2}, 2)$.

Slice Parameterization. Every slice has a unique point where the line enters the positive quadrant of $\mathbb{R}^2$, which is either its intersection with the positive x-axis, the positive y-axis, or the point $(0, 0)$. From now on, we always use this point as the origin of the slice.

We call a slice an *x-slice* if its origin lies on the positive x-axis, and call it a *y-slice* if its origin lies on the positive y-axis (slices through the origin are both x- and y-slices). Recall also that a slice is flat if its slope is less than 1, and steep if it is larger than 1. Thus, a slice belongs to one of the four types: flat x-slices, flat y-slices, steep x-slices and steep y-slices. Every slice is represented as a point $(\lambda, \mu) \in (0, 1] \times [0, \infty)$ where the interpretation of the parameters depends on the type of the slice as follows: Let $\mathcal{O} = (\mathcal{O}_x, \mathcal{O}_y)$ be the origin of L, and recall that γ is the angle of the slice with the x-axis. Then

$$\lambda = \begin{cases} \tan(\gamma), & \text{if } L \text{ is flat} \\ \cot(\gamma), & \text{if } L \text{ is steep} \end{cases}, \qquad \mu = \begin{cases} \mathcal{O}_x & \text{if } L \text{ is } x\text{-slice} \\ \mathcal{O}_y & \text{if } L \text{ is } y\text{-slice} \end{cases}.$$

In other words, λ is the slope of the line in the flat case, and the inverse of the slope in the steep case, and μ contains the non-trivial coordinate of the origin. Note that the same pair of parameters can parameterize different slices depending on the type. Figure 6 illustrates this.

Weighted pushes. We next show a simple formula for the value of $\text{wpush}(p, L)$ depending on the type of the slice.

▶ **Lemma 1.** *With the chosen parameterization and choice of origin on L, $\text{wpush}(p, L)$ is computed according to the formulas given in Table 1.*

Proof. The proof of the lemma is a series of elementary calculations. Let us consider, for example, the case of a flat y-slice. In this case the slice $L = (\lambda, \mu)$ is given by

$$\left\{ \begin{pmatrix} 0 \\ \mu \end{pmatrix} + \rho \begin{pmatrix} \cos\gamma \\ \sin\gamma \end{pmatrix} \mid \rho \in \mathbb{R} \right\},$$

If $p = (p_x, p_y)$ is above L, we consider the point $q = (q_x, q_y)$ which is the intersection of L and the line $y = p_y$. Obviously, $q_y = p_y$, and, since q lies on L, the second coordinate yields

$$\rho = \frac{q_y - \mu}{\sin\gamma} = \frac{p_y - \mu}{\sin\gamma}.$$

By definition, ρ is the push of p to L and since L is flat, we have that $w(L) = \sin\gamma$ which cancels with the denominator. The other 7 cases are proved analogously. ◀

Table 1 Formulas for weighted push of (p_x, p_y) onto a slice $L = (\lambda, \mu)$.

	y-slices		x-slices	
	flat	steep	flat	steep
p above L	$p_y - \mu$	$\lambda(p_y - \mu)$	p_y	λp_y
p below L	λp_x	p_x	$\lambda(p_x - \mu)$	$p_x - \mu$

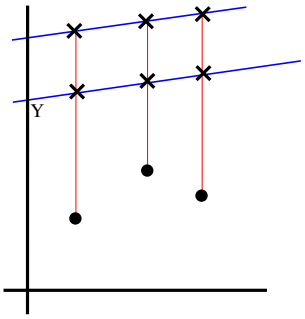


Figure 7 Illustration for the fact that slices with larger value of μ can be ignored.

All 8 expressions in Table 1 involve only addition and multiplication without trigonometric functions. Hence we can extend them continuously to $\lambda = 0$, which corresponds to horizontal lines (in the flat case) or vertical lines (in the steep case). With this interpretation, we can extend $\mathcal{L}$ to a set $\bar{\mathcal{L}}$ in (1), containing these limit cases, without changing the supremum.

Next, we observe that we can restrict our attention to a bounded range of μ-parameters. For that, let X denote the maximal x-coordinate and Y be the maximal y-coordinate among all critical values of F^1 and of F^2. For a y-slice (steep or flat) $L = (\lambda, \mu)$ with $\mu > Y$, let $L' = (\lambda, Y)$ be the parallel slice with origin at $(0, Y)$. All critical values of $F^{1,2}$ are below L and L' by construction (recall that all critical points are assumed in the upper-right quadrant), hence we obtain the push by projecting vertically upwards. Looking at the second row in Table 1, we see that the weighted pushes are independent of μ, and therefore equal for L and L'. Hence, the weighted bottleneck distances along L and L' are equal: $d_B(F^1_L, F^2_L) = d_B(F^1_{L'}, F^2_{L'})$. See Figure 7 for an illustration. We conclude that for y-slices it suffices to consider $0 \leqslant \mu \leqslant Y$ in (1) without changing the matching distance. An analogous argument shows that for x-slices, it is only necessary to consider $0 \leqslant \mu \leqslant X$.

To summarize the last two observations, we arrive at the following statement. There are sets $\mathcal{L}_1$ of flat x-slices, $\mathcal{L}_2$ of steep x-slices, $\mathcal{L}_3$ of flat y-slices, and $\mathcal{L}_4$ of steep y-slices (with each set containing some vertical/horizontal lines as limit case) such that

$$\mathrm{d_M}(F^1, F^2) = \sup_{L \in \mathcal{L}_1 \cup \ldots \cup \mathcal{L}_4} d_B(F^1_L, F^2_L) \tag{2}$$

and such that $\mathcal{L}_1$ and $\mathcal{L}_2$ are parameterized by $[0,1] \times [0,X]$ and $\mathcal{L}_3$ and $\mathcal{L}_4$ by $[0,1] \times [0,Y]$.

Approximation. We present an approximation algorithm that, given two bi-filtrations F^1 and F^2 and some $\varepsilon > 0$ returns a number δ such that

$$\mathrm{d_M}(F^1, F^2) - \varepsilon \leqslant \delta \leqslant \mathrm{d_M}(F^1, F^2).$$

We assume that the two bi-filtrations are given as simplicial complexes, i.e., a list of simplices, where each simplex is annotated with two real values denoting the critical value of the simplex. In the description, we set $T := \{x\text{-flat}, x\text{-steep}, y\text{-flat}, y\text{-steep}\}$ for the type of a slice. The algorithm is based on the following two primitives:

`Eval(`F^1, F^2, L`)` Computes $d_B(F^1_L, F^2_L)$, where L is specified by the triple (λ, μ, t) where (λ, μ) are the parameterization of L and $t \in T$ denotes its type.

`Bound(`F^1, F^2, B, t`)` If B is an axis-parallel rectangle and $t \in T$, the pair (B, t) specifies a set of slices $\mathcal{L}_0$. The primitive computes a number $\mu \in \mathbb{R}$ such that, for every $L \in \mathcal{L}_0$,

$$d_B(F^1_L, F^2_L) \leqslant \mu.$$

With these two primitives, we can state our approximation algorithm: from now on, we refer to axis-parallel rectangles as *boxes* for brevity. We start by computing maximal coordinates X and Y of critical values of F^1 and F^2 and enqueueing the four initial items $([0,1]\times[0,Y], y\text{-steep})$, $([0,1]\times[0,Y], y\text{-flat})$, $([0,1]\times[0,X], x\text{-steep})$, and $([0,1]\times[0,X], x\text{-flat})$ into a FIFO-queue. We also maintain a variable ρ storing the largest bottleneck distance encountered so far, initialized to 0.

Now, we pop items from the queue and repeat the following steps: for an item (B, t), let L denote the slice that corresponds to the center point of B. We call `Eval(`F^1`,`F^2`,`L`)` and update ρ if the computed value is bigger than the current maximum. Then, we compute $\mu \leftarrow$`Bound(`F^1`,`F^2`,`B`,`t`)`. If $\mu > \rho + \varepsilon$, we split B into 4 sub-boxes $B_1, \ldots, B_4$ of equal dimensions (using the center as splitpoint) and enqueue $(B_1, t), \ldots, (B_4, t)$. When the queue is empty, we return $\delta \leftarrow \rho$. This ends the description of the algorithm.

Assuming that the above algorithm terminates (which is unclear at this point because it depends on the implementation of the `Bound` primitive), the output is indeed an ε-approximation. This can be derived directly from the termination condition of the subdivision and the fact that ρ is non-decreasing during the algorithm. See Appendix A, [14] for details.

A variant of the above algorithm computes a relative approximation of the matching distance, that is, a number δ such that

$$\mathrm{d_M}(F^1, F^2) \leqslant \delta \leqslant (1+\varepsilon)\,\mathrm{d_M}(F^1, F^2).$$

The algorithm is analogous to the above, with the difference that a box is subdivided if $\mu > (1+\varepsilon)\rho$, and at the end of the algorithm $(1+\varepsilon)\rho$ is returned as δ. The correctness of this variant follows similarly. However, the algorithm terminates only if $\mathrm{d_M}(F^1, F^2) > 0$, and its complexity depends on the value of the matching distance.

What is needed to realize the `Eval` primitive? First, we compute the weighted pushes of each critical value of F^1 and of F^2 in time proportional to the number of critical values using Lemma 1. Then, we compute the persistence diagrams of F^1 and of F^2, and their bottleneck distance. Both steps are well-studied standard tasks in persistent homology, and several practically efficient algorithms have been studied. We use Phat [1] for computing persistence diagrams and Hera [12] for the bottleneck computation.

5 The `Bound` primitive

Recall that the input of `Bound` is (F^1, F^2, B, t), where (B, t) specifies a collection of slices of type t. In what follows, we will identify points in B with the parameterized slice, writing $L \in B$ to denote that L is obtained from a pair of parameters $(\lambda, \mu) \in B$ with respect to type t (which we skip for notational convenience).

Let L_c be the slice corresponding to the center of B. The *variation* of a point $p \in \mathbb{R}^2$ for B denotes how much the weighted push of p changes when the slice is changed within B:

$$v(p, B) := \max_{L \in B} |\operatorname{wpush}_p(L) - \operatorname{wpush}_p(L_c)|.$$

For a bi-filtration F, we define

$$v(F,B) := \max_{p \text{ critical value of } F} v(p,B).$$

The variation yields an upper bound for the bottleneck distance within a box:

▶ **Lemma 2.** *With the notation as before, we have that for two filtrations F^1, F^2 that*

$$\sup_{L \in B} d_B(F^1_L, F^2_L) \leqslant v(F^1,B) + d_B(F^1_{L_c}, F^2_{L_c}) + v(F^2,B)$$

Proof. By triangle inequality of the bottleneck distance,

$$d_B(F^1_L, F^2_L) \leqslant d_B(F^1_L, F^1_{L_c}) + d_B(F^1_{L_c}, F^2_{L_c}) + d_B(F^2_{L_c}, F^2_L).$$

Looking at the first term on the right, we have two filtrations of the same simplicial complex, and every critical values changes by at most $v(F^1,B)$ by definition of the variation. Hence, by stability of the bottleneck distance, $d_B(F^1_L, F^1_{L_c}) \leqslant v(F^1,B)$. The same argument applies to the third term which proves the theorem. ◀

Note that the second term in the bound of Lemma 2 is the value at the center slice, which is already computed in the algorithm. It remains to compute the variation of a bi-filtration within B. This, in turn, we do by analyzing the variation of a point p within B. We show

▶ **Theorem 3.** *For a box B, let $L_1, \dots, L_4$ be the four slices on the corners of B. Then*

$$v(p,B) = \max_{i=1,\dots,4} |\operatorname{wpush}_p(L_i) - \operatorname{wpush}_p(L_c)|$$

The theorem gives a direct algorithm to compute $v(p,B)$, just by computing the weighted pushes at the four corners (in constant time) and return the maximal difference to the weighted push in the center. Doing so for every critical point of a bi-filtration F yields $v(F,B)$, and with Lemma 2 an algorithm for the `Bound` primitive that runs in time proportional to the number of critical points of F^1 and F^2. We refer to this bound as *local linear bound* (where the term "linear" refers to the computational complexity), or as *L-bound*.

The proof of the theorem is presented in Appendix B, [14]. The main idea is that the expression $|\operatorname{wpush}_p(L_{\lambda,\mu}) - \operatorname{wpush}_p(L_c)|$ (with $L_{\lambda,\mu}$ the slice given by (λ,μ)) has no isolated local maxima, even for a fixed λ or a fixed μ. That implies that from any (λ,μ) in the box, there is rectilinear path to a corner on which the expression above is non-decreasing.

A coarser bound. We have derived a method to compute $v(p,B)$ exactly which takes linear time. Alternatively, we can derive an upper bound as follows:

▶ **Theorem 4.** *Let B be a box $[\lambda_{\min}, \lambda_{\max}] \times [\mu_{\min}, \mu_{\max}]$ with center (λ_c, μ_c), width $\Delta\lambda = \lambda_{\max} - \lambda_{\min}$ and height $\Delta\mu = \mu_{\max} - \mu_{\min}$. Then, for any point $p \in [0,X] \times [0,Y]$, $v(p,B)$ is at most*

$$\begin{cases} \frac{1}{2}(\Delta\mu + X\Delta\lambda) & \text{for flat } y\text{-slices} \\ \frac{1}{2}(\lambda_c\Delta\mu + (Y - \mu_{\min})\Delta\lambda)\} & \text{for steep } y\text{-slices} \\ \frac{1}{2}(\lambda_c\Delta\mu + (X - \mu_{\min})\Delta\lambda) & \text{for flat } x\text{-slices} \\ \frac{1}{2}(\Delta\mu + Y\Delta\lambda) & \text{for steep } x\text{-slices.} \end{cases}$$

Importantly, the bound is independent of p, and hence also an upper bound for $v(F,B)$ that can be computed in constant time; we refer to it as *local constant bound* or *C-bound*.

The proof of Theorem 4 is based on deriving a bound of how much $\mathrm{wpush}_p(L)$ and $\mathrm{wpush}_p(L')$ can differ for two slices $L = (\lambda, \mu)$ and $L' = (\lambda', \mu')$ in dependence of $|\lambda - \lambda'|$ and $|\mu - \mu'|$. This bound, in turn, is derived separately for all four types of boxes and involves an inner case distinction depending on whether p lies above both slices, below both slices, or in-between. In either case, the claim of the statement follows from the bound by plugging in the center slice of a box for either L or L'. See [14] for the detailed proof.

Termination and complexity. We show next that our absolute approximation algorithm terminates when realized with either the local linear bound or the local constant bound. In what follows, set $C := \max\{X, Y\}$. In the subdivision process, each box B considered is assigned a *level*, where the level of the four initial boxes is 0, and the four sub-boxes obtained from a level-k-box have level $k + 1$. Since every box is subdivided by the center, we have immediately that for a level-k-box, $\Delta\lambda = 2^{-k}$, and $\Delta\mu \leqslant C2^{-k}$. Using these estimates in Theorem 4, we obtain

$$v(F^i, B) \leqslant \frac{1}{2}(C2^{-k} + C2^{-k}) = C2^{-k}. \tag{3}$$

for $i = 1, 2$ and every level-k-box B considered by the algorithm. Note that the local constant bound yields a bound on $v(F^i, B)$ that is not worse, and so does the local linear bound (which computes $v(F^i, B)$ exactly). Hence we have

▶ **Lemma 5.** *Let B be a level-k-box considered in the algorithm. Then, μ, the result of the* `Bound` *primitive in the algorithm, satisfies*

$$\mu \leqslant d_B(F^1_{L_c}, F^2_{L_c}) + 2C2^{-k}$$

both for the local linear and local constant bound.

It follows easily that if $2C2^{-k} \leqslant \varepsilon$, or equivalently $2^k \geqslant \frac{2C}{\varepsilon}$, a box is not further subdivided in the algorithm (see Lemma 9 in [14]). Moreover, if the maximal subdivision depth is k, the algorithm visits $O(4^k)$ boxes, and requires $O(n^3)$ time per box because of the computation of two persistence diagrams and their bottleneck distance. Combining these results leads to the complexity bound.

▶ **Theorem 6.** *Our algorithm to compute an absolute ε-approximation terminates in*

$$O(n^3 \left(\frac{C}{\varepsilon}\right)^2)$$

steps in the worst case (both for the linear and constant bound).

See again [14] for more details. In there, we also derive a similar bound for the variant of computing a relative approximation.

▶ **Theorem 7.** *Our algorithm to compute a relative $(1 + \varepsilon)$-approximation terminates in*

$$O(n^3 \left(\frac{C(1+\varepsilon)}{\varepsilon\, \mathrm{d_M}(F^1, F^2)}\right)^2)$$

steps in the worst case if $\mathrm{d_M}(F^1, F^2) > 0$.

6 Experiments

Experimental setup. Our experiments were performed on a workstation with an Intel(R) Xeon(R) CPU E5-1650 v3 CPU (6 cores, 3.5GHz) and 64 GB RAM, running GNU/Linux (Ubuntu 16.04.5). The code was written in C++ and compiled with gcc-8.1.0.

We generated two datasets, which we call GH and ED, following [2] (unfortunately, we were unable to get either the code or the data used by the authors). Each of the 70 files in the datasets is a lower-star bi-filtration of a triangular mesh (2-dimensional complex), representing a 3D shape. We also generated dataset RND of larger random bi-filtrations with up to 2,000 vertices. A more detailed description of the datasets can be found in [14]. In all our experiments we used persistence diagrams in dimension 0. In the experiments with the datasets GH and ED, we computed all pairwise distances; in the experiments with RND we computed distances only between bi-filtration with the same number of vertices. We used relative error threshold, which we call ε in this section throughout (i.e., we always compute $(1+\varepsilon)$-approximation).

Comparison of different bounds. First, we experimentally compare the performance of our algorithm with the L-bound from Theorem 3 and with the C-bound from Theorem 4. Obviously, the L-bound is sharper, so it allows us to subdivide fewer boxes and in this sense is more efficient. However, it is not a priori clear that the L-bound is preferable, since its computation takes $O(n)$ time per box, in contrast to the constant bound.

Secondly, we compare the local bounds L and C with the bound provided by Equation (3), which we call the *global* bound or G-bound because it only depends on the size of the box. A bound of that sort (worse than Equation (3)) is used in [2].

Recall that the dominating step in the complexity analysis (and in practice) is the computation of persistence, and we perform two such computations when we call the `Eval` primitive. Therefore we are interested in the number of calls of `Eval`; for brevity, we refer to this number simply as the number of *calls*.

In Table 2, we give the average number of calls and timings for different datasets and values of ε. Actually, the variance behind the average in these tables is large, so we additionally provide Table 3 and Table 4, where we report the average, maximal, and minimal ratios of the number of calls and time that the algorithm needs with different bounds. For instance, the third line of Table 3 shows that for all pairs from ED that we tested with relative error $\varepsilon = 0.5$, switching from the local constant to the local linear bound reduces the number of calls by a factor between 1.78 and 4.92.

Table 4 shows that, as expected, the C-bound always performs better than the G-bound, with the average speed-up around 2. The L-bound brings an additional speed-up by a factor of 1.5-2 in terms of the running time; the number of calls is reduced more significantly, by a factor of 3. However, the second from the right column of Table 4 shows that the running time can sometimes moderately increase, if we switch to the L-bound from the C-bound. If we compare the G-bound with the L-bound directly (these numbers are not present in Table 4), the best speed-up factor is 15.6, the worst one is 1.14, and the average is between 3 and 8, depending on the dataset.

Breadth-first search, depth-first search, and error decay. In the formulation of our algorithm we used a FIFO-queue. This means that we traverse the quad-tree in breadth-first order (i.e., level by level). Other traversal strategies are possible, for instance a depth-first order, or a greedy algorithm where boxes with large bottleneck distance at the center are

Table 2 Average number of calls and average running time with the L-, C- and G-bounds for different datasets and relative error ε.

	#Calls			Time (min)		
	L	C	G	L	C	G
GH, $\varepsilon = 0.5$	938	2502	11082	2.08	3.67	18.78
ED, $\varepsilon = 0.1$	1455	3920	27529	2.58	3.26	25.96
ED, $\varepsilon = 0.5$	169	531	2112	0.28	0.42	1.67

Table 3 Comparison of number of calls between the global, local constant, and local linear bounds. G / C denotes the ratio of the G-bound compared with the C-bound; C / L denotes the ratio of the C-bound compared with the L-bound.

	#Calls: G / C			#Calls: C / L		
Dataset, ε	Avg	Min	Max	Avg	Min	Max
GH, $\varepsilon = 0.5$	1.80	1.21	3.28	3.10	1.51	7.02
ED, $\varepsilon = 0.1$	2.93	1.43	5.07	3.00	1.88	6.82
ED, $\varepsilon = 0.5$	1.94	1.17	2.81	3.29	1.78	4.92
RND, $\varepsilon = 0.1$	6.06	2.76	10.58	2.08	1.91	2.47

Table 4 Comparison of running time between the global, local constant, and local linear bounds.

	Time: G / C			Time: C / L		
Dataset, ε	Avg	Min	Max	Avg	Min	Max
GH, $\varepsilon = 0.5$	1.66	1.00	3.18	2.03	0.75	6.32
ED, $\varepsilon = 0.1$	3.12	1.44	5.21	1.64	0.81	3.40
ED, $\varepsilon = 0.5$	2.08	1.07	3.38	1.59	0.92	3.89
RND, $\varepsilon = 0.1$	5.73	2.83	10.67	1.93	1.66	2.20

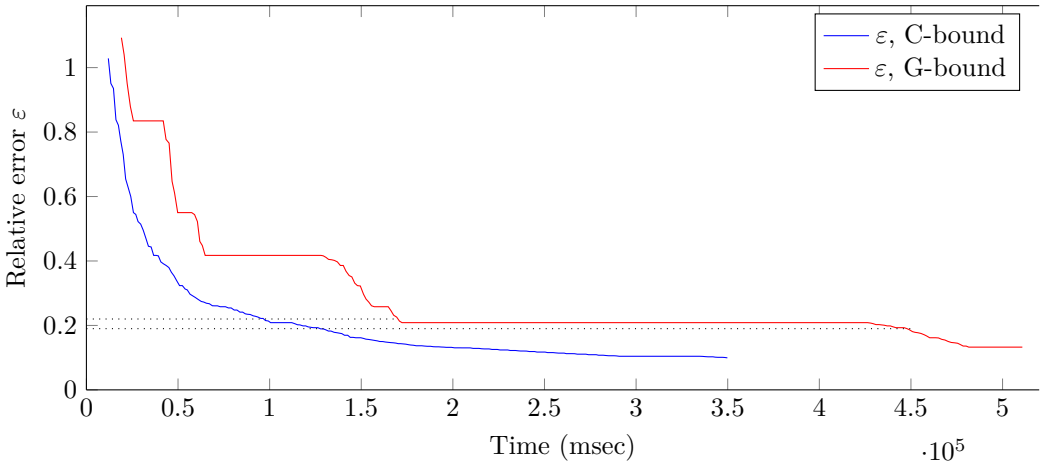


Figure 8 Error decay with time for the C- and G-bounds.

picked first. We experimented with these variants and found no significant difference. The explanation is that a good lower bound is achieved after a small number of iterations in every variant, and the remaining part of the computations is mostly to certify the answer.

A variant of our algorithm is that instead of ε, we are given a time budget and want to compute the best possible (relative) approximation in this time limit. In such a case, we propose to traverse the quad-tree by always subdividing the box with the largest upper bound (i.e., the output of the `Bound` primitive). When the time is over, it suffices to peek at the top of the priority queue to get the current upper bound, and we can output the lower bound ρ and the relative error that we can guarantee at this moment. It is instructive to plot how the relative error decreases as the algorithm runs;see Figure 8. For instance, we can see that it takes approximately 3.5 times longer to bring the relative error below 0.1 than below 0.2, if we use the constant bound. This agrees with the complexity estimate in Theorem 7.

One detail in this plot is relevant for the experiments of the previous subparagraph. If we choose a relative error ε_0 and draw a horizontal line $\varepsilon = \varepsilon_0$ in Figure 8 until it intersects the plotted curves, then the x-coordinate of the intersection is the time that our algorithm needs to guarantee a $1 + \varepsilon_0$ approximation with the corresponding bound. We can see that the difference between the time needed with the global bound and the time needed with the constant bound is not large for some values of ε_0, but a small change of ε_0 can rapidly increase it. Clearly, this is highly input-specific, and this partially explains the large variation in the improvement ratios that we observed above, when we ran experiments with fixed ε.

We provide additional experimental results in [14]. It contains the reduction rate (the measure used in [2]) for the experiments of this section, scaling results on the dataset RND, and heatmap visualization of $d_B(F_L^1, F_L^2)$.

7 Conclusion

We presented an algorithm for the matching distance that keeps subdividing boxes until a sufficiently close approximation of the matching distance can be guaranteed. This high-level description also applies to the previous approach by Biasotti et al., which raises the question of how the approaches compare in the details. Instead of pointing out similarities and differences in the technical part, we give a detailed discussion on this topic in [14].

We have restricted to the case of bi-filtrations in this work. Generalizations in several directions are possible. First of all, instead of bi-filtrations, our algorithm works the same when the input is a pair of presentations of persistence modules [16, 13, 3]. Since a minimal presentation of a bi-filtration can be of much smaller size than the bi-filtration itself, and its computation is feasible [16, 8], switching to a minimal presentation will most likely increase the performance further. We plan to investigate this in further work. Moreover, the case of k-critical bi-filtrations can be handled with our methodology, just by defining the push of a simplex as the minimal push over all its critical values (Appendix H in [14]). Our approach can also be combined with *barcode templates* as introduced in the RIVET library [17]. Finally, an extension of our approach to 3 and more parameters should be possible in principle, but we point out that the space of affine lines through $\mathbb{R}^d$ is $2(d-1)$ dimensional. Hence, already the next case of tri-filtrations requires a subdivision in $\mathbb{R}^4$, and it is questionable whether reasonably-sized instances could be handled by an extended algorithm.

As the experiments show, in some cases the cost of computing the L-bound makes the variant with the C-bound faster. However, parallelization of the L-bound is trivial, and we believe that on larger instances it will make the L-bound the best choice.

Finally, since the matching distance can be computed exactly in polynomial time [13], the question is whether there is a practical algorithm for this exact computation. Our current implementation can serve as a base-line for a comparison between exact and approximate version of matching distance computations that hopefully lead to further improvements for the computation of the matching distance.

References

1. Ulrich Bauer, Michael Kerber, Jan Reininghaus, and Hubert Wagner. Phat - persistent homology algorithms toolbox. *J. Symb. Comput.*, 78:76–90, 2017. doi:10.1016/j.jsc.2016.03.008.
2. 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.
3. Håvard Bjerkevik, Magnus Botnan, and Michael Kerber. Computing the interleaving distance is NP-hard. arXiv:1811.09165.
4. G. Carlsson. Topology and data. *Bulletin of the AMS*, 46:255–308, 2009.
5. G. Carlsson and A. Zomorodian. The theory of multidimensional persistence. *Discrete & Computational Geometry*, 42(1):71–93, 2009. doi:10.1007/s00454-009-9176-0.
6. A. Cerri, B. Di Fabio, M. Ferri, P. Frosini, and C. Landi. Betti numbers in multidimensional persistent homology are stable functions. *Mathematical Methods in the Applied Sciences*, 36(12):1543–1557, 2013.
7. D. Cohen-Steiner, H. Edelsbrunner, and J. Harer. Stability of persistence diagrams. *Discrete & Computational Geometry*, 37:103–120, 2007.
8. Tamal Dey and Cheng Xin. Generalized persistence algorithm for decomposing multi-parameter persistence modules. arXiv:1904.03766.
9. H. Edelsbrunner and J. Harer. *Computational Topology. An Introduction.* American Mathematical Society, 2010.
10. H. Edelsbrunner, D. Letscher, and A. Zomorodian. Topological persistence and simplification. *Discrete & Computational Geometry*, 28(4):511–533, 2002. doi:10.1007/s00454-002-2885-2.
11. Bryn Keller, Michael Lesnick, and Theodore L Willke. Persistent homology for virtual screening, 2018.
12. M. Kerber, D. Morozov, and A. Nigmetov. Geometry helps to compare persistence diagrams. *Journal of Experimental Algorithms*, 22:1.4:1–1.4:20, September 2017.

13 Michael Kerber, Michael Lesnick, and Steve Oudot. Exact computation of the matching distance on 2-parameter persistence modules. In *35th International Symposium on Computational Geometry (SoCG 2019)*, pages 46:1–46:15, 2019.

14 Michael Kerber and Arnur Nigmetov. Efficient approximation of the matching distance for 2-parameter persistence. *arXiv preprint*, 2019. `arXiv:1912.05826`.

15 Claudia Landi. The rank invariant stability via interleavings. In *Research in Computational Topology*, pages 1–10. Springer, 2018.

16 Michael Lesnick and Matthew Wright. Computing minimal presentations and bigraded betti numbers of 2-parameter persistent homology. `arXiv:1902.05708`.

17 Michael Lesnick and Matthew Wright. Interactive visualization of 2-D persistence modules persistence modules. *arXiv*, 2015. `arXiv:1512.00180`.

18 S. Oudot. *Persistence theory: From Quiver Representation to Data Analysis*, volume 209 of *Mathematical Surveys and Monographs*. American Mathematical Society, 2015.

Homotopy Reconstruction via the Cech Complex and the Vietoris-Rips Complex

Jisu Kim
Inria Saclay – Île-de-France, Palaiseau, France
http://pages.saclay.inria.fr/jisu.kim/
jisu.kim@inria.fr

Jaehyeok Shin
Department of Statistics & Data Science, Carnegie Mellon University, Pittsburgh, PA, USA
http://www.stat.cmu.edu/~jaehyeos/
shinjaehyeok@cmu.edu

Frédéric Chazal
Inria Saclay – Île-de-France, Palaiseau, France
https://geometrica.saclay.inria.fr/team/Fred.Chazal/
frederic.chazal@inria.fr

Alessandro Rinaldo
Department of Statistics & Data Science, Carnegie Mellon University, Pittsburgh, PA, USA
http://www.stat.cmu.edu/~arinaldo/
arinaldo@cmu.edu

Larry Wasserman
Department of Statistics & Data Science, Machine Learning Department, Carnegie Mellon University, Pittsburgh, PA, USA
http://www.stat.cmu.edu/~larry/
larry@stat.cmu.edu

Abstract

We derive conditions under which the reconstruction of a target space is topologically correct via the Čech complex or the Vietoris-Rips complex obtained from possibly noisy point cloud data. We provide two novel theoretical results. First, we describe sufficient conditions under which any non-empty intersection of finitely many Euclidean balls intersected with a positive reach set is contractible, so that the Nerve theorem applies for the restricted Čech complex. Second, we demonstrate the homotopy equivalence of a positive μ-reach set and its offsets. Applying these results to the restricted Čech complex and using the interleaving relations with the Čech complex (or the Vietoris-Rips complex), we formulate conditions guaranteeing that the target space is homotopy equivalent to the Čech complex (or the Vietoris-Rips complex), in terms of the μ-reach. Our results sharpen existing results.

2012 ACM Subject Classification Mathematics of computing → Algebraic topology; Theory of computation → Computational geometry

Keywords and phrases Computational topology, Homotopy reconstruction, Homotopy Equivalence, Vietoris-Rips complex, Čech complex, Reach, μ-reach, Nerve Theorem, Offset, Double offset, Consistency

Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.54

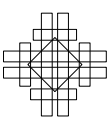
Related Version The full version of the paper is available at https://arxiv.org/abs/1903.06955 and https://hal.archives-ouvertes.fr/hal-02425686.

Supplementary Material The code is available at https://github.com/jisuk1/nerveshape.

Funding *Jisu Kim*: Partially supported by Samsung Scholarship.

Acknowledgements We want to thank André Lieutier and Henry Adams for the thoughtful discussions and comments.

36th International Symposium on Computational Geometry (SoCG 2020).
Editors: Sergio Cabello and Danny Z. Chen; Article No. 54; pp. 54:1–54:19



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

1 Introduction

A fundamental task in topological data analysis, geometric inference, and computational geometry is that of estimating the topology of a set $\mathbb{X} \subset \mathbb{R}^d$ based on a finite collection of data points $\mathcal{X}$ that lie in it or in its proximity. This problem naturally occurs in many applications area, such as cosmology [30], time series data [28], machine learning [17], and so on.

A natural way to approximate the target space is to consider an r-offset of the data points, that is, to take the union of the open balls of radius $r > 0$ centered at the data points. Under appropriate conditions, by the Nerve theorem [5] this offset is topologically equivalent to the target space $\mathbb{X}$ via the Čech complex [7, 23]. For computational reasons, the Alpha shape complex may be used instead, which is homotopy equivalent to the Čech complex [20]. To further speed up calculations, and in particular if the data are high dimensional, the Vietoris-Rips complex may be preferable as only the pairwise distances between the data points are used.

To guarantee that the topological approximation based on the data points recovers correctly the homotopy type of $\mathbb{X}$, it is necessary that the data points are dense and close to the target space, and that the radius parameter used for constructing the Čech complex or the Vietoris-Rips complex be of appropriate size.

The conditions require the offset r to be lower bounded by a constant times the Hausdorff distance between the target space and the data points, and upper bounded by another constant times a measure of the size of the topological features of the target space. Originally, the topological feature size was described as a sufficiently small number, for the Vietoris-Rips complex in [24, 25]. Then, the topological feature size was expressed in terms of the reach of $\mathbb{X}$: see, for the Čech complex, in [12, 27]. Subsequently, the notion of μ-reach was put forward to allow for more general target spaces: the condition for the Čech complex is studied in [6, 8], and the condition for the Vietoris-Rips complex is studied in [6]. Also, the radii parameters are allowed to vary across the data points in [12]. For the case when the target space equals the data points, the conditions for the Čech complex or the Vietoris-Rips complex is studied in [3, 4]. When the offset r is beyond the topological feature size so that the homotopy equivalence does not hold, the homotopy type of the Vietoris-Rips complex was studied for the circle in [2].

In this paper, we derive conditions under which the homotopy type of the target space is correctly recovered via the Čech complex or the Vietoris-Rips complex, in terms of the Hausdorff distance and the μ-reach of the target space. To tackle this problem, we provide two novel theoretical results. First, we describe sufficient conditions under which any non-empty intersection of finitely many Euclidean balls intersected with a set of positive reach is contractible, so that the Nerve theorem applies for the restricted Čech complex. Second, we demonstrate the homotopy equivalence of a positive μ-reach set and its offsets. These results are new and of independent interest.

Overall, our new bounds offer significant improvements over the previous results in [27, 6] and are sharp: in particular, they achieve the optimal upper bound for the parameter of the Čech complex and the Vietoris-Rips complex under a positive reach condition. We will provide a detailed comparison of our results with existing ones in Section 6.

2 Background

This section provides a brief introduction to simplicial complex, Nerve theorem, reach, and μ-reach. We refer to Appendix A and [23, 19, 21, 1, 8, 13, 26, 18] for further definitions and details. Throughout the paper, we let $\mathbb{X}$ and $\mathcal{X}$ be subsets of $\mathbb{R}^d$. For $x, y \in \mathbb{R}^d$, we let $d(x, y) := \|x - y\|$ be the Euclidean distance with $\|\cdot\|$ being the Euclidean norm. Let

$d(x, \mathbb{X}) = \inf_{y \in \mathbb{X}} d(x, y)$ denotes the distance from a point x to a set $\mathbb{X}$, and let $d_{\mathbb{X}} : \mathbb{R}^d \to \mathbb{R}$ be the distance function $x \mapsto d(x, \mathbb{X})$. For $r > 0$, we let $\mathbb{B}_{\mathbb{X}}(x, r) := \{y \in \mathbb{X} : d(x, y) < r\}$ be the open restricted ball centered at $x \in \mathbb{R}^d$ of radius $r > 0$. For $r > 0$, we let $\mathbb{X}^r$ be an r-offset of a set $\mathbb{X}$ defined by the collection of all points that are within r distance to $\mathbb{X}$, that is, $\mathbb{X}^r := \bigcup_{x \in \mathbb{X}} \mathbb{B}_{\mathbb{R}^d}(x, r)$. Finally, for two sets $X, Y \subset \mathbb{R}^d$, we let $d_H(X, Y) := \inf\{r > 0 : X \subset Y^r \text{ and } Y \subset X^r\}$ be the Hausdorff distance between X and Y.

2.1 Simplicial complex and Nerve theorem

A natural way to approximate the target space $\mathbb{X}$ with the data points $\mathcal{X}$ is to take the union of open balls centered at the data points. In detail, let $r = \{r_x, x \in \mathcal{X}\} \in \mathbb{R}_+^{\mathcal{X}}$ be pre-specified radii and consider the union of restricted balls

$$\bigcup_{x \in \mathcal{X}} \mathbb{B}_{\mathbb{X}}(x, r_x). \tag{1}$$

Though we allow for the points in $\mathcal{X}$ to lie outside $\mathbb{X}$, we will assume throughout that $\mathbb{B}_{\mathbb{X}}(x, r_x) \neq \emptyset$ for all $x \in \mathcal{X}$.

To infer the topological properties of the union of balls in (1), we rely on a simplicial complex, which can be seen as a high dimensional generalization of a graph. Given a set V, an *(abstract) simplicial complex* is a collection K of finite subsets of V such that $\alpha \in K$ and $\beta \subset \alpha$ implies $\beta \in K$. Each set $\alpha \in K$ is called its *simplex*, and each element of α is called a *vertex* of α.

A simplicial complex encoding the topological properties of the union of balls in (1) is the Čech complex.

▶ **Definition 1** (Čech complex). *Let $\mathcal{X}$, $\mathbb{X}$ be two subsets and $r \in \mathbb{R}_+^{\mathcal{X}}$. The (weighted)* Čech complex *$\check{C}ech_{\mathbb{X}}(\mathcal{X}, r)$ is the simplicial complex*

$$\check{C}ech_{\mathbb{X}}(\mathcal{X}, r) := \left\{ \sigma = \{x_1, \ldots, x_k\} \subset \mathcal{X} : \bigcap_{j=1}^{k} \mathbb{B}_{\mathbb{X}}(x_j, r_{x_j}) \neq \emptyset \right\}. \tag{2}$$

Computing the Čech complex requires computing all possible intersections of the balls. To further speed up the calculation, we only check the pairwise distances between the data points and instead build the Vietoris-Rips complex.

▶ **Definition 2** (Vietoris-Rips complex). *Let $\mathcal{X}$, $\mathbb{X}$ be two subsets and $r \in \mathbb{R}_+^{\mathcal{X}}$. The weighted* Vietoris-Rips complex *$Rips(\mathcal{X}, r)$ is the simplicial complex defined as*

$$Rips(\mathcal{X}, r) := \left\{ \sigma \subset \mathcal{X} : d(x_i, x_j) < r_{x_i} + r_{x_j}, \text{ for all } x_i, x_j \in \sigma \right\}. \tag{3}$$

The ambient Čech complex in (2) (that is, $\mathbb{X} = \mathbb{R}^d$) and the Vietoris-Rips complex in (3) have the following interleaving relationship when all radii are equal (e.g., see Theorem 2.5 in [16]). That is, when $r_x = r > 0$ for all $x \in \mathcal{X}$, then

$$\check{\text{C}}\text{ech}_{\mathbb{R}^d}(\mathcal{X}, r) \subset \text{Rips}(\mathcal{X}, r) \subset \check{\text{C}}\text{ech}_{\mathbb{R}^d}\left(\mathcal{X}, \sqrt{\frac{2d}{d+1}} r\right). \tag{4}$$

This interleaving relation is extended to the case of different radii in Lemma 16.

The union of balls in (1) and the Čech complex in (2) are homotopy equivalent under appropriate conditions. This remarkable result is precisely the renowned nerve theorem [5, 7, 23], which we recall next. We first introduce the *nerve*, which is a more abstract notion of the Čech complex.

▶ **Definition 3** (Nerve). *Let $\mathcal{U} = \{U_\alpha\}$ be an open cover of a given topological space $\mathbb{X}$. The nerve of $\mathcal{U}$, denoted by $\mathcal{N}(\mathcal{U})$, is the abstract simplicial complex defined as*

$$\mathcal{N}(\mathcal{U}) = \left\{ \{U_1, \ldots, U_k\} \subset \mathcal{U} : \bigcap_{j=1}^{k} U_j \neq \emptyset \right\}.$$

The nerve theorem prescribes conditions under which the nerve of an open cover of $\mathbb{X}$ is homotopy equivalent to $\mathbb{X}$ itself.

▶ **Theorem 4** (Nerve theorem). *Let $\mathbb{X}$ be a paracompact space and $\mathcal{U}$ be an open cover of $\mathbb{X}$. If every nonempty intersection of finitely many sets in $\mathcal{U}$ is contractible, then $\mathbb{X}$ is homotopy equivalent to the nerve $\mathcal{N}(\mathcal{U})$.*

Thus, in order to conclude that the $\check{\text{C}}\text{ech}_{\mathbb{X}}(\mathcal{X}, r)$ complex in (2) has the same homotopy type as $\mathbb{X}$, it is enough to show, by the nerve theorem, that the union of restricted balls $\bigcup_{x\in\mathcal{X}} \mathbb{B}_{\mathbb{X}}(x, r_x)$ covers the target space $\mathbb{X}$ and that any arbitrary non-empty intersection of restricted balls is contractible. The difficulty in establishing the latter, more technical, condition lies in the fact that it is not clear a priori what properties of $\mathbb{X}$ will imply it. If $\mathbb{X}$ is a convex set, then the nerve theorem applies straightforwardly. But for more general spaces, such as smooth lower-dimensional manifolds, it is not obvious how contractibility may be guaranteed. One of the main results of this paper, given below in Theorem 9, asserts that if $\mathbb{X}$ has positive reach and the radii of the restricted balls are small compared to the reach, then any non-empty intersection of restricted balls is contractible.

2.2 The reach

First introduced by [21], the reach is a quantity expressing the degree of geometric regularity of a set. In detail, given a closed subset $\mathbb{X} \subset \mathbb{R}^d$, the medial axis of $\mathbb{X}$, denoted by $\text{Med}(\mathbb{X})$, is the subset of $\mathbb{R}^d$ consisting of all the points that have at least two nearest neighbors in $\mathbb{X}$. Formally,

$$\text{Med}(\mathbb{X}) = \left\{ x \in \mathbb{R}^d \setminus \mathbb{X} \text{: there exist } q_1 \neq q_2 \in \mathbb{X}, ||q_1 - x|| = ||q_2 - x|| = d(x, \mathbb{X}) \right\}, \quad (5)$$

The reach of $\mathbb{X}$ is then defined as the minimal distance from $\mathbb{X}$ to $\text{Med}(\mathbb{X})$.

▶ **Definition 5.** *The reach of a closed subset $\mathbb{X} \subset \mathbb{R}^d$ is defined as*

$$\tau_{\mathbb{X}} = \inf_{q\in\mathbb{X}} d\left(q, \text{Med}(\mathbb{X})\right) = \inf_{q\in\mathbb{X}, x\in\text{Med}(\mathbb{X})} ||q - x||. \quad (6)$$

Some authors [see, e.g. 27, 29] refer to $\tau_{\mathbb{X}}^{-1}$ as the *condition number*. From the definition of the medial axis in (5), the projection $\pi_{\mathbb{X}}(x) = \arg\min_{p\in\mathbb{X}} \|p - x\|$ onto $\mathbb{X}$ is well defined (i.e. unique) outside $Med(\mathbb{X})$. In fact, the reach is the largest distance $\rho \geq 0$ such that $\pi_{\mathbb{X}}$ is well defined on the ρ-offset $\left\{x \in \mathbb{R}^d : d(x, \mathbb{X}) < \rho\right\}$. Hence, assuming the set $\mathbb{X}$ has positive reach can be seen as a generalization or weakening of convexity, since a set $\mathbb{X} \subset \mathbb{R}^d$ is convex if and only if $\tau_{\mathbb{X}} = \infty$. In the next section, we describe how to use the reach condition to ensure that the union of restricted balls is contractible, which in turn allows us to apply the Nerve theorem to recover the homotopy type of the target space $\mathbb{X}$.

For a non-smooth target space, the reach of the space can be zero. In this case, we can deploy a more general notion of feature size, called μ-reach, introduced by [8]. For any point $x \in \mathbb{R}^d \setminus \mathbb{X}$, let $\Gamma_{\mathbb{X}}(x)$ be the set of points in $\mathbb{X}$ closest to x. Let $\Theta_{\mathbb{X}}(x)$ be the center of the

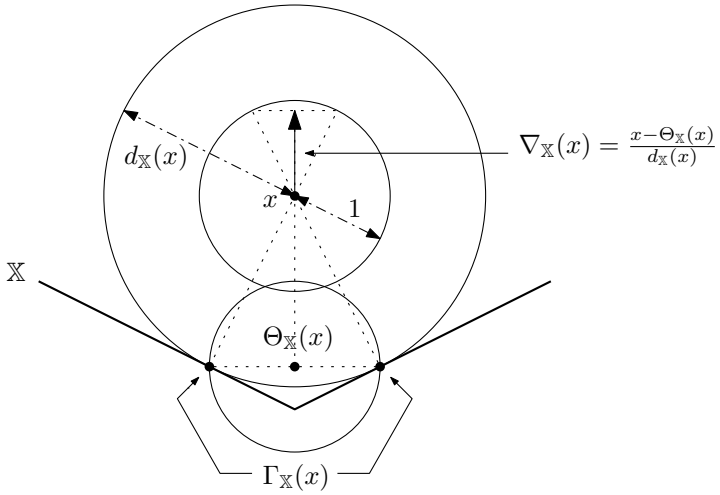


Figure 1 The graphical illustration for the generalized gradient $\nabla_{\mathbb{X}}(x)$, from [9, 8].

unique smallest closed ball enclosing $\Gamma_{\mathbb{X}}(x)$. Then, for $x \in \mathbb{R}^d \setminus \mathbb{X}$, the generalized gradient of the distance function $d_{\mathbb{X}}$ is defined as

$$\nabla_{\mathbb{X}}(x) = \frac{x - \Theta_{\mathbb{X}}(x)}{d_{\mathbb{X}}(x)}, \tag{7}$$

and set $\nabla_{\mathbb{X}}(x) = 0$ for $x \in \mathbb{X}$. See Figure 1 for a graphical illustration. Then, for $\mu \in (0, 1]$, the μ-medial axis of $\mathbb{X}$ is defined as

$$\mathrm{Med}_\mu(\mathbb{X}) = \left\{x \in \mathbb{R}^d \setminus \mathbb{X} : \|\nabla_{\mathbb{X}}(x)\| < \mu\right\}, \tag{8}$$

Finally, the μ-reach of $\mathbb{X}$ is defined as the minimal distance from $\mathbb{X}$ to $\mathrm{Med}_\mu(\mathbb{X})$.

▶ **Definition 6.** *The μ-reach of a closed subset $\mathbb{X} \subset \mathbb{R}^d$ is defined as*

$$\tau^\mu_{\mathbb{X}} = \inf_{q \in \mathbb{X}} d\left(q, \mathrm{Med}_\mu(\mathbb{X})\right) = \inf_{q \in \mathbb{X}, x \in \mathrm{Med}_\mu(\mathbb{X})} ||q - x||. \tag{9}$$

Note that if $\mu = 1$, the corresponding μ-reach equals to the reach of $\mathbb{X}$.

Two offsets $\mathbb{X}^r$ and $\mathbb{X}^s$ of the target space $\mathbb{X}$ are topologically equivalent if they are free of critical points of the distance function $d_{\mathbb{X}}$ in the sense specified below (see e.g., [22] or Proposition 3.4 in [11]).

▶ **Lemma 7** (Isotopy Lemma). *Let $\mathbb{X} \subset \mathbb{R}^d$ be a set, and for $r, s > 0$ with $s \leq r$, let $\mathbb{X}^r$ and $\mathbb{X}^s$ be two offsets of $\mathbb{X}$. Suppose the distance function $d_{\mathbb{X}}$ does not have a critical point on $\overline{\mathbb{X}^r} \setminus \mathbb{X}^s$, that is, $\nabla_{\mathbb{X}}(x) \neq 0$ for all $x \in \overline{\mathbb{X}^r} \setminus \mathbb{X}^s$ where $\nabla_{\mathbb{X}}$ is from (7). Then $\mathbb{X}^r$ and $\mathbb{X}^s$ are homeomorphic.*

Note that requiring $\nabla_{\mathbb{X}}(x) \neq 0$ for all $x \in \overline{\mathbb{X}^r} \setminus \mathbb{X}^s$ is weaker than the μ-reach condition $\tau^\mu_{\mathbb{X}} > r$ for any $\mu \in (0, 1]$. One of the main results of the paper, given in Theorem 12, generalizes this topological relation to the relation between the target space and its offset under a stronger positive μ-reach condition.

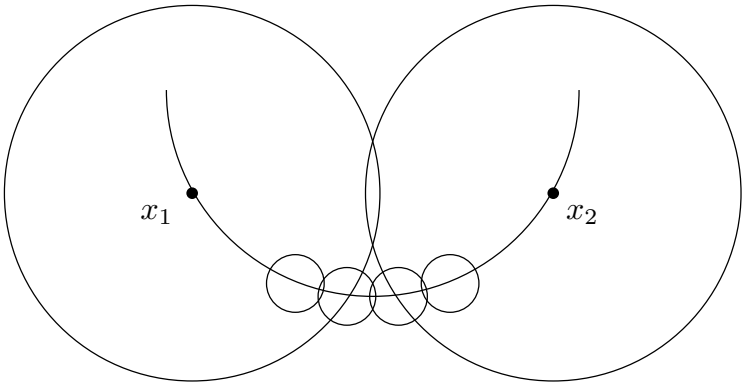


Figure 2 *An example in which the union of balls is different from the underlying space in terms of the homotopy. In the figure, the union of balls deformation retracts to a circle, hence its homotopy is different from the underlying semicircle.*

2.3 Restricted versus Ambient balls

It is important to point out that the nerve theorem needs not to be applied to the Čech complex built using ambient, as opposed, to restricted balls. In particular, the homotopy type of $\mathbb{X}$, may not be correctly recovered using unions of ambient balls even if the point cloud is dense in $\mathbb{X}$ and the radii of the balls all vanish. We elucidate this point in the next example. Below, $\mathbb{B}_{\mathbb{R}^d}(x,r)$ denotes the open ambient ball in $\mathbb{R}^d$ centered at x and of radius $r>0$.

▶ **Example 8.** Let $\mathbb{X} = (\partial\mathbb{B}_{\mathbb{R}^2}(0,1)) \cap \{x \in \mathbb{R}^2 : x_2 \geq 0\}$ be a semicircle in $\mathbb{R}^2$. Let $\epsilon \in (0,1)$ be fixed, and x_1, x_2 be points on $\mathbb{X}$ satisfying $\|x_1 - x_2\| \in \left(\epsilon\sqrt{4-\epsilon^2}, 2\epsilon\right)$. Then, $\mathbb{B}_{\mathbb{R}^2}(x_1,\epsilon) \cap \mathbb{B}_{\mathbb{R}^2}(x_2,\epsilon)$ is nonempty but has an empty intersection with $\mathbb{X}$. Now, choose $\rho < d(\mathbb{X}, \mathbb{B}_{\mathbb{R}^2}(x_1,\epsilon) \cap \mathbb{B}_{\mathbb{R}^2}(x_2,\epsilon))$ and choose $\mathcal{X}_0 \subset \mathbb{X}$ be dense enough so that $\bigcup_{x\in\mathcal{X}_0} \mathbb{B}_{\mathbb{R}^2}(x,\rho)$ covers $\mathbb{X}$. Now, consider the union of ambient balls

$$\left(\mathbb{B}_{\mathbb{R}^2}(x_1,\epsilon) \bigcup \mathbb{B}_{\mathbb{R}^2}(x_2,\epsilon)\right) \bigcup \left(\bigcup_{x\in\mathcal{X}_0} \mathbb{B}_{\mathbb{R}^2}(x,\rho)\right). \tag{10}$$

Then from the fact $\rho < d\left(\mathbb{X}, \mathbb{B}_{\mathbb{R}^2}(x_1,\epsilon) \cap \mathbb{B}_{\mathbb{R}^2}(x_2,\epsilon)\right)$ and $\bigcup_{x\in\mathcal{X}_0} \mathbb{B}_{\mathbb{R}^2}(x,\rho)$ is a covering of $\mathbb{X}$, we have that the union of balls in (10) is homotopy equivalent to a circle, hence its homotopy is different from the semicircle $\mathbb{X}$. Note that the above construction holds for all choices of $\epsilon \in (0,1)$. Since $\rho \to 0$ as $\epsilon \to 0$, $\mathcal{X}_0$ can be arbitrary dense in $\mathbb{X}$. See Figure 2.

3 The nerve theorem for Euclidean sets of positive reach

In order to apply the nerve theorem to the Čech complex built on restricted balls, it is enough to check whether any finite intersection of the restricted balls $\bigcap_{j=1}^{k} \mathbb{B}_{\mathbb{X}}(x_j, r_{x_j})$ is contractible (since $\mathbb{X}$ is a subset of $\mathbb{R}^d$ and is endowed with the subspace topology, it is paracompact.).

Theorem 9 is one of the main statements of this paper and shows that, if a subset $\mathbb{X} \subset \mathbb{R}^d$ has a positive reach $\tau > 0$, any non-empty intersection of restricted balls is contractible if the radii are small enough compared to τ.

▶ **Theorem 9.** *Let $\mathbb{X} \subset \mathbb{R}^d$ be a subset with reach $\tau > 0$ and let $\mathcal{X} \subset \mathbb{R}^d$ be a set of points. Let $\{r_x > 0 : x \in \mathcal{X}\}$ be a set of radii indexed by $x \in \mathcal{X}$. Then, if $r_x \leq \sqrt{\tau^2 + (\tau - d_{\mathbb{X}}(x))^2}$ for all $x \in \mathcal{X}$, any nonempty intersection of restricted balls $\bigcap_{x\in I} \mathbb{B}_{\mathbb{X}}(x, r_x)$ for $I \subset \mathcal{X}$ is contractible.*

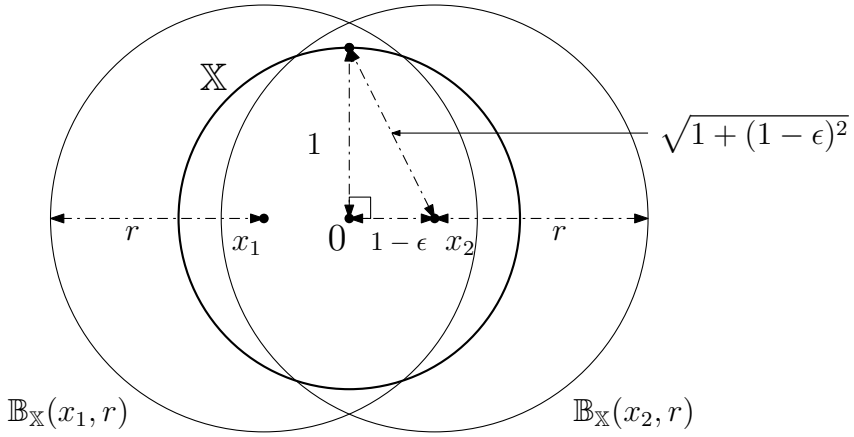


Figure 3 *An example in which* $\mathbb{B}_{\mathbb{X}}(x_1, r) \bigcup \mathbb{B}_{\mathbb{X}}(x_2, r)$ *is not homotopy equivalent to* $\check{C}ech_{\mathbb{X}}(\mathcal{X}, r)$ *where* $\mathbb{X} = \left\{x \in \mathbb{R}^2 : \|x\|_2 = 1\right\}$, $x_1 = (-1+\epsilon, 0)$, $x_2 = (1-\epsilon, 0)$, $\mathcal{X} = \{x_1, x_2\}$, *and* $r > \sqrt{1+(1-\epsilon)^2}$, *for any* $\epsilon > 0$.

Therefore, by combining Theorem 9 and the Nerve Theorem (Theorem 4), we can establish that the topology of the subspace $\mathbb{X}$ can be recovered by the corresponding restricted Čech complex $\check{C}ech_{\mathbb{X}}(\mathcal{X}, r)$, provided the radii of the balls are not too large with respect to the reach. This result is summarized in the following corollary.

▶ **Corollary 10** (Nerve Theorem on the restricted balls). *Under the same condition of Theorem 9, suppose* $r_x \leq \sqrt{\tau^2 + (\tau - d_{\mathbb{X}}(x))^2}$ *for all* $x \in \mathcal{X}$, *then the union of restricted balls* $\bigcup_{x\in\mathcal{X}} \mathbb{B}_{\mathbb{X}}(x, r_x)$ *is homotopy equivalent to the restricted Čech complex* $\check{C}ech_{\mathbb{X}}(\mathcal{X}, r)$. *If, in addition, the union of restricted balls covers the target space* $\mathbb{X}$, *that is,*

$$\mathbb{X} \subset \bigcup_{x\in\mathcal{X}} \mathbb{B}_{\mathbb{X}}(x, r_x), \tag{11}$$

then $\mathbb{X}$ *is homotopy equivalent to the restricted Čech complex* $\check{C}ech_{\mathbb{X}}(\mathcal{X}, r)$.

The reach condition $r_x \leq \sqrt{\tau^2 + (\tau - d_{\mathbb{X}}(x))^2}$ is tight as the following example shows.

▶ **Example 11.** Let $\mathbb{X}$ be the unit Euclidean sphere in $\mathbb{R}^d$, and fix $\epsilon > 0$. Let $x_1 := (1-\epsilon, 0, \ldots, 0), x_2 := (-1+\epsilon, 0, \ldots, 0) \in \mathbb{R}^d$, and set $\mathcal{X} := \{x_1, x_2\}$. For a unit Euclidean sphere, the reach is equal to its radius 1. Therefore, if $r = (r_1, r_2) \in \left(0, \sqrt{1+(1-\epsilon)^2}\right]^2$ then $\mathbb{B}_{\mathbb{X}}(x_1, r_1) \bigcup \mathbb{B}_{\mathbb{X}}(x_2, r_2)$ is homotopy equivalent to $\check{C}ech_{\mathbb{X}}(\mathcal{X}, r)$ by Corollary 10. However, if $r_1, r_2 > \sqrt{1+(1-\epsilon)^2}$, $\mathbb{B}_{\mathbb{X}}(x_1, r_1) \bigcup \mathbb{B}_{\mathbb{X}}(x_2, r_2) \simeq \mathbb{X}$ but $\check{C}ech_{\mathbb{X}}(\mathcal{X}, r) \simeq 0$. Figure 3 illustrates the 2-dimensional case.

4 Deformation retraction on positive μ-reach

The positive reach condition is critical for the nerve theorem on the restricted Čech complex. However, it is not easily generalized to the positive μ-reach condition. Instead, we find a positive reach set that approximates the positive μ-reach set. And to show their homotopy equivalence, we discover the topological relation between the positive μ-reach set and its offset.

The homeomorphic relation between two offsets $\mathbb{X}^r$ and $\mathbb{X}^s$ of the target space $\mathbb{X}$ in Lemma 7 does not hold in general between the target space and its offset, but a weakened topological relation holds under a stronger condition on the target space. Theorem 12, which

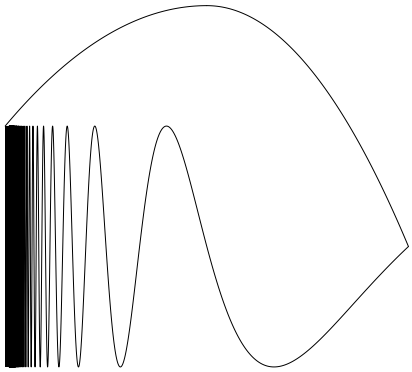


Figure 4 *An example where $\mathbb{X}^r$ does not deformation retracts to $\mathbb{X}$. $\mathbb{X}$ is a topologist's sine circle, that is, $\mathbb{X} = \mathbb{X}_0 \cup \mathbb{X}_1 \cup \mathbb{X}_2$, with $\mathbb{X}_0 = \left\{\left(x, \sin\frac{\pi}{x}\right) \in \mathbb{R}^2 : x \in [0,1]\right\}$, $\mathbb{X}_1 = \{0\} \times [-1,1]$, and $\mathbb{X}_2$ is a sufficiently smooth curve joining $(0,1)$ and $(1,0)$ and meeting $\mathbb{X}_0 \cup \mathbb{X}_1$ only at its endpoints.*

is one of the main results in our paper, asserts that if the target space $\mathbb{X}$ has a positive μ-reach, then the offset $\mathbb{X}^r$ deformation retracts to $\mathbb{X}$ when the offset size is not large, and in particular, they are homotopy equivalent.

▶ **Theorem 12.** *Let $\mathbb{X} \subset \mathbb{R}^d$ be a subset with positive μ-reach $\tau^\mu > 0$. For $r \leq \tau^\mu$, the r-offset $\mathbb{X}^r$ deformation retracts to $\mathbb{X}$. In particular, $\mathbb{X}$ and $\mathbb{X}^r$ are homotopy equivalent.*

The positive μ-reach condition $r \leq \tau^\mu$ in Theorem 12 is critical and cannot be weakened to $\nabla_{\mathbb{X}}(x) \neq 0$ for all $x \in \overline{\mathbb{X}^r} \backslash \mathbb{X}$ as in Lemma 7. Indeed, Example 13 shows that the offset does not deformation retract to the target space although $\nabla_{\mathbb{X}}(x) \neq 0$ for all $x \in \mathbb{R}^d$.

▶ **Example 13.** Let $\mathbb{X} \subset \mathbb{R}^2$ be a topologist's sine circle, that is, $\mathbb{X} = \mathbb{X}_0 \cup \mathbb{X}_1 \cup \mathbb{X}_2$, with $\mathbb{X}_0 = \left\{\left(x, \sin\frac{\pi}{x}\right) \in \mathbb{R}^2 : x \in [0,1]\right\}$, $\mathbb{X}_1 = \{0\} \times [-1,1]$, and $\mathbb{X}_2$ is a sufficiently smooth curve joining $(0,1)$ and $(1,0)$ and meets $\mathbb{X}_0 \cup \mathbb{X}_1$ only at its endpoints. See Figure 4. Then, $\tau^\mu_{\mathbb{X}} = 0$ for any $\mu \in (0,1]$ but $\nabla_{\mathbb{X}}$ is nonzero for all $x \in \mathbb{R}^2 \backslash \mathbb{X}$. Now, $H_1(\mathbb{X}) = 0$, but for any sufficiently small $r > 0$, $\mathbb{X}^r$ is homeomorphic to an annulus $\mathbb{B}_{\mathbb{R}^2}(0,2) \backslash \overline{\mathbb{B}_{\mathbb{R}^2}(0,1)}$ and hence $H_1(\mathbb{X}^r) = \mathbb{Z}$. Hence $\mathbb{X}^r$ cannot deformation retract to $\mathbb{X}$.

Using Theorem 12, we find a positive reach set that approximates the positive μ-reach set. The set we will use is the double offset [9]. Recall that, for $r > 0$, an r-offset $\mathbb{X}^r$ of a set $\mathbb{X}$ is the collection of all points that are within r distance to $\mathbb{X}$, that is, $\mathbb{X}^r := \bigcup_{x \in \mathbb{X}} \mathbb{B}_{\mathbb{R}^d}(x, r)$. The double offset is to take offset, take complement, take offset, and take complement, that is, for $s \geq t > 0$, $\mathbb{X}^{s,t} := (((\mathbb{X}^s)^{\complement})^t)^{\complement}$. Roughly speaking, it is to inflate your set first, and then deflate your set, so that sharp corners become smooth. See [9] for more details. To set up the homotopy equivalence of the positive μ-reach set and its double offset, we need another tool for finding the homotopy equivalence of the complement set. This is done in the next lemma.

▶ **Lemma 14.** *Let $\mathbb{X} \subset \mathbb{R}^d$ be a subset with positive reach $\tau > 0$. For $r \leq \tau$, $\mathbb{X}^{\complement}$ deformation retracts to $(\mathbb{X}^r)^{\complement}$. In particular, $\mathbb{X}^{\complement}$ and $(\mathbb{X}^r)^{\complement}$ are homotopy equivalent.*

Now, combining Theorem 12 and Lemma 14 gives the desired homotopy equivalence between the target set of positive μ-reach and its double offset, where the double offset has a positive reach.

▶ **Corollary 15.** *Let $\mathbb{X} \subset \mathbb{R}^d$ be a subset with positive μ-reach $\tau^\mu > 0$. For $s, t > 0$ with $t \leq s$, let $\mathbb{X}^{s,t} := (((\mathbb{X}^s)^\complement)^t)^\complement$ be the double offset of $\mathbb{X}$. If $s < \tau^\mu$ and $t < \mu s$, then $\mathbb{X}^{s,t}$ and $\mathbb{X}$ are homotopy equivalent, and the reach of $\mathbb{X}^{s,t}$ is greater than or equal to t, that is, $\tau_{\mathbb{X}^{s,t}} \geq t$.*

5 Homotopy Reconstruction via Cech complex and Vietoris-Rips complex

Next, we derive conditions under which the homotopy type of the target space is correctly recovered via the Čech complex and the Vietoris-Rips complex. We first extend the interleaving relationship of the ambient Čech complex and the Vietoris-Rips complex in (4) to the different radii case in Lemma 16.

▶ **Lemma 16.** *Let $\mathcal{X} \subset \mathbb{R}^d$ be a set of points and $r = \{r_x > 0 : x \in \mathcal{X}\}$ be a set of radii indexed by $x \in \mathcal{X}$. Then,*

$$\check{C}ech_{\mathbb{R}^d}(\mathcal{X}, r) \subset Rips(\mathcal{X}, r) \subset \check{C}ech_{\mathbb{R}^d}\left(\mathcal{X}, \sqrt{\frac{2d}{d+1}}r\right).$$

To recover the homotopy of the target set via the ambient Čech complex and the Vietoris-Rips complex, we utilize the restricted Čech complex. Hence, we set up the interleaving relationship between the restricted Čech complex and the ambient Čech complex in Lemma 17 and between the restricted Čech complex and the Vietoris-Rips complex in Corollary 18.

▶ **Lemma 17.** *Let $\mathbb{X} \subset \mathbb{R}^d$ be a subset with reach $\tau > 0$ and let $\mathcal{X} \subset \mathbb{R}^d$ be a set of points. Let $r = \{r_x > 0 : x \in \mathcal{X}\}$ be a set of radii indexed by $x \in \mathcal{X}$. Then,*

$$\check{C}ech_{\mathbb{X}}(\mathcal{X}, r) \subset \check{C}ech_{\mathbb{R}^d}(\mathcal{X}, r) \subset \check{C}ech_{\mathbb{X}}(\mathcal{X}, r'),$$

where $r' = \{r'_x > 0 : x \in \mathcal{X}\}$ with

$$r'_x = \sqrt{\frac{2\tau\left(r_x^2 + d_{\mathbb{X}}(x)\left(2\tau - d_{\mathbb{X}}(x)\right)\right)}{\tau + \sqrt{\tau^2 - \left(r_x^2 + d_{\mathbb{X}}(x)\left(2\tau - d_{\mathbb{X}}(x)\right)\right)}} - d_{\mathbb{X}}(x)\left(2\tau - d_{\mathbb{X}}(x)\right)}.$$

Equivalently,

$$\check{C}ech_{\mathbb{R}^d}(\mathcal{X}, r'') \subset \check{C}ech_{\mathbb{X}}(\mathcal{X}, r) \subset \check{C}ech_{\mathbb{R}^d}(\mathcal{X}, r),$$

where $r'' = \{r''_x > 0 : x \in \mathcal{X}\}$ with

$$r''_x = \sqrt{\tau^2 - d_{\mathbb{X}}(x)(2\tau - d_{\mathbb{X}}(x)) - \frac{(2\tau^2 - r_x^2 - d_{\mathbb{X}}(x)(2\tau - d_{\mathbb{X}}(x)))^2}{4\tau^2}}.$$

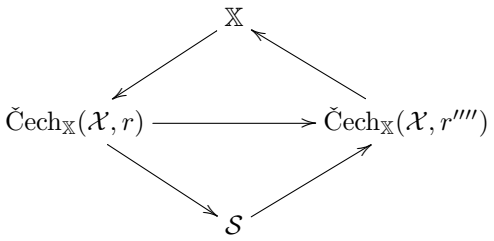
▶ **Corollary 18.** *Let $\mathbb{X} \subset \mathbb{R}^d$ be a subset with reach $\tau > 0$ and let $\mathcal{X} \subset \mathbb{R}^d$ be a set of points. Let $r = \{r_x > 0 : x \in \mathcal{X}\}$ be a set of radii indexed by $x \in \mathcal{X}$. Then,*

$$\check{C}ech_{\mathbb{X}}(\mathcal{X}, r) \subset Rips(\mathcal{X}, r) \subset \check{C}ech_{\mathbb{X}}(\mathcal{X}, r'''),$$

where $r''' = \{r'''_x > 0 : x \in \mathcal{X}\}$ with

$$r'''_x = \sqrt{\frac{2\tau\left(\frac{2d}{d+1}r_x^2 + d_{\mathbb{X}}(x)\left(2\tau - d_{\mathbb{X}}(x)\right)\right)}{\tau + \sqrt{\tau^2 - \left(\frac{2d}{d+1}r_x^2 + d_{\mathbb{X}}(x)\left(2\tau - d_{\mathbb{X}}(x)\right)\right)}} - d_{\mathbb{X}}(x)\left(2\tau - d_{\mathbb{X}}(x)\right)}.$$

Combining Nerve Theorem on the restricted balls (Corollary 10) with the covering condition (11) and Lemma 17 or Corollary 18 gives the following commutative diagram:



, (12)

where $\mathcal{S}$ is either the ambient Čech complex $\check{\mathrm{C}}\mathrm{ech}_{\mathbb{R}^d}(\mathcal{X}, r)$ or the Vietoris-Rips complex $\mathrm{Rips}(\mathcal{X}, r)$. Using this diagram, we develop the homotopy equivalence between the target space and either the ambient Čech complex or the Vietoris-Rips complex. First, Theorem 19 asserts that when the target space of positive reach is densely covered by the data points and if they are not too far apart, the ambient Čech complex can be used to recover the homotopy type.

▶ **Theorem 19.** *Let $\mathbb{X} \subset \mathbb{R}^d$ be a subset with reach $\tau > 0$ and let $\mathcal{X} \subset \mathbb{R}^d$ be a closed discrete set of points. Let $\{r_x > 0 : x \in \mathcal{X}\}$ be a set of radii indexed by $x \in \mathcal{X}$ with $r_{\min} := \min_{x\in\mathcal{X}}\{r_x\}$ and $r_{\max} := \max_{x\in\mathcal{X}}\{r_x\}$, and let $\epsilon := \max\{d_{\mathbb{X}}(x) : x \in \mathcal{X}\}$. Suppose $\mathbb{X}$ is covered by the union of balls centered at $x \in \mathcal{X}$ and radius δ as*

$$\mathbb{X} \subset \bigcup_{x\in\mathcal{X}} \mathbb{B}_{\mathbb{R}}(x, \delta). \tag{13}$$

Suppose that the maximum radius $r_{\max}$ is bounded as

$$r_{\max} \leq \tau - \epsilon. \tag{14}$$

Also, suppose δ satisfies the following condition:

$$\begin{aligned}
&\delta + \sqrt{r_{\max}^2 - \tilde{l}^2 + \epsilon(2\tau-\epsilon) - ((\tau-\epsilon)^2 - r_{\max}^2 + \tilde{l}^2 + (\tau-\epsilon_{\tilde{l}})^2)\left(\frac{\tau}{\sqrt{\tau^2 - \tilde{r}_{\delta,c}}} - 1\right)} \\
&\leq r_{\min}, \\
&\sqrt{\frac{d}{2(d+1)}}\frac{r_{\max}}{r_{\min}}\left(\sqrt{\tilde{r}_b^2 - (2\tau^2 - \tilde{r}_b^2)\left(\frac{\tau}{\sqrt{\tau^2 - \tilde{r}_{\delta,b}^2}} - 1\right)} + 2\delta\right) \leq r''_{\min},
\end{aligned} \tag{15}$$

$$\begin{aligned}
&\tilde{l} := \frac{1}{2}\left(r_{\min} - \tau + \sqrt{(\tau-\epsilon)^2 - r_{\max}^2} - \delta\right), \qquad \epsilon_{\tilde{l}} := \tau - \sqrt{(\tau-\epsilon)^2 - r_{\max}^2} + \tilde{l}, \\
&\tilde{r}_{\delta,c}^2 := \min\left\{\delta^2 + \epsilon(2\tau-\epsilon), \frac{1}{2}(r_{\max}^2 - \tilde{l}^2 + \epsilon(2\tau-\epsilon) + \epsilon_{\tilde{l}}(2\tau-\epsilon_{\tilde{l}}))\right\}, \\
&r''_{\min} := \sqrt{\tau^2 - \epsilon(2\tau-\epsilon) - \frac{(2\tau^2 - r_{\min}^2 - \epsilon(2\tau-\epsilon))^2}{4\tau^2}}, \\
&\tilde{r}_b^2 := \frac{2\tau\left((r''_{\min})^2 + \epsilon(2\tau-\epsilon)\right)}{\tau + \sqrt{\tau^2 - ((r''_{\min})^2 + \epsilon(2\tau-\epsilon))}}, \qquad \tilde{r}_{\delta,b}^2 := \min\left\{\delta^2 + \epsilon(2\beta-\epsilon), \frac{1}{2}\tilde{r}_b^2\right\}.
\end{aligned}$$

Then $\mathbb{X}$ is homotopy equivalent to the ambient Čech complex $\check{C}ech_{\mathbb{R}^d}(\mathcal{X}, r)$.

A similar approach also gives the homotopy equivalence between the target space and the Vietoris-Rips complex when the target space has positive reach.

▶ **Theorem 20.** *Let $\mathbb{X} \subset \mathbb{R}^d$ be a subset with reach $\tau > 0$ and let $\mathcal{X} \subset \mathbb{R}^d$ be a closed discrete set of points. Let $\{r_x > 0 : x \in \mathcal{X}\}$ be a set of radii indexed by $x \in \mathcal{X}$ with $r_{\min} := \min_{x\in\mathcal{X}}\{r_x\}$ and $r_{\max} := \max_{x\in\mathcal{X}}\{r_x\}$, and let $\epsilon := \max\{d_{\mathbb{X}}(x) : x \in \mathcal{X}\}$. Suppose $\mathbb{X}$ is covered by the union of balls centered at $x \in \mathcal{X}$ and radius δ as*

$$\mathbb{X} \subset \bigcup_{x\in\mathcal{X}} \mathbb{B}_{\mathbb{R}}(x, \delta). \tag{16}$$

Suppose that the maximum radius $r_{\max}$ is bounded as

$$r_{\max} \leq \sqrt{\frac{d+1}{2d}}\,(\tau - \epsilon)\,. \tag{17}$$

Also, suppose δ satisfies the following condition:

$$\begin{gathered}\sqrt{\tilde{r}_b^2(r_{\max}) - (2\tau^2 - \tilde{r}_b^2(r_{\max}))\left(\frac{\tau}{\sqrt{\tau^2 - \tilde{r}_{\delta,b}^2(r_{\max})}} - 1\right)} + 2\delta \leq 2r_{\min},\\ \sqrt{\frac{d}{2(d+1)}}\left(\sqrt{\tilde{r}_b^2(r''_{\min}) - (2\tau^2 - \tilde{r}_b^2(r''_{\min}))\left(\frac{\tau}{\sqrt{\tau^2 - \tilde{r}_{\delta,b}^2(r''_{\min})}} - 1\right)} + 2\delta\right) \leq r''_{\min},\end{gathered} \tag{18}$$

where

$$\begin{aligned} r''_{\min} &:= \sqrt{\tau^2 - \epsilon(2\tau-\epsilon) - \frac{(2\tau^2 - r_{\min}^2 - \epsilon(2\tau-\epsilon))^2}{4\tau^2}},\\ \tilde{r}_b^2(t) &:= \frac{2\tau\left(t^2 + \epsilon(2\tau-\epsilon)\right)}{\tau + \sqrt{\tau^2 - (t^2 + \epsilon(2\tau-\epsilon))}}, \qquad \tilde{r}_{\delta,b}^2(t) := \min\left\{\delta^2 + \epsilon(2\tau-\epsilon), \frac{1}{2}\tilde{r}_b^2(t)\right\}. \end{aligned}$$

Then $\mathbb{X}$ is homotopy equivalent to the Vietoris-Rips complex $\mathrm{Rips}(\mathcal{X}, r)$.

▶ Remark 21. Compared to the restricted Čech complex (Corollary 10), the covering condition in (13) or (16) is more critical for the ambient Čech complex (Theorem 19) or the Vietoris-Rips complex (Theorem 20). Although the restricted Čech complex $\check{\mathrm{C}}\mathrm{ech}_{\mathbb{X}}(\mathcal{X}, r)$ is still homotopy equivalent to the union of restricted balls $\bigcup_{x\in\mathcal{X}} \mathbb{B}_{\mathbb{X}}(x, r_x)$ without the covering condition in (11), such homotopy equivalence does not hold for the ambient Čech complex or the Vietoris-Rips complex. This is since the upper triangle of the diagram in (12) only holds under the covering condition in (13) or (16). Furthermore, the covering condition in (13) or (16) is denser in that $\delta < r_x$ for all $x \in \mathcal{X}$, to construct an additional homotopy equivalence on the lower triangle of the diagram in (12).

The homotopy equivalences in Theorem 19 and 20 for the positive reach case is extended to the positive μ-reach case by applying Corollary 15 with the double offset of the target space. Corollary 22 shows that when the double offset of the target space of positive μ-reach is densely covered by the data points and if they are not too far apart, either the ambient Čech complex or the Vietoris-Rips complex can be used to recover the homotopy type of $\mathbb{X}$.

▶ **Corollary 22.** *Let $\mathbb{X} \subset \mathbb{R}^d$ be a subset with positive μ-reach $\tau^\mu > 0$ and let $\mathcal{X} \subset \mathbb{R}^d$ be a closed discrete set of points. Let $\{r_x > 0 : x \in \mathcal{X}\}$ be a set of radii indexed by $x \in \mathcal{X}$ with $r_{\min} := \min_{x\in\mathcal{X}}\{r_x\}$ and $r_{\max} := \max_{x\in\mathcal{X}}\{r_x\}$. Let $s, t, \epsilon \geq 0$ with $\frac{t}{\mu} < s < \tau^\mu$, and let $\mathbb{Y} := (((\mathbb{X}^s)^{\complement})^t)^{\complement}$ be the double offset, with $d_{\mathbb{Y}}(x) \leq \epsilon$ for all $x \in \mathcal{X}$. Suppose $\mathbb{Y}$ is covered by the union of balls centered at $x \in \mathcal{X}$ and radius δ as*

$$\mathbb{Y} \subset \bigcup_{x\in\mathcal{X}} \mathbb{B}_{\mathbb{R}}(x, \delta).$$

(i) *Suppose $r_{\max} \leq t - \epsilon$, and δ satisfies the following condition:*

$$\delta + \sqrt{r_{\max}^2 - \tilde{l}^2 + \epsilon(2t-\epsilon) - ((t-\epsilon)^2 - r_{\max}^2 + \tilde{l}^2 + (t-\epsilon_{\tilde{l}})^2)\left(\frac{t}{\sqrt{t^2 - \tilde{r}_{\delta,c}^2}} - 1\right)}$$
$$\leq r_{\min},$$
$$\sqrt{\frac{d}{2(d+1)}}\frac{r_{\max}}{r_{\min}}\left(\sqrt{\tilde{r}_b^2 - (2t^2 - \tilde{r}_b^2)\left(\frac{t}{\sqrt{t^2 - \tilde{r}_{\delta,b}^2}} - 1\right)} + 2\delta\right) \leq r''_{\min},$$

where

$$\tilde{l} := \frac{1}{2}\left(r_{\min} - t + \sqrt{(t-\epsilon)^2 - r_{\max}^2} - \delta\right), \qquad \epsilon_{\tilde{l}} := t - \sqrt{(t-\epsilon)^2 - r_{\max}^2} + \tilde{l},$$
$$\tilde{r}_{\delta,c}^2 := \min\left\{\delta^2 + \epsilon(2t-\epsilon), \frac{1}{2}(r_{\max}^2 - \tilde{l}^2 + \epsilon(2t-\epsilon) + \epsilon_{\tilde{l}}(2t-\epsilon_{\tilde{l}}))\right\},$$
$$r''_{\min} := \sqrt{t^2 - \epsilon(2t-\epsilon) - \frac{(2t^2 - r_{\min}^2 - \epsilon(2t-\epsilon))^2}{4t^2}},$$
$$\tilde{r}_b^2 := \frac{2t\left((r''_{\min})^2 + \epsilon(2t-\epsilon)\right)}{t + \sqrt{t^2 - ((r''_{\min})^2 + \epsilon(2t-\epsilon))}}, \qquad \tilde{r}_{\delta,b}^2 := \min\left\{\delta^2 + \epsilon(2t-\epsilon), \frac{1}{2}\tilde{r}_b^2\right\}.$$

Then $\mathbb{X}$ is homotopy equivalent to the ambient Čech complex $\check{C}ech_{\mathbb{R}^d}(\mathcal{X}, r)$.

(ii) *Suppose $r_{\max} \leq \sqrt{\frac{d+1}{2d}}\,(t-\epsilon)$, and δ satisfies the following condition:*

$$\sqrt{\tilde{r}_b^2(r_{\max}) - (2t^2 - \tilde{r}_b^2(r_{\max}))\left(\frac{t}{\sqrt{t^2 - \tilde{r}_{\delta,b}^2(r_{\max})}} - 1\right)} + 2\delta \leq 2r_{\min},$$
$$\sqrt{\frac{d}{2(d+1)}}\left(\sqrt{\tilde{r}_b^2(r''_{\min}) - (2t^2 - \tilde{r}_b^2(r''_{\min}))\left(\frac{t}{\sqrt{t^2 - \tilde{r}_{\delta,b}^2(r''_{\min})}} - 1\right)} + 2\delta\right) \leq r''_{\min},$$

where

$$r''_{\min} := \sqrt{t^2 - \epsilon(2t-\epsilon) - \frac{(2t^2 - r_{\min}^2 - \epsilon(2t-\epsilon))^2}{4t^2}},$$
$$\tilde{r}_b^2(t) := \frac{2t\left(t^2 + \epsilon(2t-\epsilon)\right)}{t + \sqrt{t^2 - (t^2 + \epsilon(2t-\epsilon))}}, \qquad \tilde{r}_{\delta,b}^2(t) := \min\left\{\delta^2 + \epsilon(2t-\epsilon), \frac{1}{2}\tilde{r}_b^2(t)\right\}.$$

Then $\mathbb{X}$ is homotopy equivalent to the Vietoris-Rips complex $Rips(\mathcal{X}, r)$.

We end this section by introducing a sampling condition in which we can guarantee the covering conditions in Corollary 10 and Theorem 19, 20 are satisfied. Let P be the sampling distribution on $\mathbb{X}$. We assume that there exist positive constants a, b and ϵ_0 such that, for all $x \in \mathbb{X}$, the following inequality holds:

$$P(\mathbb{B}_{\mathbb{R}^d}(x, \epsilon)) \geq a\epsilon^b, \quad \text{for all } \epsilon \in (0, \epsilon_0). \tag{19}$$

This condition on P is also known as the (a, b)-condition or the standard condition [15, 14, 10]. It is satisfied, for example, if $\mathbb{X}$ is a smooth manifold of dimension b and P has a density with respect to the Hausdorff measure on it bounded from below by a.

Under this condition, we have the following covering lemma.

▶ **Lemma 23.** *Let $\{X_1, \ldots, X_n\}$ be an i.i.d. sample from the distribution P and let $\{r_n = (r_{n,1}, \ldots, r_{n,n})\}_{n\in\mathbb{N}}$ be a triangular array of positive numbers such that, for each n,*

$$2\left(\frac{\log n}{an}\right)^{1/b} \leq \min_i r_{n,i} \leq 2\epsilon_0. \tag{20}$$

Then, the probability that the sample is a r_n-covering of $\mathbb{X}$ is bounded as

$$P\left(\mathbb{X} \subset \bigcup_{i=1}^{n} \mathbb{B}_{\mathbb{R}^d}(X_i, r_{n,i})\right) \geq 1 - \frac{1}{2^b \log n}. \tag{21}$$

5.1 Conditions for homotopy reconstruction

In this subsection, we discuss the tightness of the conditions we have identified for guaranteeing the homotopy equivalence of the target space and the Čech complex and the Vietoris-Rips complex. We first argue that the maximum radius conditions in (14) and (17) are tight, as Example 24 shows that the Čech complex fails to be homotopy equivalent to $\mathbb{X}$ when $r_{\max} > \tau - d_{\mathbb{X}}(x)$ and the Vietoris-Rips complex fails to be homotopy equivalent to $\mathbb{X}$ when $r_{\max} > \sqrt{\frac{d+1}{2d}}\,(\tau - d_{\mathbb{X}}(x))$ and $d \leq 2$.

▶ **Example 24.** Let $\epsilon \in [0, 1)$ be fixed. Let $\mathbb{X} \subset \mathbb{R}^d$ be the unit sphere in $\mathbb{R}^d$, and let $\mathcal{X} = \{x_1, \ldots, x_n\} \subset (1-\epsilon)\mathbb{X}$ be a finite set of points on the sphere centered at 0 and of radius $1 - \epsilon$. Suppose that for some $\delta > \epsilon$, $\mathbb{X}$ is covered by δ-balls centered at $\mathcal{X}$, that is, $\mathbb{X} \subset \bigcup_{x\in\mathcal{X}} \mathbb{B}_{\mathbb{R}}(x, \delta)$. The reach of $\mathbb{X}$ equals to its radius 1.

For the ambient Čech complex, if $r \in (0, 1-\epsilon]^n$ and condition (15) is satisfied, then $\mathbb{X}$ is homotopy equivalent to $\check{\mathrm{C}}\mathrm{ech}_{\mathbb{X}}(\mathcal{X}, r)$ by Theorem 19. Now, suppose that $r_{\min} > 1 - \epsilon$. Then $0 \in \mathbb{B}_{\mathbb{R}^d}(x_i, r_{x_i})$ for all i, hence for any $y \in \bigcup_{i=1}^n \mathbb{B}_{\mathbb{R}^d}(x_i, r_{x_i})$, a line segment connecting 0 and y is contained in $\bigcup_{i=1}^n \mathbb{B}_{\mathbb{R}^d}(x_i, r_{x_i})$ as well. Hence $\bigcup_{i=1}^n \mathbb{B}_{\mathbb{R}^d}(x_i, r_{x_i})$ is contractible, and then from the usual Nerve Theorem, $\check{\mathrm{C}}\mathrm{ech}_{\mathbb{R}^d}(\mathcal{X}, r) \simeq \bigcup_{i=1}^n \mathbb{B}_{\mathbb{R}^d}(x_i, r_{x_i}) \simeq 0$. On the other hand, the $d-1$-th homology group of $\mathbb{X}$ is $H_{d-1}(\mathbb{X}) = \mathbb{Z}$, so $\mathbb{X}$ and $\check{\mathrm{C}}\mathrm{ech}_{\mathbb{R}^d}(\mathcal{X}, r)$ are not homotopy equivalent.

For the Vietoris-Rips complex, if $r \in \left(0, \sqrt{\frac{d+1}{2d}}(1-\epsilon)\right]^{d+1}$ and condition (18) is satisfied, then $\mathbb{X}$ is homotopy equivalent to $\mathrm{Rips}_{\mathbb{X}}(\mathcal{X}, r)$ by Theorem 20. Now, suppose each r_{x_i} is equal to some $r > \sqrt{\frac{d+1}{2d}}(1-\epsilon)$, and further suppose that $d \leq 2$ and $\delta < \frac{1}{2(1-\epsilon)} r_0 - \frac{\sqrt{3}}{4}$. When $d = 1$, then the Vietoris-Rips complex equals the ambient Čech complex, hence from the above argument, $\mathrm{Rips}(\mathcal{X}, r) = \check{\mathrm{C}}\mathrm{ech}_{\mathbb{R}^d}(\mathcal{X}, r) \simeq 0$. When $d = 2$, then $\mathrm{Rips}(\mathcal{X}, r) \cong \mathrm{Rips}\left(\frac{1}{1-\epsilon}\mathcal{X}, \frac{1}{1-\epsilon} r_0\right)$ and $\frac{1}{1-\epsilon}\mathcal{X} \subset \mathbb{X} \subset \bigcup_{i=1}^n \mathbb{B}_{\mathbb{R}^d}(\frac{1}{1-\epsilon}x_i, \delta)$ holds. Then $\frac{1}{1-\epsilon} r_0 - 2\delta > \frac{\sqrt{3}}{2}$,

and hence from Proposition 3.8, Corollary 4.5, Proposition 5.2 of [2], either $\mathrm{Rips}\,(\mathcal{X}, r) \simeq S^{2l+1}$ for some $l \geq 1$ or $\mathrm{Rips}\,(\mathcal{X}, r) \simeq \vee^c S^{2l}$ for some $l \geq 1$ and $c \geq 0$. In either case, $H_1(\mathrm{Rips}\,(\mathcal{X}, r)) = 0$. However, the $d-1$-th homology group of $\mathbb{X}$ is $H_{d-1}(\mathbb{X}) = \mathbb{Z}$, so $\mathbb{X}$ and $\mathrm{Rips}\,(\mathcal{X}, r)$ are not homotopy equivalent.

We then rephrase the conditions on $\epsilon := \max\{d_{\mathbb{X}}(x) : x \in \mathcal{X}\}$ and the covering radius δ in (15) and (18) in terms of the Hausdorff distance $d_H(\mathbb{X}, \mathcal{X})$. For simplicity, we consider the case when all the radii r_x's are equal, and we denote that common value as r. In general, the Hausdorff distance $d_H(\mathbb{X}, \mathcal{X})$ gives a bound for both ϵ and δ, that is, $\epsilon, \delta \leq d_H(\mathbb{X}, \mathcal{X})$. Let $\rho := \frac{d_H(\mathbb{X}, \mathcal{X})}{\tau}$. For the Čech complex, a sufficient condition for (15) is that for some $\frac{r}{\tau} \in (0, 1]$,

$$\rho + \sqrt{(\frac{r}{\tau})^2 - \tilde{l}^2 + \rho(2-\rho) - ((1-\rho)^2 - (\frac{r}{\tau})^2 + \tilde{l}^2 + (1-\rho_{\tilde{l}})^2)\left(\frac{1}{\sqrt{1-\tilde{r}^2_{\delta,c}}} - 1\right)} \leq \frac{r}{\tau},$$
$$\sqrt{\frac{d}{2(d+1)}}\left(\sqrt{\tilde{r}_b^2 - (2-\tilde{r}_b^2)\left(\frac{1}{\sqrt{1-\tilde{r}^2_{\delta,b}}} - 1\right)} + 2\rho\right) \leq r''_{\min}, \tag{22}$$

where

$$\tilde{l} := \frac{1}{2}\left(\frac{r}{\tau} - 1 + \sqrt{(1-\rho)^2 - (\frac{r}{\tau})^2} - \rho\right), \qquad \rho_{\tilde{l}} := 1 - \sqrt{(1-\rho)^2 - (\frac{r}{\tau})^2} + \tilde{l},$$
$$\tilde{r}^2_{\delta,c} := \min\left\{2\rho, \frac{1}{2}((\frac{r}{\tau})^2 - \tilde{l}^2 + \rho(2-\rho) + \rho_{\tilde{l}}(2-\rho_{\tilde{l}}))\right\},$$
$$r''_{\min} := \sqrt{1 - \rho(2-\rho) - \frac{(2 - (\frac{r}{\tau})^2 - \rho(2-\rho))^2}{4}},$$
$$\tilde{r}_b^2 := \frac{2\left((r''_{\min})^2 + \rho(2-\rho)\right)}{1 + \sqrt{1 - ((r''_{\min})^2 + \rho(2-\rho))}}, \qquad \tilde{r}^2_{\delta,b} := \min\left\{2\rho, \frac{1}{2}\tilde{r}_b^2\right\}.$$

And for the Vietoris-Rips complex, the sufficient condition for (18) is

$$\sqrt{\tilde{r}_b^2(r_0) - (2-\tilde{r}_b^2(r_0))\left(\frac{1}{\sqrt{1-\tilde{r}^2_{\delta,b}(r_0)}} - 1\right)} + 2\rho \leq 2r_0,$$
$$\sqrt{\frac{d}{2(d+1)}}\left(\sqrt{\tilde{r}_b^2(r''_{\min}) - (2-\tilde{r}_b^2(r''_{\min}))\left(\frac{1}{\sqrt{1-\tilde{r}^2_{\delta,b}(r''_{\min})}} - 1\right)} + 2\rho\right) \leq r''_{\min}, \tag{23}$$

where

$$r_0 := \sqrt{\frac{d+1}{2d}}(1-\rho), \qquad r''_{\min} := \sqrt{1 - \rho(2-\rho) - \frac{(2 - \frac{d+1}{2d}(1-\rho)^2 - \rho(2-\rho))^2}{4}},$$
$$\tilde{r}_b^2(t) := \frac{2\left(t^2 + \rho(2-\rho)\right)}{1 + \sqrt{1 - (t^2 + \rho(2-\rho))}}, \qquad \tilde{r}^2_{\delta,b}(t) := \min\left\{2\rho, \frac{1}{2}\tilde{r}_b^2(t)\right\}.$$

With the aid of a computer program, we can check that (22) is equivalent to $\rho \leq 0.01591\cdots$, and (23) is equivalent to $\rho \leq 0.07856\cdots$.

Now, we consider two specific cases. First, we consider the noiseless case $\mathcal{X} \subset \mathbb{X}$, that is, the data points lie in the target space. For that case, $\epsilon = 0$ and $\delta \leq d_H(\mathbb{X}, \mathcal{X})$. For the Čech complex, the sufficient condition for (15) is that for some $\frac{r}{\tau} \in (0, 1]$,

$$\rho + \sqrt{(\frac{r}{\tau})^2 - \tilde{l}^2 - (1-(\frac{r}{\tau})^2 + \tilde{l}^2 + (1-\rho_{\tilde{l}})^2)\left(\frac{1}{\sqrt{1-\tilde{r}^2_{\delta,c}}} - 1\right)} \leq \frac{r}{\tau},$$
$$\sqrt{\frac{d}{2(d+1)}}\left(\sqrt{\tilde{r}_b^2 - (2-\tilde{r}_b^2)\left(\frac{1}{\sqrt{1-\tilde{r}^2_{\delta,b}}} - 1\right)} + 2\rho\right) \leq r''_{\min}, \tag{24}$$

where

$$\tilde{l} := \frac{1}{2}\left(\frac{r}{\tau} - 1 + \sqrt{1-(\frac{r}{\tau})^2} - \rho\right), \qquad \rho_{\tilde{l}} := 1 - \sqrt{1-(\frac{r}{\tau})^2} + \tilde{l},$$
$$\tilde{r}^2_{\delta,c} := \min\left\{\rho^2, \frac{1}{2}((\frac{r}{\tau})^2 - \tilde{l}^2 + \rho_{\tilde{l}}(2-\rho_{\tilde{l}}))\right\},$$
$$r''_{\min} := \sqrt{1 - \frac{(2-(\frac{r}{\tau})^2)^2}{4}}, \qquad \tilde{r}_b^2 := \frac{2(r''_{\min})^2}{1+\sqrt{1-(r''_{\min})^2}}, \qquad \tilde{r}^2_{\delta,b} := \min\left\{\rho^2, \frac{1}{2}\tilde{r}_b^2\right\}.$$

For the Vietoris-Rips complex, a sufficient condition for (18) is

$$\sqrt{\tilde{r}_b^2(r_0) - (2-\tilde{r}_b^2(r_0))\left(\frac{1}{\sqrt{1-\tilde{r}^2_{\delta,b}(r_0)}} - 1\right)} + 2\rho \leq 2r_0,$$
$$\sqrt{\frac{d}{2(d+1)}}\left(\sqrt{\tilde{r}_b^2(r''_{\min}) - (2-\tilde{r}_b^2(r''_{\min}))\left(\frac{1}{\sqrt{1-\tilde{r}^2_{\delta,b}(r''_{\min})}} - 1\right)} + 2\rho\right) \leq r''_{\min}, \tag{25}$$

where

$$r_0 := \sqrt{\frac{d+1}{2d}}(1-\rho), \qquad r''_{\min} := \sqrt{1 - \frac{(2-\frac{d+1}{2d}(1-\rho)^2)^2}{4}},$$
$$\tilde{r}_b^2(t) := \frac{2t^2}{1+\sqrt{1-t^2}}, \qquad \tilde{r}^2_{\delta,b}(t) := \min\left\{\rho^2, \frac{1}{2}\tilde{r}_b^2(t)\right\}.$$

With the aid of a computer program, we can check that (24) is equivalent to $\rho \leq 0.02994\cdots$, and (25) is equivalent to $\rho \leq 0.1117\cdots$.

Second, we consider the asymptotic case, where we sample more and more points and $\mathcal{X}$ forms a dense cover of $\mathbb{X}$, that is, $\sup_{y\in\mathbb{X}} \inf_{x\in\mathcal{X}} \|x-y\| \to 0$. Still, we have a noisy sample distribution, that is, $\sup_{x\in\mathcal{X}} \inf_{y\in\mathbb{X}} \|x-y\| \nrightarrow 0$, so the Hausdorff distance $d_H(\mathbb{X},\mathcal{X})$ need not go to 0. In this case, $\delta \to 0$ and $\epsilon \leq d_H(\mathbb{X},\mathcal{X})$. For the Čech complex, a sufficient condition for (15) is that for some $\frac{r}{\tau} \in (0,1]$,

$$\sqrt{(\frac{r}{\tau})^2 - \tilde{l}^2 + \rho(2-\rho) - ((1-\rho)^2 - (\frac{r}{\tau})^2 + \tilde{l}^2 + (1-\rho_{\tilde{l}})^2)\left(\frac{1}{\sqrt{1-\tilde{r}^2_{\delta,c}}} - 1\right)} \leq \frac{r}{\tau},$$
$$\sqrt{\frac{d}{2(d+1)}}\sqrt{\tilde{r}_b^2 - (2-\tilde{r}_b^2)\left(\frac{1}{\sqrt{1-\tilde{r}^2_{\delta,b}}} - 1\right)} \leq r''_{\min}, \tag{26}$$

where

$$\tilde{l} := \frac{1}{2}\left(\frac{r}{\tau} - 1 + \sqrt{(1-\rho)^2 - (\frac{r}{\tau})^2}\right), \qquad \rho_{\tilde{l}} := 1 - \sqrt{(1-\rho)^2 - (\frac{r}{\tau})^2} + \tilde{l},$$
$$\tilde{r}^2_{\delta,c} := \min\left\{\rho(2-\rho), \frac{1}{2}((\frac{r}{\tau})^2 - \tilde{l}^2 + \rho(2-\rho) + \rho_{\tilde{l}}(2-\rho_{\tilde{l}}))\right\},$$
$$r''_{\min} := \sqrt{1 - \rho(2-\rho) - \frac{(2 - (\frac{r}{\tau})^2 - \rho(2-\rho))^2}{4}},$$
$$\tilde{r}^2_b := \frac{2\left((r''_{\min})^2 + \rho(2-\rho)\right)}{1 + \sqrt{1 - ((r''_{\min})^2 + \rho(2-\rho))}}, \qquad \tilde{r}^2_{\delta,b} := \min\left\{\rho(2-\rho), \frac{1}{2}\tilde{r}^2_b\right\}.$$

And for the Vietoris-Rips complex, a sufficient condition for (18) is

$$\sqrt{\tilde{r}^2_b(r_0) - (2 - \tilde{r}^2_b(r_0))\left(\frac{1}{\sqrt{1 - \tilde{r}^2_{\delta,b}(r_0)}} - 1\right)} \le 2r_0,$$
$$\sqrt{\frac{d}{2(d+1)}}\sqrt{\tilde{r}^2_b(r''_{\min}) - (2 - \tilde{r}^2_b(r''_{\min}))\left(\frac{1}{\sqrt{1 - \tilde{r}^2_{\delta,b}(r''_{\min})}} - 1\right)} \le r''_{\min}, \tag{27}$$

where

$$r_0 := \sqrt{\frac{d+1}{2d}}(1-\rho), \qquad r''_{\min} := \sqrt{1 - \rho(2-\rho) - \frac{(2 - \frac{d+1}{2d}(1-\rho)^2 - \rho(2-\rho))^2}{4}},$$
$$\tilde{r}^2_b(t) := \frac{2\left(t^2 + \rho(2-\rho)\right)}{1 + \sqrt{1 - (t^2 + \rho(2-\rho))}}, \qquad \tilde{r}^2_{\delta,b}(t) := \min\left\{\rho(2-\rho), \frac{1}{2}\tilde{r}^2_b(t)\right\}.$$

With the aid of a computer program, we can check that (26) is equivalent to $\rho \le 0.03440\cdots$, and (27) is equivalent to $\rho \le 0.07712\cdots$.

6 Discussion and Conclusion

Above we have provided conditions under which the ambient Čech complex $\check{\mathrm{C}}\mathrm{ech}_{\mathbb{R}^d}(\mathcal{X}, r)$ and the Rips complex $\mathrm{Rips}(\mathcal{X}, r)$ are homotopy equivalent to the target space $\mathbb{X}$ when the target space $\mathbb{X}$ has positive μ-reach τ^μ and the data points $\mathcal{X}$ being contained in the ϵ-offset $\mathbb{X}^\epsilon$ of $\mathbb{X}$. In this section, we further discuss our results and compare them with existing ones. For the comparison purpose, we consider the case when all the radii r_x's are equal, and we denote the common value as r. In these settings, an analogous homotopy equivalence between the ambient Čech complex $\check{\mathrm{C}}\mathrm{ech}_{\mathbb{R}^d}(\mathcal{X}, r)$ and the target space $\mathbb{X}$ is presented in [6] and [27].

First, we compare the upper bound for the maximum parameter value r in $\check{\mathrm{C}}\mathrm{ech}_{\mathbb{R}^d}(\mathcal{X}, r)$ or $\mathrm{Rips}(\mathcal{X}, r)$. When $\mu = 1$ so that $\tau^\mu = \tau$, our result suggests that the homotopy equivalences hold when $r \le \tau - \epsilon$ for $\check{\mathrm{C}}\mathrm{ech}_{\mathbb{R}^d}(\mathcal{X}, r)$ and $r \le \sqrt{\frac{d+1}{2d}}(\tau - \epsilon)$ for $\mathrm{Rips}(\mathcal{X}, r)$. As we have seen in Example 24, these bounds are optimal bounds. In [27], such a bound for $\check{\mathrm{C}}\mathrm{ech}_{\mathbb{R}^d}(\mathcal{X}, r)$ is $\frac{(\tau+\epsilon)+\sqrt{\tau^2+\epsilon^2-6\tau\epsilon}}{2}$ (see Proposition 7.1). Then our bound is strictly sharper than this when $\epsilon > 0$ since

$$\frac{(\tau+\epsilon) + \sqrt{\tau^2+\epsilon^2-6\tau\epsilon}}{2} < \frac{(\tau+\epsilon) + \sqrt{\tau^2+9\epsilon^2-6\tau\epsilon}}{2} = \tau - \epsilon.$$

In [6], a necessary condition for $\check{\mathrm{C}}\mathrm{ech}_{\mathbb{R}^d}(\mathcal{X}, r)$ in Section 5.3 is $r \le \tau - 3\epsilon$, so our upper bound is strictly better when $\epsilon > 0$.

Second, we compare the condition for the maximum possible ratio of the Hausdorff distance $d_H(\mathbb{X}, \mathcal{X})$ and the μ-reach τ^μ. For this case, as we have seen in Section 5.1, we can check that $\check{\text{C}}\text{ech}_{\mathbb{R}^d}(\mathcal{X}, r)$ is homotopy equivalent to $\mathbb{X}$ when $\frac{d_H(\mathbb{X},\mathcal{X})}{\tau} \leq 0.01591\cdots$. This result is worse than $3-\sqrt{8} \approx 0.1716\cdots$ in Proposition 7.1 in [27] or $\frac{-3+\sqrt{22}}{13} \approx 0.1300\cdots$ in Section 5.3 in [6]. Again from Section 5.1, we can check that $\text{Rips}(\mathcal{X}, r)$ is homotopy equivalent to $\mathbb{X}$ when $\frac{d_H(\mathbb{X},\mathcal{X})}{\tau} \leq 0.07856\cdots$. This result is better than $\frac{2\sqrt{2-\sqrt{2}}-\sqrt{2}}{2+\sqrt{2}} \approx 0.03412\cdots$ in Section 5.3 in [6].

Then we consider two specific cases. In the noiseless case $\mathcal{X} \subset \mathbb{X}$, the data points lie in the target space. In this case, as we have seen in Section 5.1, we can verify that $\check{\text{C}}\text{ech}_{\mathbb{R}^d}(\mathcal{X}, r)$ is homotopy equivalent to $\mathbb{X}$ when $\frac{d_H(\mathbb{X},\mathcal{X})}{\tau} \leq 0.02994\cdots$, and $\text{Rips}(\mathcal{X}, r)$ is homotopy equivalent to $\mathbb{X}$ when $\frac{d_H(\mathbb{X},\mathcal{X})}{\tau} \leq 0.1117\cdots$.

In the asymptotics case, as we sample more and more points from the target space, $\mathcal{X}$ forms a dense cover on $\mathbb{X}$, that is, $\sup_{y\in\mathbb{X}} \inf_{x\in\mathcal{X}} \|x-y\| \to 0$. For this case, as we have seen in Section 5.1, we can check that $\check{\text{C}}\text{ech}_{\mathbb{R}^d}(\mathcal{X}, r)$ is homotopy equivalent to $\mathbb{X}$ when $\frac{d_H(\mathbb{X},\mathcal{X})}{\tau} \leq 0.03440\cdots$, and $\text{Rips}(\mathcal{X}, r)$ is homotopy equivalent to $\mathbb{X}$ when $\frac{d_H(\mathbb{X},\mathcal{X})}{\tau} \leq 0.07712\cdots$.

Finally, we emphasize that our result also allows the radii $\{r_x\}_{x\in\mathcal{X}}$ to vary across the points $x \in \mathcal{X}$. Considering different radii is of practical interest if each data point has different importance. For example, one might want to use large radii on the flat and sparse region, while to use small radii on the spiky and dense region. However, there remain significant technical difficulties to allow for a different radius per each data point. As it can be seen in Figure 2, an uneven distribution of radii might lead to nonhomotopic between the Čech complex (or the Vietoris-Rips complex) and the target space. This situation has been studied in [12] for the union of balls under the reach condition, but not the Vietoris-Rips complex or under the μ-reach case. Theorem 20 or Corollary 22 first tackles this homotopy reconstruction problem with different radii for the Vietoris-Rips complex or under the μ-reach condition.

References

1 Eddie Aamari, Jisu Kim, Frédéric Chazal, Bertrand Michel, Alessandro Rinaldo, and Larry Wasserman. Estimating the reach of a manifold. *Electronic Journal of Statistics*, 13(1):1359–1399, 2019. doi:10.1214/19-EJS1551.

2 Michał Adamaszek and Henry Adams. The Vietoris-Rips complexes of a circle. *Pacific Journal of Mathematics*, 290(1):1–40, 2017. doi:10.2140/pjm.2017.290.1.

3 Michał Adamaszek, Henry Adams, and Florian Frick. Metric reconstruction via optimal transport. *SIAM Journal on Applied Algebra and Geometry*, 2(4):597–619, 2018. doi:10.1137/17M1148025.

4 Henry Adams and Joshua Mirth. Metric thickenings of euclidean submanifolds. *Topology and its Applications*, 254:69–84, 2019. doi:10.1016/j.topol.2018.12.014.

5 Paul Alexandroff. Über den allgemeinen Dimensionsbegriff und seine Beziehungen zur elementaren geometrischen Anschauung. *Mathematische Annalen*, 98(1):617–635, 1928. doi:10.1007/BF01451612.

6 Dominique Attali, André Lieutier, and David Salinas. Vietoris–rips complexes also provide topologically correct reconstructions of sampled shapes. *Computational Geometry*, 46(4):448–465, 2013. 27th Annual Symposium on Computational Geometry (SoCG 2011). doi:10.1016/j.comgeo.2012.02.009.

7 Anders Björner. *Topological Methods*, page 1819–1872. MIT Press, Cambridge, MA, USA, 1996.

8 Frédéric Chazal, David Cohen-Steiner, and André Lieutier. A sampling theory for compact sets in euclidean space. *Discrete & Computational Geometry*, 41(3):461–479, 2009. doi:10.1007/s00454-009-9144-8.

9 Frédéric Chazal, David Cohen-Steiner, André Lieutier, and Boris Thibert. Shape smoothing using double offsets. In *Proceedings of the 2007 ACM Symposium on Solid and Physical Modeling*, SPM '07, page 183–192, New York, NY, USA, 2007. Association for Computing Machinery. doi:10.1145/1236246.1236273.

10 Frédéric Chazal, Brittany Fasy, Fabrizio Lecci, Bertr, Michel, Aless, ro Rinaldo, and Larry Wasserman. Robust topological inference: Distance to a measure and kernel distance. *Journal of Machine Learning Research*, 18(159):1–40, 2018. URL: http://jmlr.org/papers/v18/15-484.html.

11 Frédéric Chazal and André Lieutier. Weak feature size and persistent homology: Computing homology of solids in $\mathbb{R}^n$ from noisy data samples. In *Proceedings of the Twenty-First Annual Symposium on Computational Geometry*, SCG '05, page 255–262, New York, NY, USA, 2005. Association for Computing Machinery. doi:10.1145/1064092.1064132.

12 Frédéric Chazal and André Lieutier. Smooth manifold reconstruction from noisy and non-uniform approximation with guarantees. *Computational Geometry*, 40(2):156–170, 2008. doi:10.1016/j.comgeo.2007.07.001.

13 Frédéric Chazal and Steve Yann Oudot. Towards persistence-based reconstruction in euclidean spaces. In *Proceedings of the Twenty-Fourth Annual Symposium on Computational Geometry*, SCG '08, page 232–241, New York, NY, USA, 2008. Association for Computing Machinery. doi:10.1145/1377676.1377719.

14 Antonio Cuevas. Set estimation: another bridge between statistics and geometry. *Boletín de Estadística e Investigación Operativa. BEIO*, 25(2):71–85, 2009.

15 Antonio Cuevas and Alberto Rodríguez-Casal. On boundary estimation. *Advances in Applied Probability*, 36(2):340–354, 2004. doi:10.1239/aap/1086957575.

16 Vin de Silva and Robert Ghrist. Coverage in sensor networks via persistent homology. *Algebraic & Geometric Topology*, 7:339–358, 2007. doi:10.2140/agt.2007.7.339.

17 Tamal K. Dey. *Curve and Surface Reconstruction: Algorithms with Mathematical Analysis*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, 2006. doi:10.1017/CBO9780511546860.002.

18 Jürgen Eckhoff. Chapter 2.1 - Helly, Radon, and Carathéodory type theorems. In P.M. GRUBER and J.M. WILLS, editors, *Handbook of Convex Geometry*, pages 389–448. North-Holland, Amsterdam, 1993. doi:10.1016/B978-0-444-89596-7.50017-1.

19 Herbert Edelsbrunner and John Harer. *Computational Topology: An Introduction*. American Mathematical Society, 2010. URL: http://www.ams.org/bookstore-getitem/item=MBK-69.

20 Herbert Edelsbrunner and Ernst P. Mücke. Three-dimensional alpha shapes. *ACM Transactions on Graphics*, 13(1):43–72, January 1994. doi:10.1145/174462.156635.

21 Herbert Federer. Curvature measures. *Transactions of the American Mathematical Society*, 93:418–491, 1959. doi:10.2307/1993504.

22 Karsten Grove. Critical point theory for distance functions. In *Differential geometry: Riemannian geometry (Los Angeles, CA, 1990)*, volume 54 of *Proc. Sympos. Pure Math.*, pages 357–385. Amer. Math. Soc., Providence, RI, 1993.

23 Allen Hatcher. *Algebraic topology*. Cambridge University Press, Cambridge, 2002.

24 Jean-Claude Hausmann. On the Vietoris-Rips complexes and a cohomology theory for metric spaces. In *Prospects in topology (Princeton, NJ, 1994)*, volume 138 of *Annals of Mathematics Studies*, pages 175–188. Princeton Univ. Press, Princeton, NJ, 1995.

25 Janko Latschev. Vietoris-Rips complexes of metric spaces near a closed Riemannian manifold. *Archiv der Mathematik*, 77(6):522–528, 2001. doi:10.1007/PL00000526.

26 John M. Lee. *Introduction to smooth manifolds*, volume 218 of *Graduate Texts in Mathematics*. Springer, New York, second edition, 2013.

27 Partha Niyogi, Stephen Smale, and Shmuel Weinberger. Finding the homology of submanifolds with high confidence from random samples. *Discrete & Computational Geometry*, 39(1-3):419–441, 2008. doi:10.1007/s00454-008-9053-2.

28 Vanessa Robins, James D. Meiss, and Elizabeth Bradley. Computing connectedness: disconnectedness and discreteness. *Physica D: Nonlinear Phenomena*, 139(3):276–300, 2000. doi:10.1016/S0167-2789(99)00228-6.

29 Amit Singer and Hau-Tieng Wu. Vector diffusion maps and the connection laplacian. *Communications on Pure and Applied Mathematics*, 65(8):1067–1144, 2012. doi:10.1002/cpa.21395.

30 Jean-Luc Starck, Vicent Martínez, D. Donoho, Ofer Levi, Philippe Querre, and Enn Saar. Analysis of the spatial distribution of galaxies by multiscale methods. *EURASIP Journal on Applied Signal Processing*, 2005(15):2455–2469, 2005. doi:10.1155/ASP.2005.2455.

A Quasi-Polynomial Algorithm for Well-Spaced Hyperbolic TSP

Sándor Kisfaludi-Bak
Max Planck Institute for Informatics, Saarbrücken, Germany
sandor.kisfaludi-bak@mpi-inf.mpg.de

Abstract

We study the traveling salesman problem in the hyperbolic plane of Gaussian curvature -1. Let α denote the minimum distance between any two input points. Using a new separator theorem and a new rerouting argument, we give an $n^{O(\log^2 n)\max(1,1/\alpha)}$ algorithm for Hyperbolic TSP. This is quasi-polynomial time if α is at least some absolute constant, and it grows to $n^{O(\sqrt{n})}$ as α decreases to $\log^2 n/\sqrt{n}$. (For even smaller values of α, we can use a planarity-based algorithm of Hwang et al. (1993), which gives a running time of $n^{O(\sqrt{n})}$.)

2012 ACM Subject Classification Theory of computation → Computational geometry; Theory of computation → Parameterized complexity and exact algorithms

Keywords and phrases Computational geometry, Hyperbolic geometry, Traveling salesman

Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.55

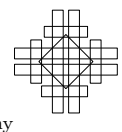
Related Version A full version of the paper is available at [19], `https://arxiv.org/abs/2002.05414`.

1 Introduction

The Traveling Salesman Problem (or TSP for short) is very widely studied in combinatorial optimization and computer science in general, with a long history. In the general formulation, we are given a complete graph G with positive weights on its edges. The task is to find a cycle through all the vertices (i.e., a Hamiltonian cycle) of minimum weight. The first non-trivial algorithm (with running time $O(2^n n^2)$) was given by Held and Karp [11], and independently by Bellman [3]. The problem was among the first problems to be shown NP-hard by Karp [16].

A very important case of TSP concerns metric weight functions, where the edge weights satisfy the triangle inequality. The problem has a (3/2)-approximation due to Christofides [6], which is still unbeaten. On the other hand, it is NP-hard to approximate Metric TSP within a factor of 123/122 [17]. Fortunately, the problem is more tractable in low-dimensional geometric spaces. Arora [1] and independently, Mitchell [21] gave the first polynomial time approximation schemes (PTASes) for the low-dimensional Euclidean TSP problem, where vertices correspond to points in $\mathbb{R}^d$ and the weights are defined by the Euclidean distance between the given points. The PTAS was later improved by Rao and Smith [23], and after two decades, several more general approximation schemes are known. In particular, there is a PTAS in metric spaces of bounded doubling dimension by Bartal *et al.* [2], and in metric spaces of negative curvature by Krauthgamer and Lee [20]. The PTAS of [20] applies in the hyperbolic plane.

Turning to the exact version of the problem in the geometric setting, we can again get significant improvements over the best known $O(2^n \operatorname{poly}(n))$ running time for the general version. In the Euclidean case, the first set of improved algorithms were proposed in the plane by Kann [15] and by Hwang *et al.* [12] with running time $n^{O(\sqrt{n})}$. Later, an algorithm in $\mathbb{R}^d$ with running time $n^{O(n^{1-1/d})}$ was given by Smith and Wormald [24]. The latest



improvement to $2^{O(n^{1-1/d})}$ by De Berg *et al.* [7] came with a matching lower bound under the Exponential Time Hypothesis (ETH) [13]. To our knowledge, the exact version of the problem in hyperbolic space has not been studied yet.

Given the history of the problem, the PTAS results and the Euclidean exact algorithm, one might expect that the hyperbolic case is very similar to the Euclidean, and a good hyperbolic TSP algorithm will have a running time of $n^{O(n^\delta)}$ for some constant δ. In this paper, we show that we can often get significantly faster algorithms. Let $\mathbb{H}^2$ denote the hyperbolic plane of Gaussian curvature -1. The first hopeful sign is that $\mathbb{H}^2$ exhibits special properties when it comes to intersection graphs. Recently, the present author has given quasi-polynomial algorithms for several classic graph problems in certain hyperbolic intersection graphs of ball-like objects [18]. The studied problems include Independent Set, Dominating Set, Steiner Tree, Hamiltonian Cycle and several other problems that are NP-complete in general graphs. Interestingly, a polynomial time algorithm was given for the Hamiltonian Cycle problem in hyperbolic unit disk graphs. The question arises whether a quasi-polynomial algorithm is available for TSP in $\mathbb{H}^2$? Given that the best running times for Hamiltonian Cycle in unit disk graphs in $\mathbb{R}^2$ and for Euclidean TSP are identical, perhaps even polynomial time is achievable for Hyperbolic TSP?

Unfortunately, a quasi-polynomial algorithm is unlikely to exist for the general Hyperbolic TSP problem: the lower bound of [8] in grids can be carried over to $\mathbb{H}^2$, which rules out a $2^{o(\sqrt{n})}$ algorithm under the Exponential Time Hypothesis (ETH) [13]. This however relies on embedding a grid-like structure in $\mathbb{H}^2$ efficiently, which seems to be possible only if the points are densely placed. Since $\mathbb{H}^2$ is locally Euclidean, it comes as no surprise that we cannot beat the Euclidean running time for dense point sets.

For this reason, we use a parameter measuring the density of the input point set. We say that the input point set P is *α-spaced* if for any pair of distinct points $p, p' \in P$, we have that $\operatorname{dist}(p, p') \geqslant \alpha$. Our main contribution is the following theorem.

▶ **Theorem 1.** *Let $P \subset \mathbb{H}^2$ be an α-spaced set of points. Then the shortest traveling salesman tour of P can be computed in $n^{O(\log^2 n)\cdot\max(1,1/\alpha))}$ time.*

Note that for $\alpha \geqslant 1$, this is a quasi-polynomial algorithm. In the full version [19] we show that for very dense inputs, it is unlikely that our running time can be improved significantly: we prove that there is no $2^{o(\sqrt{n})}$ algorithm for point sets of spacing $\Theta(1/\sqrt{n})$, unless the Exponential Time Hypothesis (ETH) fails.

Adapting algorithms from the Euclidean plane

Most algorithms for Euclidean TSP are difficult to adapt to the hyperbolic setting. The majority of known subexponential algorithms for Euclidean TSP (see [15, 24, 7]) are based on some version of the so-called *Packing Property* [7], which roughly states that for any disk δ of radius r, the number of segments in an optimal tour of length at least r that intersect δ is at most some absolute constant. This starting point is not available to us, since a direct adaptation of the Packing Property as stated above is false in $\mathbb{H}^2$. For example, we can create a regular n-gon where the length of each side is $c \log n$ for some constant c, and the inscribed circle has radius $r < c \log n$. The boundary of the n-gon is an optimal tour of its vertices, and the inscribed disk is intersected more than a constant times with tour segments of length at least r.

The only exact Euclidean TSP algorithm that directly carries over to $\mathbb{H}^2$ is the algorithm of Hwang, Chang and Lee [12], as it only relies on the fact that any optimal tour in the plane is crossing-free. Unfortunately, this algorithm has a running time of $n^{O(\sqrt{n})}$, which is far from our goal. Nonetheless, we can use this algorithm for the case when the point set P has close point pairs, that is, when $\alpha \leqslant \log^2 n/\sqrt{n}$. This is discussed further in Section 2.

Our techniques

To get a quasi-polynomial algorithm for $\alpha = \Omega(1)$, we need to prove our own separator theorem. The separator itself is fairly simple: it is a line segment of length $O(\log n)$. Due to the special properties of $\mathbb{H}^2$, optimal tours cannot go "around" this segment. The difficulty is to show that the line segment is crossed only $O(\log n)$ times by an optimal tour. We show that having a pair of "nearby"[1] tour edges crossing a certain neighborhood R of the segment can be ruled out with a rerouting argument that is reminiscent of the proof of the Packing Property in $\mathbb{R}^2$. This limits the number of segments crossing both R and the segment to $O(\log n)$. All other tour edges crossing the segment must have an endpoint in R. Since R is "narrow", it can contain at most $O(\log n)$ points from P, as P is α-spaced. These bounds together limit the number of tour edges crossing our segment to $O(\log n)$. With the separator at hand, we use a standard divide-and-conquer algorithm to prove Theorem 1. For values $\alpha \leqslant \log^2 n/\sqrt{n}$, we suggest using the algorithm of Hwang *et al.* [12].

Computational model

As our input, we get a list of points P with rational coordinates in the Poincaré disk model (which we briefly introduce in Section 2) and a rational number x. The goal is to decide if there is a tour of length at most x.

It is a common issue in computational geometry that one needs to be able to compare sums of distances. In geometric variants of TSP, this directly impacts the output, and unfortunately no method is known to tackle this in a satisfactory manner on a word-RAM machine. For this reason, most work in the area assumes that the computation is done on a real-RAM machine that can compute square roots exactly. Perhaps even less is known about comparing sums of distances in hyperbolic space. For this article, we work in a real-RAM that, in addition to taking square roots, is also capable of computing the natural logarithm $\ln(.)$.

2 Preliminaries

The hyperbolic plane and the Poincaré disk model

Introducing the hyperbolic plane properly is well beyond the scope of this section, but we list some important properties that we will be using. A detailed exposition can be found in several textbooks [4, 26, 10, 22].

The hyperbolic plane $\mathbb{H}^2$ is a homogeneous metric space with the key property that the area and circumference of disks grows exponentially with the radius, that is, a disk of radius r has area $4\pi \sinh^2(r/2)$ and circumference $2\pi \sinh(r)$. For $r > 1$, both the area and circumference are $\Theta(e^r)$. On the other hand, a small neighborhood of any point in the hyperbolic plane is very similar to a small neighborhood of a point in the Euclidean plane. More precisely, the disk of radius ε around a point in $\mathbb{H}^2$ and $\mathbb{R}^2$ have a smooth bijective mapping that preserve distances up to a multiplicative factor of $1 + f(\varepsilon)$, where $\lim_{\varepsilon \to 0} f(\varepsilon) = 0$.

The hyperbolic plane itself can be defined in many ways, but it is most convenient to take some region of $\mathbb{R}^2$, and equip it with a custom metric. Such definitions are also called *models* of the hyperbolic plane. In this article, we use the Poincaré disk model for all of the figures, however, most of the claims and proofs are model-independent.

[1] The absolute distance of crossing edges cannot be bounded; we use a special definition of "nearby".

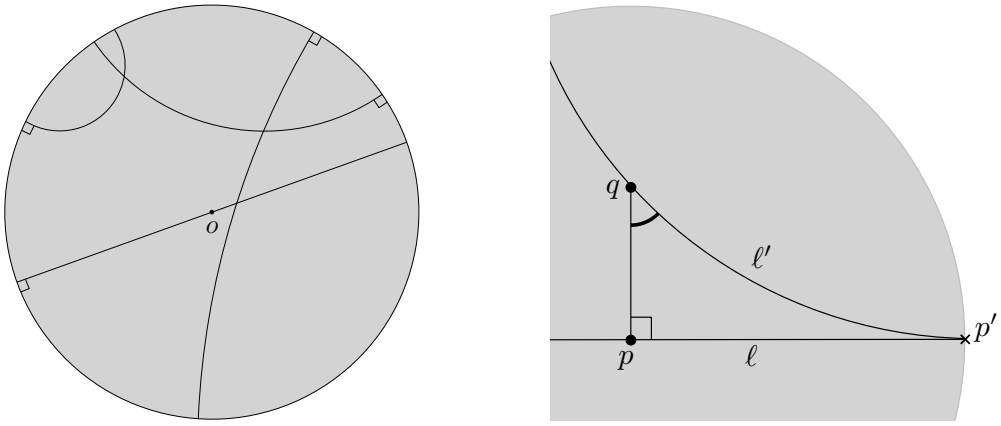


Figure 1 Left: lines in the Poincaré model. Right: the angle of parallelism for the length $|pq|$.

The Poincaré disk model is the open unit disk of $\mathbb{R}^2$ equipped with the distance function

$$\operatorname{dist}(u,v) = \cosh^{-1}\left(1 + 2\frac{\|u-v\|^2}{(1-\|u\|^2)(1-\|v\|^2)}\right),$$

where $\|.\|$ is the Euclidean norm.[2] The precise function here is irrelevant; we present the formula just as an example of defining a custom metric space.[3] We list some further properties of $\mathbb{H}^2$ used in the article.

- *Lines, angles, and ideal points.*
  In the Poincaré disk model hyperbolic lines appear as Euclidean circular arcs that are perpendicular to the unit circle, as illustrated on the left of Figure 1. In particular, hyperbolic lines through the center of the disk are diametrical segments of the unit disk. The model is *conformal*, that is, the angle of a pair of lines in $\mathbb{H}^2$ is the same as the angle of the corresponding arcs in $\mathbb{R}^2$. The points on the boundary of the disk are called *ideal points*.
- *Angle of parallelism.*
  Let $p, q \in \mathbb{H}^2$ and let ℓ be a line through p that is perpendicular to pq, and let p' be an ideal point of ℓ, see the right hand side of Figure 1. Let ℓ' be the line through q and p'. Note that ℓ and ℓ' are disjoint lines in the open disk; they are called *limiting parallels*. The angle $\sphericalangle pqp'$ is called the *angle of parallelism*, which only depends on the length of the segment pq the following way [22].

$$\tan(\sphericalangle pqp') = \frac{1}{\sinh(|pq|)} \tag{1}$$

- *Hypercycles or equidistant curves.*
  The set of points at a given distance ϱ from a line ℓ is not a line, but it forms a *hypercycle* in $\mathbb{H}^2$. A hypercycle has two arcs, one on each side of ℓ. In the Poincaré model, a hypercycle for a line ℓ consists of two circular arcs, ending at the same ideal points as ℓ.

[2] As $\cosh^{-1}(x) = \ln(x + \sqrt{x^2-1})$, the distance of two points in the Poincaré disk model with given Euclidean coordinates can be computed on a real-RAM machine which is capable of taking square roots and computing ln(.).

[3] If we need to calculate angles, curve length, and area, we should define the metric tensor instead: $ds^2 = 4\frac{\|dx\|^2}{(1-\|x\|^2)^2}$ [5].

- *Optimal tours in $\mathbb{H}^2$ and crossings.*
  An optimal traveling salesman tour will consist of geodesics between pairs of input points, i.e., hyperbolic segments, just as in $\mathbb{R}^2$. Moreover, the triangle inequality implies that any self-crossing tour (where two segments pp' and qq' cross) can be shortened. Thus, optimal tours in $\mathbb{H}^2$ are non-crossing.

Getting a subexponential algorithm for all values of α

We can give the following more general formulation of the result of [12].

▶ **Theorem 2** (Hwang, Chang and Lee [12], stated generally). *Let P be a set of n points in $\mathbb{R}^2$, and let $w : \binom{P}{2} \to \mathbb{R}$ be a weight function on the (straight) segments defined by the point pairs. Suppose that the optimal TSP tour of P with respect to w is crossing-free. Then there is an algorithm to compute this optimal tour in $n^{O(\sqrt{n})}$ time.*

We convert our initial point set P in the Poincaré model to the Beltrami-Klein model of $\mathbb{H}^2$ to get a point set P_{BK}. In the Beltrami-Klein model, Euclidean segments inside the open unit disk are (geodesic) segments of $\mathbb{H}^2$. Since the optimal hyperbolic TSP tour is crossing-free, the tour in the Beltrami-Klein model is a polygon with vertex set P_{BK}. The hyperbolic distances can be used as weights on all segments with endpoints from P_{BK}, and we can apply Theorem 2 to get an $n^{O(\sqrt{n})}$ algorithm regardless of the value of α.

3 A separator for Hyperbolic TSP

Centerpoint and a separating line

It has already been observed in [18] that for any set $P \subset \mathbb{H}^2$ of n points there exists a point $q \in \mathbb{H}^2$ such that for any line ℓ through q the two open half-planes with boundary ℓ both contain at most $\frac{2}{3}n$ points from P, that is, the line ℓ is a 2/3-*balanced separator* of P. Such a point q is called a *centerpoint* of P. It has been observed in [18] that given P, a centerpoint of P can be computed using a Euclidean centerpoint algorithm, which takes linear time [14].

It is now easy to prove that we can find a balanced line separator that has a small neighborhood empty of input points.

▶ **Lemma 3.** *Given a point set $P \subset \mathbb{H}^2$, there exists a point $q \in \mathbb{H}^2$ and there exists a line ℓ through q such that P is disjoint from the open double cone with center q, axis ℓ and half-angle $\frac{\pi}{2n}$. Any such line ℓ is a 2/3-balanced separator of P, and given P, a suitable point q and line ℓ can be found in linear time.*

Proof. Let q be a centerpoint of P. For each point $p \in P$, let ℓ_p be the line through q and p. Since we have defined n lines through q, there is a pair of consecutive lines $\ell_p, \ell_{p'}$ whose acute angle is at least π/n. Let ℓ be the angle bisector of ℓ_p and $\ell_{p'}$. Then ℓ clearly has the desired properties, and the centerpoint q, the lines $\ell_p, \ell_{p'}$ and the line ℓ can all be computed in linear time. ◀

We can extend Lemma 3 to get balance with respect to a subset $B \subseteq P$, that is, both half-planes bounded by ℓ would contain at most $\frac{2}{3}|B|$ points of B. One only needs to set q to be the centerpoint of B instead of P.

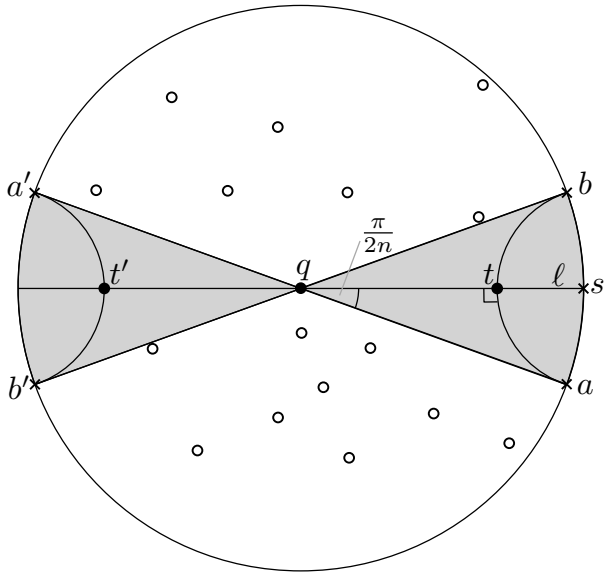


Figure 2 The empty double cone with axis ℓ.

Defining a region around the separator

From this point onwards, q denotes a centerpoint of P, and ℓ is a line through q with the properties from Lemma 3. Let C denote the double cone of center q, axis ℓ and half-angle $\frac{\pi}{2n}$, see Figure 2. Note that by Lemma 3, we have that $C \cap P = \emptyset$. Let s be an ideal point of ℓ, and let a, b be ideal points on the boundary of C, such that $\sphericalangle aqs = \sphericalangle sqb = \frac{\pi}{2n}$. Let $t = ab \cap \ell$. Notice that qta is a right-angle triangle with ideal point a, and it has angle $\frac{\pi}{2n}$ at q. Therefore, $\frac{\pi}{2n}$ is the angle of parallelism for the distance $|qt|$, and it satisfies

$$\sinh(|qt|) = \frac{1}{\tan(\frac{\pi}{2n})}. \tag{2}$$

The line ab splits $\mathbb{H}^2$ into two open half-planes: the side H_q containing q, and the side H_s that has s on its boundary. Note that $H_s \subset C$, therefore $P \subset H_q$. Consequently, all segments of the tour are contained in H_q. We mirror a, b and t to the point q; let a', b' and t' denote the resulting points respectively. By our earlier observation, the entire tour is contained in the geodesically convex region between the lines ab and $a'b'$, and any tour segment intersecting ℓ will intersect it somewhere on the segment tt'.

Let a_t and b_t be the points on $a'b'$ at distance ϱ from t, where $\varrho \in (0, \alpha/2)$ is a suitable number that will be defined later. Let a'_t and b'_t denote the analogous points on $a'b'$, see Figure 3. Let R denote the region of the hyperbolic plane consisting of all points between ab and $a'b'$ whose distance from tt' is at most ϱ. The resulting shape R is geodesically convex; its boundary consists of two segments (a_tb_t and $a'_tb'_t$), and two hypercycle arcs, denoted by $\widehat{a_tb'_t}$ and $\widehat{b_ta'_t}$. In general, for two points u, v on one of these hypercycle arcs, let $\widehat{uv}$ denote the arc between them, and let $|\widehat{uv}|$ be the length of this arc.

Note that any tour segment that connects points on two different sides of ℓ also intersects R. A tour segment that intersects R can have $0, 1$ or 2 endpoints in R. A segment with exactly 1 endpoint in R is called *entering*. As R is geodesically convex, segments with both endpoints in R are entirely contained in R. All other tour segments crossing ℓ must intersect at least one of $\widehat{a_tb'_t}$ and $\widehat{b_ta'_t}$. We say that a segment *crosses* R if it intersects both $\widehat{a_tb'_t}$ and $\widehat{b_ta'_t}$. (It is possible that a segment whose endpoints lie outside R on the same side of ℓ intersect one of these arcs twice. These segments are not relevant for our algorithm.)

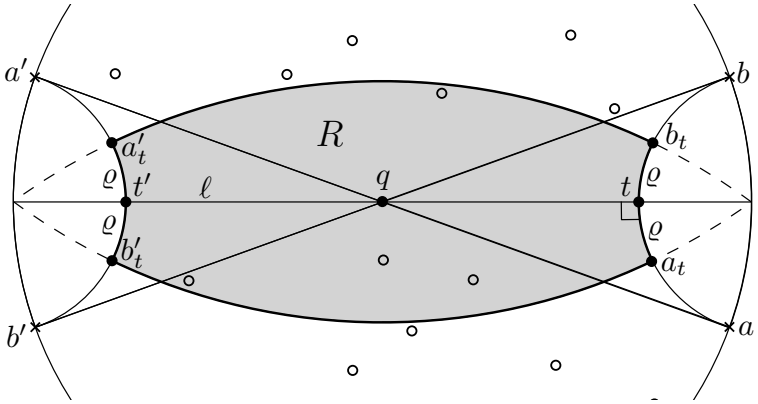


Figure 3 The construction of the region R.

The rest of this section focuses on the following main lemma.

▶ **Lemma 4.** *The region R has the following properties:*

(i) $|R \cap P| < n_{in} \stackrel{\text{def}}{=} 1 + \frac{2(\ln n+1)}{\alpha - 2\varrho}$

(ii) *There are less than* $s_{cr} \stackrel{\text{def}}{=} 2 + \frac{2(\ln n+1)\cosh \varrho}{\varrho}$ *tour segments that cross R.*

The proof requires that we explore the geometry of R more thoroughly.

▶ **Lemma 5.** *We have* $|qt| < \ln n + 1$, *and* $|\widehat{a_t b'_t}| = |\widehat{b_t a'_t}| < 2(\ln n + 1)\cosh \varrho$.

Proof. We first prove our bound on $|qt|$. Note that $\sinh(.)$ is monotone increasing and $\sinh(|qt|) = \frac{1}{\tan(\frac{\pi}{2n})}$ by (2), so it suffices to show that $\sinh(\ln n + 1) > \frac{1}{\tan(\frac{\pi}{2n})}$. Indeed,

$$\sinh(\ln n + 1) = \frac{en - \frac{1}{en}}{2} > n \quad \text{and} \quad \frac{1}{\tan(\frac{\pi}{2n})} < \frac{1}{\frac{3}{2n}} < n.$$

The arc length of the equidistant hypercycle of base b and distance ϱ is $b\cosh\varrho$ according to [25], therefore $|\widehat{a_t b'_t}| = |tt'|\cosh(\varrho) = 2|qt|\cosh(\varrho) < 2(\ln n + 1)\cosh\varrho$. ◀

Ruling out dense crossings

Our next ingredient for the proof is to show that if two segments cross R very close to each other, then they cannot both be in an optimal tour.

▶ **Lemma 6.** *Let $p_1 p_2 \dots p_i p_{i+1} \dots p_{n-1} p_n$ be an optimal tour on P where both p_1p_2 and p_ip_{i+1} cross R, and where p_1, p_i and a_t lie on the same side of ℓ. Let $p'_1 = p_1p_2 \cap \widehat{a_t b'_t}$, and define p'_2, p'_i and p'_{i+1} analogously. Then $|\widehat{p'_1 p'_i}| + |\widehat{p'_2 p'_{i+1}}| \geqslant 4\varrho$.*

Proof. We can create a new tour by removing the segments $p'_1p'_2$ and $p'_ip'_{i+1}$, and replacing them with $p'_1p'_i$ and $p'_2p'_{i+1}$, see Figure 4. The resulting tour is

$$p_1 p'_1 p'_i p_i p_{i-1} p_{i-2} \dots p_2 p'_2 p'_{i+1} p_{i+1} p_{i+2} \dots p_n.$$

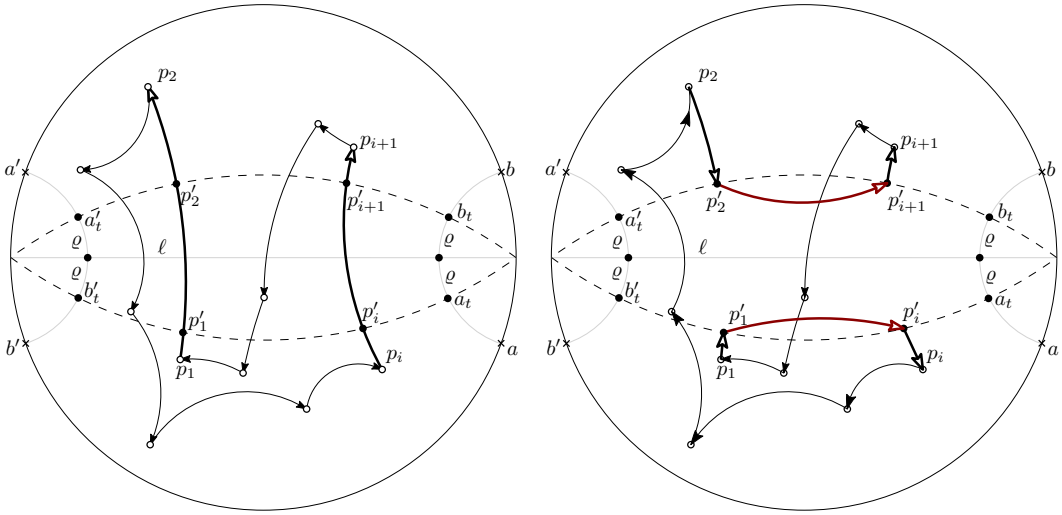


Figure 4 Rerouting two crossing edges (p_1p_2 and p_ip_{i+1}) into a different tour.

Note that this tour contains all the input points.[4] Since the only difference between the tours is that $p'_1p'_2$ and $p'_ip'_{i+1}$ are only present in the optimal tour and $p'_1p'_i$ and $p'_2p'_{i+1}$ are only present in the new tour, by the optimality of $p_1 \dots p_n$ we have that

$$0 \geqslant |p'_1p'_2| + |p'_ip'_{i+1}| - |p'_1p'_i| - |p'_2p'_{i+1}|.$$

Note that $|p'_1p'_2| \geqslant 2\varrho$ by the definition of R, and analogously $|p'_ip'_{i+1}| \geqslant 2\varrho$. Therefore we have

$$0 \geqslant |p'_1p'_2| + |p'_ip'_{i+1}| - |p'_1p'_i| - |p'_2p'_{i+1}| \geqslant 4\varrho - \widehat{|p'_1p'_i|} - \widehat{|p'_2p'_{i+1}|},$$

which concludes the proof. ◀

We can now prove Lemma 4.

Proof of Lemma 4.

(i) For a point $p \in P \cap R$, let p_ℓ denote the point on ℓ for which pp_ℓ is perpendicular to ℓ. Let $p, p' \in P \cap R$ be points such that p_ℓ, p'_ℓ are consecutive on ℓ (i.e., there is no $p'' \in P \cap R$ such that $p''_\ell \in p_\ell p'_\ell$). By the triangle inequality, $|pp_\ell| + |p_\ell p'_\ell| + |p'_\ell p'| \geqslant |pp'|$, and $|pp'| \geqslant \alpha$ since P is α-spaced. By the definition of R and ϱ, we also have that $|pp_\ell| \leqslant \varrho$ and $|p'_\ell p'| \leqslant \varrho$. Consequently,

$$|p_\ell p'_\ell| \geqslant \alpha - 2\varrho. \tag{3}$$

We can apply this inequality to all consecutive pairs $p_\ell p'_\ell$. Since all the points p_ℓ lie on the segment tt', the total length of the segments $p_\ell p'_\ell$ cannot exceed $|tt'|$. It follows that

$$|P \cap R| \leqslant 1 + \left\lfloor \frac{|tt'|}{\alpha - 2\varrho} \right\rfloor < 1 + \frac{2(\ln n + 1)}{\alpha - 2\varrho},$$

where the second inequality uses our bound from Lemma 5.

[4] This is generally not an optimal tour as it can be further shortened into $p_1p_ip_{i-1}p_{i-2} \dots p_2p_{i+1}p_{i+2} \dots p_n$.

(ii) Let $p_1 \dots, p_n$ be an optimal tour, and let p_ip_{i+1} be an edge crossing R. (Indices are defined modulo n.) Note that p_ip_{i+1} can cross R in two directions: either from the side of a to the side of b or the other way around. By Lemma 6, consecutive crossings p_ip_{i+1} and p_jp_{j+1} in the same direction use at least a total arc length of 4ϱ on the arcs $\overset{\frown}{a_tb'_t}$ and $\overset{\frown}{b_ta'_t}$. Since the total length of these arcs is less than $4(\ln n + 1)\cosh\varrho$ by Lemma 5, the number of crossings in one direction is less than

$$1 + \left\lfloor \frac{4(\ln n + 1)\cosh\varrho}{4\varrho} \right\rfloor \leqslant 1 + \frac{(\ln n + 1)\cosh\varrho}{\varrho}.$$

Consequently, the total number of crossings (in both directions) is less than

$$2 + \frac{2(\ln n + 1)\cosh\varrho}{\varrho}.$$

This concludes the proof. ◀

4 A divide-and-conquer algorithm

In order for a divide-and-conquer approach to work for Euclidean TSP, one should be able to solve subproblems with partial tours. We follow the terminology and definitions of De Berg *et al.* [7] here. Let M be a perfect matching on a set $B \subseteq P$ of so-called *boundary points*. We say that a collection $\mathcal{P} = \{\pi_1, \dots, \pi_{|B|/2}\}$ of paths *realizes* M *on* P if (i) for each pair $(p, q) \in M$ there is a path $\pi_i \in \mathcal{P}$ with p and q as endpoints, and (ii) the paths together visit each point $p \in P$ exactly once. We define the length of a path π_i to be the sum of the lengths of its edges, and we define the total length of $\mathcal{P}$ to be the sum of the lengths of the paths $\pi_i \in \mathcal{P}$. The subproblems that arise in our divide-and-conquer algorithm can be defined as follows.

Hyperbolic Path Cover
Input: A point set $P \subset \mathbb{H}^2$, a set of boundary points $B \subseteq P$, and a perfect matching M on B.
Task: Find a collection of paths of minimum total length that realizes M on P.

Let $PathTSP(P, B, M)$ be the optimal tour length for the instance (P, B, M). Note that we can solve Hyperbolic TSP on a point set P by solving Hyperbolic Path Cover $n - 1$ times on P with $B := \{p, q\}$ and $M := \{(p, q)\}$ for each $q \in P \setminus \{p\}$, and answering

$$\min_{q \in P \setminus \{p\}} \Big(PathTSP\big(P, \{p, q\}, \{(p, q)\}\big) + |pq| \Big).$$

4.1 Algorithm

Our algorithm is a standard divide and conquer algorithm that is very similar to [24] and [7]. The algorithm requires knowledge of the initial value of α; we can compute this before the first call in $O(n^2)$ time. We give a pseudocode and also explain the steps below. In the explanation, we sometimes regard sets of segments with endpoints in P as subgraphs of the complete graph with vertex set P.

As a first step, we run a brute-force algorithm (comparing all path covers of P) if the input points set P has size at most the threshold t, where t will be a large constant. On line 2, we check the size of the boundary. If it is less than $\max\left(\frac{40\ln|P|}{\alpha}, 8\ln|P|\right)$, then we

Algorithm 1 *HyperbolicTSP*(P, B, M, α).

Input: A set $P \subset \mathbb{R}^d$, a subset $B \subseteq P$, a perfect matching $M \subseteq \binom{B}{2}$, and initial spacing α
Output: The minimum length of a path cover of P realizing the matching M on B

1: **if** $|P| \leqslant t$ **then return** *BruteForceTSP*(P, B, M)
2: **if** $|B| < \max\left(\frac{40 \ln |P|}{\alpha}, 8 \ln |P|\right)$ **then**
3: Compute a centerpoint q of P, the line ℓ through q and the region R.
4: **else**
5: Compute a centerpoint q of B, the line ℓ through q and the region R.
6: $\mathsf{Cr} \leftarrow \{pp' \mid p, p' \in P,\ pp' \text{ crosses } R\}$, $\mathsf{End} \leftarrow \{pp' \mid p \in R \cap P,\ p' \in P, pp' \text{ intersects } \ell\}$
7: $mincost \leftarrow \infty$
8: **for all** $S_{cr} \subseteq \mathsf{Cr}$, $|S_{cr}| \leqslant s_{cr}$ **do**
9: **for all** $S_{end} \subseteq \mathsf{End}$, the maximum degree of S_{end} is at most 2 **do**
10: $P_1, P_2 \leftarrow$ uncovered vertices on each side of ℓ
11: $B_1, B_2 \leftarrow$ boundary vertices of $S_{cr} \cup S_{end}$ and points of B in P_1 (resp., P_2).
12: **for all** perfect matchings M_1 on B_1 and M_2 on B_2 **do**
13: **if** $M_1 \cup M_2 \cup S_{cr} \cup S_{end}$ realize M **then**
14: $c_1 \leftarrow HyperbolicTSP(P_1, B_1, M_1, \alpha)$
15: $c_2 \leftarrow HyperbolicTSP(P_2, B_2, M_2, \alpha)$
16: **if** $c_1 + c_2 + \mathbf{length}(S_{cr} \cup S_{end}) < mincost$ **then**
17: $mincost \leftarrow c_1 + c_2 + \mathbf{length}(S_{cr} \cup S_{end})$
18: **return** *mincost*

compute the centerpoint of P, the line ℓ with the empty cone according to Lemma 3, and the region R. Otherwise (similarly to [7]), we need to shrink the boundary, so we use a line ℓ through the centerpoint of B instead. Next, we define the segment set Cr as the set of segments pp' that cross R, and End as the set of segments intersecting ℓ that have at least one endpoint in R. We initialize the returned value *mincost* to infinity.

On line 8, we iterate over all segment sets $S_{cr} \subseteq \mathsf{Cr}$ with $|S_{cr}| \leqslant s_{cr}$, where s_{cr} is our bound on the number of crossing segments from Lemma 4. The algorithm considers S_{cr} to be the set of segments crossing R. Next, we iterate over all the sets $S_{end} \subseteq \mathsf{End}$ where each point of P has at most two incident segments from S_{end}. The algorithm considers S_{end} to be the set of segments crossing ℓ with at least one endpoint in R.

Each point in B needs to have one adjacent segment in the optimum tour $\mathcal{P}$, and each point in $P \setminus B$ needs two such points. We say that a point $p \in B$ (resp., $p \in P \setminus B$) is *uncovered* if its degree in $S_{cr} \cup S_{end}$ is less than 1 (resp., 2). We denote by P_1 and P_2 the set of uncovered points on each side of ℓ. A point $p \in P_1$ is a boundary point if $p \in B$ and p is not an endpoint of $S_{cr} \cup S_{end}$, or $p \in P \setminus B$ and it has degree 1 in $S_{cr} \cup S_{end}$. We let B_1 denote the boundary points in P_1. Similarly, B_2 is the set of boundary points in P_2.

Line 13 proceeds by iterating over all perfect matchings M_1 on B_1 and M_2 on B_2. If the graph on $B_1 \cup B_2 \cup B$ formed by $M_1 \cup M_2 \cup S_{cr} \cup S_{end}$ is a set of paths such that contracting edges with an endpoint in $(B_1 \cup B_2) \setminus B$ results in M, then we say that $M_1 \cup M_2 \cup S_{cr} \cup S_{end}$ realize M. If this is the case for a particular choice M_1, M_2, then on lines 14 and 15 we recurse on both P_1 and P_2. The resulting path covers together with $S_{cr} \cup S_{end}$ form a path cover realizing M: if their length is shorter than *mincost*, then we update *mincost*. After the loops have ended, we return *mincost*.

We can also compute the optimum tour itself with a small modification of the algorithm.

Correctness

The same algorithmic strategy has been used several times in the literature [24, 7], so we only give a brief justification. Given an optimal path cover $\mathcal{P}$, the set S_{cr} of segments in $\mathcal{P}$ crossing R has size at most s_{cr} by Lemma 4. The set of segments S_{end} with one endpoint in R has degree at most two at each point of $R \cap P$. Consequently, both sets will be considered in Line 8 and Line 9. The segments of $\mathcal{P}$ not in $S_{cr} \cup S_{end}$ form a path cover of P_1 and P_2 with boundary set B_1 and B_2. These path covers realize some perfect matchings M_1 and M_2 on B_1 and B_2 respectively. The matchings M_1 and M_2 together with $S_{cr} \cup S_{end}$ realize M, therefore M_1 and M_2 will be considered in the loop at line 13. These path covers must be optimal by the optimality of $\mathcal{P}$.

4.2 Analyzing the running time

All non-recursive steps can be handled in $O(n^2)$ time. The number of segment sets S_{cr} to be considered in line 8 is at most $\binom{|\mathsf{Cr}|}{s_{cr}} = O(n^{2s_{cr}})$, since $|\mathsf{Cr}| = O(n^2)$. The number of segment sets S_{end} to be considered is at most $O(n^{2|R\cap P|}) \leqslant O(n^{2n_{in}})$. By Lemma 4, the loop in line 8 has at most

$$O\left(n^{2\left(1+\frac{2(\ln n+1)}{\alpha-2\varrho}+2+\frac{2(\ln n+1)\cosh\varrho}{\varrho}\right)}\right) = O\left(n^{4(\ln n+1)\left(\frac{1}{\alpha-2\varrho}+\frac{\cosh\varrho}{\varrho}\right)+O(1)}\right) \tag{4}$$

iterations. Instead of trying to minimize this expression by our choice of ϱ, we settle for something that is easy to handle. Let

$$\varrho \stackrel{\text{def}}{=} \min\left(\frac{3}{10}\alpha, \frac{12}{10}\right).$$

The exponent of (4) can be bounded the following way. If $\alpha < 4$, then $\varrho = \frac{3}{10}\alpha$, and $\cosh(\varrho) < 1.82$, so we get

$$\begin{aligned} 2(s_{cr}+n_{in}) &= 4(\ln n+1)\left(\frac{1}{\alpha-2\varrho}+\frac{\cosh\varrho}{\varrho}\right)+O(1) \\ &< 4(\ln n+1)\left(\frac{1}{\frac{4}{10}\alpha}+\frac{1.82}{\frac{3}{10}\alpha}\right)+O(1) \\ &< 4(\ln n)\left(\frac{8.57}{\alpha}\right)+O(1/\alpha) \\ &< 35\frac{\ln n}{\alpha}, \end{aligned} \tag{5}$$

where the last step uses that n is large enough, which we can ensure by setting the threshold t in Line 1 large enough. If $\alpha \geqslant 4$, then $\varrho = 1.2$:

$$\begin{aligned} 2(s_{cr}+n_{in}) &= 4(\ln n+1)\left(\frac{1}{\alpha-2\varrho}+\frac{\cosh\varrho}{\varrho}\right)+O(1) \\ &< 4(\ln n+1)\left(\frac{1}{\alpha-2.4}+\frac{1.82}{1.2}\right)+O(1) \\ &< 7\ln n. \end{aligned} \tag{6}$$

Next, we will analyze the loop at line 13, but this will require a bound on the size of the boundary set B.

▶ **Lemma 7.** *The size of the boundary set B is at most* $\max(\frac{60\ln|P|}{\alpha}, 12\ln|P|)$ *at every recursion level of HyperbolicTSP.*

Proof. The statement holds for the initial call as we have $|B| = 2$ and $|P| = n$ there.

Notice that if $|B| < \max(\frac{40\ln|P|}{\alpha}, 8\ln|P|)$, then we use the branch on line 3. Consequently, the boundary set B_1 (and B_2) in the new recursive call always has size at most $|B|+(s_{cr}+n_{in})$. So by induction and the bounds (5) and (6), we have that

$$\begin{aligned}|B_1| &\leqslant \max\left(\frac{40\ln|P|}{\alpha}, 8\ln|P|\right) + \max\left(\frac{17.5\ln|P|}{\alpha}, 3.5\ln|P|\right)\\ &< \max\left(\frac{57.5\ln|P|}{\alpha}, 11.5\ln|P|\right)\\ &< \max\left(\frac{60\ln|P_1|}{\alpha}, 12\ln|P_1|\right),\end{aligned}$$

where we use $|P_1| \geqslant |P|/3 \Rightarrow \ln(|P|) < \ln(P_1) + 1.1$; therefore, the last inequality holds if we set the threshold t large enough.

In case of $|B| > \max(\frac{40\ln|P|}{\alpha}, 8\ln|P|)$, we use the branch on line 5. We have that $|B_1| \leqslant \frac{2}{3}|B| + (s_{cr} + n_{in})$. By induction, we still have $|B| < \max(\frac{60\ln|P|}{\alpha}, 12\ln|P|)$, so

$$\begin{aligned}|B_1| &\leqslant \frac{2}{3}\max\left(\frac{60\ln|P|}{\alpha}, 12\ln|P|\right) + \max\left(\frac{17.5\ln|P|}{\alpha}, 3.5\ln|P|\right)\\ &< \max\left(\frac{60\ln|P_1|}{\alpha}, 12\ln|P_1|\right).\end{aligned}$$

◀

The number of perfect matchings on a boundary set B_1 is at most $|B_1|^{O(|B_1|)}$. Let $b \stackrel{\text{def}}{=} \max(\frac{60\ln|P|}{\alpha}, 12\ln|P|)$ be the bound acquired above. The number of iterations of the loop at line 13 is at most $b^{O(b)}$. If $\alpha \geqslant 4$, then this is $(\ln|P|)^{O(\ln|P|)} = |P|^{O(\ln\ln|P|)} < |P|^{\epsilon\ln|P|}$ for any $\epsilon > 0$, as long as $|P|$ is large enough. If $\alpha < 4$, then we get

$$b^{O(b)} = \left(\frac{\ln n}{\alpha}\right)^{O(\frac{\ln n}{\alpha})} = n^{O(\frac{1}{\alpha}(\ln\ln n + \ln(1/\alpha))}.$$

As long as $1/\alpha = n^{o(1)}$, this term is insignificant compared to the iterations of the other loop. Otherwise, we have $\frac{1}{\alpha} \leqslant \sqrt{n}$, and therefore

$$b^{O(b)} = n^{O(\frac{1}{\alpha}(\ln\ln n + \ln(1/\alpha))} = n^{O(\frac{\log n}{\alpha})}.$$

▶ Remark 8. If one wants to optimize the leading coefficient in the exponent of the eventual running time, then it is possible to modify the algorithm to use only $c^{|B_1|}$ matchings for M_1 as all other matchings lead to crossings. See for example the technique in [9]. As a consequence, the leading coefficient will not be influenced by the second loop at all. However, this effort would be in vain if there exists a significantly better algorithm for $\alpha \leqslant 1$, say $n^{O(\log n\cdot(1/\alpha))}$ or even $n^{O(1/\alpha)}$, which we cannot rule out yet.

The following lemma finishes the proof of Theorem 1.

▶ **Lemma 9.** *The running time of HyperbolicTSP on our initial call is* $n^{O(\log^2 n)\max(1,1/\alpha)}$.

Proof. By the analysis above, the running time for an instance (P, B, M, α) with $|P| = n$ satisfies the following recursion.

$$T(n) \leqslant n^{O(\max(\frac{\log n}{\alpha}, \log n))} T\left(\frac{2}{3}n\right)$$

Therefore, there exists a constant c such that the running time is at most

$$\begin{aligned} T(n) &\leqslant n^{\max(1,1/\alpha)\cdot c\log n}\left(\frac{2}{3}n\right)^{\max(1,1/\alpha)\cdot c(\log(\frac{2}{3}n))}\cdot\left(\frac{4}{9}n\right)^{\max(1,1/\alpha)\cdot c(\log(\frac{4}{9}n))}\cdot\ldots \\ &= n^{\max(1,1/\alpha)\cdot c(\log n+\log(\frac{2}{3}n)+\log(\frac{4}{9}n)+\ldots)} \\ &= n^{\max(1,1/\alpha)\cdot O(\log^2 n)}. \end{aligned}$$

◀

5 Conclusion

We have devised a separator theorem in $\mathbb{H}^2$ that led to a quasi-polynomial algorithm for Hyperbolic TSP on constant-spaced point sets. For α-spaced point sets with spacing $\alpha \geqslant \log^2 n/\sqrt{n}$ our algorithm runs in $n^{O(\log^2 n)\max(1,1/\alpha)}$ time. When the point set has spacing only $\Theta(\log^2 n/\sqrt{n})$, the algorithm's performance degrades to the point of reaching (roughly) the performance of the Euclidean algorithm. If the point set has even closer point pairs, then the algorithm of Hwang *et al.* [12] can be used to obtain a running time of $n^{O(\sqrt{n})}$. We have shown that our algorithm's dependence on density is necessary and for spacing $1/\sqrt{n}$, it cannot be significantly improved under ETH. There are several intriguing questions that are left open. We list some of these questions below.

- **Improving the running time, lower bounds.** There is a considerable gap between the running time for Hamiltonian Cycle in hyperbolic unit disk graphs (which is polynomial) and our Hyperbolic TSP algorithm, which for constant α runs in $n^{O(\log^2(n))}$ time. Is there an $n^{O(\log n)}$ or a polynomial algorithm for $\alpha \geqslant 1$? Alternatively, can we prove a (conditional) superpolynomial lower bound? Such a lower bound would have to go beyond the quasi-polynomial lower bound for Independent Set seen in [18], as that relies heavily on dense point sets which are not allowed for $\alpha = \Omega(1)$. Another approach would be to use the naïve grid embedding of [18] directly, but that does not lead to a superpolynomial lower bound here.
- **Higher dimensions.** The grid-based lower-bound framework of [8] can be used in $\mathbb{H}^{d+1}$, see [18]. In particular, the ETH-based lower bound of [7] for Euclidean TSP implies that there is no $2^{o(n^{1-1/(d-1)})}$ algorithm for Hyperbolic TSP in $\mathbb{H}^d$ under ETH. Can we extend our algorithmic techniques to constant-spaced point sets in $\mathbb{H}^d$ and gain algorithms with running time $2^{n^{1-1/(d-1)}}\operatorname{poly}(\log n)$? What happens for denser point sets? As observed in [7], the techniques of Hwang *et al.* [12] do not even seem to extend to $\mathbb{R}^d$ for $d \geqslant 3$. Is a running time of $2^{n^{1-1/d}}\operatorname{poly}(\log n)$ possible for all point sets in $\mathbb{H}^d$?
- **A less forgiving parameter.** Our usage of the spacing parameter α may be too restrictive. Is there a better algorithm that can handle more general inputs that can contain a few close point pairs?

References

1 Sanjeev Arora. Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems. *Journal of the ACM*, 45(5):753–782, 1998. doi:10.1145/290179.290180.

2 Yair Bartal, Lee-Ad Gottlieb, and Robert Krauthgamer. The traveling salesman problem: Low-dimensionality implies a polynomial time approximation scheme. *SIAM J. Comput.*, 45(4):1563–1581, 2016. doi:10.1137/130913328.

3 Richard Bellman. Dynamic programming treatment of the travelling salesman problem. *Journal of the ACM*, 9(1):61–63, 1962. doi:10.1145/321105.321111.

4 Riccardo Benedetti and Carlo Petronio. *Lectures on hyperbolic geometry*. Springer Science & Business Media, 2012.

5 James W Cannon, William J Floyd, Richard Kenyon, Walter R Parry, et al. Hyperbolic geometry. *Flavors of geometry*, 31:59–115, 1997.

6 Nicos Christofides. Worst-case analysis of a new heuristic for the travelling salesman problem. Technical report, Graduate School of Industrial Administration, Carnegie Mellon University, 1976.

7 Mark de Berg, Hans L. Bodlaender, Sándor Kisfaludi-Bak, and Sudeshna Kolay. An ETH-tight exact algorithm for Euclidean TSP. In *Proceedings of FOCS 2018*, pages 450–461. IEEE Computer Society, 2018. doi:10.1109/FOCS.2018.00050.

8 Mark de Berg, Hans L. Bodlaender, Sándor Kisfaludi-Bak, Dániel Marx, and Tom C. van der Zanden. A framework for ETH-tight algorithms and lower bounds in geometric intersection graphs. In *Proceedings of STOC 2018*, pages 574–586, 2018. doi:10.1145/3188745.3188854.

9 Vladimir G. Deineko, Bettina Klinz, and Gerhard J. Woeginger. Exact algorithms for the Hamiltonian cycle problem in planar graphs. *Operations Research Letters*, 34(3):269–274, 2006. doi:10.1016/j.orl.2005.04.013.

10 Marvin J Greenberg. *Euclidean and non-Euclidean geometries: Development and history*. Macmillan, 1993.

11 Michael Held and Richard M. Karp. A dynamic programming approach to sequencing problems. In *Proceedings of the 1961 16th ACM National Meeting*, ACM '61, pages 71.201–71.204, New York, NY, USA, 1961. ACM.

12 R. Z. Hwang, R. C. Chang, and Richard C. T. Lee. The searching over separators strategy to solve some NP-hard problems in subexponential time. *Algorithmica*, 9(4):398–423, 1993. doi:10.1007/BF01228511.

13 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.

14 Shreesh Jadhav and Asish Mukhopadhyay. Computing a centerpoint of a finite planar set of points in linear time. *Discrete & Computational Geometry*, 12:291–312, 1994. doi:10.1007/BF02574382.

15 Viggo Kann. *On the approximability of NP-complete optimization problems*. PhD thesis, Royal Institute of Technology Stockholm, 1992.

16 Richard M. Karp. Reducibility among combinatorial problems. In *50 Years of Integer Programming*, pages 219–241. Springer, 2010.

17 Marek Karpinski, Michael Lampis, and Richard Schmied. New inapproximability bounds for TSP. *J. Comput. Syst. Sci.*, 81(8):1665–1677, 2015. doi:10.1016/j.jcss.2015.06.003.

18 Sándor Kisfaludi-Bak. Hyperbolic intersection graphs and (quasi)-polynomial time. In *Proceedings of SODA 2020*, pages 1621–1638. SIAM, 2020. doi:10.1137/1.9781611975994.100.

19 Sándor Kisfaludi-Bak. A quasi-polynomial algorithm for well-spaced hyperbolic TSP. *CoRR*, abs/2002.05414, 2020. arXiv:2002.05414.

20 Robert Krauthgamer and James R. Lee. Algorithms on negatively curved spaces. In *Proceedings of FOCS 2006*, pages 119–132, 2006. doi:10.1109/FOCS.2006.9.

21 Joseph S. B. Mitchell. Guillotine subdivisions approximate polygonal subdivisions: A simple polynomial-time approximation scheme for geometric TSP, k-MST, and related problems. *SIAM Journal on Computing*, 28(4):1298–1309, 1999. doi:10.1137/S0097539796309764.

22 Arlan Ramsay, Robert Davis Richtmyer, and Robert D. Richtmyer. *Introduction to hyperbolic geometry*. Universitext. Springer, New York, 1995.

23 Satish Rao and Warren D. Smith. Approximating geometrical graphs via "spanners" and "banyans". In *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing*, pages 540–550. ACM, 1998. doi:10.1145/276698.276868.

24 Warren D. Smith and Nicholas C. Wormald. Geometric separator theorems & applications. In *Proceedings of FOCS 2018*, pages 232–243. IEEE Computer Society, 1998. doi:10.1109/SFCS.1998.743449.

25 Aleksandr S. Smogorževskij. *Lobatschewskische Geometrie*. Mathematische Schülerbücherei 96. Teubner, Leipzig, 1. aufl. edition, 1978.

26 William P. Thurston. *Three-Dimensional Geometry and Topology*, volume 1. Princeton University Press, 1997.

Intrinsic Topological Transforms via the Distance Kernel Embedding

Clément Maria
INRIA Sophia Antipolis-Méditerranée, Valbonne, France
clement.maria@inria.fr

Steve Oudot
INRIA Saclay, Palaiseau, France
steve.oudot@inria.fr

Elchanan Solomon
Department of Mathematics, Duke University, Durham, NC USA
yitzchak.solomon@duke.edu

Abstract

Topological transforms are parametrized families of topological invariants, which, by analogy with transforms in signal processing, are much more discriminative than single measurements. The first two topological transforms to be defined were the Persistent Homology Transform (PHT) and Euler Characteristic Transform (ECT), both of which apply to shapes embedded in Euclidean space. The contribution of this paper is to define topological transforms for abstract metric measure spaces. Our proposed pipeline is to pre-compose the PHT or ECT with a Euclidean embedding derived from the eigenfunctions and eigenvalues of an integral operator. To that end, we define and study an integral operator called the distance kernel operator, and demonstrate that it gives rise to stable and quasi-injective topological transforms. We conclude with some numerical experiments, wherein we compute and compare the eigenfunctions and eigenvalues of our operator across a range of standard 2- and 3-manifolds.

2012 ACM Subject Classification Mathematics of computing → Algebraic topology

Keywords and phrases Topological Transforms, Persistent Homology, Inverse Problems, Spectral Geometry, Algebraic Topology, Topological Data Analysis

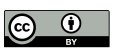
Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.56

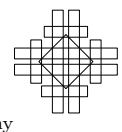
Related Version `https://arxiv.org/abs/1912.02225`

1 Introduction

One way of viewing the success of convolutional neural networks in the classification and analysis of large, multi-channel images is that these neural networks learn an optimal set of coordinates for representing sets of images in Euclidean space. However, when the data consist of shapes whose underlying topologies may vary, it becomes apparent that new tools and techniques must be brought to bear. One approach is to use topological transforms to represent these shapes as collections of topological summaries.

The topological summaries we use are persistence diagrams. Given a real-valued filter function f on a space X, the associated persistence diagram $\mathrm{Diag}(f)$ describes how the topology of the sublevel sets $X_\alpha = \{f(x) \leq \alpha \mid x \in X\}$ evolves as α increases. When the filter-function f measures something about the geometry of X, such as its curvature, the resulting persistence diagram contains both topological and geometric information. If we use not one, but a family of filter functions, the resulting collection of persistence diagrams is called a topological transform. Although the space of persistence diagrams (with the Bottleneck of Wasserstein distances) is not an inner product space, there are many methods






for mapping persistence diagrams to a Hilbert space. The composition of any such map with our topological transform results in a collection of vectors that can be concatenated and fed into any standard machine learning model.

Turner et al. [20] first introduced topological transforms for shapes embedded in Euclidean space. Their work, and the work in various subsequent papers, demonstrates that this transform completely captures the geometric structure of its defining shape. Our goal in this article is to extend these topological transforms to intrinsic metric spaces by pre-composing them with a suitable Euclidean embedding. For this pipeline to be successful in practice, the chosen Euclidean embedding must preserve the geometric structure of our metric spaces. In this paper, we make the case that a particular integral operator, the distance kernel operator, is well suited to this task. We show that the eigenfunctions and eigenvalues of this operator are stable with respect to discretization and perturbations of the underlying shape, that they define a Euclidean embedding which encodes the large-scale geometry of our metric space, and that the resulting topological transforms enjoys favorable stability and quasi-injectivity properties.

Related Work. In [20], Turner et al. defined the Persistent Homology Transform (PHT) and Euler Characteristic Transform (ECT). These transforms take as input sufficiently regular subsets S of Euclidean space $\mathbb{R}^d$, and associate to every vector $v \in \mathbb{S}^{d-1}$ of the sphere in $\mathbb{R}^d$ the persistence diagram, or Euler characteristic curve, of the sublevel-set filtration of S induced by the function $f_v : S \to \mathbb{R}$: $f_v(x) = v \cdot x$. It was subsequently proven in [9] and [12] that these topological transforms are injective in all dimensions[1]. Moreover, it was shown in [5] and [9] that, for certain families of embedded shapes, these topological transforms can be computed in finitely many steps[2]. Complimenting these theoretical results, Crawford et al. [8] demonstrated how to use these topological transforms to build an improved classifier for glioblastoma patient outcomes.

In [16], Oudot and Solomon defined a topological transform for intrinsic metric spaces (X, d_X). This transform associates to every basepoint $x_0 \in X$ the extended persistence diagram of the function $f_{x_0} : X \to \mathbb{R}$: $f_{x_0}(x) = d_X(x_0, x)$. The resulting invariant, called the Intrinsic Persistent Homology Transform (IPHT), is the collection of all persistence diagrams arising from basepoints in X. By computing Euler characteristic curves instead of persistence diagrams, one obtains the Intrinsic Euler Characteristic Transform (IECT). This invariant was first studied, in the case of metric graphs, by Dey, Shi, and Wang in [10], where they proved stability and computability results and ran some experiments. The main result of [16] demonstrated that these invariants are injective[3] on an appropriately generic subset of the space of metric graphs. For a detailed survey on related problems in applied topology, we refer the reader to [17]. Another line of research, at the intersection between persistent homology and spectral geometry, can be found in the work of Polterovich et al. [18], where they study and bound various functionals on persistence diagrams arising from Laplacian eigenfunctions on compact surfaces.

As this paper is concerned with both applied topology and spectral geometry, let us now consider some results, both classical and modern, in the latter field. To begin, the data of a weighted graph can be encoded via its adjacency matrix, and the spectral theory of

[1] By "injective", we mean that two subsets of Euclidean space have the same transform if and only if they are identical. Thus, the transform is injective on the space of admissible subsets.

[2] That is, finitely many directions determine the entire transform, and these directions can be identified with finitely many geometric computations.

[3] As with the PHT and ECT, this means that two graphs having the same transform must be isometric.

these matrices is deep and of great utility, seeing application in, e.g., graph clustering and Google's PageRank algorithm. Another matrix associated to a graph is its Laplacian, whose eigendecomposition forms the basis for the Laplacian Eigenmaps technique studied by Belkin and Niyogi in [2, 3, 4], as well as the diffusion maps of Coifman and Lafon [7]. Spectral analysis of the Gram matrix of the distances gives rise to the classical Multi-Dimensional Scaling embedding and its extension by Tenenbaum et al. [19] to non-linear embeddings: IsoMap. Lastly, the X-ray transform of [14] takes as input a continuous, compactly supported function f on $\mathbb{R}^d$, and outputs a function on the space of lines in $\mathbb{R}^d$ that encodes the corresponding line integral of f.

Contributions and Outline. The structure of this article is as follows. In Section 2, we introduce a general framework for producing topological transforms on intrinsic metric objects via embedding these shapes in Euclidean space, and then applying the extrinsic topological transforms of [20]. To that end, we are charged with identifying a Euclidean embedding whose associated topological transforms have desirable stability and inverse properties. We observe that the eigenfunctions of the Laplacian are not well suited to this task, and so, in Section 3, we define a new operator, called the distance kernel operator, which we prove gives rise to a Euclidean embedding (which we call distance kernel embedding, or DKE). In Section 4, we show that the DKE is sufficiently regular to allow for the computation of topological invariants, in addition to being stable under discrete sampling and perturbation of the metric. We also show that the regularity of the DKE implies stability results for its associated topological transforms. In Section 5, we prove inverse results for the DKE and its topological transforms. We conclude, in Section 6, with a range of experiments illustrating the discriminative power of the DKE for discrete samples of various 2- and 3-manifolds.

2 Intrinsic Topological Transforms from Compact Operators

In this section we introduce a general framework for combining the existing extrinsic topological transforms with Euclidean embeddings of intrinsic metric spaces via compact operators.

2.1 Extrinsic Topological Transforms

▶ **Definition 1.** *Let f be a real-valued function on a topological space X. We write $PH(X, f)$ to denote the* graded sublevel set persistence diagram *of (X, f), which contains the sublevel set persistence diagrams of (X, f) for each homological degree. We write* **GrDiag** *to refer to the space of such graded persistence diagrams. For a fixed degree k, we write $\beta_k(X, f)$ to denote the corresponding Betti curve, which is the sum of the indicator functions of intervals in the persistence diagram. Lastly, we write $\chi(X, f)$ to denote the Euler curve, which is the alternating sum of the Betti curves in all degrees.*

▶ **Definition 2.** *Let $\mathbb{S}^k$ be the k-dimensional sphere, and $L(\mathbb{R}^{k+1}, \mathbb{R})$ the space of linear maps from $\mathbb{R}^{k+1}$ to $\mathbb{R}$. Define the map $\Theta : \mathbb{S}^k \to L(\mathbb{R}^{k+1}, \mathbb{R})$ which sends $v \in \mathbb{S}^k$ to the map $x \mapsto \langle x, v \rangle$.*

▶ **Definition 3** ([9]). *Let $X \subset \mathbb{R}^d$ be a compact, definable set*[4]*. For every $v \in \mathbb{S}^{d-1}$, the sublevel set persistence diagram and Euler curve of the pair $(X, \Theta(v))$ exist. The* Persistent Homology Transform *is the map $PHT(X) : \mathbb{S}^{d-1} \to \mathbf{GrDiag}$ defined by:*

$$PHT(X)(v) = PH(X, \Theta(v)).$$

If one computes Euler curves instead of persistence diagrams, one obtains the Euler Characteristic Transform *$ECT(X)$.*

Intuitively, the PHT and ECT probe an embedded subset of Euclidean space like a multi-directional MRI scanner, recording how the topology evolves along each direction. The following injectivity result demonstrates the rich geometric content of these transforms.

▶ **Theorem 4** ([9, 12]). *The PHT and ECT are injective for all k. That is, for any definable sets $X, Y \subset \mathbb{R}^d$, if $PHT(X) = PHT(Y)$ or $ECT(X) = ECT(Y)$, then $X = Y$ as sets.*

2.2 Compact Operators and their Embeddings

▶ **Definition 5.** *Let H be a Hilbert space. A linear operator $T : H \to H$ is bounded if there exists a constant M such that, for all $v \in H$, $\|Tv\| \leq M\|v\|$. A bounded operator is compact if the image of any bounded subset of H under T is relatively compact. A bounded operator T is self-adjoint if T is equal to its adjoint T^*; equivalently, $\langle Tx, y\rangle = \langle x, Ty\rangle$ for all $x, y \in H$.*

The spectral theorem for compact, self-adjoint operators on a Hilbert space asserts that these operators can be diagonalized.

▶ **Theorem 6** (Spectral Theorem). *Let T be a compact, self-adjoint operator on a Hilbert space H. Then H admits a finite or countably infinite basis $\{\phi_i\}$ of eigenvectors of T with real eigenvalues $\{\lambda_i\}$, where $\lim_{i\to\infty} \lambda_i = 0$.*

Given a compact metric (Borel) measure space (X, d_X, μ_X), we can consider compact, self-adjoint operators on the Hilbert space $L^2(X)$. The eigenfunctions $\{\phi_i\}$ and eigenvalues $\{\lambda_i\}$ arising from the spectral theorem can then be used to define embeddings of X into Euclidean space. This requires the adoption of various conventions and generic assumptions:

1. The spectral theorem asserts the existence of the eigenfunctions ϕ_i, but it does not guarantee their uniqueness. Indeed, the choice is never unique. If the eigenvalue λ_i has geometric multiplicity one, then there are two choices of unit norm eigenfunctions: $\{\phi_i, -\phi_i\}$. If the eigenvalue has geometric multiplicity greater than one, then there are infinitely many choices. In the rest of the paper, we make the generic assumption that all the eigenvalues have multiplicity one[5].
2. We adopt the convention of dropping the eigenfunctions in the zero-eigenspace, and, to fix the choice of sign, we pick ϕ_i such that $\langle \phi_i, |\phi_i|\rangle > 0$ for all i[6].
3. We order the eigenvalues (and hence eigenfunctions) in decreasing order of absolute value, $|\lambda_1| \geq |\lambda_2| \geq |\lambda_3| \geq \ldots$, breaking the tie between positive-negative pairs by listing the positive eigenvalue first.

---

[4] The notion of definability is always understood to be relative to a choice of o-minimal structure on $\mathbb{R}^d$, which is an algebra of sets satisfying certain membership conditions. Examples include the collection of semi-algebraic or analytic subsets of $\mathbb{R}^d$. See [9] §2 for details.

[5] We can always infinitesimally perturb our space to make this true.

[6] We make the generic assumption that this dot product is nonzero.

4. To take advantage of many useful results in operator theory, we restrict ourselves to operators that are *Hilbert-Schmidt*, which means that $\sum_{i=1}^{\infty} \lambda_i^2 < \infty$. Every Hilbert-Schmidt operator on $L^2(X)$ can be represented as an integral operator with square integrable kernel $K(\cdot,\cdot)$[7], i.e., an operator of the form:

$$T\colon L^2(X) \to L^2(X) \qquad (Tf)(x) = \int_X K(x,y)f(y)d\mu_X(y).$$

We thus assume our operators are of this form.

▶ **Definition 7.** *Given a compact metric (Borel) measure space (X, d_X, μ_X), let T be a compact, self-adjoint operator on $L^2(X)$ with spectral decomposition $\{\phi_i, \lambda_i\}$, following the conventions above. We define coordinate functions on X as follows: $\alpha_i(x) = \sqrt{\lambda_i}\phi_i(x)$. Note that the eigenvalue λ_i may be negative (we have not assumed that the operator is positive definite), so the coordinate function takes values in $\mathbb{C}$. When λ_i is negative, we adopt the convention of taking the square root with positive imaginary part. By identifying $\mathbb{C}$ with $\mathbb{R}^2$, we can also think of this coordinate function as taking a pair of real values.*

Our rationale for scaling the eigenfunctions by the square root of their eigenvalues is that, for an integral operator T with kernel $K(\cdot,\cdot)$, the sum $\sum_{k=1}^{\infty} \lambda_i\phi_i(x)\phi_i(x') = \sum_{k=1}^{\infty}(\sqrt{\lambda_i}\phi_i(x))(\sqrt{\lambda_i}\phi_i(x')) = \sum_{i=1}^{\infty} \alpha_i(x)\alpha_i(x')$ converges to $K(x,x')$ in $L^2(X,X)$[8]. Using these coordinates, we define a kernel embedding[9]:

▶ **Definition 8.** *Let (X, d_X, μ_X), T, and $\{\phi_i, \lambda_i\}$ be as in Definition 7. For $k \geq 1$, we define $\Phi_k : X \to \mathbb{C}^k \cong \mathbb{R}^{2k}$ to be the map sending a point $x \in X$ to $(\alpha_1(x), \cdots, \alpha_k(x)) \in \mathbb{C}^k \cong \mathbb{R}^{2k}$. Setting $k = \infty$ gives us a map $\Phi : X \to \mathbb{C}^\infty \cong \mathbb{R}^\infty$. When Φ is continuous and injective, the image of Φ (resp. Φ_k) is called the* kernel embedding *(resp.* truncated kernel embedding*) associated to T.*

2.3 Topological Kernel Transforms

By post-composing this embedding with the PHT or the ECT, we obtain topological transforms that are defined intrinsically on metric measure spaces[10].

▶ **Definition 9.** *Let X and Φ be as in Definition 8. For k finite, the* embedded persistence kernel transform *e-$PKT_k(X)$ is the PHT applied to the image of the embedding $\Phi_k(X) \subset \mathbb{R}^{2k}$, which takes as input vectors in $\mathbb{S}^{2k-1}$ and takes values in* **GrDiag**. *Using Euler curves in place of persistent diagrams gives rise to the embedded* Euler kernel transform *e-EKT_k.*

The following meta-theorem motivates the constructions and results to follow.

▶ **Theorem 10.** *Fix a positive integer k. Let $\mathcal{M}$ be a class of metric measure spaces with integral kernels $\{K^M\}_{M\in\mathcal{M}}$, giving definable embeddings $\{\Phi_k^M\}_{M\in\mathcal{M}}$. If $\Phi_k^M(M) \neq \Phi_k^{M'}(M')$ for any pair of non-isometric spaces $M \neq M' \in \mathcal{M}$, then the e-$PKT_k$ and e-EKT_k are injective on $\mathcal{M}$.*

Proof. This is an immediate consequence of Theorem 4. ◀

[7] This is the Hilbert-Schmidt Kernel Theorem. See Appendix B of [13], where the result is proven for subsets of $\mathbb{R}$. The proof for compact metric (Borel) measure spaces is identical.
[8] This convergence is not necessarily uniform, unless $K(\cdot,\cdot)$ is positive semidefinite (Mercer's theorem).
[9] The use of the term *embedding* comes from the injectivity properties of this map, proved in §5.1
[10] We implicitly assume here that these persistence diagrams and Euler curves exist. Later on in this article, we verify this explicitly for the integral operator of interest.

For most classes $\mathcal{M}$ of interest, such as the set of Riemannian manifolds of a given dimension, there are no known integral operators whose associated embeddings in finite dimensions are injective in the sense of Theorem 10. However, if we relax the assertion of injectivity, and ask only that $\Phi_k^M(M) = \Phi_k^{M'}(M')$ implies a bound on the Gromov-Hausdorff distance between M and M', we can construct such an integral operator. Before defining this operator, which is the subject of the next section, we note that the diffusion operator, which is related to the *local* geometry of a space, does not enjoy *global* guarantees of this kind. Indeed, although one can recover the metric on a Riemannian manifold from the exact knowledge of all its Laplacian eigenfunctions and eigenvalues (cf. [21]), this reconstruction is asymptotic, cannot be approximated with finitely many eigenfunctions and eigenvalues, and is unstable to noise.

3 The Distance Kernel Operator and its Embedding

We now define our proposed integral operator, and prove that it is compact and self-adjoint. Recall that a compact metric measure space (X, d_X, μ_X) has finite volume $\mu_X(X) < +\infty$. We write $\mathrm{Vol}(X) := \mu_X(X)$.

▶ **Definition 11.** *Let (X, d_X, μ_X) be a compact metric measure space*[11]*. We define the following operator D^X on $L^2(X)$, called the* distance kernel operator *(DKO):*

$$(D^X f)(x) = \int_X f(y)\, \mathrm{d}_X(x,y) d\mu_X(y).$$

▶ **Proposition 12.** *D^X is a self-adjoint operator.*

Proof. By convention, μ_X is Radon. Since X is compact, this implies that $\mu_X(X) < \infty$, and hence (X, μ_X) is σ-finite. We can thus apply Fubini's theorem, and the symmetry of the distance function d_X, to observe that, for two integrable functions f and g,

$$\begin{aligned}
\langle D^X f, g\rangle &= \int_X \left(\int_X f(y)\, \mathrm{d}_X(x,y) d\mu_X(y) \right) g(x) d\mu_X(x) \\
&= \int_X \int_X f(y) g(x)\, \mathrm{d}_X(x,y) d\mu_X(x) d\mu_X(y) \\
&= \int_X f(y) \left(\int_X g(x)\, \mathrm{d}_X(y,x) d\mu_X(x) \right) d\mu_X(y) \quad = \quad \langle f, D^X g\rangle,
\end{aligned}$$

demonstrating self-adjointness. ◀

▶ **Proposition 13.** *D^X is a compact operator.*

Proof. Let $f_n \in L^2(X)$ be a bounded sequence of functions, $||f_n||_{L^2} \leq C$ for all n.

For all $\mathrm{d}_X(x,x') \leq \varepsilon$ and all n,

$$\begin{aligned}
|D^X f_n(x) - D^X f_n(x')| &= \left| \int_X (\mathrm{d}_X(x,y) f_n(y) - \mathrm{d}_X(x',y) f_n(y))\, d\mu_X(y) \right| \\
&\leq \int_X |\, \mathrm{d}_X(x,y) - \mathrm{d}_X(x',y)||f_n(y)| d\mu_X(y) \\
\text{(Cauchy-Schwarz)} &\leq \|\, \mathrm{d}_X(x,y) - \mathrm{d}_X(x',y)\|_{L^2(y)} \cdot \|f_n(y)\|_{L^2(y)} \\
\text{(triangle inequality)} &\leq \|\varepsilon\|_{L^2(y)} \cdot \|f_n(y)\|_{L^2(y)} \\
&\leq \varepsilon \sqrt{\mathrm{Vol}(X)} \cdot C,
\end{aligned}$$

[11] For the rest of the paper, we assume that μ_X is a Radon measure.

where $\mathrm{Vol}(X)$ is finite. Thus, $D^X f_n$ is an equicontinuous family of functions on X, so, by the Arzelà-Ascoli theorem, it contains a uniformly convergent, and hence L^2-convergent, subsequence. This demonstrates compactness. ◀

Note that, as a consequence of the proof of this proposition, the eigenfunctions of the distance kernel operator are always continuous. We can thus define an embedding as in Definition 8, which we call the *distance kernel embedding* (DKE)[12]. From now on, we write Φ and Φ_k to exclusively denote the distance kernel embedding, and the e-PKT and e-EKT likewise refer exclusively to the resulting transforms. In the following sections, we study the stability and inverse properties of these embeddings and transforms.

4 Stability for the DKE and its Topological Transforms

Let (X, d_X, μ_X) be a compact metric (Borel) measure space. The eigenfunctions of the distance kernel operator with nonzero eigenvalue are Lipschitz continuous, with the Lipschitz constant being inversely proportional to the absolute value of the eigenvalue.

▶ **Lemma 14.** *For every $i \in \mathbb{N}_{>0}$, The function $\lambda_i\phi_i$ is $\sqrt{\mathrm{Vol(X)}}$-Lipschitz. Hence, if $\lambda_i \neq 0$, ϕ_i is $(\sqrt{\mathrm{Vol(X)}}/|\lambda_i|)$-Lipschitz. Note that X being compact, $\mathrm{Vol(X)} < +\infty$ and these Lipschitz constants are indeed finite.*

Proof. Let $x, y \in X$ and $\varepsilon = \mathrm{d}_X(x, y)$. By the fact that $\lambda_i\phi_i = D^X\phi_i$, we have

$$\begin{aligned}
|\lambda_i\phi_i(x) - \lambda_i\phi_i(y)|^2 &= \left|(D^X\phi_i)(x) - (D^X\phi_i)(y)\right|^2 \\
&= \left|\int_X (\mathrm{d}_X(x,z) - \mathrm{d}_X(y,z))\phi_i(z)d\mu_X(z)\right|^2 \\
\text{(Cauchy-Schwarz)} &\leq \int_X \underbrace{(\mathrm{d}_X(x,z) - \mathrm{d}_X(y,z))}_{\leq \mathrm{d}_X(x,y)=\varepsilon}{}^2 d\mu_X(z) \cdot \underbrace{\int_X \phi_i^2(z)d\mu_X(z)}_{=1} \\
&\leq \varepsilon^2 \,\mathrm{Vol}(X).
\end{aligned}$$

Thus, $|\lambda_i\phi_i(x) - \lambda_i\phi_i(y)| \leq \varepsilon\sqrt{\mathrm{Vol(X)}}$, so $\lambda_i\phi_i$ is $\sqrt{\mathrm{Vol(X)}}$-Lipschitz. ◀

This regularity result on eigenfunctions has many implications for our topological transforms, which are given below (the proofs can be found in the full version of the paper [15], but are omitted here due to their complexity and length). The result implies in particular that persistence diagrams exist and are well-defined (which is not the case for an arbitrary continuous function on a compact topological space), and, under the additional assumption that the space X *implies bounded degree-q total persistence*[13], that Euler curves exist:

▶ **Proposition 15.** *Let (X, d_X, μ_X) be a compact metric measure space homeomorphic to the geometric realization of a finite simplicial complex. Then, any finite linear combination $f = \sum_{i=1}^n c_i\phi_i$ of eigenfunctions of the distance kernel D^X has a well-defined sublevel set graded persistence diagram $PH(X, f)$. Now, suppose further that X implies bounded degree-q total persistence. Let $p = 1/q$. Then for any homological degree k, the sum defining $\beta_k(X, f)$ converges in L^p. Moreover, the sum defining $\chi(X, f)$ is finite, so the Euler curve exists as a function in L^p.*

[12] The matter of injectivity will be established in Lemma 20.

[13] Intuitively, this technical condition means that, for any graded persistence diagram $PH(X, f)$, the sum of the qth powers of the persistences of the points across all degrees is finite. The bilipschitz image of a finite dimensional Euclidean simplicial complex has bounded degree-q total persistence for q sufficiently large. See the full version of this paper for details.

In addition, we obtain the following stability result for the resulting topological transforms:

▶ **Theorem 16.** *Suppose X is homeomorphic to the geometric realization of a finite simplicial complex. If we equip* **GrDiag** *with the graded bottleneck distance and the sphere $\mathbb{S}^{2k-1}$ with the ℓ^1 distance, then the e-PKT_k is Lipschitz continuous. Now suppose further that X implies bounded degree-q total persistence for some $q > 0$, and that there is a uniform bound on the number of points in the persistence diagrams obtained when evaluating the e-PKT_k at an arbitrary vector $v \in \mathbb{S}^{2k-1}$. If we equip the sphere $\mathbb{S}^{2k-1}$ with the ℓ^1 distance, and the space of Euler curves with the $L^{1/q}$ distance, then the e-EKT_k is q-Hölder continuous on $\mathbb{S}^{2k-1}$.*

We also have two more stability results for the DKE, for which we do not yet have analogues for the topological transforms. The first result asserts that the distance kernel embedding of a discrete sample of a space converges almost surely to the distance kernel embedding of the underlying space:

▶ **Theorem 17.** *Let (X, d_X, μ_X) be a compact metric measure space with (a, b)-standard Borel measure*[14]*. For an i.i.d sample X_n of X of size n, call $\hat{\Phi}_k(X_n)$ the empirical DKE defined on the metric measure space $(X_n, \mathrm{d}_X, \mu_n)$ with uniform measure $\mu_n(\hat{x}) = \mu_X(X)/n$ for all $\hat{x} \in X_n$. Writing $\mathrm{d}_H^{L^2}$ for the Hausdorff distance for the L^2 norm in $\mathbb{C}^k$, we have:*

$$\mathrm{d}_H^{L^2}(\Phi_k(X), \hat{\Phi}_k(X_n)) \xrightarrow{a.s.} 0 \text{ as } n \to +\infty.$$

In addition, the distance kernel embedding is stable on the space of Riemannian manifolds. The following is a simplified version of the result contained in the full version of the paper, which gives an explicit form for the function F.

▶ **Theorem 18.** *Let (X, d_X, μ_X) and (Y, d_Y, μ_Y) be compact finite-dimensional Riemannian manifolds equipped with their volume measures, such that $\mu_X(X) = \mu_Y(Y)$. Let $\varepsilon = \mathrm{d}_{G\bar{P}}(X, Y)$ be the modified Gromov-Prokhorov distance*[15] *between X and Y, and let $|\lambda_1| > \ldots > |\lambda_k| > 0$ and $|\nu_1| > \ldots > |\nu_k| > 0$ be the k largest (in absolute value) eigenvalues of the distance kernel operators of X and Y respectively, all non-zero with distinct absolute values. Let $\Phi_k(X)$ and $\Psi_k(Y)$ be the induced DKE. If we write Λ for the set $\{\lambda_1, \cdots, \lambda_k, \nu_1, \cdots \nu_k\}$, then there is a function $F(\Lambda, \varepsilon)$, depending on the magnitude of the elements of Λ and the gaps $|\lambda_i^2 - \nu_i^2|$ between corresponding eigenvalues, such that:*

$$\mathrm{d}_H^{L^2}(\Phi_k(X), \Psi_k(Y)) \leq F(\Lambda, \varepsilon) \quad \text{and} \quad \lim_{\varepsilon \to 0} F(\Lambda, \varepsilon) = 0.$$

5 Injectivity for the DKE and its Topological Transforms

We now demonstrate some inverse results for the distance kernel embedding and transforms. We stress that these results apply specifically when the integral kernel is taken to be $\mathrm{d}_X(\cdot, \cdot)$.

5.1 Injectivity of Φ

Our first result, in Corollary 21 below, is that, under the mild hypothesis of strict positivity (defined below), the infinite-dimensional embedding Φ is a homeomorphism of the metric measure space onto its image in $\mathbb{C}^\infty$.

[14] This ensures a lower bound on the volume of metric balls. See the full paper for a precise definition.
[15] This is a slight modification of the Gromov-Prokhorov distance, introduced by Burago et al. in §8 of [6].

▶ **Definition 19.** *For a topological space X equipped with its Borel σ-algebra, we call a measure μ_X* strictly positive *if the measure of any nonempty open set is strictly positive.*

▶ **Lemma 20.** *Let (X, d_X, μ_X) be a compact, strictly positive metric measure space. Then the map $\Phi : X \to \mathbb{C}^\infty$ is injective.*

Proof. Suppose that there are $x \neq y \in X$ such that $\Phi(x) = \Phi(y)$. This implies that $\alpha_i(x) = \alpha_i(y)$ and, in turn, $\lambda_i\phi_i(x) = \lambda_i\phi_i(y)$ for all i. Let d_x and d_y be the distance functions associated to x and y respectively. Using the L^2-convergence of the eigenfunction expansion, we know that:

$$\| \mathrm{d}_x - \sum_{i=1}^{n} \lambda_i\phi_i(x)\phi_i \|_{L^2} \xrightarrow{n\to\infty} 0 \quad \text{and} \quad \| \mathrm{d}_y - \sum_{i=1}^{n} \lambda_i\phi_i(y)\phi_i \|_{L^2} \xrightarrow{n\to\infty} 0.$$

Since $\sum_{i=1}^{n} \lambda_i\phi_i(x)\phi_i = \sum_{i=1}^{n} \lambda_i\phi_i(y)\phi_i$ for all n, the triangle inequality implies that $\| \mathrm{d}_x - \mathrm{d}_y \|_{L^2} = 0$. Let now $r = \mathrm{d}_X(x, y)/3 > 0$, and let U be the open neighborhood of radius r around x. The function $| \mathrm{d}_x - \mathrm{d}_y |$ is bounded below by r on U, and since U is not empty (it contains x), it has strictly positive measure. This then implies $\| \mathrm{d}_x - \mathrm{d}_y \|_{L^2} > 0$, a contradiction. Thus, $\Phi(x) \neq \Phi(y)$ for $x \neq y$. ◀

▶ **Corollary 21.** *By Lemma 14, every component of the map Φ is continuous. Meanwhile, any metric on $\mathbb{C}^\infty$ gives it a Hausdorff topological structure. Thus, for any such choice of metric, Φ is a continuous injection from a compact space to a Hausdorff space. Hence, Φ is a homeomorphism.*

We also have the following injectivity result, where the domain of interest is the space of compact metric measure spaces. As a consequence of Corollary 21, it suffices to consider pairs of metric measure spaces that are defined on a common topological space.

▶ **Theorem 22.** *Fix a compact topological space Z. Let μ and μ' be strictly positive measures for the Borel σ-algebra on Z, with μ absolutely continuous with respect to μ', and d and d' metrics on X, both consistent with the topology on Z, making $X = (Z, \mathrm{d}, \mu)$ and $X' = (Z, \mathrm{d}', \mu')$ metric measure spaces. If $\Phi(X) = \Phi(X')$, then $\mathrm{d} = \mathrm{d}'$.*

Proof. By assuming that both metric measure spaces induce the same topology, we can work with a single σ-algebra: their common Borel σ-algebra. This will prove essential in the following proof, where we take various unions and complements of measurable sets for μ and μ', respectively. Next, let D and D' be the integral operators with kernels d and d', respectively. The equality $\Phi(X) = \Phi(X')$ implies that D and D' have the same scaled eigenfunctions α_i. The distance functions d, d' thus have the same eigenfunction expansion:

$$(x_1, x_2) \mapsto \sum_{i=1}^{\infty} \alpha_i(x_1)\alpha_i(x_2).$$

This converges to d in $L_2(\mu \otimes \mu)$ and to d' in $L_2(\mu' \otimes \mu')$ to d'. Let us denote by S_n the partial sums of this expansion:

$$S_n = \sum_{i=1}^{n} \alpha_i(x_1)\alpha_i(x_2).$$

It is a standard result in measure theory that any L^2-convergent sequence admits a subsequence that converges pointwise a.e.[16] Thus, one can extract a subsequence S_{n_k} that converges to d pointwise on $(Z \times Z) \setminus N_1$, for some set $N_1 \subset Z$ such that $(\mu \otimes \mu)(N_1) = 0$. We can then extract a further subsequence $S_{n_{k_j}}$ that converges pointwise to d' on $((Z \times Z) \setminus N_1) \setminus N_2$, where $(\mu' \otimes \mu')(N_2) = 0$. Since μ is absolutely continuous to μ', if we set $N = N_1 \cup N_2$ then $(\mu \otimes \mu)(N) = 0$. Since μ is strictly positive, the set N cannot contain any open sets, hence N^c is dense in $Z \times Z$. We see then that $\mathrm{d} = \mathrm{d}'$ on a dense subset of $Z \times Z$; since these functions are both continuous in the same topology Z, they are equal everywhere. ◀

5.2 Quasi-Injectivity of Φ_k

While the truncated embedding Φ_k may not be injective, we can get control over the diameter of its fibers (Corollary 26). The bounds are expressed in terms of the error of the approximation of the metrics by its truncated expansion:

▶ **Definition 23.** *For a compact metric measure space (X, d_X, μ_X) and a positive integer k, we define the error function $E_{X,k}$, which measures the pointwise distance between d_X and its truncated eigenfunction expansion:*

$$E_{X,k}(x, x') = |\sum_{i=1}^{k} \alpha_i(x)\alpha_i(x') - \mathrm{d}_X(x, x')|.$$

▶ **Theorem 24.** *Let (X, d_X, μ_X) and (Y, d_Y, μ_Y) be compact metric measure spaces, with eigenvalues $\{\lambda_i\}$ and $\{\nu_i\}$. Let $k \in \mathbb{N}_{>0}$ be and integer, and $\varepsilon := \mathrm{d}_H^{L^2}(\Phi_k(X), \Phi_k(Y))$. Then:*

$$d_{GH}(X, Y) \leq 2\varepsilon \min\left\{\max_{x \in X} \|\Phi_k(x)\|_2, \max_{y \in Y} \|\Phi_k(y)\|_2\right\} + \|E_{X,k}\|_\infty + \|E_{Y,k}\|_\infty + \varepsilon^2.$$

In the special case where X and Y are finite metric measure spaces, Theorem 24 assumes a more precise form:

▶ **Theorem 25.** *Let (X, d_X, μ_X) and (Y, d_Y, μ_Y), with eigenvalues $\{\lambda_i\}$ and $\{\nu_i\}$, and let $\theta = \min\{\min_{x \in X} \mu_X(x), \min_{y \in Y} \mu_X(y)\}$. Take $k \leq |X|, |Y|$, and suppose that $\mathrm{d}_H^{L^2}(\Phi_k(X), \Phi_k(Y)) \leq \varepsilon$. Then,*

$$\mathrm{d}_{GH}(X, Y) \leq 2\varepsilon \frac{\min(\sqrt{|\lambda_1|}, \sqrt{|\nu_1|})}{\theta} + \varepsilon^2 + \frac{|\lambda_{k+1}| + |\nu_{k+1}|}{\theta}.$$

We thus obtain the following bound on the diameter of the fibers of the DKE.

▶ **Corollary 26.** *Let (X, d_X, μ_X) and (Y, d_Y, μ_Y) be compact metric measure spaces, with eigenvalues $\{\lambda_i\}$ and $\{\nu_i\}$. Let $k \in \mathbb{N}$ be a positive integer, and suppose that $\Phi_k(X) = \Phi_k(Y)$. Then, $d_{GH}(X, Y) \leq \|E_{X,k}\|_\infty + \|E_{Y,k}\|_\infty$. If X and Y are finite metric spaces, and we set $\theta = \min\{\min_{x \in X} \mu_X(x), \min_{y \in Y} \mu_X(y)\}$, then $d_{GH}(X, Y) \leq \frac{1}{\theta}(|\lambda_{k+1}| + |\nu_{k+1}|)$.*

The effectiveness of these results depends on the magnitude of the quantities $\|E_{X,k}\|_\infty$ and the decay of the eigenvalues λ_i. It remains to identify general metric and measure-theoretic criteria that imply bounds on these spectral statistics. The rest of this section is devoted to

[16] Chebyshev's inequality proves that L^2 convergence implies convergence in measure. See Theorem 2.15(c) in [11] for the implication that convergence in measure implies the existence of pointwise a.e. convergent subsequence.

the proof of Theorem 24. Theorem 25 follows from Theorem 24, and we refer the reader to the full version of the paper for the details of this derivation [15]. The proof of Theorem 24 requires us to define the following algebraic operation, and to prove some technical lemmas regarding it.

▶ **Definition 27.** *For vectors $v, w \in \mathbb{C}^k$, define the following bilinear form:*

$$[v, w] = \sum_{i=1}^{k} v_i w_i \in \mathbb{C}.$$

This form is symmetric but not a dot product.

The utility of the bilinear form $[\cdot, \cdot]$ comes from the following equality:

$$[\Phi_k(x), \Phi_k(x')] = \sum_{i=1}^{k} \alpha_i(x)\alpha_i(x') = \sum_{i=1}^{k} \left(\sqrt{\lambda_i}\phi_i(x)\right)\left(\sqrt{\lambda_i}\phi_i(x')\right) = \sum_{i=1}^{k} \lambda_i \phi_i(x)\phi_i(x').$$

That is, when applied to the distance kernel embedding, $[\cdot, \cdot]$ gives the first k terms of the eigenfunction expansion of the distance function d_X.

▶ **Lemma 28.** *The bilinear form $[\cdot, \cdot]$ satisfies the following Cauchy-Schwarz inequality:*

$$|[v, w]| \leq \|v\|_2 \|w\|_2.$$

Proof Sketch. By the triangle inequality for complex numbers, we have $|[v, w]| \leq [\tilde{v}, \tilde{w}] = \langle \tilde{v}, \tilde{w} \rangle$, where $\tilde{v}, \tilde{w}$ are obtained from v and w by replacing each coordinate with its modulus. The result then follows by applying the standard Cauchy-Schwarz inequality. ◀

The following lemma asserts that pairs of nearby vectors have similar bilinear products.

▶ **Lemma 29.** *Let $v_1, v_2, w_1, w_2 \in \mathbb{C}^k$ be such that $\|v_1 - w_1\|_2 \leq \varepsilon$ and $\|v_2 - w_2\|_2 \leq \varepsilon$. Then*

$$|[v_1, v_2] - [w_1, w_2]| \leq \varepsilon \min\{\|v_1\|_2 + \|v_2\|_2, \|w_1\|_2 + \|w_2\|_2\} + \varepsilon^2.$$

Proof. By bilinearity,

$$[w_1, w_2] = [v_1, v_2] + [v_1, (w_2 - v_2)] + [(w_1 - v_1), v_2] + [(w_1 - v_1), (w_2 - v_2)].$$

Thus,

$$|[v_1, v_2] - [w_1, w_2]| \leq |[v_1, (w_2 - v_2)]| + |[(w_1 - v_1), v_2]| + |[(w_1 - v_1), (w_2 - v_2)]|.$$

By a symmetric argument, switching v_1 and v_2 with w_1 and w_2, one obtains:

$$|[v_1, v_2] - [w_1, w_2]| \leq |[w_1, (v_2 - w_2)]| + |[(v_1 - w_1), w_2]| + |[(v_1 - w_1), (v_2 - w_2)]|.$$

The result then follows by applying the Cauchy-Schwarz inequality to each term on the right-hand sides of both inequalities, and by taking the minimum of the two sums. ◀

We can now prove Theorem 24:

Proof. Let C be an optimal Hausdorff correspondence between $\Phi_k(X)$ and $\Phi_k(Y)$. Let $(x, x') \in X \times X$ and $(y, y') \in Y \times Y$ with $(\Phi_k(x), \Phi_k(y)), (\Phi_k(x'), \Phi_k(y')) \in C$. Lemma 29, together with the bounds $\|\Phi_k(x) - \Phi_k(y)\|_{L^2} \leq \varepsilon$ and $\|\Phi_k(x') - \Phi_k(y')\|_{L^2} \leq \varepsilon$, gives

$$|[\Phi_k(x), \Phi_k(x')] - [\Phi_k(y), \Phi_k(y')]| \leq 2\varepsilon \min\left\{\max_{x \in X} \|\Phi_k(x)\|_2, \max_{y \in Y} \|\Phi_k(y)\|_2\right\} + \varepsilon^2.$$

Using the triangle inequality, we can replace $[\Phi_k(x), \Phi_k(x')]$ with $\mathrm{d}_X(x, x')$ and $[\Phi_k(y), \Phi_k(y')]$ with $\mathrm{d}_Y(y, y')$, at the cost of adding an additive error of at most $\|E_{X,k}\|_\infty$ and $\|E_{Y,k}\|_\infty$ respectively, giving the following inequality, from which the result follows:

$$|\,\mathrm{d}_X(x,x') - \mathrm{d}_Y(y,y')| \leq 2\varepsilon \min\left\{\max_{x\in X} \|\Phi_k(x)\|_2, \max_{y\in Y} \|\Phi_k(y)\|_2\right\} + \|E_{X,k}\|_\infty + \|E_{Y,k}\|_\infty + \varepsilon^2.$$

◀

5.3 Quasi-Injectivity of the e-PKT_k and e-EKT_k

Corollary 26, taken together with Theorem 4, implies the following result, which bounds the diameter of the fibers of the topological transforms. For general metric measure spaces, the diameter depends on the error functions $E_{X,k}$ and $E_{Y,k}$. As k goes to infinity, these functions go to zero in the L^2 norm, but we do not have any general guarantees that this also holds in the L^∞ norm[17]. For finite metric spaces, the diameter does indeed go to 0 as k goes to infinity.

▶ **Theorem 30.** *Let (X, d_X, μ_X) and (Y, d_Y, μ_Y) be compact metric measure spaces, with eigenvalues $\{\lambda_i\}$ and $\{\nu_i\}$ respectively, giving rise to definable distance kernel embeddings. Let $k \in \mathbb{N}$ be a positive integer, and suppose that e-PKT$_k(X)$ = e-PKT$_k(Y)$ or e-EKT$_k(X)$ = e-EKT$_k(Y)$. Then $d_{GH}(X, Y) \leq \|E_{X,k}\|_\infty + \|E_{Y,k}\|_\infty$. If X and Y are finite spaces, and we set $\theta = \min\{\min_{x\in X} \mu_X(x), \min_{y\in Y} \mu_X(y)\}$, then $d_{GH}(X, Y) \leq \frac{1}{\theta}\left(|\lambda_{k+1}| + |\nu_{k+1}|\right)$.*

The condition that the DKEs be definable is always satisfied when the spaces are finite. It remains to work out the correct hypotheses to ensure definability more generally; this is work in progress.

6 Experiments

The goal of this section is to illustrate the results of Sections 4 and 5. In the following experiments, we compute the DKE for a variety of discrete samples on the torus and 2-sphere, with metric induced by their embedding in Euclidean space, on the 3-sphere, and on the Lens spaces $L(7, 1)$ and $L(7, 4)$, with spherical geometry. The measures on these samples are uniform. These spaces have distinct integer homology, except for the two Lens spaces that have the same homotopy type but are not homeomorphic, and therefore not isometric. This makes $L(7, 1)$ and $L(7, 4)$ difficult to distinguish by purely topological methods. We see that the DKE (and, therefore, the resulting topological transforms) is capable of distinguishing these Lens spaces.

Spectra of various manifolds. In Figure 1a, we have plotted the first 8 eigenvalues of five discrete metric spaces, sampled from each of these five manifolds, normalized by the number of points in each sample. We can observe the following: (1) the two Lens spaces have relatively similar eigenvalues, (2) the 2- and 3-sphere have many similar eigenvalues, but their first and fourth eigenvalues are significantly different, and (3) the torus has the most distinct spectrum.

[17] However, experimental studies suggest that this is the case for a variety of manifolds [15].

Spectra of Lens spaces for various samples. In Figure 1b, we compare the spectra of a number of different random i.i.d. samples of the two Lens spaces $L(7,1)$ and $L(7,4)$. To be precise, for each Lens space we compute the spectra of two distinct random samples with 2000 points, and a third sample with 5000 points. The spectra for the different samples of the same Lens space are virtually impossible to distinguish, and only two curves – one for the spectrum of $L(7,1)$, and one of the spectrum of $L(7,4)$ – are visible in Figure 1b. This attests to the stability of the eigenvalues of the distance kernel operator under random i.i.d. sampling, in line with Theorem 17. Notably, the two Lens spaces are distinguished by the first, third, and fourth eigenvalues of their distance kernel operators. $L(7,1)$ and $L(7,4)$ having same homotopy type, this illustrates the ability of the operator to capture geometric information and distinguish between non-isometric spaces.

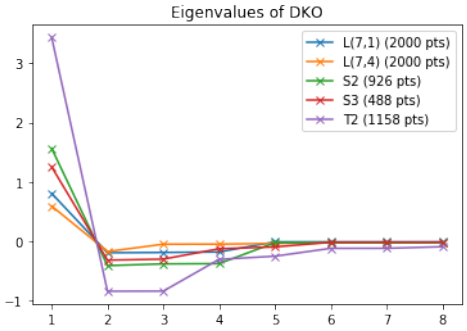
Hausdorff distance between DKEs. Finally, in Figure 1c, we compare the Hausdorff distances between various pairs of distance kernel embeddings. We observe the following: (1) The two samples of the same size coming from the $L(7,1)$ Lens space are the closest in Hausdorff distance, and that distance is close to zero up to dimension $k = 4$. Indeed, if we had taken samples of sufficiently high resolution, we would see the Hausdorff distances going to zero for larger values of k, as proven in Theorem 17. (2) The second closest pair of spaces are the Lens spaces $L(7,1)$ and $L(7,4)$, that have same homotopy type and have both spherical geometry. (3) The third closest pair of spaces are the Lens space $L(7,1)$ and the 3-Sphere, both with spherical geometry (Lens spaces are constructed as quotients of 3-Spheres). (4) The manifold that appears to be most distinct from the rest is the torus. (5) For all pairs of manifolds, the Hausdorff distance stabilizes at around $k = 10$, after which eigenvalues are close to 0.

In conclusion, these experiments illustrate that the spectra and embedding of the distance kernel operator can be approximated by finite samples, as predicted by Theorem 17. Moreover, by combining the DKE with the Hausdorff metric on Euclidean space, we obtain a pseudo-metric on the space of compact metric measure spaces that succeeds in distinguishing a variety of diverse manifolds.

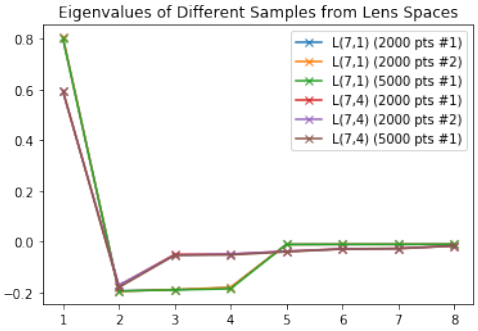
7 Open Problems

This work introduces new techniques, at the crossroads of topological data analysis and spectral geometry, to study general metric measure spaces. It also raises a number of interdisciplinary questions in persistence theory, optimal transport, spectral geometry, and o-minimal geometry, that, due to their specialized and technical nature, have not been resolved in this article:

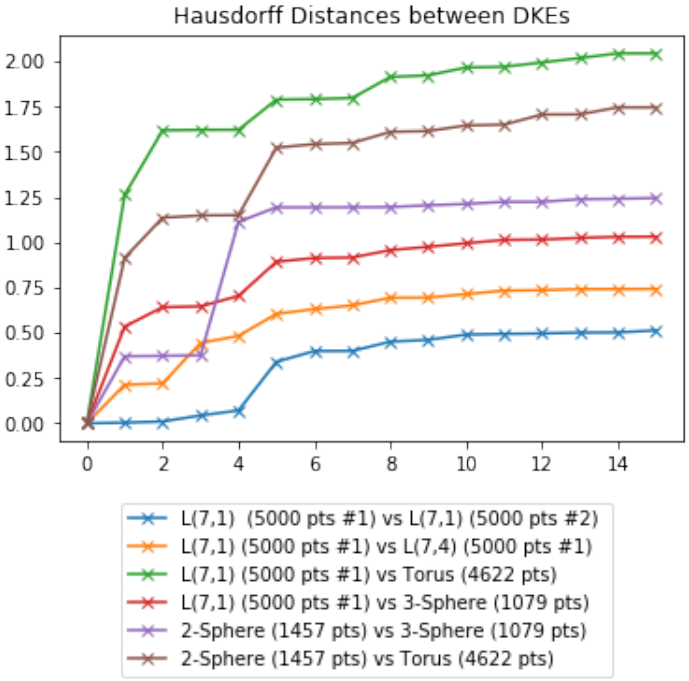
- Using the sampling and stability results for the distance kernel operator (Theorems 17 and 18) to provide analogous results for the topological transforms.
- Proving that the truncated distance kernel embedding is an injection for k sufficiently large. This for example the case for Laplacian eigenfunctions on manifolds, as shown by Bates [1], whose proof relies on deeper results in spectral geometry.
- Providing general hypotheses that ensure the definability of the DKE.
- Obtaining experimental results for these topological transforms in line with the distance kernel embedding experiments of Section 6. These experiments will hinge on a principled method for choosing which vector directions should be used for the computation of topological invariants; this is a question of interest in the TDA community, and we expect some heuristics and theoretical guarantees to emerge on this topic in the near future.



(a) Eigenvalues of the DKO for a variety of spaces, normalized by the number of points in the sample.



(b) A comparison of the eigenvalues of various samples, at different resolutions, of these two Lens spaces.



(c) A comparison of Hausdorff distances between various samples of 2- and 3-manifolds.

Figure 1 Spectra and DKE for samples of various manifolds. In subfigures (a) and (b), the x-represents the index of the eigenvalues in the sorted sequence of eigenvalues. In subfigure (c), the x-axis represents the embedding dimension (over $\mathbb{C}$).

References

1 Jonathan Bates. The embedding dimension of Laplacian eigenfunction maps. *Applied and Computational Harmonic Analysis*, 37(3):516–530, 2014.

2 Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in neural information processing systems*, pages 585–591, 2002.

3 Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396, 2003.

4 Mikhail Belkin and Partha Niyogi. Convergence of laplacian eigenmaps. In *Advances in Neural Information Processing Systems*, pages 129–136, 2007.

5 Robin Lynne Belton, Brittany Terese Fasy, Rostik Mertz, Samuel Micka, David L Millman, Daniel Salinas, Anna Schenfisch, Jordan Schupbach, and Lucia Williams. Learning simplicial complexes from persistence diagrams. *arXiv preprint*, 2018. `arXiv:1805.10716`.

6 Dmitri Burago, Sergei Ivanov, and Yaroslav Kurylev. A graph discretization of the laplace-beltrami operator. *arXiv preprint*, 2013. `arXiv:1301.2222`.

7 Ronald R Coifman and Stéphane Lafon. Diffusion maps. *Applied and computational harmonic analysis*, 21(1):5–30, 2006.

8 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*, 2016. `arXiv:1611.06818`.

9 Justin Curry, Sayan Mukherjee, and Katharine Turner. How many directions determine a shape and other sufficiency results for two topological transforms. *arXiv preprint*, 2018. `arXiv:1805.09782`.

10 Tamal K Dey, Dayu Shi, and Yusu Wang. Comparing graphs via persistence distortion. *arXiv preprint*, 2015. `arXiv:1503.07414`.

11 Gerald B Folland. *A Guide to Advanced Real Analysis*. Number 37 in Dolciani Mathematical Expositions. MAA, 2009.

12 Robert Ghrist, Rachel Levanger, and Huy Mai. Persistent homology and euler integral transforms. *Journal of Applied and Computational Topology*, 2(1-2):55–60, 2018.

13 Christopher Heil. Compact and hilbert–schmidt operators. In *A Basis Theory Primer*, pages 481–490. Springer, 2011.

14 Fritz John et al. The ultrahyperbolic differential equation with four independent variables. *Duke Mathematical Journal*, 4(2):300–322, 1938.

15 Clément Maria, Steve Oudot, and Elchanan Solomon. Intrinsic topological transforms via the distance kernel embedding. *arXiv preprint*, 2019. `arXiv:submit/2957368`.

16 Steve Oudot and Elchanan Solomon. Barcode embeddings for metric graphs. *arXiv preprint*, 2017. `arXiv:1712.03630`.

17 Steve Oudot and Elchanan Solomon. Inverse problems in topological persistence. *arXiv preprint*, 2018. `arXiv:1810.10813`.

18 Iosif Polterovich, Leonid Polterovich, and Vukašin Stojisavljević. Persistence barcodes and laplace eigenfunctions on surfaces. *Geometriae Dedicata*, 201(1):111–138, 2019.

19 Joshua B Tenenbaum, Vin De Silva, and John C Langford. A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500):2319–2323, 2000.

20 Katharine Turner, Sayan Mukherjee, and Doug M Boyer. Persistent homology transform for modeling shapes and surfaces. *Information and Inference: A Journal of the IMA*, 3(4):310–344, 2014.

21 Sathamangalam R Srinivasa Varadhan. On the behavior of the fundamental solution of the heat equation with variable coefficients. *Communications on Pure and Applied Mathematics*, 20(2):431–455, 1967.

Long Alternating Paths Exist

Wolfgang Mulzer
Institut für Informatik, Freie Universität Berlin, Germany
mulzer@inf.fu-berlin.de

Pavel Valtr
Department of Applied Mathematics, Faculty of Mathematics and Physics,
Charles University, Prague, Czech Republic
valtr@kam.mff.cuni.cz

Abstract

Let P be a set of $2n$ points in convex position, such that n points are colored red and n points are colored blue. A non-crossing alternating path on P of length ℓ is a sequence $p_1, \ldots, p_\ell$ of ℓ points from P so that (i) all points are pairwise distinct; (ii) any two consecutive points p_i, p_{i+1} have different colors; and (iii) any two segments $p_i p_{i+1}$ and $p_j p_{j+1}$ have disjoint relative interiors, for $i \neq j$.

We show that there is an absolute constant $\varepsilon > 0$, independent of n and of the coloring, such that P always admits a non-crossing alternating path of length at least $(1+\varepsilon)n$. The result is obtained through a slightly stronger statement: there always exists a non-crossing bichromatic separated matching on at least $(1+\varepsilon)n$ points of P. This is a properly colored matching whose segments are pairwise disjoint and intersected by common line. For both versions, this is the first improvement of the easily obtained lower bound of n by an additive term linear in n. The best known published upper bounds are asymptotically of order $4n/3 + o(n)$.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Non-crossing path, bichromatic point sets

Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.57

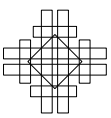
Related Version A full version is available on the arXiv (`https://arxiv.org/abs/2003.13291`).

Funding *Wolfgang Mulzer*: Supported in part by ERC StG 757609 and by the German Research Foundation within the collaborative DACH project *Arrangements and Drawings* as DFG Project MU-3501/3-1.
Pavel Valtr: Supported by the grant no. 18-19158S of the Czech Science Foundation (GAČR).

Acknowledgements This work was initiated at the second DACH workshop on Arrangements and Drawings which took place 21.–25. January 2019 at Schloss St. Martin, Graz, Austria. We would like to thank the organizers and all the participants of the workshop for creating a conducive research atmosphere and for stimulating discussions. Part of this work was done on the Seventh Annual Workshop on Geometry and Graphs, Bellairs Research Institute, Holetown, Barbados, 10.–15. March 2019. We also thank Zoltán Király for pointing out the reference [14] to us.

1 Introduction

We study a family of problems that were discovered independently in two different (but essentially equivalent) settings. Researchers in discrete and computational geometry found a geometric formulation, while researchers in computational biology and stringology studied circular words. Around 1989, Erdős asked the following geometric question [4, p. 409]: given a set P of n red and n blue points in convex position, how many points of P can always be collected by a non-intersecting polygonal path π with vertices in P such that the vertex-color along π alternates between red and blue. Taking every other segment of π, we obtain a properly colored set of pairwise disjoint segments with endpoints in P. A closely related



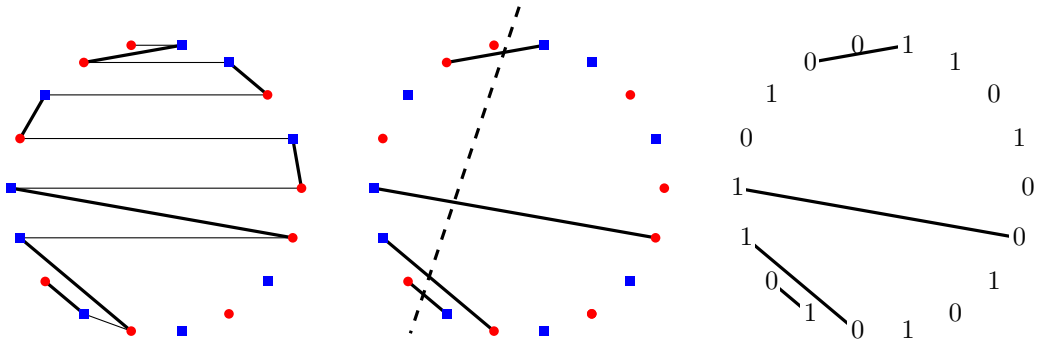


Figure 1 Left: a set P of 18 points in convex position, 9 of them red and 9 of them blue, with an alternating path of length 15 (this is not a longest such path). Taking every other segment, we obtain a properly colored disjoint matching on P. Middle: a separated matching on P, with a dashed line that intersects all matching edges (this is not a maximum such matching). Right: an antipalindromic subsequence on a circular word of 18 bits, 9 of them 0 and 9 of them 1.

problem asks for a large *separated matching*, a collection of such segments with the extra property that all of them are intersected by a common line. This is equivalent to finding a long *antipalindromic subsequence* in a circular sequence of $2n$ bits, where n bits are 0 and n bits are 1, see Figure 1. This formulation was stated in 1999 in a paper on protein folding [10]. Similar questions were also studied for *palindromic* subsequences [14]. One such question is equivalent to finding many disjoint monochromatic segments with endpoints in P, a problem that was also studied by the geometry community.

An easy lower bound for alternating paths is n, and the best known lower bound is $n + \Omega(\sqrt{n})$ [11]. We increase this to $cn + o(n)$, for a constant $c > 1$. Similarly, for the other mentioned problems, we improve the lower bounds by an additive term of εn, for some fixed $\varepsilon > 0$. Also here, this constitutes the first $\Omega(n)$ improvement over the trivial lower bounds.

The (geometric) setting. We have a set P of $2n$ points $p_0, p_1, \ldots, p_{2n-1}$ in convex position, numbered in clockwise order. The points in P are colored red and blue, so that there are exactly n red points and n blue points. The goal is to find a long *non-crossing alternating path* in P. That is, a sequence $\pi : q_0, q_1, \ldots, q_{\ell-1}$ of points in P such that (i) each point from P appears at most once in π; (ii) π is *alternating*, i.e., for $i = 0, \ldots, \ell - 2$, we have that q_i is red and q_{i+1} is blue or that q_i is blue and q_{i+1} is red; (iii) π is *non-crossing*, i.e., for $i, j \in \{0, \ldots, \ell - 2\}$, $i \neq j$, the two segments q_iq_{i+1} and q_jq_{j+1} intersect only in their endpoints and only if they are consecutive in π, see Figure 1(left). We will also just say *alternating path* for π. Alternating paths for planar point sets in general (not just convex) position have been studied in various previous papers, e.g., [1–3, 5, 6].

For most of this work, we will focus on another, closely related, structure. A *non-crossing separated bichromatic matching* M in P is a set $\{p_1q_1, p_2q_2, \ldots, p_kq_k\}$ of k pairs of points in P, such that (i) all points $p_1, \ldots, p_k, q_1, \ldots, q_k$ are pairwise distinct; (ii) the segments p_iq_i and p_jq_j are disjoint, for all $1 \leq i < j \leq k$; (iii) for $i = 1, \ldots, k$, the points p_i and q_i have different colors; and (iv) there exists a line that intersects all segments $p_1q_1, p_2q_2, \ldots, p_kq_k$, see Figure 1(middle). Often, we will just use the term *separated bichromatic matching* or simply *separated matching* for M.

Previous results. The following basic lemma says that a large separated matching immediately yields a long alternating path. The (very simple) proof was given by Kynčl, Pach, and Tóth [9, Section 3].

▶ **Lemma 1.** *Suppose that a bichromatic convex point set P admits a separated matching with k segments. Then, P has an alternating path of length $2k$.*

Let $l(n)$ be the largest number such that for every set P of n red and n blue points in convex position, there is an alternating path of length at least $l(n)$. Around 1989, Erdős and others [9] conjectured that $\lim_{n\to\infty} l(n)/n = 3/2$. Abellanas, García, Hurtado, and Tejel [1] and, independently, Kynčl, Pach, and Tóth [9, Section 3] disproved this by showing the upper bound $l(n) \leq 4n/3 + O(\sqrt{n})$. Kynčl, Pach, and Tóth [9] also improved the (almost trivial) lower bound $l(n) \geq n$ to $l(n) \geq n + \Omega(\sqrt{n/\log n})$. They conjectured that in fact $l(n) = 4n/3 + o(n)$. In her PhD thesis [11] (see also [8, 12, 13]), Mészáros improved the lower bound to $l(n) \geq n + \Omega(\sqrt{n})$, and she described a wide class of configurations where every separated matching has at most $2n/3 + O(\sqrt{n})$ edges. This also implies the upper bound $l(n) \leq 4n/3 + O(\sqrt{n})$ mentioned above [1, 9]. It was announced to us in personal communication that E. Csóka, Z. Blázsik, Z. Király, and D. Lenger constructed configurations with an upper bound of $cn + o(n)$ on the size of the largest separated matching, where $c = 2 - \sqrt{2} \approx 0.5858$ [7].

Our results. We improve the almost trivial lower bound $n/2$ for separated matchings to $n/2 + \varepsilon n$.

▶ **Theorem 2.** *There is a fixed $\varepsilon > 0$ such that any convex point set P with n red and n blue points admits a separated matching with at least $n/2 + \varepsilon n$ edges.*

By Lemma 1, we obtain the following corollary about long alternating paths.

▶ **Theorem 3.** *There is a fixed $\varepsilon > 0$ such that any convex point set P with n red and n blue points admits an alternating path with at least $n + \varepsilon n$ vertices.*

A variant of Theorem 2 also holds for the *monochromatic* case. The definition of a *non-crossing separated monochromatic matching*, or simply *separated monochromatic matching*, is obtained from the definition of a separated bichromatic matching by changing condition (iii) to (iii') for $i = 1, \ldots, k$, the points p_i and q_i have the same color. Some of the upper bound constructions for separated bichromatic matchings apply to the monochromatic setting, also giving the upper bound $2n/3 + O(\sqrt{n})$. Here is a monochromatic version of Theorem 2. Due to space reasons, the proof of Theorem 4 has been omitted from this extened abstract. It can be found in the full version of this paper.

▶ **Theorem 4.** *There are constants $\varepsilon > 0$ and $n_0 \in \mathbb{N}$ such any convex point set P with $n \geq n_0$ points, colored red and blue, admits a separated monochromatic matching with at least $n/2 + \varepsilon n$ vertices.*

There are two differences between the statement of Theorem 2 and Theorem 4: we do not require that the number of red and blue points in P is equal (and hence the size of the matching is stated in terms of vertices instead of edges), and we need a lower bound on the size of P. This is necessary, because Theorem 4 does not always hold for, e.g., $n = 4$. It was announced to us in a personal communication that the construction of E. Csóka, Z. Blázsik, Z. Király and D. Lenger from above also gives the upper bound $cn + o(n)$ on the size of a largest separated monochromatic matching, where $c = 2 - \sqrt{2} \approx 0.5858$ [7].

Our results in the setting of finite words. As we already said, the problems in this paper were independently discovered by researchers in computational biology and stringology. In a study on protein folding algorithms, Lyngsø and Pedersen [10] formulated a conjecture

that is equivalent to saying that the bound in Theorem 2 can be improved to $2n/3$ (for n divisible by 3). Müllner and Ryzhikov [14, p. 461] write that this conjecture "has drawn substantial attention from the combinatorics of words community". For the convenience of readers from this community, we rephrase our theorems for separated matchings in the finite words setting. We use the terminology of Müllner and Ryzhikov [14], without introducing it here. The following corresponds to Theorem 2.

▶ **Theorem 5.** *There is a fixed $\varepsilon > 0$ such that for any even $n \in \mathbb{N}$, every binary circular word of length n with equal number of zeros and ones has an antipalindromic subsequence of length at least $n/2 + \varepsilon n$.*

The following corresponds to Theorem 4.

▶ **Theorem 6.** *There are constants $\varepsilon > 0$ and $n_0 \in \mathbb{N}$ so that for any $n \in \mathbb{N}$, $n \geq n_0$, every binary circular word of length n has a palindromic subsequence of length at least $n/2 + \varepsilon n$.*

2 Existence of large separated bichromatic matchings

In this section, we prove our main result: large separated bichromatic matchings exist.

2.1 Runs and separated matchings

A *run* of P is a maximal sequence $p_i, p_{i+1}, \ldots, p_{i+\ell}$ of consecutive points with the same color.[1] That is, for $j = i, \ldots, i + \ell - 1$, the color of p_j and of p_{j+1} are the same, and the colors of p_{i-1} and p_i and the colors of $p_{i+\ell}$ and $p_{i+\ell+1}$ are different. The number of runs is always even. Kynčl, Pach, and Tóth showed that if P contains t runs, then P admits an alternating path of length $n + \Omega(t)$ [9, Lemma 3.2]. We will need the following analogous result for separated matchings. The proof can be found in the full version.

▶ **Theorem 7.** *Let $c_1 = 1/32$ and $t \geq 4$. Let P be a bichromatic convex point set with $2n$ points, n red and n blue, and suppose that P has t runs. Then, P admits a separated matching with at least $n/2 + c_1 t^2/n$ edges.*

2.2 Chunks, partitions, and configurations

Let $k \in \{1, \ldots, n\}$. A *k-chunk* is a sequence of consecutive points in P with exactly k points of one color and less than k points of the other color. Hence, a k-chunk has at least k and at most $2k - 1$ points. A *clockwise k-chunk with starting point p_i* is the shortest k-chunk that starts from p_i in clockwise order. A *counterclockwise k-chunk* with starting point p_i is defined analogously, going in the counterclockwise direction. For a k-chunk C, we denote by $r(C)$ the number of red points and by $b(C)$ the number of blue points in C. We call C a *red chunk* if $r(C) = k$ (and hence $b(C) < k$) and a *blue chunk* if $b(C) = k$ (and hence $r(C) < k$). The *index* of C is $b(C)/k$ for a red chunk and $r(C)/k$ for a blue chunk. Thus, the index of C lies between 0 and $(k-1)/k$, and it measures how "mixed" C is.

Next, let $k \in \{1, \ldots, n\}$ and $\lambda \in \mathbb{N} \cup \{0\}$. We define a *$(k, \lambda)$-partition*. Suppose that k is odd. First, we construct a maximum sequence $C_0, C_1, \ldots$ of clockwise disjoint k-chunks, as follows: we begin with the clockwise k-chunk C_0 with starting point p_0, and we let ℓ_0 be the number of points in C_0. Next, we take the clockwise k-chunk C_1 with starting point

[1] When calculating with indices of points in P, we will always work modulo $2n$.

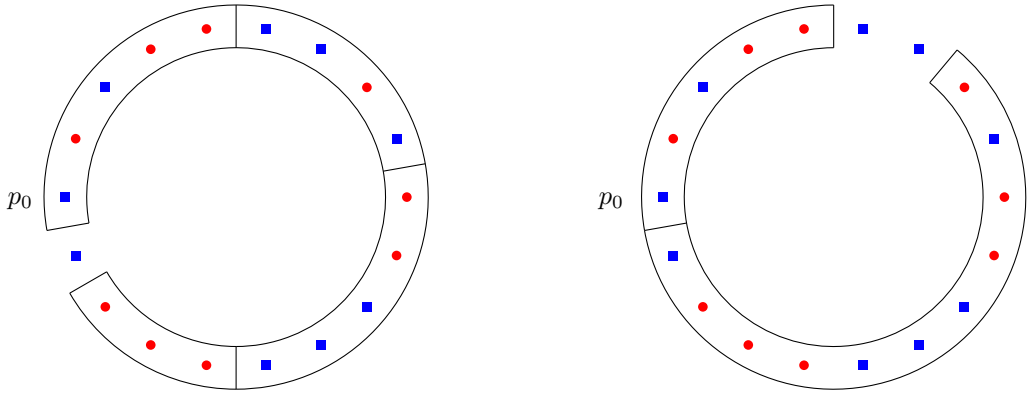


Figure 2 A set of 18 points and its $(3,0)$-partition (left) and $(3,1)$-partition (right). In the $(3,0)$-partition, the first chunk is red with index $2/3$, the second chunk is blue with index $1/3$, the third chunk is blue with index $2/3$, and the fourth chunk is red with index 0. The average red index is $1/3$, the average blue index is $1/2$. The index of the $(3,0)$-partition is $1/2$. The $(3,1)$-partition has one clockwise 3-chunk and one counterclockwise 6-chunk. The 3-chunk is red with index $2/3$, the 6-chunk is red with index $5/6$. The average red index is $3/4$, the average blue index is 0 The index of the $(3,1)$-partition is $3/4$.

p_{ℓ_0}, and let ℓ_1 be the number of points in C_1. After that, we take the clockwise k-chunk C_2 with starting point $p_{\ell_0+\ell_1}$, and so on. We stop once we reach the last k-chunk that does not overlap with C_0. Next, we construct a maximum sequence $D_0, D_1, \dots$ of *counterclockwise* $(k+3)$-chunks, starting with the point p_{2n-1}, in an analogous manner. Let λ' be the minimum of λ and the number of $(k+3)$-chunks D_i. Now, to obtain the (k,λ)-partition, we take λ' counterclockwise $(k+3)$-chunks $D_0, \dots, D_{\lambda'-1}$ and a maximum number of clockwise k-chunks $C_0, C_1, \dots$ that do not overlap with $D_0, \dots, D_{\lambda'-1}$. If k is even, the (k,λ)-partition is defined analogously, switching the roles of the clockwise and the counterclockwise direction. There may be some points that do not lie in any chunk of the (k,λ)-partition. We call these points *uncovered*.

The *average red index* of Γ is the average index in a red chunk of Γ (0, if there are no red chunks). The *average blue index* of Γ is defined analogously. The *index* of Γ is the maximum of the average red index and the average blue index of Γ. The *max-index color* is the color whose average index achieves the index of Γ, the other color is called the *min-index color*, see Figure 2 for an illustration of the concepts so far. The following simple proposition helps us bound the number of chunks. The (somewhat technical) proof can be found in the full version.

▶ **Proposition 8.** *Let P be a convex bichromatic point set with $2n$ points, n red and n blue, and let Γ be a (k,λ)-partition of P. In Γ, there are at most $2k-2$ uncovered points, at most $k-1$ of them red and at most $k-1$ of them blue. Furthermore, let R be the number of red chunks and B the number of blue chunks in Γ, and let α be the index of Γ. Then,*

$$R + B \leq \frac{2n}{k} \quad \text{and} \quad \max\{R,B\} \leq \frac{n}{k}. \tag{1}$$

Furthermore, we have

$$R + B \geq \left\lfloor \frac{2n}{2k+5} \right\rfloor > \frac{2n}{7k} - 1, \quad \max\{R,B\} \geq \frac{1}{2}\left\lfloor \frac{2n}{2k+5} \right\rfloor > \frac{n}{7k} - \frac{1}{2},$$
$$\text{and} \quad \min\{R,B\} \geq \frac{1-\alpha}{2}\left\lfloor \frac{2n}{2k+5} \right\rfloor - \frac{k-1}{k+3} > (1-\alpha)\frac{n}{7k} - 2. \tag{2}$$

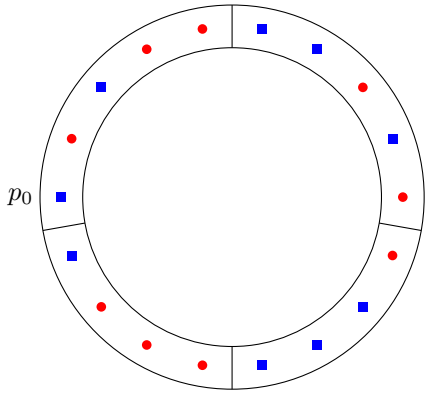


Figure 3 A set of 18 points and a 3-configuration for it. The chunk from p_0 is red with index 2/3, the next clockwise chunk is blue with index 2/3, followed by another blue chunk of index 1/3 and a final red chunk of index 1/3. The average blue index and the average red index are both 1/2. Note that the chunks are not minimal.

If $\lambda = 0$, the lower bounds improve to

$$R + B \geq \left\lfloor \frac{2n}{2k-1} \right\rfloor \geq \frac{n}{k} - 1, \quad \max\{R, B\} \geq \frac{1}{2} \left\lfloor \frac{2n}{2k-1} \right\rfloor > \frac{n}{2k} - \frac{1}{2},$$
$$\text{and} \quad \min\{R, B\} \geq \frac{1-\alpha}{2} \left\lfloor \frac{2n}{2k-1} \right\rfloor - \frac{k-1}{k} > (1-\alpha)\frac{n}{2k} - 2. \quad (3)$$

The purpose of the (k, λ)-partitions is to transition smoothly between the $(k, 0)$-partition and the $(k+3, 0)$-partition. In our proof, this will enable us to gradually increase the chunk-sizes, while keeping the index under control.

A *k-configuration* of P is a partition of P into k-chunks, leaving no uncovered points, see Figure 3. In contrast to a (k, λ)-partition, the chunks in a k-configuration are not necessarily minimal. Note that while P always has a (k, λ)-partition, it does not necessarily admit a k-configuration. The average red index, the average blue index, etc. of a k-configuration are defined as for a (k, λ)-partition. The following proposition helps us bound the number of chunks in a k-configuration. The proof can be found in the full version.

▶ **Proposition 9.** *Let P be a convex bichromatic point set with $2n$ points, n red and n blue, and let Γ be a k-configuration of P. Let R be the number of red chunks, B the number of blue chunks,α the average red index and β the average blue index of Γ. Then,*

$$n = kR + \beta kB = kB + \alpha kR. \quad (4)$$

Furthermore, $R + B \geq n/k$, $\max\{R, B\} \geq n/2k$, and $\min\{R, B\} \geq (1 - \max\{\alpha, \beta\})n/2k$. Finally, $\max\{R, B\} = R$ if and only if $\alpha \geq \beta$.

In our proof, the key challenge will be to analyze k-configurations with small constant index (say, around 0.1).

2.3 From (k, λ)-partitions to k-configurations

Our first goal is to show that we can focus on (k, λ)-partitions with large k and constant, but not too large index. We begin by noting that if the $(k, 0)$-partition of P for a constant k has a large index, then we can find a long alternating path in P. The proof can be found in the full version.

▶ **Lemma 10.** *Set $c_2 = 1/12800$. Let $k, n \in \mathbb{N}$ with $8k^2 \leq n$. Let P be a convex bichromatic point set with $2n$ points, n red and n blue. If the $(k, 0)$-partition Γ of P has index at least 0.1, then P admits a separated matching of size at least $(1/2 + c_2/k^4)n$.*

Next, we show that if the $(k, 0)$-partition still has a small index for $k = \Omega(n)$, then we can find a large separated matching. The proof, which is inspired by a simlar argument of Kynčl, Pach, and Tóth [9, Lemma 3.1], can be found in the full version.

▶ **Lemma 11.** *Set $c_3 = 1/81$. Let $k, n \in \mathbb{N}$ with $k \leq n$ and $6480n \leq k^2$. Let P be a convex bichromatic point set with $2n$ points. If the $(k, 0)$-partition Γ of P has index at most 0.1, then P admits a separated matching of size at least $(1/2 + c_3(k/n)^2)n$.*

Our goal now is to show that we can focus on k-configurations with k neither too small nor too large, and of index approximately 0.1. Here, we only sketch the argument, and we will make it more precise below, once all the lemmas have been stated formally: we choose $k_1 = O(1)$ and $k_2 = \Omega(n)$ to satisfy the previous two lemmas, and we consider the sequence of the $(k_1, 0)$-partition, the $(k_1, 1)$-partition, the $(k_1, 2)$-partition, ..., up to the $(k_2, 0)$-partition of P. By Lemma 10 and Lemma 11, we can assume that the first partition in the sequence has index less than 0.1 and the last partition in the sequence has index larger than 0.1. Thus, at some point the index has to jump over 0.1. Our definition of (k, λ)-partition ensures that this jump is gradual. The proof can be found in the full version.

▶ **Lemma 12.** *Let $k, n \in \mathbb{N}$ with $n \geq 210000k$. Let P be a convex bichromatic point set with $2n$ points, n red and n blue. Let Γ_1 be the (k, λ)-partition and Γ_2 the $(k, \lambda + 1)$-partition of P. Suppose that the index of Γ_1 is at most 0.1. Then, the average red index and the average blue index of Γ_1 and Γ_2 each differ by at most 0.001.*

It follows that we can assume that we are dealing with a (k, λ)-partition of index approximately 0.1. Actually, we will see that it suffices to consider *k-configurations* of index 0.1. This will be the focus of the next section.

2.4 Random chunk-matchings in k-configurations

In this section, we will focus on convex bichromatic point sets P that admit a k-configuration Γ with special properties. Later, we will see how to reduce to this case.

Let $C_0, C_1, \ldots, C_{\ell-1}$ be the chunks of the k-configuration Γ. We define a notion of *chunk-matching*, as illustrated in Figures 4 and 5. A chunk matching pairs each of the ℓ chunks with another chunk (possibly itself). Our goal is to define chunk matchings in such a way that we can easily derive from a chunk matching a separated matching between the points in P.

Formally, we define ℓ matchings $M_0, \ldots, M_{\ell-1}$ by saying that for $i, j = 0, \ldots, \ell - 1$, the matching M_i pairs the chunks C_j and $C_{(i-j) \bmod \ell}$. Again, refer to Figures 4 and 5 for examples. The matching rule is symmetric, i.e., if C_a is matched to C_b then C_b is matched to C_a. Note that if $j \equiv (i - j) \pmod{\ell}$, the chunk C_j is matched to itself in M_i. If ℓ is even, this happens only for even i, namely for $j = i/2$ and for $j = i/2 + \ell/2$. If ℓ is odd, this happens in every matching, namely for $j \equiv (\ell + 1)i/2 \pmod{\ell}$. By construction, for every M_i, if we connect the matched chunks by straight line edges, we obtain a set of plane segments such that there is one line that intersects all segments. Furthermore, every pair C_i, C_j of chunks, $0 \leq i \leq j \leq \ell - 1$ appears in exactly one chunk matching. In essence, these matchings correspond to partitioning the chunks of Γ with a line, where the line can possibly pass through one or two chunks of Γ that are then matched to themselves.

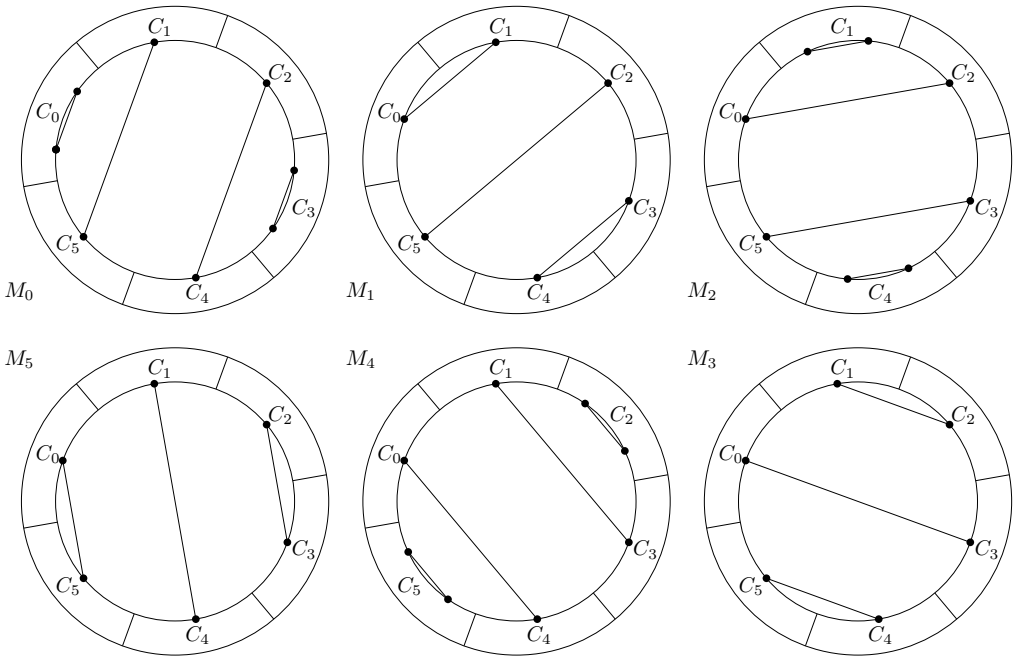


Figure 4 The six chunk matchings $M_0, \ldots, M_5$ for a set of six chunks. If i is even, the chunks $C_{i/2}$ and $C_{i/2+3}$ are matched to themselves. If i is odd, every chunk is matched to a different chunk.

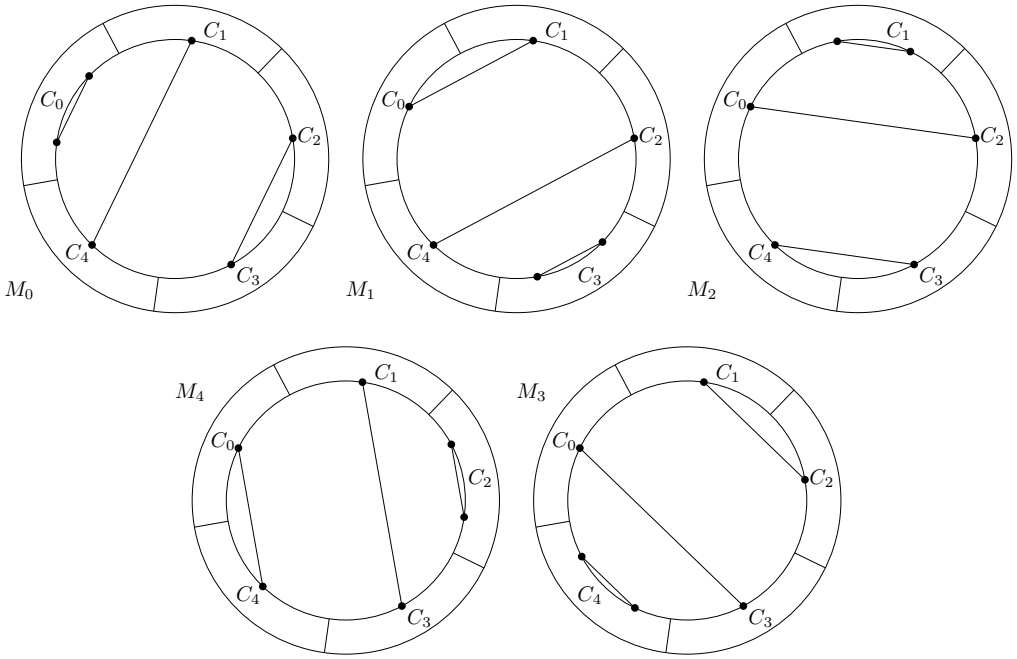


Figure 5 The five chunk matchings $M_0, M_1, \ldots, M_4$ for a set of five chunks. In matching M_i, the chunk $C_{(3i \bmod 5)}$ is matched to itself. Every other chunk is matched to a different chunk.

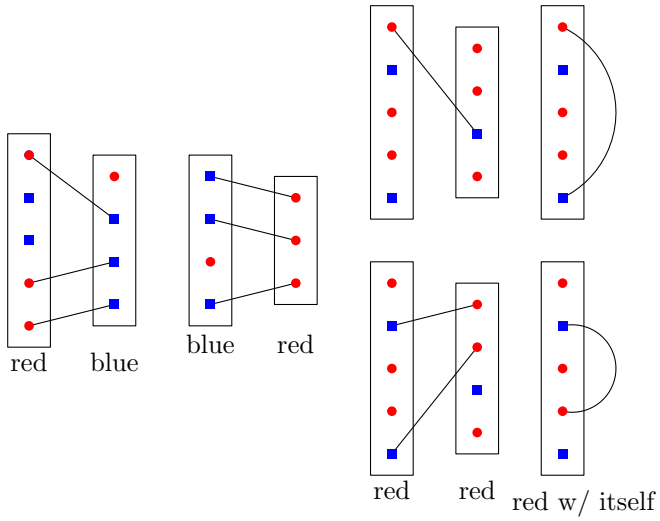


Figure 6 Going from a matched pair of chunks to a separated matching. If the two chunks have different colors, we can match k edges. If the two colors are the same, there are two reasonable options, matching the red points in one chunk with the blue points in the other chunk. We choose the one that matches more edges. A special case occurs if a chunk is matched to itself. In this case, we split the majority color into half and match between the halves.

Next, we describe how to derive from a given chunk matching M a separated matching on P, see Figure 6 for an illustration. We look at every two chunks C and D paired my M (possibly, $C = D$). If C is red and D blue, we match the k red points in C to the k blue points in D, getting k matched edges. The case that C is blue and D is red is analogous. If $C \neq D$ and both C and D are red, we could match the k red points in C to the $b(D) < k$ blue points in D, or vice versa. We choose the option that gives more edges, yielding $\max\{b(C), b(D)\}$ matched edges. The case that $C \neq D$ and both are blue is similar. Finally, suppose that $C = D$, and for concreteness, suppose that C is red. In this case, we split the points in C into two parts, containing $\lceil k/2 \rceil$ red points each (if k is odd, the median point belongs to both parts). In one part, we have at least $\lceil b(C)/2 \rceil$ blue points, and we match these blue points to the red points in the other part. This yields $\lceil b(C)/2 \rceil \geq b(C)/2$ matched edges. Thus, a chunk matching M gives a separated matching with at least

$$\frac{1}{2}\Bigg(\sum_{\substack{(C,D)\in M \\ C \text{ red}, D \text{ red}}} \max\{b(C), b(D)\} + \sum_{\substack{(C,D)\in M \\ C \text{ red}, D \text{ blue}}} k + \sum_{\substack{(C,D)\in M \\ C \text{ blue}, D \text{ red}}} k + \sum_{\substack{(C,D)\in M \\ C \text{ blue}, D \text{ blue}}} \max\{r(C), r(D)\} \Bigg) \quad (5)$$

matched edges, where the sums go over all ordered pairs of matched chunks in M, i.e., a matched pair (C, D) with $C \neq D$ appears twice (which is compensated by the leading factor of $1/2$) and a matched pair (C, C) appears once. The next lemma shows that a chunk matching that is chosen uniformly at random usually matches half the points of P.

▶ **Lemma 13.** *Let Γ be a k-configuration of P and M a random chunk matching in Γ. The expected number of matched edges in the corresponding separated matching is at least $n/2$.*

Proof. Let R be the number of red chunks in Γ and B the number of blue chunks in Γ. Let α be the average index of the red chunks, and β the average index of the blue chunks. We sum (5) over all $R+B$ possible chunk matchings and take the average. This gives the expected number of matched edges (the sums range over all ordered pairs of chunks in Γ).

$$\frac{1}{2(R+B)}\Bigg(\sum_{C \text{ red}, D \text{ red}} \max\{b(C), b(D)\} + 2\sum_{C \text{ red}, D \text{ blue}} k + \sum_{C \text{ blue}, D \text{ blue}} \max\{r(C), r(D)\}\Bigg)$$

Since there are R red chunks and B blue chunks, this is

$$= \frac{1}{2(R+B)}\left(\sum_{C \text{ red}, D \text{ red}} \max\{b(C), b(D)\} + 2kRB + \sum_{C \text{ blue}, D \text{ blue}} \max\{r(C), r(D)\}\right)$$

We lower bound the maximum by the average to estimate this as

$$\geq \frac{1}{2(R+B)}\left(\sum_{C \text{ red}, D \text{ red}} \frac{b(C)+b(D)}{2} + 2kRB + \sum_{C \text{ blue}, D \text{ blue}} \frac{r(C)+r(D)}{2}\right) \qquad (**)$$

Simplifying the sums, this is

$$= \frac{1}{2(R+B)}\left(R\sum_{C \text{ red}} b(C) + 2kRB + B\sum_{C \text{ blue}} r(C)\right)$$

Since the total number of blue points in red chunks is αkR and the total number of red points in blue chunks is βkB, this equals

$$= \frac{\alpha kR^2 + 2kRB + \beta kB^2}{2(R+B)}$$

Regrouping the terms and using (4), this becomes

$$= \frac{R(\alpha kR + kB) + B(\beta kB + kR)}{2(R+B)} = \frac{(R+B)n}{2(R+B)} = \frac{n}{2}.$$ ◀

2.5 Taking advantage of k-configurations

One inefficiency in the calculation in Lemma 13 is that we bound the maximum by the average in inequality (**). If these two quantities often differ significantly, we can gain an advantage over Lemma 13. This is made precise in the next lemma.

▶ **Lemma 14.** *Set $c_4 = 1/40$. Let $\delta > 0$ and let P be a convex bichromatic point set with $2n$ points, n red and n blue, and Γ a k-configuration for P with index at most 0.11 that contains at least $\delta(n/k)$ red chunks or at least $\delta(n/k)$ blue chunks with index at least 0.22. Then, P admits a separated matching of size at least $(1/2 + c_4\delta^2)n$.*

Proof. Suppose without loss of generality that there are at least $\delta(n/k)$ red chunks with index at least 0.2. Let R be the number of red chunks and B the number of blue chunks. The average red index of Γ is at most 0.11. Thus, if writing $\gamma_1(n/k)$ for the number of red chunks with index in $(0.11, 0.22)$ and $\gamma_2(n/k) \geq \delta(n/k)$ for the number of red chunks with index in $[0.22, 1)$, we have

$$0.11R \geq 0.11\gamma_1\frac{n}{k} + 0.22\gamma_2\frac{n}{k} = 0.11(\gamma_1 + 2\gamma_2)\frac{n}{k}.$$

It follows that $R \geq (\gamma_1 + 2\gamma_2)(n/k)$, and there must be at least $\gamma_2(n/k) \geq \delta(n/k)$ red chunks of index in $[0, 0.11]$. Now, consider the following sum over all ordered pairs (C, D) of red chunks, where one chunk (C or D) has red index at most 0.11 and the other chunk (D or C) has red index at least 0.22:

$$\frac{1}{2(R+B)} \left(\sum_C \sum_D \max\{b(C), b(D)\} - \frac{b(C) + b(D)}{2} \right)$$

Since $2\max\{a, b\} - a - b = \max\{a, b\} - \min\{a, b\}$, for all $a, b \in \mathbb{R}$, this equals

$$= \frac{1}{4(R+B)} \sum_C \sum_D (\max\{b(C), b(D)\} - \min\{b(C), b(D)\})$$

One chunk in each summand contains at least $0.22k$ blue points, the other chunk contains at most $0.11k$ blue points, so we can lower bound this as

$$\geq \frac{1}{4(R+B)} \sum_C \sum_D (0.22 - 0.11)k$$
$$\geq \frac{1}{4(R+B)} \sum_C \sum_D \frac{k}{10} \geq \frac{\delta^2(n/k)^2}{R+B} \frac{k}{20} \geq \frac{\delta^2}{40} n,$$

since we are adding over at least $2\delta^2(n/k)^2$ ordered pairs (C, D) (recall that each ordered pair (C, D) has a partner (D, C) in the sum) and since by (1), we have $R + B \leq 2n/k$. Thus, comparing with (**), the lemma follows. ◀

Lemma 14 shows that we can assume that few chunks in the k-configuration Γ of P have index larger than 0.22. In fact, suppose now that Γ contains no chunk of index at least 0.3 (this will be justified below). From now on, we will also assume that k is divisible by 3. We subdivide each chunk in our k-configuration Γ into three $(k/3)$-*subchunks*. Since all k-chunks have index less than 0.3, the subchunks have the same color as the original chunk. Let C be a k-chunk. The *middle subchunk* of C, denoted by C_M, is the $(k/3)$-subchunk of C that lies in the middle of the three subchunks. Now, we consider the middle subchunks. If the middle subchunks of the max-index color contain many points of the min-index color, we can gain an advantage by considering two *cross-matchings* between chunks of the max-index color.

▶ **Lemma 15.** *Set $c_5 = 1/4$. Let $\delta > 0$ and let P be a convex bichromatic point set with $2n$ points, n red and n blue. Let Γ be a k-configuration for P such that (i) k is divisible by 3; (ii) every chunk in Γ has index less than 0.3; and (iii) the middle subchunks of the max-index color contain in total at least δn points of the min-index color. Then P admits a separated matching of size at least $(1/2 + c_5\delta)n$.*

Proof. Suppose that the max-index color is red. We take a random chunk matching M of Γ, and we derive a separated matching from M as described above. However, when considering a pair (C, D) of two red chunks, we proceed slightly differently. First, suppose that $C \neq D$, and let C_1, C_2, C_3 be the three subchunks of C, and D_1, D_2, D_3 be the three subchunks of D (in clockwise order). We have $r(C_i) = r(D_i) = k/3$, for $i = 1, 2, 3$; and $b(C_1) + b(C_2) + b(C_3) < k/3$ and $b(D_1) + b(D_2) + b(D_3) < k/3$. We consider two separated matchings between C and D (see Figure 7(left): (a) match all blue points in C_1 and C_2 to red points in D_3 and all blue points in D_1 and D_2 to red points in C_3; and (b) match all blue

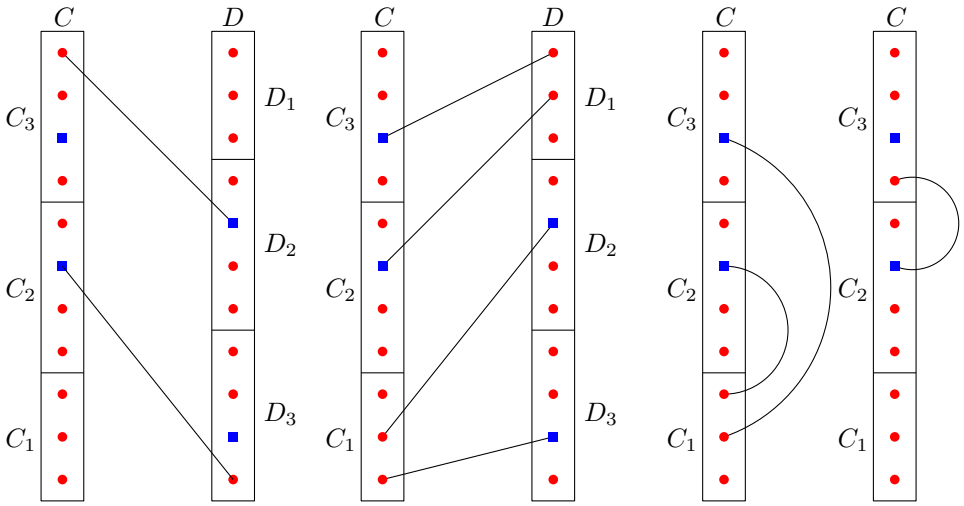


Figure 7 The two different separated matchings between two distinct red chunks (left) and the same red chunk (right).

points in D_2 and D_3 to red points in C_1 and all blue points in C_2 and C_3 to red points in D_1. We take the better of the two matchings. The number of matched edges $\text{matched}(C, D)$ is lower-bounded by the average, so

$$\begin{aligned}\text{matched}(C,D) &\geq \frac{1}{2}\left(b(C_1)+b(C_2)+b(D_1)+b(D_2)+b(D_3)+b(D_2)+b(C_2)+b(C_3)\right)\\ &= \frac{1}{2}\left(b(C)+b(D)+b(C_2)+b(D_2)\right). \qquad (6)\end{aligned}$$

Second, if $C = D$, we subdivide C into the three subchunks C_1, C_2, C_3 with $(C_1) = r(C_2) = r(C_3) = k/3$ and $b(C_1) + b(C_2) + b(C_3) < k/3$. Again, we consider two different matchings for C (see Figure 7(right): (a) match the blue points in C_1 and C_2 to the red points in C_3, and (b) match the blue points in C_2 and C_3 to the red points in C_1. Again, the number of matched edges $\text{matched}(C, C)$ is at least

$$\text{matched}(C,C) \geq \frac{1}{2}(b(C_1)+b(C_2)+b(C_2)+b(C_3)) = \frac{1}{2}(b(C)+b(C_2)). \qquad (7)$$

Now, we set R to the number of red chunks and B to the number of blue chunks in Γ. Then, in a random chunk matching, the expected number of edges in the separated matchings between the pairs (C, D) of red chunks is

$$\frac{1}{2(R+B)}\left(\sum_{C\neq D, C, D \text{ red}} \text{matched}(C,D) + \sum_{C \text{ red}} 2\,\text{matched}(C,C)\right). \qquad (8)$$

Note that in the first sum, each unordered pair $\{C, D\}$ of distinct red chunks appears twice, even though it appears once in a random chunk matching. This is compensated by the leading factor of $1/2$, which again leads to a coefficient of 2 for the expected number of edges in the separated matching in a chunk that is paired with itself. Using (6, 7), we can write

$$(8) \geq \frac{1}{2(R+B)}\left(\sum_{C \text{ red}}\sum_{D \text{ red}} \frac{b(C)+b(D)+b(C_M)+b(D_M)}{2}\right),$$

where we sum over all ordered pairs (C, D) of red chunks and C_M and D_M denote the middle chunks of C and D. Now we compare with (**).

$$\frac{1}{2(R+B)}\left(\sum_{C \text{ red}}\sum_{D \text{ red}} \frac{b(C)+b(D)+b(C_M)+b(D_M)}{2} - \frac{b(C)+b(D)}{2}\right)$$
$$= \frac{1}{2(R+B)}\left(\sum_{C \text{ red}}\sum_{D \text{ red}} \frac{b(C_M)+b(D_M)}{2}\right)$$

In the sum, every middle chunk C_M and every middle chunk D_M appears exactly R times, and by assumption, the total number of blue points in the red middle chunks is at least δn. Thus, this is lower-bounded as

$$\geq \frac{1}{2(R+B)} R\delta n \geq \frac{1}{4R} R\delta n = \frac{\delta}{4} n,$$

since red is the max-index color and hence by Proposition 9, we have $B \leq R$ and $R+B \leq 2R$. Thus, the lemma follows. ◀

Finally, we consider the case that the middle subchunks of the max-index color contain relatively few points. Since the index of Γ is relatively small, it means that the indices of the middle subchunks of the max-index color have a large variance. As in Lemma 14, this leads to a large separated matching. The proof is very similar to the proof of Lemma 14, and it can be found in the full version.

▶ **Lemma 16.** *Set $\delta = 10^{-4}$ and $\varepsilon = 10^{-5}$. Let P be a convex bichromatic point set with $2n$ points, n red and n blue, and let Γ be a k-configuration for P such that (i) k is divisible by 3; (ii) Γ has index at least 0.09; and (iii) every chunk in Γ has index less than 0.3. Then, if the middle subchunks of the max-index color contain in total at most δn points of the min-index color, P admits a separated matching of size at least $(1/2+\varepsilon)n$.*

2.6 Putting it together

From Theorem 7, it follows that if P has at least four runs, there is always a separated matching with strictly more than $n/2$ edges. Moreover, if P has two runs, then P has a separated matching with $n > n/2$ edges. Therefore, the following theorem implies Theorem 2.

▶ **Theorem 17.** *There exist constants $\varepsilon_* > 0$ and $n_0 \in \mathbb{N}$ with the following property: let P be a convex bichromatic point set with $2n \geq 2n_0$ points, n red and n blue. Then, P admits a separated matching on at least $(1+\varepsilon_*)n$ vertices.*

Proof. Set $n_0 = 10^{100}$ and $\varepsilon = 10^{-5}$, as in Lemma 16. Let k_1 the smallest integer larger than $10^3\varepsilon^{-3} = 10^{18}$ that is divisible by 3. Since $n \geq 10^{100} \geq 8k_1^2$, Lemma 10 shows that if the $(k_1, 0)$-partition Γ_1 of P has index at least 0.1, the theorem follows with $\varepsilon_* = \Omega(1/k_1^4) = \Omega(1)$. Thus, we may assume the following claim:

▷ Claim 18. The $(k_1, 0)$-partition Γ_1 of P has index less than 0.1, where k_1 is a fixed constant with $k_1 \geq 10^3\varepsilon^{-3} = 10^{18}$.

Next, let k_2 be the largest integer in the interval $[10^{-4}\varepsilon^3 n, 10^{-3}\varepsilon^3 n]$ that is divisible by 3. Since $n \geq 10^{100}$, it follows that k_2 exists. Furthermore, since $n \geq k_2$ and $6480n \leq 10^{-8}\varepsilon^6 n^2 \leq k_2^2$, Lemma 11 implies that if the $(k_2, 0)$-partition Γ_2 of P has index at most 0.1, the theorem follows with $\varepsilon_* = \Omega((k_2/n)^2) = \Omega(1)$. Hence, we may assume the following claim:

▷ Claim 19. The $(k_2, 0)$-partition Γ_2 of P has index more than 0.1, where k_2 is the largest integer in the interval $[10^{-4}\varepsilon^3 n, 10^{-3}\varepsilon^3 n]$ that is divisible by 3.

We now interpolate between Γ_1 and Γ_2. Consider the sequence of (k, λ)-partitions of P for the parameter pairs

$$(k_1, 0), (k_1, 1), \ldots, (k_1, \lambda(k_1)), (k_1 + 3, 0), (k_1 + 3, 1), \ldots, \\ (k_1 + 3, \lambda(k_1 + 3)), (k_1 + 6, 0), \ldots, (k_2, 0),$$

where $\lambda(k)$ denotes the largest λ for which the (k, λ)-partition of P still contains a k-chunk. Let (k_*, λ_*) be the first parameter pair for which the index of the (k_*, λ_*)-partition Γ_3 of P is larger than 0.1. This parameter pair exists, because $(k_2, 0)$ is a candidate.

▷ Claim 20. The (k_*, λ_*)-partition Γ_3 of P has index in $[0.1, 0.101]$. Here, k_* is divisible by 3 and lies in the interval $[10^3\varepsilon^{-3}, 10^{-3}\varepsilon^3 n]$.

Proof. The claim on k_* and the fact that Γ_3 has index at least 0.1 follow by construction. Furthermore, let (k_{**}, λ_{**}) be such that Γ_3 is the $(k_{**}, \lambda_{**} + 1)$ partition of P (we either have $k_{**} = k_*$ and $\lambda_{**} = \lambda_* - 1$; or $k_{**} = k_* - 1$ and $\lambda_{**} = \lambda(k_{**})$). Since $210000k_{**} \leq 10^6 \cdot 10^{-3}\varepsilon^3 n \leq n$, Lemma 12 implies that the index of Γ_3 is at most 0.101. ◁

We rearrange P to turn Γ_3 into a k_*-configuration Γ_4 of a closely related point set P_2.

▷ Claim 21. There exists a convex bichromatic point set P_2 with $2n$ points, n red and n blue, and a k_*-configuration Γ_4 of P_2 such that (i) P_2 differs from P in at most $10^{-1}\varepsilon^3 n$ points; and (ii) the index of Γ_4 lies in $[0.097, 0.103]$.

Proof. We remove from P all the uncovered points of Γ_3 as well as 3 points of the majority color from each $(k_* + 3)$-chunk of Γ_3 (and, if necessary, up to 3 points of the minority color, to keep chunk structure valid). If we consider a single red $(k_* + 3)$-chunk C and denote the original number of blue points in C by $b(C)$ and the resulting number of blue points by $b'(C)$, then the index of C changes by at most

$$\left|\frac{b(C)}{k_* + 3} - \frac{b'(C)}{k_*}\right| = \left|\frac{k_* b(C) - (k_* + 3)b'(C)}{k_*(k_* + 3)}\right| \leq \frac{|b(C) - b'(C)|}{k_* + 3} + \frac{3b'(C)}{k_*(k_* + 3)} \leq \frac{6}{k_* + 3},$$

since $|b(C) - b(C')| \leq 3$ and $b'(C) \leq k_*$. A similar bound holds for a blue $(k_* + 3)$-chunk.

By (1), there are at most $2n/k_* \leq 2 \cdot 10^{-3}\varepsilon^3 n$ many $(k_* + 3)$-chunks, and by Proposition 8, there at most $2k_* - 1 \leq 2 \cdot 10^{-3} \cdot \varepsilon^3 n$ uncovered points, so in total we remove at most $14 \cdot 10^{-3}\varepsilon^3 n \leq 10^{-1}\varepsilon^3 n$ points. We arrange these points into as many pure chunks of k_* red points or of k_* blue points as possible. This creates at most $10^{-1}\varepsilon^3(n/k_*)$ new k^*-chunks, all of which have index 0. Now, less than k_* red points and less than k_* blue points remain. By (2), there are at least

$$(1 - 0.101)\frac{n}{7k_*} - 2 \geq 10^{-1} \cdot 10^3\varepsilon^{-3} - 2 \geq 10^3$$

chunks of each color in Γ_3. Thus, we can partition the remaining red points into at most 10^3 groups of size at most $10^{-3}k_*$ and add each group to a single blue chunk; and similarly for the remaining blue points. This changes the index of each chunk by at most 10^{-3}.

We call the resulting rearranged point set P_2 and the resulting k_*-configuration Γ_4. As mentioned, P_2 was obtained from P by moving at most $10^{-1} \cdot \varepsilon^3 n$ points. We change the index of any existing chunk by at most $6/(k^* + 3) + 10^{-3} \leq 2 \cdot 10^{-3}$. Furthermore, we create at most $10^{-1}\varepsilon^3(n/k_*)$ new k_*-chunks (all of index 0) and by (2), we have at least $(1 - 0.101)n/(7k_*) - 2 \geq (10^{-1} - 10^{-2} \cdot \varepsilon^3)(n/k_*)$ original chunks of each color in Γ_3. Thus,

if we denote by α the average index of the existing red chunks after the rearrangement, by R the number of existing red chunks, and by R' the number of new red chunks, the average red index of Γ_4 can differ from α by at most

$$\alpha - \frac{R}{R+R'}\alpha = \alpha\frac{R'}{R+R'} \leq \alpha\frac{R'}{R} \leq 0.102\frac{10^{-1}\varepsilon^3}{10^{-1}-10^{-2}\varepsilon^3} \leq 10^{-3},$$

and similarly for the average blue index of Γ_5. It follows that Γ_4 has index in $[0.097, 0.103]$. ◁

Now, using Lemma 14 with $\delta = 10^{-1}\varepsilon$, we get that if the k^*-configuration Γ_4 contains at least $\delta(n/k_*)$ red chunks or at least $\delta(n/k_*)$ blue chunks with index at least 0.22, then the rearranged point set P_2 admits a separated matching of size at least

$$\left(\frac{1}{2} + \frac{1}{40}\cdot 10^{-2}\varepsilon^2\right)n \geq \left(\frac{1}{2} + 10^{-4}\cdot\varepsilon^2\right)n.$$

By Claim 21, P_2 differs from P by at most $10^{-1}\varepsilon^3 n$ points. Since $\varepsilon = 10^{-5}$, it follows that after deleting all matching edges incident to a rearranged point, we obtain the theorem. Thus, we may assume the following claim:

▷ Claim 22. At most $10^{-1}\cdot\varepsilon(n/k_*)$ red chunks and at most $10^{-1}\cdot\varepsilon(n/k_*)$ blue chunks in Γ_4 have index more than 0.22.

We again rearrange the point set P_2 to obtain a point set P_3 and a k^*-configuration Γ_5 for P_3 such that every k^*-chunk in Γ_5 has index less than 0.3.

▷ Claim 23. There exists a convex bichromatic point set P_3 with $2n$ points, n red and n blue, and a k_*-configuration Γ_5 of P_3 such that (i) P_3 differs from P_2 in at most $2\cdot 10^{-1}\varepsilon n$ points; (ii) the index of Γ_5 is at least 0.096; (iii) all chunks in Γ_5 have index less than 0.3; and (iv) k_* is divisible by 3.

Proof. We remove all the blue points from red chunks of index at least 0.22 and all the red points from all blue chunks of index at least 0.22. These are at most $2\cdot 10^{-1}\cdot\varepsilon n$ points in total. By removing these points, we decrease the index of at most $10^{-1}\varepsilon(n/k_*)$ existing chunks of each color to 0. By Proposition 9, there are at least

$$(1-0.103)\frac{n}{2k_*} \geq 10^{-1}\cdot\frac{n}{k_*} \tag{9}$$

existing chunks of each color, so this step decreases the average index by at most ε.

We rearrange the deleted points into as many pure chunks with k_* red points or with k_* blue points as possible. Less than k_* red points and less than k_* blue points remain. By (9), there are at least $10^{-1}(n/k_*) \geq 10^3$ chunks of each color, so we group the remaining points into blocks of size $10^{-3}\cdot k_*$ and distribute the blocks over the existing red and blue chunks. This increases the average index of the existing chunks by at most 10^{-3}.

Finally, we create at most $10^{-1}\cdot\varepsilon(n/k_*)$ new chunks of each color (all with index 0), and the existing number of chunks of the max-index color of Γ_4 is at least $n/2k_*$, by Proposition 9. Suppose for concreteness that the max-index color of Γ_4 is red, and let R be the number of existing red chunks, R' the number of new red chunks, and α the average index of the existing red chunks after the rearrangement. Then, the average red index after the rearrangement differs from α be at most

$$\alpha - \frac{R}{R+R'}\alpha \leq \alpha\frac{R'}{R} \leq 0.104\cdot\frac{10^{-1}\varepsilon}{1/2} \leq \varepsilon.$$

Thus, the red index in the resulting k_*-configuration Γ_5 is at least $0.097 - 2\varepsilon \geq 0.096$. This implies that the index of Γ_5 is at least 0.096. ◁

Now, we consider the k^*-configuration Γ_5. By Lemma 16, if in Γ_5 the middle-chunks of the max-index color contain in total at most $10^{-4}n$ points of the min-index color, we get a separated matching for P_3 of size at least $(1/2+\varepsilon)n$. By deleting all the matching edges that are incident to the at most $2 \cdot 10^{-1}\varepsilon n + 10^{-1}\varepsilon^3 n \leq 0.3\varepsilon n$ points that were moved to obtain P_3 from P, the theorem follows. Similarly, if in Γ_5 the middle-chunks of the max-index color contain in total more than $10^{-4}n$ points of the min-index color, by Lemma 15, we get a separated matching for P_3 of size at least $(1/2+10^4/4)n \geq (1/2+\varepsilon)n$. Again, we obtain the theorem after deleting edges that are incident to the rearranged points. ◀

References

1 Manuel Abellanas, Alfredo García, Ferran Hurtado, and Javier Tejel. Caminos alternantes. In *X Encuentros de Geometría Computational*, pages 7–12, 2003.

2 Oswin Aichholzer, Carlos Alegría, Irene Parada, Alexander Pilz, Javier Tejel, Csaba D. Tóth, Jorge Urrutia, and Birgit Vogtenhuber. Hamiltonian meander paths and cycles on bichromatic point sets. In *XVIII Spanish Meeting on Computational Geometry*, pages 35–38, 2019.

3 Jin Akiyama and Jorge Urrutia. Simple alternating path problem. *Discrete Mathematics*, 84(1):101–103, 1990. doi:10.1016/0012-365X(90)90276-N.

4 Peter Brass, William O. J. Moser, and János Pach. *Research problems in discrete geometry*. Springer, 2005.

5 Josef Cibulka, Jan Kynčl, Viola Mészáros, Rudolf Stolař, and Pavel Valtr. Universal Sets for Straight-Line Embeddings of Bicolored Graphs. In János Pach, editor, *Thirty Essays on Geometric Graph Theory*, pages 101–119, New York, NY, 2013. Springer New York. doi:10.1007/978-1-4614-0110-0_8.

6 Merce Claverol, Delia Garijo, Ferran Hurtado, Dolores Lara, and Carlos Seara. The alternating path problem revisited. In *XV Spanish Meeting on Computational Geometry*, pages 115–118, 2013.

7 Endre Csóka, Zoltán L. Blázsik, Zoltán Király, and Dániel Lenger. The necklace folding problem. Manuscript in preparation, 2020.

8 Peter Hajnal and Viola Mészáros. Note on noncrossing path in colored convex sets. unpublished preprint, 2010. URL: http://infoscience.epfl.ch/record/175677.

9 Jan Kynčl, János Pach, and Géza Tóth. Long alternating paths in bicolored point sets. *Discrete Mathematics*, 308(19):4315–4321, 2008.

10 Rune Lyngsø and Christian Pedersen. Protein Folding in the 2D HP Model. *BRICS Report Series*, 6(16), January 1999. doi:10.7146/brics.v6i16.20073.

11 Viola Mészáros. *Extremal problems on planar point sets*. PhD thesis, University of Szeged, Bolyai Institute, 2011.

12 Viola Mészáros. Separated matchings and small discrepancy colorings. In Alberto Márquez, Pedro Ramos, and Jorge Urrutia, editors, *Computational Geometry - XIV Spanish Meeting on Computational Geometry, EGC 2011, Dedicated to Ferran Hurtado on the Occasion of His 60th Birthday, Alcalá de Henares, Spain, June 27-30, 2011, Revised Selected Papers*, volume 7579 of *Lecture Notes in Computer Science*, pages 236–248. Springer, 2011. doi:10.1007/978-3-642-34191-5_23.

13 Viola Mészáros. An upper bound on the size of separated matchings. *Electronic Notes in Discrete Mathematics*, 38:633–638, 2011. doi:10.1016/j.endm.2011.10.006.

14 Clemens Müllner and Andrew Ryzhikov. Palindromic subsequences in finite words. In *Proc 13th Int. Conf. Language and Automata Theory and Applications (LATA)*, pages 460–468, 2019. doi:10.1007/978-3-030-13435-8_34.

k-Median Clustering Under Discrete Fréchet and Hausdorff Distances

Abhinandan Nath[1]
Mentor Graphics, Fremont, CA, USA
abnath@mentor.com

Erin Taylor
Duke University, Durham, NC, USA
ect15@cs.duke.edu

Abstract

We give the first near-linear time $(1+\varepsilon)$-approximation algorithm for k-median clustering of polygonal trajectories under the discrete Fréchet distance, and the first polynomial time $(1+\varepsilon)$-approximation algorithm for k-median clustering of finite point sets under the Hausdorff distance, provided the cluster centers, ambient dimension, and k are bounded by a constant. The main technique is a general framework for solving clustering problems where the cluster centers are restricted to come from a *simpler* metric space. We precisely characterize conditions on the simpler metric space of the cluster centers that allow faster $(1+\varepsilon)$-approximations for the k-median problem. We also show that the k-median problem under Hausdorff distance is NP-Hard.

2012 ACM Subject Classification Theory of computation → Computational geometry; Theory of computation → Design and analysis of algorithms

Keywords and phrases Clustering, k-median, trajectories, point sets, discrete Fréchet distance, Hausdorff distance

Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.58

Related Version A full version of the paper is available at `https://arxiv.org/abs/2004.00722`.

Funding *Erin Taylor*: Work on this paper was supported by NSF under grants CCF-15-13816, CCF-15-46392, IIS-14-08846, and by an ARO grant W911NF-15-1-0408.

Acknowledgements The authors would like to thank Pankaj K. Agarwal, Kamesh Munagala, and anonymous reviewers for helpful discussions and feedback.

1 Introduction

We study the k-median problem for an arbitrary metric space $\mathcal{X} = (X, \mathtt{d})$, where the cluster centers are restricted to come from a (possibly infinite) subset $C \subseteq X$. We call it the (k, C)-median problem. We prove general conditions on the structure of C that allow us to get efficient $(1+\varepsilon)$-approximation algorithms for the (k, C)-median problem for any $\varepsilon > 0$. As applications of our framework, we give $(1+\varepsilon)$-approximation algorithms for the metric space defined over polygonal trajectories and finite point sets in $\mathbb{R}^d$ under the discrete Fréchet and Hausdorff distance respectively, where the cluster centers have bounded complexity. For trajectories, our algorithm runs in near-linear time in the number of input points (Theorem 13) and is exponentially faster than the previous best algorithm ([10], Theorem 11). For point sets, ours is the first $(1+\varepsilon)$-approximation algorithm that runs in time polynomial in the number of input points (Theorem 15) for bounded dimensions and cluster complexity. Our results are summarized in Table 1. We also show that the k-median problem under Hausdorff distance problem is NP-Hard.

[1] Part of the work was done when the author was a graduate student at Duke University.

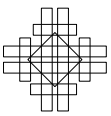


Table 1 Our results for $(1+\varepsilon)$-approximate k-median for n trajectories/point sets in $\mathbb{R}^d$, each having at most m points. Each cluster center can have at most l points. While stating running times, we assume k, l, d are constants independent of n, m, and $\tilde{O}$ hides logarithmic factors in n, m.

Metric space	Our result	Previous best
Polygonal traj., discrete Fréchet	$\tilde{O}(nm)$	$\tilde{O}\left(n^{dkl+1}m\right)$ [10]
Point sets, Hausdorff	$nm^{O(dl)}$	–

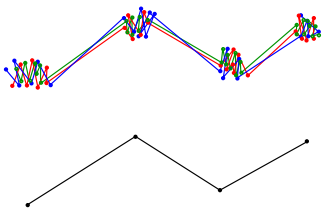


Figure 1 The red, green and blue trajectories are similar to each other and form a single cluster. If the cluster center trajectory is restricted to have four vertices, it will look like the black trajectory at the bottom. However if the center is unrestricted, it will contain a lot of vertices inside the four noticeable noisy *clumps* of vertices of the input trajectories, thereby overfitting to the input.

The k-median problem has been very widely studied. We are given a set P of n elements from a metric space. The goal is now to select k *centers* so that the sum of the distances of each point to the nearest cluster center is minimized. In the simplest setting, $P \subseteq \mathbb{R}^d$ under the Euclidean metric. In this paper, each individual element of P is itself a collection of points in $\mathbb{R}^d$, e.g., a curve traced by a moving object, or a point cloud. Since the objective of clustering is to group *similar* objects into the same cluster and to summarize each cluster using its cluster center, it is important that a meaningful distance function is used to compare two input elements. In our work, we look at the widely used discrete Fréchet and Hausdorff distances for trajectories and point sets respectively.

Trajectories can model a variety of systems that change with time. As such, trajectory data is being collected at enormous scales. As a first step, clustering is hugely important in understanding and summarizing the data. It involves partitioning a set of trajectories into clusters of similar trajectories, and computing a representative trajectory (a center) per cluster. It can be viewed as a compression scheme for large trajectory datasets, effectively performing non-linear dimension reduction. If the centers have low complexity, this representation can reduce uncertainty and noise found in individual trajectories. Information provided by the set of centers is useful for trajectory analysis applications such as similarity search and anomaly detection [32]. The Hausdorff distance is another widely used shape-based distance [8, 23]. In shape matching applications, we may want to cluster similar shapes into one group (where a shape is represented by a point cloud).

The k-median problem is hard to solve exactly, even in Euclidean space [16, 29]. There is a long line of work on both constant factor and $(1+\varepsilon)$-approximations, with varying running time dependence on k and the ambient dimension (if applicable). Many of these algorithms require the underlying metric space to have bounded doubling dimension (e.g., see [1]). However, it can be shown that both the discrete Fréchet and Hausdorff distances do not have doubling dimension bounded by a constant [30, Appendix A]. We circumvent this problem by considering cluster centers from a somewhat *simpler* metric space compared to the input metric space. For trajectories and point sets, we restrict the centers to have low complexity,

i.e., a bounded number of points. This approach has been used before ([9, 10, 13]). It has the added benefit of preventing the cluster center from overfitting to the elements of its cluster. This is crucial, since real-life measurements are noisy and error-prone, and without any restrictions the cluster center can inherit noise and high complexity from the input (see Fig. 1). As another example, in clustering financial time-series data using Hausdorff distance [6], frequent intra-day fluctuations may not be useful in capturing long-term trends, and we want to avoid them by retricting the cluster centers' complexity. However, our work differs from previous approaches in that we precisely characterize general conditions on the simpler metric space for the cluster centers, which leads to faster $(1+\varepsilon)$-approximation algorithms for the k-median problem for the discrete Fréchet and Hausdorff distances.

Problem definition. A **metric space** $\mathcal{X} = (X, \mathtt{d})$ consists of a set X and a distance function $\mathtt{d} : X \times X \to \mathbb{R}_{\geq 0}$ that satisfies the following properties : (i) $\mathtt{d}(x, x) = 0$ for all $x \in X$; (ii) $\mathtt{d}(x, y) = \mathtt{d}(y, x)$ for all $x, y \in X$; and (iii) $\mathtt{d}(x, z) \leq \mathtt{d}(x, y) + \mathtt{d}(y, z)$ for all $x, y, z \in X$.

Given subsets $P, C \subseteq X$, the **(k, C)-median** problem is to compute a set $C' \subseteq C$ of k *center* points that minimizes

$$\sum_{p \in P} \mathtt{d}(p, C'),$$

where $\mathtt{d}(p, C') = \min_{c \in C'} \mathtt{d}(p, c)$. Here P is finite, but C need not be.

Let T^l (resp. U^l) be the set of all trajectories (resp. point sets) in $\mathbb{R}^d$, where each trajectory (resp. point set) has at most l points. Thus, $T = \bigcup_{l>0} T^l$ and $U = \bigcup_{l>0} U^l$ are the set of all trajectories and finite point sets in $\mathbb{R}^d$ respectively. As special cases of the (k, C)-median problem, we discuss the (k, T^l)-median and the (k, U^l)-median problems for the metric spaces $\mathcal{T} = (T, \mathtt{d}_F)$ and $\mathcal{U} = (U, \mathtt{d}_H)$ respectively, i.e., each center trajectory or point set can have at most l points. Here, $\mathtt{d}_F$ and $\mathtt{d}_H$ denote the discrete Fréchet and Hausdorff distances respectively.

Challenges and ideas. As mentioned before, both the discrete Fréchet and Hausdorff metrics do not have low doubling dimension, so a number of previous techniques do not directly apply to our setting to yield efficient algorithms. One approach would be to embed these metrics into other metric spaces. Backurs and Sidiropoulos [4] give an embedding of the Hausdorff metric over point sets of size s in d-dimensional Euclidean space, into $l_\infty^{s^{O(s+d)}}$ with distortion $s^{O(s+d)}$. However, both the distortion and the resultant dimension are too high for many applications. It is not known if the Fréchet distance can be embedded into an l_p space using finite dimension.

We circumvent the problem by restricting the cluster centers to come from a subset C of the original space X, namely trajectories and point sets defined by a bounded number of points each. We show that if every metric ball in C can be *covered* by a small number of metric balls of a fixed smaller radius, then we can use a sampling-based algorithm similar to the one in Ackermann *et al.* [1]; we make this precise by introducing the notion of *coverability* of C. We crucially show that the centers of the balls in the *cover* can be arbitrary, and need not come from C. This is more general than C having bounded doubling dimension. This allows us to approximate the optimal $(1, C)$-median of P using the $(1, C)$-median of a constant sized random sample of P, allowing us to use the framework of [1].

It is not known how to efficiently compute the optimal $(1, C)$-median under the discrete Fréchet and Hausdorff distances. However, we show that all we need is to compute a constant number of candidate centers in time independent of the size of the input, at least one of which

is a good approximation to the optimal $(1, C)$-median of the input. We show how to compute these candidates for a *coverable* set C. Then, we can apply the sampling technique of [1] and recursively use this property to find k centers that approximate the cost of the (k, C)-median optimal solution. Although our work heavily relies on the framework of Ackermann et al. [1], it is a significant improvement from existing work on clustering under the discrete Fréchet distance, and the first such result for clustering under the Hausdorff distance.

Previous work. Trajectory clustering has a lot of applications, e.g., finding frequent movement patterns in trajectory data. As such, there has been work on trajectory clustering [17, 22, 33], and possibly computing a representative trajectory for each cluster. Many proposed algorithms and models have no provable performance gaurantees and are experimental in nature.

Driemel *et al.* [13] started the rigorous study of clustering trajectories under the continuous Fréchet distance under the classic k-clustering objectives. However, they only deal with 1D-trajectories. They introduce the (k, l)-clustering problem, i.e., clustering trajectories with k cluster centers such that each center can have at most l points. For $1D$-trajectories, they give $(1 + \varepsilon)$-approximation algorithms for both the (k, l)-center and (k, l)-median problem that run in near-linear time for constant ε, k, l. Buchin *et al.* [9] study the (k, l)-center clustering problem for trajectories in $\mathbb{R}^d$ under the discrete and continuous Fréchet distances, and give both upper and lower bounds. The most closely related work to ours is the one by Buchin, Driemel and Struijs [10], where they give algorithms for the (k, l)-median problem under discrete Fréchet; however their running times are much slower (see Table 1). They also show that the 1-median problem under discrete Fréchet distance is NP-Hard, and W[1]-Hard in the number of input trajectories.

On the other hand, clustering under Hausdorff distance has received much less attention. There is work on hierarchical clustering of financial time series data using Hausdorff distance [6]. Chen *et al.* [11] use the DBSCAN algorithm [15] while Qu *et al.* [31] use spectral clustering for trajectories and using the Hausdorff distance. We are not aware of any theoretical analysis for k-median clustering of point sets under the Hausdorff distance.

In general metric spaces, a polynomial time $(1 + \sqrt{3} + \varepsilon)$-approximation algorithm to the k-median problem exists [28], whereas no polynomial time algorithm can achieve an approximation ratio less than $(1+2/e)$ unless NP $\subseteq$ DTIME $[n^{O(\log \log n)}]$ [24]. For Euclidean k-median in d dimensions, Guruswami and Indyk [18] showed that there is no PTAS if both k and d are part of the input. Arora *et al.* [3] gave the first PTAS when d is fixed, whose running time was subsequently improved in [25]. Kumar *et al.* [27] gave a $(1+\varepsilon)$-approximate algorithm with runtime $2^{(k/\varepsilon)^{O(1)}} dn$, this was extended by Ackermann *et al.* [1] to those metric spaces for which the optimal 1-median can be approximated using a constant-sized random sample; this holds true for doubling metric spaces. There are coreset-based approaches with running times linear in n and either exponential in d and polynomial in k [20, 19], or vice versa [5]. Recently, Cohen-Addad *et al.* [12] gave a PTAS for k-median in low-dimensional Euclidean and minor-free metrics using local search.

2 Preliminaries, definitions and an overview

We formally define the discrete Fréchet and Hausdorff distances. We also define two properties on C and `d` which allow us to design efficient clustering algorithms. Finally we give an overview of our algorithm.

Discrete Fréchet and Hausdorff distance. Consider two finite sets ζ_1 and ζ_2. A **correspondence** $\mathcal{C}$ between ζ_1 and ζ_2 is a subset of $\zeta_1 \times \zeta_2$ such that every element of ζ_1 and ζ_2 appears in at least one pair in $\mathcal{C}$. For $\zeta_1, \zeta_2 \subseteq \mathbb{R}^d$, the **Hausdorff distance** [21] is defined as

$$\mathtt{d}_H(\zeta_1, \zeta_2) = \min_{\mathcal{C} \in \Xi(\zeta_1, \zeta_2)} \max_{(p,q) \in \mathcal{C}} \|p - q\|,$$

where $\|.\|$ is the l_2 norm, and $\Xi(\zeta_1, \zeta_2)$ is the set of all correspondences between ζ_1 and ζ_2.

A trajectory γ is a finite sequence of points $\langle p_1, p_2 \ldots \rangle$ in $\mathbb{R}^d$. A correspodence can be defined for a pair of trajectories $\gamma_1 = \langle p_1, p_2, \ldots \rangle$ and $\gamma_2 = \langle q_1, q_2, \ldots \rangle$ by treating each trajectory as a point sequence; such a correspondence is said to be **monotonic** if it also respects the ordering of points in the trajectories, i.e., if $(p_{i_1}, q_{j_1}), (p_{i_2}, q_{j_2}) \in \mathcal{C}$, then $i_2 \geq i_1 \Rightarrow j_2 \geq j_1$. The **discrete Fréchet distance** [14] between γ_1 and γ_2 is defined as

$$\mathtt{d}_F(\gamma_1, \gamma_2) = \min_{\mathcal{C} \in \Xi_M(\gamma_1, \gamma_2)} \max_{(p,q) \in \mathcal{C}} \|p - q\|,$$

where $\Xi_M(\gamma_1, \gamma_2)$ is the set of all monotone correspondences between γ_1 and γ_2. Our algorithms cluster in the metric space defined by the discrete Fréchet and Hausdorff distance.

Strong and weak sampling properties. We define two properties that make efficient clustering algorithms possible. These are generalizations of the *strong* and *weak* sampling properties defined by Ackermann *et al.*(see Theorem 1.1 and Property 4.1 in [1]). The major difference is that the cluster centers are restricted to a subset C of the metric space X. These properties allow fast approximation of the $(1, C)$-median using only a constant sized random sample of the input. We later show how to get an efficient (k, C)-median algorithm using the fast $(1, C)$-median algorithm as a subroutine. We denote the optimal $(1, C)$-median of any set $P \subseteq X$ by c_P.

▶ **Definition 1** (Strong sampling property). *Let $0 < \varepsilon, \delta < 1$ be arbitrary. $(X, C, \mathtt{d})$ is said to satisfy the strong sampling property for ε, δ iff*

- **(i)** *For any finite $P \subseteq X$, c_P can be computed in time depending only on $|P|$.*
- **(ii)** *There exists a positive integer $m_{\delta,\varepsilon}$ depending on δ, ε such that for any $P \subseteq X$, the optimal $(1, C)$-median c_S of a uniform random multiset $S \subseteq P$ of size $m_{\delta,\varepsilon}$ satisfies*

$$\Pr\left[\sum_{p \in P} \mathtt{d}(p, c_S) \leq (1 + \varepsilon) \sum_{p \in P} \mathtt{d}(p, c_P)\right] \geq 1 - \delta.$$

The strong sampling property characterizes those instances in which the optimal $(1, C)$-median of a constant-sized random sample is a good approximation to the optimal $(1, C)$-median of the whole set. However, in many cases it is impossible to efficiently solve the $(1, C)$-median exactly (e.g., when $C, X = \mathbb{R}^d$ and $\mathtt{d}$ is the Euclidean metric). This is also true for the discrete Fréchet and Hausdorff distances for polygonal trajectories and finite point sets respectively. The following definition becomes helpful then.

▶ **Definition 2** (Weak sampling property). *Let $0 < \varepsilon, \delta < 1$ be arbitrary. $(X, C, \mathtt{d})$ is said to satisfy the weak sampling property for ε, δ iff there exist positive integers $m_{\delta,\varepsilon}$ and $t_{\delta,\varepsilon}$ depending on δ, ε such that for any $P \subseteq X$ and a uniform random multiset $S \subseteq X$ of size $m_{\delta,\varepsilon}$, there exists a set $\Gamma(S) \subseteq C$ of size $t_{\delta,\varepsilon}$ that satisfies*

$$\Pr\left[\exists c \in \Gamma(S) \mid \sum_{p \in P} \mathtt{d}(p, c) \leq (1 + \varepsilon) \sum_{p \in P} \mathtt{d}(p, c_P)\right] \geq 1 - \delta.$$

Furthermore, $\Gamma(S)$ can be computed in time depending on $\delta, \varepsilon, |S|$ but independent of $|P|$.

The weak sampling property characterizes those instances in which one can generate a constant number of candidate centers in time independent of the size of the input set, and at least one of which is guaranteed to be a good approximation to the optimal 1-center of the input set. We later show that the discrete Fréchet and Hausdorff distances satisfy the weak sampling property. We use $m_{\delta,\varepsilon}$ to denote the size of the random sample for both the strong and weak sampling properties.

Algorithm overview. We give an overview of our algorithm, denoted Cluster, in Algorithm 1. It is similar to the algorithm Cluster from [1], which approximates the k-median problem by recursively taking a random sample of constant size and solving the 1-median problem on the sample. The small but crucial difference in our setting is that the set of candidate cluster centers $\overline{C}_S$ comes from C; this also changes how the candidates are generated. We show that with a careful choice of C, our instance satisfies one of the sampling properties (Definitions 1, 2), and we can apply the framework of Ackermann et al. [1].

The algorithm takes as input the set of points $\overline{P} \subseteq P$ that are yet to be assigned cluster centers, the number of cluster centers $\overline{k}$ still to be computed, and the centers $\overline{C} \subseteq C$ already computed. It returns the final set of cluster centers. To solve the (k, C)-median problem for P, we call Cluster$(P, k, \{\})$ with values of $\alpha, m_{\delta,\varepsilon}, F$ that we will specify later.

Briefly, the algorithm has two phases. In the pruning phase no new centers are added. Rather, the set N containing half of the points of $\overline{P}$ closest to $\overline{C}$ are removed from $\overline{P}$, and the algorithm is called recursively on $\overline{P} \setminus N$. In the sampling phase, new centers are added. The algorithm first samples a uniformly random multiset S of $\overline{P}$ of size $\frac{2}{\alpha} m_{\delta,\varepsilon}$ for some constant α and $m_{\delta,\varepsilon}$ to be defined later. Then for each subset $S' \subset S$ of size $m_{\delta,\varepsilon}$, a set of candidate centers $F(S')$ is generated; the function F varies depending on certain conditions satisfied by $(X, C, \mathtt{d})$; in particular $F(S') = \{c_{S'}\}$ for the strong sampling property, and $F(S') = \{\Gamma(S')\}$ for the weak sampling property. Each candidate center is in turn added to $\overline{C}$, and the algorithm is run recursively. Finally, the solution with the lowest cost is returned.

Algorithm 1 Algorithm Cluster computes a (k, C)-median clustering.

Cluster$(\overline{P}, \overline{k}, \overline{C})$:

input: Point set $\overline{P}$, remaining number of centers $\overline{k}$, computed centers $\overline{C}$
if $\overline{k} = 0$: return $\overline{C}$
else:
  if $\overline{k} \geq |\overline{P}|$: return $\overline{C} \cup \overline{P}$
  else:
    /* *Pruning phase* */
    $N \leftarrow$ set of $\frac{1}{2}|\overline{P}|$ minimal points $p \in \overline{P}$ w.r.t $\mathtt{d}(p, \overline{C})$
    $C^* \leftarrow$ Cluster$(\overline{P} \setminus N, \overline{k}, \overline{C})$
    /* *Sampling phase* */
    $S \leftarrow$ uniform random multisubset of $\overline{P}$ of size $\frac{2}{\alpha} m_{\delta,\varepsilon}$
    $\overline{C}_S \leftarrow \bigcup_{S' \subset S, |S'| = m_{\delta,\varepsilon}} F(S')$
    for all $\overline{c} \in \overline{C}_S$:
      $C^{\overline{c}} \leftarrow$ Cluster$(\overline{P}, \overline{k} - 1, \overline{C} \cup \{\overline{c}\})$
    return $C^{\overline{c}}$ or C^* with the lowest cost

The rest of the paper is organized as follows. In Section 3, we show that for $(X, C, \mathtt{d})$ satisfying the strong and weak sampling properties, the algorithm Cluster (using the appropriate $\alpha, m_{\delta,\varepsilon}, F$) computes a $(1 + \varepsilon)$-approximation to the optimal (k, C)-median.

In Section 4, we prove sufficient conditions on C for the sampling properties to hold. In Section 5, we give clustering algorithms for the discrete Fréchet and Hausdorff distances using the framework developed in previous sections. In Section 6, we show that the k-median problem for Hausdorff distance is NP-Hard.

3 Clustering via sampling

We show that the Cluster algorithm (Algorithm 1) computes an approximate (k, C)-median for instances satisfying the sampling properties and for appropriate $\alpha, m_{\delta,\varepsilon}$ and F, i.e., if we use $F(S') = \{c_{S'}\}$ for the strong sampling property and $F(S') = \Gamma(S')$ for the weak sampling property. The analysis closely follows that of Ackermann *et al.* [1], and detailed proofs are provided in the full version [30].

The *superset sampling lemma* [30, Lemma 17] shows how to draw a uniform random multiset from $P' \subseteq P$ while only knowing P (without explicitly knowing P'), provided P' contains a constant fraction of the points of P. Using this lemma and the strong and weak sampling properties, we have the following. See [30, Appendices B.1,B.2] for proofs of the *superset sampling lemma* and the following lemma.

▶ **Lemma 3.** *Let $\alpha < \frac{1}{4k}$ be an arbitrary positive constant. Suppose $(X, C, \mathtt{d})$ satisfies the strong or weak sampling property (Definitions 1, 2) for some $\varepsilon, \delta \in (0, 1)$. Given $P \subseteq X$, algorithm Cluster run with input $(P, k, \{\})$ and appropriate F computes a set $\tilde{C} \subseteq C$ of size k such that*

$$\Pr\left[\sum_{p\in P} \mathtt{d}(p, \tilde{C}) \leq (1 + 8\alpha k^2)(1 + \varepsilon) \sum_{p\in P} \mathtt{d}(p, C^*)\right] \geq \left(\frac{1-\delta}{5}\right)^k,$$

where C^* is an optimal solution to the (k, C)-median problem for P.

Running time. We characterize the running time of Cluster in terms of $\mathtt{d}$, C and F. Let $h(C)$ denote the maximum time required to compute the nearest neighbor of x in C, i.e., $\arg\min_{y\in C} \mathtt{d}(x, y)$, for any $x \in X$. Let $t(C)$ denote the maximum time required to compute $\mathtt{d}(x, y)$ for any $x \in X, y \in C$. Finally, let $w(m) = \max_{S\subseteq X, |S|=m} |F(S)|$, and let $f(m)$ be the maximum number of operations needed to compute $F(S)$ for any $S \subseteq X$ of size m, where computing $\mathtt{d}$ between points in X and C, and computing the closest point in C to any point in X count as one operation each. The proof is similar to the running time analysis from [1], and is given in [30, Appendix B.3].

▶ **Lemma 4.** *Suppose $(X, C, \mathtt{d})$ satisfies the strong or weak sampling properties (Definitions 1 and 2) for some $\epsilon, \delta \in (0, 1)$. Given $P \subseteq X$ containing n points, algorithm Cluster runs in time*

$$n \cdot 2^{O\left(km_{\delta,\varepsilon} \log\left(\frac{1}{\alpha} m_{\delta,\varepsilon}\right)\right)} \cdot \left(w(m_{\delta,\varepsilon}) \cdot f(m_{\delta,\varepsilon})\right)^{O(k)} \cdot (h(C) + t(C)).$$

By setting $\alpha = \frac{\varepsilon}{8k^2}$, the approximation factor in Lemma 3 becomes $(1 + 3\varepsilon)$. Moreover, the error probability can be made arbitrarily small by running the Cluster algorithm $2^{\Theta(k)}$ times and taking the minimum cost solution, without changing the asymptotic running time. We thus get the following. Note that $w(m_{\delta,\varepsilon})$ takes on values 1 and $t_{\delta,\varepsilon}$ for the strong and weak sampling properties respectively.

▶ **Theorem 5.** *Suppose $(X, C, \mathtt{d})$ satisfies the strong sampling property (Definition 1) for some $\varepsilon, \delta \in (0,1)$. Further, suppose c_S can be computed in $a(m_{\delta,\varepsilon})$ operations, where computing $\mathtt{d}$ between points in X and C, and computing the closest point in C to any point in X count as one operation each. Given $P \subseteq X$ having n points and $k \in \mathbb{N}$, with probability $\geq 1-\delta$, a $(1+3\varepsilon)$-approximate solution to the (k, C)-median problem for P can be computed in time*

$$n \cdot 2^{O\left(k m_{\delta,\varepsilon} \log\left(\frac{k}{\varepsilon} m_{\delta,\varepsilon}\right)\right)} \cdot a(m_{\delta,\varepsilon})^{O(k)} \cdot (h(C) + t(C)).$$

▶ **Theorem 6.** *Suppose $(X, C, \mathtt{d})$ satisfies the weak sampling property (Definition 2) for some $\varepsilon, \delta \in (0,1)$. Further, suppose $\Gamma(S)$ can be computed in $b(m_{\delta,\varepsilon})$ operations, where computing $\mathtt{d}$ between points in X and C, and computing the closest point in C to any point in X count as one operation each. Given $P \subseteq X$ having n points and $k \in \mathbb{N}$, with probability $\geq 1-\delta$, a $(1+3\varepsilon)$-approximate solution to the (k, C)-median problem for P can be computed in time*

$$n \cdot 2^{O\left(k m_{\delta,\varepsilon} \log\left(\frac{k}{\varepsilon} m_{\delta,\varepsilon}\right)\right)} \cdot \left(t_{\delta,\varepsilon} \cdot b(m_{\delta,\varepsilon})\right)^{O(k)} \cdot (h(C) + t(C)).$$

4 Covering metric spaces

We specify sufficient conditions on C for the sampling properties to hold. These conditions characterize how well can certain subsets of C be *covered* using a small number of sets.

Let $\mathcal{X} = (X, \mathtt{d})$ be a metric space. Given $x \in X$, let $\mathcal{B}_{\mathtt{d}}(x, r) = \{x' \in X \mid \mathtt{d}(x, x') \leq r\}$ denote the ball of radius r (under $\mathtt{d}$) centered at x; we will drop the subscript $\mathtt{d}$ if it is clear from the context. An **r-cover** of a subset $X' \subseteq X$ for some $r > 0$ is a set $Y \subseteq X$ such that $X' \subseteq \bigcup_{y \in Y} \mathcal{B}(y, r)$. Note that the elements of Y need not be in X'. Also note that if Y is an r-cover for X', it is also an r-cover for any subset of X'.

A subset $Y \subseteq X$ is said to be **g-coverable** for some non-decreasing function $g : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ iff for all $y \in Y$ and $r > r' > 0$, there exists an r'-cover of $\mathcal{B}(y, r) \cap Y$ of size at most $g(r/r')$. Note that if Y is g-coverable then any subset $Y' \subseteq Y$ is also g-coverable.

Intuitively, if C has a small cover, then for any metric ball in C there exists a small set of points (not necessarily from C), termed the cover, such that the distance from any point in the ball to a point in the cover is smaller than the radius of the ball.

Sufficient conditions for the strong sampling property. The following theorem gives sufficient conditions for the strong sampling property to hold in terms of coverability of C. The proof is similar to Lemma 3.4 of [1] but has been adapted to our setting, see [30, Appendix C] for more details.

▶ **Theorem 7.** *If C is g-coverable, and for any $P \subseteq X$, c_P can be computed in time depending only on $|P|$, then $(X, C, \mathtt{d})$ satisfies the strong sampling property (Definition 1) for any $\varepsilon, \delta \in (0,1)$. Here, the constant $m_{\delta,\varepsilon} = m_{\delta,\varepsilon,g}$ also depends on g.*

From Theorems 5 and 7, we get the following.

▶ **Corollary 8.** *Suppose C is g-coverable and the optimal $(1, C)$-median of any subset of X can be computed in time depending on the size of the subset. Let $\varepsilon, \delta \in (0,1)$. Given $P \subseteq X$ having n points and $k \in \mathbb{N}$, with probability $\geq 1-\delta$, a $(1+3\varepsilon)$-approximate solution to the (k, C)-median problem for P can be computed in time*

$$n \cdot 2^{O\left(k m_{\delta,\varepsilon,g} \log\left(\frac{k}{\varepsilon} m_{\delta,\varepsilon}\right)\right)} \cdot a(m_{\delta,\varepsilon,g})^{O(k)} \cdot (h(C) + t(C)),$$

where $m_{\delta,\varepsilon,g}$ is a constant depending only on ε, δ, g, and $a(m)$ is the number of operations needed to compute the optimal $(1, C)$-median of m points in X, where computing $\mathtt{d}$ between points in X and C, and computing the closest point in C to any point in X count as one operation each.

Sufficient conditions for the weak sampling property. We give sufficient conditions for the weak sampling property to hold in terms of coverability of C.

For $Y \subseteq X$, let $\theta_Y\left(\frac{r}{r'}\right)$ be the number of operations required to compute an r'-cover of $\mathcal{B}(y, r) \cap Y$ for any $y \in Y$ (if such a cover exists) where computing $\mathtt{d}$ between points in X and C, and computing the closest point in C to any point in X count as one operation each (we assume that the number of operations can be expressed in terms of $\frac{r}{r'}$).

The following lemma will be helpful, and states that if C has a small cover and if we have a good estimate of the *cost* of the optimal $(1, C)$-median, then we can construct a small set of points in C such that at least one of them is a good approximation to the optimal $(1, C)$-median. Further, this can be done in time independent of $|P|$. Both of these properties are necessary for the weak sampling property (Definition 2).

▶ **Lemma 9.** *Let $P \subseteq X$. Suppose C is g-coverable. Then given a, b such that $a \leq \frac{1}{|P|}\sum_{p\in P} \mathtt{d}(c_P, p) \leq b$, we can compute a set $Q \subseteq C$ of size $O(g\left(\frac{4b}{\varepsilon\delta a}\right))$ such that*

$$Pr\left[\exists q \in Q \mid \sum_{p\in P} \mathtt{d}(p, q) \leq (1+\varepsilon)\sum_{p\in P} \mathtt{d}(p, c_P)\right] \geq 1 - \delta.$$

Further, Q can be computed in $O\left(\theta_C\left(\frac{4b}{\varepsilon\delta a}\right) + g\left(\frac{4b}{\varepsilon\delta a}\right)\right)$ operations, where computing $\mathtt{d}$ between points in X and C, and computing the closest point in C to any point in X count as one operation each.

Proof. For any $x \in X$, we define $x' = \arg\min_{y\in C} \mathtt{d}(x, y)$. Consider a point $q \in P$ chosen uniformly at random. By Markov's inequality, $\mathtt{d}(q, c_P) \leq \frac{1}{\delta|P|}\sum_{p\in P} \mathtt{d}(p, c_P)$ with probability $\geq 1 - \delta$. In such a case,

$$\mathtt{d}(q', c_P) \leq \mathtt{d}(q', q) + \mathtt{d}(q, c_p) \leq 2\mathtt{d}(q, c_P) \leq \frac{2}{\delta|P|}\sum_{p\in P} \mathtt{d}(p, c_P).$$

Thus, $c_P \in \mathcal{B}(q', \frac{2b}{\delta})$ with probability at least $1 - \delta$.

Let C' be an $\left(\frac{\varepsilon a}{2}\right)$-cover of $\mathcal{B}(q', \frac{2b}{\delta}) \cap C$. Since C is g-coverable, $|C'| = g\left(\frac{4b}{\varepsilon\delta a}\right)$. We will argue that $Q = \{q'\} \cup \{c' \mid c \in C'\}$ is the required solution. Let $x = \arg\min_{y\in C'} \mathtt{d}(y, c_P)$. Since C' is an $\left(\frac{\varepsilon a}{2}\right)$-cover, $\mathtt{d}(x, c_P) \leq \frac{\varepsilon a}{2}$. Also, $\mathtt{d}(x, x') \leq \mathtt{d}(x, c_P) \leq \frac{\varepsilon a}{2}$. Thus, $\mathtt{d}(x', c_P) \leq \mathtt{d}(x, x') + \mathtt{d}(x, c_P) \leq \varepsilon a$. We then have

$$\begin{aligned}\sum_{p\in P} \mathtt{d}(p, x') &\leq \sum_{p\in P} (\mathtt{d}(p, c_P) + \mathtt{d}(x', c_P)) \leq \left(\sum_{p\in P} \mathtt{d}(p, c_P)\right) + \varepsilon a|P| \\ &\leq (1+\varepsilon)\sum_{p\in P} \mathtt{d}(p, c_P).\end{aligned}$$

Computing C' takes $\theta_C\left(\frac{4b}{\varepsilon\delta a}\right)$ operations. Computing the output set takes $|C'| + 1 = O\left(g\left(\frac{4b}{\varepsilon\delta a}\right)\right)$ operations. ◀

The next theorem shows that with a small random sample of P, one of two things can happen. Either one of the samples is close to an approximate $(1, C)$-median, or we can approximate the cost of the optimal $(1, C)$-median in time independent of $|P|$. This along

with Lemma 9 shows that the weak sampling property holds if C is g-coverable. The proof is inspired by the proof of Theorem 1 in [26], and it also shows how to compute Γ for the weak sampling property (Definition 2).

▶ **Theorem 10.** *If C is g-coverable, then $(X, C, \mathtt{d})$ satisfies the weak sampling property (Definition 2) for $0 < \varepsilon < \frac{4}{9}$ and $\left(1 - \frac{5}{18}\varepsilon\right) < \delta < 1$. Further, the constants $m_{\delta,\varepsilon} = 1 + \frac{4}{\varepsilon}$ and $t_{\delta,\varepsilon} = O\left(g\left(\frac{2048}{\delta_1 \varepsilon^5}\right)\right)$, and the number of operations needed to compute $\Gamma(S)$ is $O\left(\frac{1}{\varepsilon} + \theta_C\left(\frac{2048}{\delta_1\varepsilon^5}\right) + g\left(\frac{2048}{\delta_1\varepsilon^5}\right)\right)$, where $\delta_1 = \frac{\varepsilon}{2} - \frac{9}{5}(1-\delta)$; and computing $\mathtt{d}$ between points in X and C, and computing the closest point in C to any point in X count as one operation each.*

Proof. Let $\varepsilon_1 = \frac{\varepsilon}{4}$ and $\bar{r} = \frac{1}{|P|}\sum_{p\in P} \mathtt{d}(p, c_P)$. Also, let $x' = \arg\min_{y\in C} \mathtt{d}(x, y)$ for any $x \in X$.

Let $Q \subseteq P$ be a uniform random multiset of size $\frac{1}{\varepsilon_1}$, and $q \in P$ be another point chosen uniformly at random. We will show that $Q \cup \{q\}$ plays the role of S in Definition 2.

Using Markov's inequality and union bound, we have

$$\Pr\left[\mathtt{d}(q, c_P) > \frac{\bar{r}}{2\varepsilon_1^2}\right] < 2\varepsilon_1^2 \text{ and } \Pr\left[\exists p \in Q \mid \mathtt{d}(p, c_P) > \frac{\bar{r}}{2\varepsilon_1^2}\right] < \left(\frac{1}{\varepsilon_1}\right) 2\varepsilon_1^2 = 2\varepsilon_1.$$

Thus with probability $\geq 1 - 2\varepsilon_1 - 2\varepsilon_1^2$, q and Q are in $\mathcal{B}\left(c_P, \frac{\bar{r}}{2\varepsilon_1^2}\right)$. We assume that this event happens. Now, by definition of q', we have $\mathtt{d}(q, q') \leq \mathtt{d}(q, c_P)$. Hence,

$$\mathtt{d}(q', c_P) \leq \mathtt{d}(q, q') + \mathtt{d}(q, c_P) \leq 2\mathtt{d}(q, c_P) \leq \frac{\bar{r}}{\varepsilon_1^2}.$$

Let $\mathcal{B}_1 = \mathcal{B}\left(c_P, \frac{\bar{r}}{\varepsilon_1^2}\right)$, $\mathcal{B}_2 = \mathcal{B}\left(q', \varepsilon_1 \bar{r}\right)$ and $P' = P \cap \mathcal{B}_1$. Then, $q' \in \mathcal{B}_1$ and $Q \subseteq P'$. We consider two cases now.

Case 1: P' has at least $2\varepsilon_1|P'|$ points outside $\mathcal{B}_2$. For any $p \in Q$, the probability p is outside $\mathcal{B}_2$ is $2\varepsilon_1$. Thus, with probability at least $2\varepsilon_1$, there exists $p \in Q$ such that $\mathtt{d}(p, q') \geq \varepsilon_1 \bar{r}$ and hence $\sum_{p\in Q} \mathtt{d}(p, q') \geq \varepsilon_1 \bar{r}$. Also, $\mathtt{d}(p, q') \leq \frac{2\bar{r}}{\varepsilon_1^2}$ for any $p \in Q$. Hence, $\sum_{p\in Q} \mathtt{d}(p, q') \leq \frac{2\bar{r}}{\varepsilon_1^3}$.

Let $\delta_1 = 2\varepsilon_1 - \frac{9}{5}(1-\delta)$. We can now use Lemma 9 with $a = \frac{\varepsilon_1^3}{2}\sum_{p\in Q} \mathtt{d}(p, q')$ and $b = \frac{1}{\varepsilon_1}\sum_{p\in Q} \mathtt{d}(p, q')$ to compute a set $Q_1 \subseteq C$ of $O\left(g\left(\frac{4b}{\varepsilon\delta_1 a}\right)\right) = O\left(g\left(\frac{2048}{\delta_1\varepsilon^5}\right)\right)$ candidate centers, one of which is a $(1+\varepsilon)$-approximate center with probability at least $1 - \delta_1$. The total probability of getting a good set of candidate centers is $(2\varepsilon_1 - \delta_1)(1 - 2\varepsilon_1 - 2\varepsilon_1^2) > 1 - \delta$.

Case 2: P' has at most $2\varepsilon_1|P'|$ points putside $\mathcal{B}_2$. We further consider two cases.

Case 2(a): $\mathtt{d}(q', c_P) \leq 4\varepsilon_1\bar{r}$. Then

$$\sum_{p\in P} \mathtt{d}(p, q') \leq \sum_{p\in P} \left(\mathtt{d}(p, c_p) + \mathtt{d}(q', c_p)\right) \leq (1 + 4\varepsilon_1) \sum_{p\in P} \mathtt{d}(p, c_P) \leq (1+\varepsilon) \sum_{p\in P} \mathtt{d}(p, c_P).$$

Case 2(b): $\mathtt{d}(q', c_P) > 4\varepsilon_1\bar{r}$. Suppose we assign all points from c_P to q'. By an averaging argument, we have $|P'| \geq (1 - \varepsilon_1^2)|P|$. Then, the number of points of P that are outside $\mathcal{B}_2$ is at most

$$\begin{aligned} |P \setminus P'| + 2\varepsilon_1|P'| &= |P| - (1 - 2\varepsilon_1)|P'| \\ &\leq |P| - (1 - 2\varepsilon_1)(1 - \varepsilon_1^2)|P| \\ &\leq (\varepsilon_1^2 + 2\varepsilon_1(1 - \varepsilon_1^2))|P|. \end{aligned}$$

Thus, $|P \cap \mathcal{B}_2| \geq (1 - \varepsilon_1^2 - 2\varepsilon_1(1-\varepsilon_1^2))|P|$. Now, for $p \in P \cap \mathcal{B}_2$, the decrease in cost on switching from c_P to q' is at least

$$\mathtt{d}(p, c_P) - \mathtt{d}(p, q') \geq \mathtt{d}(p, c_P) - \varepsilon_1 \bar{r} \geq \mathtt{d}(q', c_P) - 2\varepsilon_1 \bar{r}.$$

For $p \in P \setminus \mathcal{B}_2$, the increase in cost on switching from c_P to q' is at most

$$\mathtt{d}(p, q') - \mathtt{d}(p, c_P) \leq \mathtt{d}(q', c_P).$$

The overall decrease in cost is

$$|P \cap \mathcal{B}_2|(\mathtt{d}(q', c_P) - 2\varepsilon_1 \bar{r}) - |P \setminus \mathcal{B}_2|\mathtt{d}(q', c_P) > 0$$

for our choice of ε_1 and $\mathtt{d}(q', c_P) > 4\varepsilon_1 \bar{r}$. But c_P is the optimal $(1, C)$-median of P, a contradiction. Hence case 2(b) cannot occur.

From the two cases above, we can see that $Q_1 \cup \{q'\}$ plays the role of $\Gamma(S)$ in Definition 2.

Sampling Q, q takes time $O(\frac{1}{\varepsilon})$. Computing $\sum_{p \in P} \mathtt{d}(p, q')$ involves $O(\frac{1}{\varepsilon})$ computations of $\mathtt{d}$ between a pair of points from X, at least one of which comes from C. Computing the set of candidates takes $O\left(\theta_C\left(\frac{2048}{\delta_1 \varepsilon^5}\right) + g\left(\frac{2048}{\delta_1 \varepsilon^5}\right)\right)$ operations. Thus, total number of operations needed is $O\left(\frac{1}{\varepsilon} + \theta_C\left(\frac{2048}{\delta_1 \varepsilon^5}\right) + g\left(\frac{2048}{\delta_1 \varepsilon^5}\right)\right)$. Moreover, $m_{\delta,\varepsilon} = |Q \cup \{q\}| = 1 + \frac{1}{\varepsilon_1} = 1 + \frac{4}{\varepsilon}$, and $t_{\delta,\varepsilon} = |Q_1 \cup \{q'\}| = O\left(g\left(\frac{2048}{\delta_1 \varepsilon^5}\right)\right)$. ◀

From Theorems 6 and 10, we get the following.

▶ **Corollary 11.** *Suppose C is g-coverable. Let $\varepsilon \in (0, \frac{4}{9}), \delta \in (1 - \frac{5}{18}\varepsilon, 1)$. Given $P \subseteq X$ of size n and $k \in \mathbb{N}$, with probability $\geq 1-\delta$, a $(1+3\varepsilon)$-approximate solution to the (k, C)-median problem for P can be computed in time*

$$n \cdot 2^{O\left(\frac{k}{\varepsilon}\log\left(\frac{k}{\varepsilon}\right)\right)} \cdot (b(m_{\delta,\varepsilon}) \cdot t_{\delta,\varepsilon})^{O(k)} \cdot (h(C) + t(C)),$$

where $b(m_{\delta,\varepsilon}) = O\left(\frac{1}{\varepsilon} + \theta_C\left(\frac{2048}{\delta_1 \varepsilon^5}\right) + g\left(\frac{2048}{\delta_1 \varepsilon^5}\right)\right)$, $t_{\delta,\varepsilon} = O\left(g\left(\frac{2048}{\delta_1 \varepsilon^5}\right)\right)$, and $\delta_1 = \frac{\varepsilon}{2} - \frac{9}{5}(1-\delta)$.

5 Clustering discrete Fréchet and Hausdorff distances

In this section, we show how the results from the previous section can be used to cluster trajectories and point sets under the discrete Fréchet and Hausdorff distances respectively.

Clustering under discrete Fréchet distance. Recall that T^l is the set of all trajectories in $\mathbb{R}^d$ having at most l points each; thus $T = \bigcup_{l>0} T^l$ is the set of all trajectories in $\mathbb{R}^d$. Given $\mathcal{T} = (T, \mathtt{d}_F)$ and trajectories $P \subseteq T$, the (k, l)-median problem [13, 10] is equivalent to the (k, T^l)-median problem in our setting, i.e., the center trajectories contain at most l points. We show that T^l is g-coverable for some g that depends on l.

▶ **Lemma 12.** *T^l is g-coverable under $\mathtt{d}_F$ for $g(x) = l^{2l} \cdot x^{O(dl)}$. Further, an r'-cover of $\mathcal{B}_{\mathtt{d}_F}(\gamma, r) \cap T^l$ for $r > r' > 0$ and $\gamma \in T^l$ can be computed in $l^{2l} \cdot \left(\frac{r}{r'}\right)^{O(dl)}$ time.*

Proof. Let $r > r' > 0$ be arbitrary. Let $\gamma = \langle p_1, \ldots, p_{l'} \rangle \in T^l$ for some $l' \leq l$. Since the Euclidean metric in $\mathbb{R}^d$ has doubling dimension $O(d)$, for any $p \in \mathbb{R}^d$ there exist $\left(\frac{r}{r'}\right)^{O(d)}$ points in the Euclidean ball $\mathcal{B}_E(p, r)$ centered at p such that any point in $\mathcal{B}_E(p, r)$ is at most r' distance away from one of these points; denote these points by $B_p(r, r')$. Consider the set of points $\bigcup_{i=1}^{l'} B_{p_i}(r, r')$; this set has cardinality $l' \cdot \left(\frac{r}{r'}\right)^{O(d)}$.

Next, consider the set of all trajectories T' defined by at most $2l$ points from $\bigcup_{i=1}^{l'} B_{p_i}(r, r')$ and containing at least one point from $B_{p_i}(r, r')$ for every i; further these points respect the ordering of the sets that they belong to, i.e., if $p \in B_{p_i}(r, r')$, $q \in B_{p_j}(r, r')$, and $i < j$, then p appears before q in the trajectory (for points coming from the same set $B_{p_i}(r, r')$ all possible orderings are considered). Note that $|T'| \leq \left(l' \cdot \left(\frac{r}{r'}\right)^{O(d)}\right)^{2l}$. Further, $\mathtt{d}_F(\gamma, \gamma') \leq r$ for all $\gamma' \in T'$.

We will show that for any $\gamma'' \in \mathcal{B}_{\mathtt{d}_F}(\gamma, r) \cap T^l$, there exists $\gamma' \in T'$ such that $\mathtt{d}_F(\gamma', \gamma'') \leq r'$. Thus T' is the desired cover, completing the first part of our proof. Let $\gamma'' = \langle q_1, \ldots, q_{l''} \rangle$ for some $l'' \leq l$. By definition of $\mathtt{d}_F$ and the fact that $\mathtt{d}_F(\gamma, \gamma'') \leq r$, each q_i has a corresponding sequence of points $\langle p_{j_i}, p_{j_i+1}, \ldots, p_{j'_i} \rangle$ each of which is at most r distance away from q_i. Moreover, for all $1 \leq i < l''$ we have $j'_i \leq j_{i+1} \leq j'_i + 1$, and $j_1 = 1, j'_{l''} = l'$.

For each q_i and $j \in \{j_i, j_i + 1, \ldots, j'_i\}$, let u_j denote the point in $B_{p_j}(r, r')$ that is closest to q_i. Note that $q_i \in \mathcal{B}_E(p_j, r)$ and $\|q_i - u_j\| \leq r'$. Consider the sequence of points $\gamma(q_i) = \langle u_{j_i}, u_{j_i+1}, \ldots, u_{j'_i} \rangle$. Then, $\mathtt{d}_F(\langle q_i \rangle, \gamma(q_i)) \leq r'$. Let γ' be the trajectory obtained by concatenating $\gamma(q_1), \ldots, \gamma(q_{l''})$ in order. Then we get $\mathtt{d}_F(\gamma', \gamma'') \leq r'$. Further, by construction $\gamma'' \in T'$.

As far as running time is concerned, computing the set $\bigcup_{i=1}^{l'} B_{p_i}(r, r')$ takes time $l' \cdot \left(\frac{r}{r'}\right)^{O(d)}$. From this set, computing T' takes time $l^{2l} \cdot \left(\frac{r}{r'}\right)^{O(dl)}$. ◀

For a trajectory having m points, computing $\mathtt{d}_F$ to any trajectory in T^l takes time $O(ml)$ using the standard dynamic programming algorithm, whereas computing the closest trajectory in T^l under $\mathtt{d}_F$ takes time $O\left(lm \log m \log\left(\frac{m}{l}\right)\right)$ time (see [7], Theorem 3). This, along with Corollary 11 and Lemma 12 give the following.

▶ **Theorem 13.** *Let $\varepsilon \in (0, \frac{4}{9}), \delta \in (1 - \frac{5}{18}\varepsilon, 1)$. Given a set of n trajectories $P \subseteq T$ each having at most m points and $k \in \mathbb{N}$, with probability $\geq 1 - \delta$, the algorithm* Cluster *(Algorithm 1) computes a $(1 + 3\varepsilon)$-approximate solution to the (k, T^l)-median problem for P under the discrete Fréchet distance in time*

$$nm \log m \log\left(\frac{m}{l}\right) \cdot 2^{O\left(\frac{k}{\varepsilon} \log\left(\frac{k}{\varepsilon}\right)\right)} \cdot \left(\frac{l}{\delta_1 \varepsilon}\right)^{O(kdl)},$$

where $\delta_1 = \frac{\varepsilon}{2} - \frac{9}{5}(1 - \delta)$.

Clustering under Hausdorff distance. Recall that U^l is the set of all point sets in $\mathbb{R}^d$ containing at most l points each. Thus, $U = \bigcup_{l>0} U^l$ is the set of all finite point sets of $\mathbb{R}^d$. Given $\mathcal{U} = (U, \mathtt{d}_H)$ and subsets $P \subseteq U$, we show how to approximately solve the (k, U^l)-clustering problem for P and $k, l > 0$.

▶ **Lemma 14.** *U^l is g-coverable under $\mathtt{d}_H$ for $g(x) = l^l \cdot x^{O(dl)}$. Further, an r'-cover of $\mathcal{B}_{\mathtt{d}_H}(\zeta, r) \cap U^l$ for $r > r' > 0$ and $\zeta \in U^l$ can be computed in $l^l \cdot \left(\frac{r}{r'}\right)^{O(dl)}$ time.*

Proof. Let $r > r' > 0$ be arbitrary, and let $\zeta = \{p_1, p_2, \ldots, p_{l'}\}$ for some $l' \leq l$. Since the Euclidean metric has doubling dimension $O(d)$, there exist $\left(\frac{r}{r'}\right)^{O(d)}$ points in the Euclidean ball $\mathcal{B}_E(p, r)$ such that any point in $\mathcal{B}_E(p, r)$ is at most r' distance away from one of these points; denote these points by $B_p(r, r')$.

Consider all subsets of size l of the set $\bigcup_{i=1}^{l'} B_{p_i}(r, r')$, denote this set by U'. Note that $|U'| = \left(l' \cdot \left(\frac{r}{r'}\right)^{O(d)}\right)^{l}$.

Next, consider any point set $\zeta' \in \mathcal{B}_{\mathtt{d}_H}(\zeta, r) \cap U^l$. By definition of Hausdorff distance, the points of ζ' (there are at most l of them) must lie in $\bigcup_{i=1}^{l'} \mathcal{B}_E(p_i, r)$. Thus, for each $p \in \zeta'$, there is some $i \in \{1, \ldots, l'\}$ such that $\|p - q\| \leq r'$ for some $q \in B_{p_i}(r, r')$. Thus, there exists $\zeta'' \in U'$ such that $\mathtt{d}_H(\zeta', \zeta'') \leq r'$. Then, U' is the desired cover, completing the first part of our proof.

Computing $\bigcup_{i=1}^{l'} B_{p_i}(r, r')$ takes time $l' \cdot \left(\frac{r}{r'}\right)^{O(d)}$. Computing U' from it takes time $\left(l' \cdot \left(\frac{r}{r'}\right)^{O(d)}\right)^l$. ◀

Computing $\mathtt{d}_H$ between two point sets of size m_1 and m_2 in $\mathbb{R}^d$ takes time $O(m_1 m_2)$. Given $\zeta \in U$ of size m, computing its nearest neighbor in U^l (under $\mathtt{d}_H$) boils down to finding l disks in $\mathbb{R}^d$ of minimum radius such that all the points of ζ lie in the union of these disks; the centers of these disks give the desired set in U^l. This is the l-center problem in $\mathbb{R}^d$ for the Euclidean metric, which is NP-Hard when l is part of the input. Note that the total number of subsets of ζ induced by disks in $\mathbb{R}^d$ is $m^{O(d)}$. By looking at l such subsets at a time, we can pick the one that covers ζ and minimizes the radius of the largest disk; this takes total time $m^{O(dl)}$. This along with Corollary 11 and Lemma 14 give the following.

▶ **Theorem 15.** *Let $\varepsilon \in (0, \frac{4}{9}), \delta \in (1 - \frac{5}{18}\varepsilon, 1)$. Given a set of n point sets $P \subseteq U$ each having at most m points and $k \in \mathbb{N}$, with probability $\geq 1 - \delta$, the algorithm Cluster (Algorithm 1) computes a $(1 + 3\varepsilon)$-approximate solution to the (k, U^l)-median problem for P under the Hausdorff distance can be computed in time*

$$nm^{O(dl)} \cdot 2^{O\left(\frac{k}{\varepsilon} \log\left(\frac{k}{\varepsilon}\right)\right)} \cdot \left(\frac{l}{\delta_1 \varepsilon}\right)^{O(kdl)},$$

where $\delta_1 = \frac{\varepsilon}{2} - \frac{9}{5}(1 - \delta)$.

▶ Remark. The running time of Theorem 15 can be improved to be similar to that of Theorem 13, since we believe a $(1 + O(\varepsilon))$-nearest neighbor in C for any $x \in X$ should suffice. For the Hausdorff case, this involves using a fast approximation for the l-center problem [2].

6 Hardness of k-median clustering under Hausdorff distance

▶ **Theorem 16.** *The k-median clustering problem for finite point sets under the Hausdorff distance is NP-Hard.*

Proof. We reduce the Euclidean k-median problem, which is known to be NP-Hard [29]. The reduction is fairly straightforward – for each input point p of an instance of the Euclidean k-median problem, we have a singleton set $\{p\}$ as input to the Haudorff k-median problem. Any solution to the Euclidean k-median problem is also a solution to the Hausdorff k-median problem of the same cost – we replace cluster center c in the Euclidean version by the cluster center $\{c\}$ for the Haudorff version, and for each p assigned to c, we assign $\{p\}$ to $\{c\}$.

On the other hand, consider a solution to the instance of Hausdorff k-median problem. In particular, let S be a cluster center that is assigned the sets $\{\{p_1\}, \ldots, \{p_n\}\}$. The cost of this single cluster is

$$\sum_{i=1}^{n} \mathtt{d}_H(\{p_i\}, S) = \sum_{i=1}^{n} \max_{s \in S} \|s - p_i\|,$$

by the definition of $\mathtt{d}_H$. Thus, replacing S by a singleton set $\{s\}$ for any $s \in S$ does not increase the cost of clustering. Hence we can assume that all cluster centers are singleton sets. We can then construct a solution for the Euclidean k-median problem by assigning p to s, where $\{s\}$ is the cluster center that $\{p\}$ was assigned to in the Hausdorff clustering solution. This does not increase the cost of the clustering as well. ◀

7 Conclusion

We have given a framework for clustering where the cluster centers are restricted to belong to a simpler metric space. We characterized general conditions on this simpler space that allow us to obtain efficient $(1+\varepsilon)$-approximation algorithms for the k-median problem. As special cases, we gave efficient algorithms for clustering trajectories and point sets under the discrete Fréchet and Hausdorff distances respectively.

It would be interesting to extend the general framework to other metric spaces, e.g., the continuous Fréchet distance, and to non-metric distance measures such as dynamic time warping. Another interesting next step is to provide other characterizations on the metric space for the cluster centers (as alternatives to the notion of covering discussed in this paper) that are amenable to efficient clustering algorithms.

References

1 Marcel R. Ackermann, Johannes Blömer, and Christian Sohler. Clustering for metric and nonmetric distance measures. *ACM Trans. Alg.*, 6(4):59:1–59:26, 2010.

2 Pankaj K Agarwal and Cecilia M Procopiuc. Exact and approximation algorithms for clustering. *Algorithmica*, 33(2):201–226, 2002.

3 Sanjeev Arora, Prabhakar Raghavan, and Satish Rao. Approximation schemes for Euclidean k-medians and related problems. In *Proc. ACM Symp. Th. Computing*, volume 98, pages 106–113, 1998.

4 Arturs Backurs and Anastasios Sidiropoulos. Constant-distortion embeddings of hausdorff metrics into constant-dimensional l_p spaces. In *APPROX/RANDOM*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.

5 Mihai Bādoiu, Sariel Har-Peled, and Piotr Indyk. Approximate clustering via core-sets. In *Proc. ACM Symp. Th. Computing*, pages 250–257. ACM, 2002.

6 Nicolas Basalto, Roberto Bellotti, Francesco De Carlo, Paolo Facchi, Ester Pantaleo, and Saverio Pascazio. Hausdorff clustering of financial time series. *Physica A: Statistical Mechanics Applications*, 379(2):635–644, 2007.

7 Sergey Bereg, Minghui Jiang, Wencheng Wang, Boting Yang, and Binhai Zhu. Simplifying 3D polygonal chains under the discrete Fréchet distance. In *Lat. Amer. Symp. Theoret. Informatics*, pages 630–641. Springer, 2008.

8 P. C. Besse, B. Guillouet, J. Loubes, and F. Royer. Review and perspective for distance-based clustering of vehicle trajectories. *IEEE Tran. Intell. Transportation Sys.*, 17(11):3306–3317, 2016.

9 Kevin Buchin, Anne Driemel, Joachim Gudmundsson, Michael Horton, Irina Kostitsyna, Maarten Löffler, and Martijn Struijs. Approximating (k,l)-center clustering for curves. In *Proc. ACM-SIAM Symp. Disc. Alg.*, pages 2922–2938. SIAM, 2019.

10 Kevin Buchin, Anne Driemel, and Martijn Struijs. On the hardness of computing an average curve. *arXiv preprint*, 2019. `arXiv:1902.08053`.

11 Jinyang Chen, Rangding Wang, Liangxu Liu, and Jiatao Song. Clustering of trajectories based on hausdorff distance. In *Proc. Int. Conf. Electronics Comm. Control*, pages 1940–1944. IEEE, 2011.

12 Vincent Cohen-Addad, Philip N Klein, and Claire Mathieu. Local search yields approximation schemes for k-means and k-median in euclidean and minor-free metrics. *SIAM J. Computing*, 48(2):644–667, 2019.

13 Anne Driemel, Amer Krivošija, and Christian Sohler. Clustering time series under the Fréchet distance. In *Proc. ACM-SIAM Symp. Disc. Alg.*, pages 766–785. SIAM, 2016.

14 Thomas Eiter and Heikki Mannila. Computing discrete Fréchet distance. Technical report, Information Systems Dept., Technical University of Vienna, 1994.

15 Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. ACM Int. Conf. Know. Disc. Data Mining*, volume 96, pages 226–231, 1996.

16 Tomás Feder and Daniel Greene. Optimal algorithms for approximate clustering. In *Proc. ACM Symp. Th. Computing*, pages 434–444. ACM, 1988.

17 Scott Gaffney and Padhraic Smyth. Trajectory clustering with mixtures of regression models. In *Proc. ACM Int. Conf. Know. Disc. Data Mining*, volume 99, pages 63–72, 1999.

18 Venkatesan Guruswami and Piotr Indyk. Embeddings and non-approximability of geometric problems. In *Proc. ACM-SIAM Symp. Disc. Alg.*, volume 3, pages 537–538, 2003.

19 Sariel Har-Peled and Akash Kushal. Smaller coresets for k-median and k-means clustering. *Disc. Computat. Geom.*, 37(1):3–19, 2007.

20 Sariel Har-Peled and Soham Mazumdar. On coresets for k-means and k-median clustering. In *Proc. ACM Symp. Th. Computing*, pages 291–300. ACM, 2004.

21 Felix Hausdorff. *Grundzuge der mengenlehre*, volume 61. American Mathematical Society, 1978.

22 Chih-Chieh Hung, Wen-Chih Peng, and Wang-Chien Lee. Clustering and aggregating clues of trajectories for mining trajectory patterns and routes. *Int. J. Very Large Databases*, 24(2):169–192, 2015.

23 Daniel P Huttenlocher, Gregory A Klanderman, and William J Rucklidge. Comparing images using the hausdorff distance. *IEEE Trans. Pattern Anal. Machine Intell.*, 15(9):850–863, 1993.

24 Kamal Jain, Mohammad Mahdian, and Amin Saberi. A new greedy approach for facility location problems. In *Proc. ACM Symp. Th. Computing*, pages 731–740. ACM, 2002.

25 Stavros G Kolliopoulos and Satish Rao. A nearly linear-time approximation scheme for the euclidean k-median problem. *SIAM J. Computing*, 37(3):757–782, 2007.

26 Amit Kumar, Yogish Sabharwal, and Sandeep Sen. Linear time algorithms for clustering problems in any dimensions. In *Int. Coll. Automata Lang. Programming*, pages 1374–1385. Springer, 2005.

27 Amit Kumar, Yogish Sabharwal, and Sandeep Sen. Linear-time approximation schemes for clustering problems in any dimensions. *J. ACM*, 57(2):5, 2010.

28 Shi Li and Ola Svensson. Approximating k-median via pseudo-approximation. *SIAM J. Computing*, 45(2):530–547, 2016.

29 Nimrod Megiddo and Kenneth J Supowit. On the complexity of some common geometric location problems. *SIAM J. Computing*, 13(1):182–196, 1984.

30 Abhinandan Nath and Erin Taylor. k-Median clustering under discrete Fréchet and Hausdorff distances. *CoRR*, 2020. `arXiv:2004.00722`.

31 Lin Qu, Fan Zhou, and YW Chen. Trajectory classification based on hausdorff distance for visual surveillance system. *J. Jilin University*, 6:1618–1624, 2009.

32 Cynthia Sung, Dan Feldman, and Daniela Rus. Trajectory clustering for motion prediction. In *IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pages 1547–1552. IEEE, 2012.

33 Hongteng Xu, Yang Zhou, Weiyao Lin, and Hongyuan Zha. Unsupervised trajectory clustering via adaptive multi-kernel-based shrinkage. In *Proc. IEEE Int. Conf. Comp. Vision*, pages 4328–4336, 2015.

Four-Dimensional Dominance Range Reporting in Linear Space

Yakov Nekrich
Department of Computer Science, Michigan Technological University, Houghton, MI, USA
yakov.nekrich@googlemail.com

Abstract

In this paper we study the four-dimensional dominance range reporting problem and present data structures with linear or almost-linear space usage. Our results can be also used to answer four-dimensional queries that are bounded on five sides. The first data structure presented in this paper uses linear space and answers queries in $O(\log^{1+\varepsilon} n + k \log^{\varepsilon} n)$ time, where k is the number of reported points, n is the number of points in the data structure, and ε is an arbitrarily small positive constant. Our second data structure uses $O(n \log^{\varepsilon} n)$ space and answers queries in $O(\log n + k)$ time.

These are the first data structures for this problem that use linear (resp. $O(n \log^{\varepsilon} n)$) space and answer queries in poly-logarithmic time. For comparison the fastest previously known linear-space or $O(n \log^{\varepsilon} n)$-space data structure supports queries in $O(n^{\varepsilon} + k)$ time (Bentley and Mauer, 1980). Our results can be generalized to $d \geq 4$ dimensions. For example, we can answer d-dimensional dominance range reporting queries in $O(\log \log n (\log n / \log \log n)^{d-3} + k)$ time using $O(n \log^{d-4+\varepsilon} n)$ space. Compared to the fastest previously known result (Chan, 2013), our data structure reduces the space usage by $O(\log n)$ without increasing the query time.

2012 ACM Subject Classification Theory of computation → Computational geometry; Theory of computation → Data structures design and analysis

Keywords and phrases Range searching, geometric data structures, word RAM

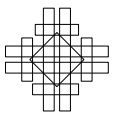
Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.59

Related Version A full version of this paper[30] is available at `https://arxiv.org/abs/2003.06742`.

1 Introduction

In the orthogonal range searching problem we keep a set S of multi-dimensional points in a data structure so that for an arbitrary axis-parallel query rectangle Q some information about points in $Q \cap S$ can be computed efficiently. Range searching is one of the most fundamental and widely studied problems in computational geometry. Typically we want to compute some aggregate function on $Q \cap S$ (range aggregate queries), generate the list of points in S (reporting queries) or determine whether $Q \cap S = \emptyset$ (emptiness queries). In this paper we study the complexity of four-dimensional orthogonal range reporting and orthogonal range emptiness queries in the case of dominance queries and in the case when the query range is bounded on five sides. We demonstrate for the first time that in this scenario both queries can be answered in poly-logarithmic time using linear or almost-linear space.

Range trees, introduced by Lueker [23] in 1978 and Bentley [7] in 1980, provide a solution for the range reporting problem in $O(n \log^d n)$ space and $O(\log^d n + k)$ time for any constant dimension d. Henceforth k denotes the number of points in the answer to a reporting query and n denotes the number of points in the data structure. A number of improvements both in time and in space complexity were obtained in the following decades. See e.g., [24, 12, 17, 18, 16, 13, 32, 14, 15, 34, 35, 5, 6, 26, 25, 27, 22, 31, 11, 10] for a selection of previous works on range reporting and related problems. Surveys of previous results can be found in [3, 4, 29]. We say that a range query is f-sided if the query range is bounded on f sides, i.e., a query can be specified with f inequalities; see Fig. 1 on p. 2. Researchers noticed



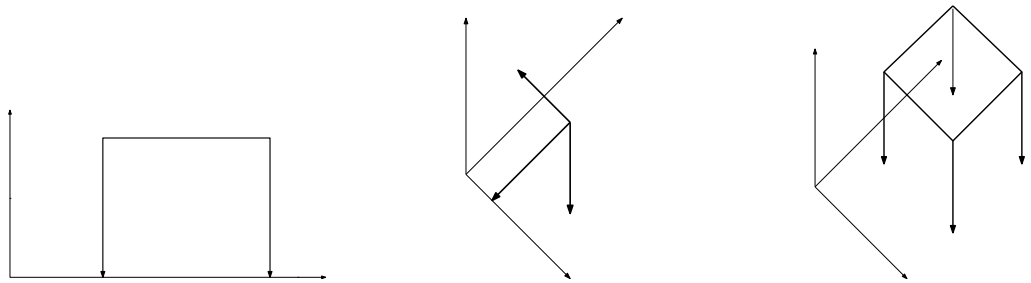


Figure 1 Examples of queries in two and three dimensions. (a) A two-dimensional 3-sided query (b) A three-dimensional 3-sided (dominance) query (c) A three-dimensional 5-sided query.

that the space and time complexity of range reporting depends not only on the dimensionality: the number of sides that bound the query range is also important. Priority search tree, introduced by McCreight [24], provides an $O(n)$ space and $O(\log n + k)$ time solution for 3-sided range reporting queries in two dimensions. In [16] Chazelle and Edelsbrunner have demonstrated that three-dimensional 3-sided queries (aka three-dimensional dominance queries) can be answered in $O(\log^2 n + k)$ time using an $O(n)$ space data structure. In 1985 Chazelle [13] described a compact version of the two-dimensional range tree that uses $O(n)$ space and supports general (4-sided) two-dimensional range reporting queries in $O(\log n + k \log^{\varepsilon} n)$ time, where ε denotes an arbitrarily small positive constant. In [13] the author also presented an $O(n)$ space data structure that supports 5-sided three-dimensional reporting queries in $O(\log^{1+\varepsilon} n + k \log^{\varepsilon} n)$ time. Bentley and Mauer [8] described a linear-space data structure that supports d-dimensional range reporting queries for any constant d; however, their data structure has prohibitive query cost $O(n^{\varepsilon} + k)$.

Summing up, we can answer range reporting queries in poly-logarithmic time using an $O(n)$ space data structure when the query is bounded on at most 5 sides and the query is in two or three dimensions. Significant improvements were achieved on the query complexity of this problem in each case; see Table 1. However, surprisingly, linear-space and polylog-time data structures are known only for the above mentioned special cases of the range reporting. For example, to answer four-dimensional 4-sided queries (four-dimensional dominance queries) in polylogarithmic time using previously known solutions one would need $\Omega(n \log n / \log \log n)$ space. This situation does not change when we increase the space usage to $O(n \log^{\varepsilon} n)$ words: data structures with poly-logarithmic time are known for the above described special cases only. See Table 2.

Previous results raise the question about low-dimensional range reporting. What determines the complexity of range reporting data structures in $d \geq 4$ dimensions: the dimensionality or the number of sides in the query range? The lower bound of Patrascu [33] resolves this question with respect to query complexity. It is shown in [33] that any data structure using $O(n\texttt{polylog}(n))$ space needs $\Omega(\log n / \log \log n)$ time to answer four-dimensional dominance (4-sided) queries. On the other hand, two- and three-dimensional 4-sided queries can be answered in $O(\log \log n + k)$ time using $O(n\texttt{polylog}(n))$ space. In this paper we address the same question with respect to space complexity.

We demonstrate that four-dimensional 5-sided queries can be answered in $O(\log^{1+\varepsilon} n + k \log^{\varepsilon} n)$ time using an $O(n)$ space data structure. Our data structure can also support 5-sided emptiness queries in $O(\log^{1+\varepsilon} n)$ time. If the space usage is slightly increased to $O(n \log^{\varepsilon} n)$, then we can answer reporting and emptiness queries in $O(\log n + k)$ and $O(\log n)$

time respectively. For comparison, the fastest previous method [9] requires $O(n \log^{1+\varepsilon} n)$ space and supports queries in $O(\log n + k)$ time. Since dominance queries are a special case of 5-sided queries, our results can be used to answer four-dimensional dominance queries within the same time and space bounds. Using standard techniques, our results can be generalized to d dimensions for any constant $d \geq 4$: We can answer d-dimensional dominance queries in $O(\log\log n(\log n/\log\log n)^{d-3} + k)$ time and $O(n \log^{d-4+\varepsilon} n)$ space. We can also answer arbitrary $(2d-3)$-sided d-dimensional range reporting queries within the same time and space bounds.

Our base data structure is the range tree on the fourth coordinate and every tree node contains a data structure that answers three-dimensional dominance queries. We design a space-efficient representation of points stored in tree nodes, so that each point uses only $O(\log\log n)$ bits. Since each point is stored $O(\log n/\log\log n)$ times in our range tree, the total space usage is $O(n \log n)$ bits. Using our representation we can answer three-dimensional queries on tree nodes in $O(\log^{\varepsilon} n)$ time; we can also "decode" each point, i.e., obtain the actual point coordinates from its compact representation, in $O(\log^{\varepsilon} n)$ time. Efficient representation of points is the core idea of our method: we keep a geometric construct called *shallow cutting* in each tree node and exploit the relationship between shallow cuttings in different nodes. Shallow cuttings were extensively used in previous works to decrease the query time. But, to the best of our knowledge, this paper is the first that uses shallow cuttings to reduce the space usage. The novel part of our construction is a combination of several shallow cuttings that allows us to navigate between the nodes of the range tree and "decode" points from their compact representations. We recall standard techniques used by our data structure in Section 2. The linear-space data structure is described in Section 3. In Section 4 we show how the decoding cost can be reduced to $O(1)$ by slightly increasing the space usage. The data structure described in Section 4 supports queries in $O(\log n + k)$ time and uses $O(n \log^{\varepsilon} n)$ space. Previous and new results for 4-sided queries in four dimensions are listed in Table 3. Compared to the only previous linear-space data structure [8], we achieve exponential speed-up in query time. Compared to the fastest previous result [10], our data structure reduces the space usage by $O(\log n)$ without increasing the query time.

The model of computation used in this paper is the standard RAM model. The space is measured in words of $\log n$ bits and we can perform standard arithmetic operations, such as addition and subtraction, in $O(1)$ time. Our data structures rely on bit operations, such as bitwise AND or bit shifts or identifying the most significant bit in a word. However such operations are performed only on small integer values (i.e., on positive integers bounded by n) and can be easily implemented using look-up tables. Thus our data structures can be implemented on the RAM model with standard arithmetic operations and arrays.

We will assume in the rest of this paper that all point coordinates are bounded by n. The general case can be reduced to this special case using the reduction to rank space, described in Section 2. All results of this paper remain valid when the original point coordinates are real numbers.

2 Preliminaries

In this paper ε will denote an arbitrarily small positive constant. We will consider four-dimensional points in a space with coordinate axes denoted by x, y, z, and z'. The following techniques belong to the standard repertoire of geometric data structures.

Table 1 Linear-space data structures for different types of queries. The second column provides the reference to the first data structure achieving linear space and the year of the first publication. The third column contains the query time of the first data structure. The fourth and the fifth columns contain the same information about the best (fastest) currently known data structure. Result marked with E supports only emptiness queries. Data structures in rows 4 and 5 also support 4-sided queries.

Query Type	First		Best	
	Ref	Query Time	Ref	Query Time
2-D 3-sided	[24], 1985	$O(\log n + k)$	[5], 2000	$O(k+1)$
2-D 4-sided	[13], 1985	$O(\log n + k \log^{\varepsilon} n)$	[11], 2011	$O((k+1)\log^{\varepsilon} n)$
3-D 3-sided	[16], 1986	$O(\log^2 n + k)$	[9], 2011	$O(\log\log n + k)$
3-D 5-sided	[13], 1985	$O(\log^{1+\varepsilon} n + k \log^{\varepsilon} n)$	[19], 2012	$O(\log n \log\log n)^{\mathrm{E}}$
4-D 5-sided	New	$O(\log^{1+\varepsilon} n + k \log^{\varepsilon} n)$		

Table 2 $O(n \log^{\varepsilon} n)$-space data structures for orthogonal range reporting. The second and the third columns contain the reference to and the query time of the first data structure. The fourth and the fifth columns contain the reference to and the query time of the best (fastest) currently known data structure. Data structures in rows 2 and 3 also support 4-sided queries.

Query Type	First		Best	
	Ref	Query Time	Ref	Query Time
2-D 4-sided	[13], 1985	$O(\log n + k)$	[5], 2000	$O(\log\log n + k)$
3-D 5-sided	[13], 1985	$O(\log n + k)$	[11], 2011	$O(\log\log n + k)$
4-D 5-sided	New	$O(\log n + k)$		

Shallow Cuttings. A point q dominates a point p if and only if every coordinate of q is larger than or equal to the corresponding coordinate of p. The *level* of a point q in a set S is the number of points p in S, such that q dominates p (the point q is not necessarily in S). We will say that a *cell* C is a region of space dominated by a point q_c and we will call q_c the apex point of C. A t-shallow cutting of a set S is a collection of cells $\mathcal{C}$, such that (i) every point in $\mathbb{R}^d$ with level at most t (with respect to S) is contained in some cell C_i of $\mathcal{C}$ and (ii) if a point p is contained in some cell C_j of $\mathcal{C}$, then the level of p in S does not exceed $2t$. The size of a shallow cutting is the number of its cells. We can uniquely identify a shallow cutting $\mathcal{C}$ by listing its cells and every cell can be identified by its apex point. Since the level of any point in a cell C_j does not exceed $2t$, every cell C_j contains at most $2t$ points from S, $|C_j \cap S| \leq 2t$ for any C_j in $\mathcal{C}$.

There exists a t-shallow cutting of size $O(n/t)$ for $d = 2$ [35] or $d = 3$ dimensions [1]. Shallow cuttings and related concepts are frequently used in data structures for three-dimensional dominance range reporting queries.

Consider a three-dimensional point $q_3 = (a, b, c)$ and the corresponding dominance range $Q_3 = (-\infty, a] \times (-\infty, b] \times (-\infty, c]$. We can find a cell C of a t-shallow cutting $\mathcal{C}$ that contains q_3 (or report that there is no such C) by answering a point location query in a planar rectangular subdivision of size $O(n/t)$. Point location queries in a rectangular subdivision of size n can be answered in $O(\log\log n)$ time using an $O(n)$-space data structure [10].

Range Trees. A range tree is a data structure that reduces d-dimensional orthogonal range reporting queries to a small number of $(d-1)$-dimensional queries. Range trees provide a general method to solve d-dimensional range reporting queries for any constant dimension d.

Table 3 Previous and new results on dominance range reporting in four dimensions. Results in lines 2, 3, and 7 can be modified so that space is decreased to $O(n \log n / \log\log n)$ and the query time is increased by $O(\log^{\varepsilon} n)$ factor.

Ref.	Space	Query Time
[8]	$O(n)$	$O(n^{\varepsilon} + k)$
[16]	$O(n \log n)$	$O(\log^2 n + k)$
[21]	$O(n \log n)$	$O(\log^2 n / \log\log n + k)$
[25]	$O(n \log^{2+\varepsilon} n)$	$O(\log n \log\log n + k)$
[1]	$O(n \log^{1+\varepsilon} n)$	$O(\log n \log\log n + k)$
[9]	$O(n \log^{1+\varepsilon} n)$	$O(\log n + k)$
[9]	$O(n \log n)$	$O(\log n \log\log n + k)$
New	$O(n \log^{\varepsilon} n)$	$O(\log n + k)$
New	$O(n)$	$O(\log^{1+\varepsilon} n + k \log^{\varepsilon} n)$

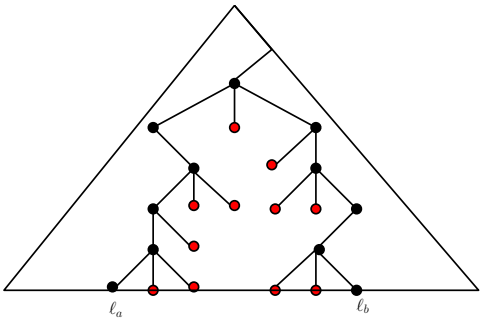


Figure 2 Example of a query in a range tree with node degree $\rho = 3$. Canonical nodes are shown in red. Only nodes of $\pi_a \cup \pi_b$ and $\pi'_a \cup \pi'_b$ are shown.

In this paper we use this data structure to reduce four-dimensional 5-sided reporting queries to three-dimensional 3-sided queries. A range tree for a set S is a balanced tree that holds the points of S in the leaf nodes, sorted by their z'-coordinate. We associated a set $S(u)$ with every internal node u. $S(u)$ contains all points p that are stored in the leaf descendants of u. We assume that each internal node of T has $\rho = \log^{\varepsilon} n$ children. Thus every point is stored in $O(\log n / \log\log n)$ sets $S(u)$. We keep a data structure that supports three-dimensional reporting queries on $S(u)$ for every node u of the range tree.

Consider a query $Q = Q_3 \times [a, b]$, where Q_3 denotes a 3-sided three-dimensional query range. Let ℓ_a be the rightmost leaf that holds some z'-coordinate $a' < a$ and let ℓ_b be the leftmost leaf that holds some $b' > b$. Let v_{ab} denote the lowest common ancestor of ℓ_a and ℓ_b. We denote by π_a (resp. π_b) the set of nodes on the path from ℓ_a (ℓ_b) to v_{ab}, excluding the node v_{ab}. We will say that u is a left (right) sibling of v iff u and v have the same parent node and u is to the left (respectively, to the right) of v. The set π'_a consists of all nodes u that have some left sibling $v \in \pi_a$ and π'_b consists of all nodes u that have a right sibling $v \in \pi_b$. The set π''_a (π''_b) consists of all nodes in π'_a (resp. in π'_b) that are not children of v_{ab}. The set π'_{ab} consists of all children of v_{ab} that are in $\pi'_a \cap \pi'_b$. For any point $p \in S$, $a \le p.z' \le b$ iff $p \in S(u)$ for some u in $\pi''_a \cup \pi''_b \cup \pi'_{ab}$. Nodes $u \in \pi''_a \cup \pi''_b \cup \pi'_{ab}$ are called canonical nodes for the range $[a, b]$. See an example on Fig. 2. In order to answer a four-dimensional query Q we visit every canonical node u and report all points $p \in S(u) \cap Q_3$.

Thus we can answer a four-dimensional 5-sided query by answering $O(\rho \cdot \log n / \log\log n)$ three-dimensional 3-sided queries in canonical nodes.

Rank Space. Let E be a set of numbers. The *rank* of a number x in E is the number of elements in E that do not exceed x: $\mathrm{rank}(x, E) = |\{\, e \in E \mid e \le x \,\}|$. Let $\mathrm{pred}(x, E) = \max\{\, e \in E \mid e \le x \,\}$ and $\mathrm{succ}(x, E) = \min\{\, e \in E \mid e \ge x \,\}$. An element $e \in E$ is in the range $[a, b]$, $a \le e \le b$, iff its rank satisfies the inequality $a' \le \mathrm{rank}(e, E) \le b'$ where $a' = \mathrm{rank}(\mathrm{succ}(a, E), E)$ and $b' = \mathrm{rank}(b, E)$. Hence we can report all $e \in E$ satisfying $a \le e \le b$ by finding all elements e satisfying $a' \le \mathrm{rank}(e, E) \le b'$.

The same approach can be also extended to multi-dimensional range reporting. A three-dimensional transformation is implemented as follows. We say that a three-dimensional point $p \in S$ is reduced to rank space (or p is in the rank space) if each coordinate of p is replaced by its rank in the set of corresponding coordinates. That is, each point $p = (p.x, p.y, p.z)$ is replaced with $\xi(p) = (\mathrm{rank}(p.x, S_x), \mathrm{rank}(p.y, S.y), \mathrm{rank}(p.z, S_z))$, where S_x, S_y, and S_z denote the sets of x-, y-, and z-coordinates of points in S. For a point $p \in S$ we have

$$p \in [a, b] \times [c, d] \times [e, f] \Leftrightarrow \xi(p) \in [a', b'] \times [c', d'] \times [e', f']$$

where $a' = \mathrm{rank}(\mathrm{succ}(a, S_x), S_x)$, $c' = \mathrm{rank}(\mathrm{succ}(c, S_y), S_y)$, $e' = \mathrm{rank}(\mathrm{succ}(e, S_z), S_z)$, $b' = \mathrm{rank}(b, S_x)$, $d' = \mathrm{rank}(d, S_y)$, and $f' = \mathrm{rank}(f, S_z)$. Thus we can reduce three-dimensional queries on a set S to three-dimensional queries on a set $\{\, \xi(p) \mid p \in S \,\}$. Suppose that we store the set $\xi(S)$ in a data structure that answers queries in time $t(n)$ and uses space $s(n)$. Suppose that we also keep data structures that answers predecessor queries on S_x, S_y, and S_z. Then we can answer orthogonal range reporting queries on S in time $t(n) + O(t_{\mathrm{pred}}(n))$ using space $s(n) + O(s_{\mathrm{pred}}(n))$. Here $s_{\mathrm{pred}}(n)$ and $t_{\mathrm{pred}}(n)$ are the space usage and query time of the predecessor data structure. Additionally we need a look-up table that computes $\xi^{-1}(p)$, i.e., computes the coordinates of a point p from its coordinates in the rank space. As we will show later, in some situations this table is not necessary. Summing up, reduction to rank space enables us to reduce the range reporting problem on a set $S \subset \mathbb{R}^3$ to a special case when all point coordinates are positive integers bounded by $|S|$.

The same rank reduction technique can be applied to range reporting in any constant dimension d. In the rest of this paper we will assume for simplicity that all points of S are in the rank space.

3 Five-Sided Range Reporting in Linear Space

Base Structure. We keep all points in a range tree that is built on the fourth coordinate. Each tree node has $\rho = \log^{\varepsilon} n$ children; thus the tree height is $O(\log n / \log\log n)$. Let $S(u)$ denote the set of points assigned to a node u. To simplify the notation, we will not distinguish between points in $S(u)$ and their projections onto (x, y, z)-space. We will say for example that a point p is in a range $Q = (-\infty, a] \times (-\infty, b] \times (-\infty, c]$ if the projection of p onto (x, y, z)-space is in Q.

Since we aim for a linear-space data structure, we cannot store sets $S(u)$ in the nodes of the range tree. We keep a t_0-shallow cutting $\mathcal{C}(u)$ of $S(u)$ where $t_0 = \log^2 n$. For every cell $C_i(u)$ of the shallow cutting we store all points from $S(u) \cap C_i$ in a data structure supporting three-dimensional dominance queries. We do not store the original (real) coordinates of points[1] in C_i. Instead we keep coordinates in the rank space of $S(u) \cap C_i$. Since $S(u) \cap C_i$ contains $O(\log^2 n)$ points, we need only $O(\log\log n)$ bits per point to answer three-dimensional dominance queries in the rank space.

[1] To avoid clumsy notation, we will sometimes omit the node specification when the node is clear from the context. Thus we will sometimes write C_i instead of $C_i(u)$ and $\mathcal{C}$ instead of $\mathcal{C}(u)$. The same simplification will be used for other shallow cuttings.

We can answer a 5-sided query $(-\infty, q_x] \times (-\infty, q_y] \times (-\infty, q_z] \times [z'_l, z'_r]$ by visiting all canonical nodes that cover the range $[z'_l, z'_r]$. In every visited node we answer a three-dimensional dominance query, i.e., report all points dominated by $q_3 = (q_x, q_y, q_z)$ in two steps: first, we search for some cell $C_i(u)$ that contains q_3. If such a cell $C_i(u)$ exists, then we answer the dominance query in the rank space of $S(u) \cap C_i(u)$ in $O(1)$ time per point. We describe the data structure for dominance queries on t_0 rank-reduced points in the full version of this paper [30].

We must address several issues in order to obtain a working solution: How can we transform a three-dimensional query to the rank space of $C_i(u)$? A data structure for cell $C_i(u)$ reports points in the rank space of $C_i(u)$. How can we obtain the original point coordinates? Finally how do we answer a query on $S(u)$ if q_3 is not contained in any cell $C_i(u)$? First, we will explain how to decode points from a cell $C(u)$ and obtain their original coordinates. We also show how to transform a query to the rank space of a cell. Next we will describe a complete procedure for answering a query. Finally we will improve the query time and achieve the main result of this section.

Decoding Points. This is the crucial component of our construction. We will need additional structures to obtain the original coordinates of points from $\mathcal{C}(u)$. To this end we keep an additional $(4t_0)$-shallow cutting $\mathcal{C}'(u)$ in every node of the range tree. For each cell $C'_i(u)$ of $\mathcal{C}'(u)$ we create a separate $(2t_0)$-shallow cutting of $S(u) \cap C'_i(u)$, called $\mathcal{D}_i(u)$.

▶ **Lemma 1.** *Let $\mathcal{A}$ be an f-shallow cutting for a set S and let $\mathcal{B}$ be an (f')-shallow cutting for a set $S' \subseteq S$ so that $f' \geq 2f$. Every cell A_i of $\mathcal{A}$ is contained in some cell B_j of $\mathcal{B}$.*

Proof. Consider an apex point a_i of A_i (i.e., the point with maximum x- y-, and z-coordinates in A_i). The level of a_i in S is at most $2f$ by definition of a shallow cutting. Since $S' \subseteq S$, the level of a_i in S' does not exceed $2f$. Hence there exists a cell B_j of $\mathcal{B}$ that contains a_i. The apex b_j of B_j dominates a_i. Hence b_j also dominates all points from A_i and B_j contains A_i. ◀

Lemma 1 will be extensively used in our decoding procedure. We will say that a point p in $C'_i(u)$ is *interesting* if p is contained in some $C_k(w)$, where w is an ancestor of u. Each interesting point $p \in S(u)$ can be uniquely represented by (a) a cell $C'_i(u)$ of $\mathcal{C}'(u)$ that contains p and (b) coordinates of p in the rank space of $C'_i(u)$. The following relationship between shallow cuttings provides the key insight.

▶ **Lemma 2.**

- **(i)** *Every cell $C_i(u)$ of $\mathcal{C}(u)$ is contained in some cell $C'_j(u)$ of $\mathcal{C}'(u)$*
- **(ii)** *Let u_r be a child of an internal node u. Every cell $D_{ij}(u)$ of every $\mathcal{D}_i(u)$ is contained in some cell $C'_k(u_r)$ of $\mathcal{C}'(u_r)$.*
- **(iii)** *Every interesting point from $C'_i(u)$ is stored in some cell $D_{ij}(u)$ of $\mathcal{D}_i(u)$.*

Proof. (i) Immediately follows from Lemma 1.(ii) Consider the apex point p_a of $D_{ij}(u)$. The point p_a dominates at most $4t_0$ points from $S(u)$ and at most $4t_0$ points from $S(u_r)$ because $S(u_r) \subset S(u)$. Hence both p_a and $D_{ij}(u)$ are contained in some cell of $C'(u_r)$. (iii) Suppose that a point $p \in S(w)$ is stored in some cell $C_k(w)$ of $\mathcal{C}(w)$ where w is an ancestor of u. The point p dominates at most $2t_0$ points from $S(w)$. Since $S(u)$ is a subset of $S(w)$, p dominates at most $2t_0$ points in $S(u)$. Hence p is contained in some cell $C'_i(u)$ of the shallow cutting $\mathcal{C}'(u)$. Every point of $C'_i(u) \cap S(u)$ that dominates at most $2t_0$ points of $S(u)$ is contained in some cell $D_{ij}(u)$ of $\mathcal{D}_i(u)$. ◀

Consider an arbitrary point p in a cell $C_i(u)$ of $\mathcal{C}(u)$. Our decoding procedure finds a representation of p in $\mathcal{C}'(u)$. That is, we find the cell $C'_j(u)$ of $\mathcal{C}'(u)$, such that $p \in C'_j(u)$, and the rank of p in $C'_j(u)$. The key observation is that $C_i(u)$ is contained in some $C'_j(u)$ (Lemma 2, item (i)) Therefore we need to store a pointer to $C'_j(u)$ only once for all points p in $C_i(u)$. For every p from $C_i(u)$, we can store its rank in $C'_j(u)$ using $O(\log\log n)$ bits. Next, our decoding procedure moves from a node u to its child u_f, such that $p \in S(u_f)$, and computes a representation of p in $\mathcal{C}'(u_f)$. This is done in two steps: First we examine the shallow cutting $\mathcal{D}_j(u)$ and find the cell $D_{jl}(u)$ that contains p. By Lemma 2, item (iii), such a cell always exists. The shallow cutting $\mathcal{D}_{jl}(u)$ consists of $O(1)$ cells. Therefore we can store, for any interesting point p, the cell $D_{jl}(u)$ containing p and the rank of p in $D_{jl}(u)$ using $O(\log\log n)$ bits. Then we move from $D_{jl}(u)$ to $\mathcal{C}(u_f)$: by Lemma 2, item (ii), $D_{jl}(u)$ is contained in some $C'_k(u_f)$. Thus we need to store the pointer to $C'_k(u_f)$ only once for all interesting points p in $D_{jl}(u)$. We can store the rank of p in $C'_k(u_f)$ using $O(\log\log n)$ bits. When the representation of p in $\mathcal{C}'(u_f)$ is known, we move to the child u_d of u_f, such that $p \in S(u_d)$ and compute a representation of p in $\mathcal{C}'(u_d)$. We continue in the same way until a leaf node is reached. Every leaf node ℓ contains original (real) coordinates of points in $S(\ell)$. Hence we can obtain the coordinates of p when a leaf is reached. Summing up, shallow cuttings $\mathcal{C}'(u)$ and $\mathcal{D}_i(u)$ allow us to move from a node u to a child of u using only $O(\log\log n)$ additional bits per point. A more detailed description of auxiliary data structures needed for decoding is given in the next paragraph.

For each cell $C_i(u)$ of $\mathcal{C}(u)$ we keep a pointer to the cell $C'_{\texttt{cont}(i)}(u)$ of $\mathcal{C}'(u)$ that contains $C_i(u)$. For every cell $D_{ij}(u) \in \mathcal{D}_i(u)$ and for each child u_r of u, we keep a pointer to the cell $C'_{\texttt{down}(i,j,r)}(u_r) \in \mathcal{C}'(u_r)$, such that $C'_{\texttt{down}(i,j,r)}(u_r)$ contains $D_{ij}(u)$. We can identify a point in each cell of a shallow cutting $\mathcal{C}(u)$ (resp. $\mathcal{C}'(u)$ or $\mathcal{D}'_i(u)$) with $O(\log\log n)$ bits because each cell contains a poly-logarithmic number of points. The x-rank of a point in a cell will be used as its identifier. We keep a mapping from points in a cell C_i to the corresponding points in a containing cell $C'_{\texttt{cont}(i)}$. The array $F_X(C_i)$ maps x-ranks of points in C_i to their x-ranks in $C'_{\texttt{cont}(i)}$: if the x-rank of a point $p \in C_i$ is equal to f, then $F_X[f] = g$ where g is the x-rank of p in $C'_{\texttt{cont}(i)}$. The array $F''_{X,r}$ for a cell $D_{ij}(u)$ and a child u_r of u maps x-ranks of points in $D_{ij}(u)$ to their x-ranks in $C'_{\texttt{down}(i,j,r)}$. If the x-rank of a point $p \in C_i$ is equal to f, then $F''_{X,r}[f] = g$ where g is the x-rank of p in $C'_{\texttt{down}(i,j,r)}$. We also keep a mapping from $C'_i(u)$ to cells of $\mathcal{D}_i(u)$: for every point $p \in C_i(u)$ we store the cell D_{ij} that contains p and the x-rank of p in C_{ij} (or $NULL$ if p is not in $\mathcal{C}_{ij}$). For every point p in each cell $D_{ij}(u)$ of $\mathcal{C}'_i(u)$, we store the index of the child u_r such that $p \in S(u_r)$. Our method requires $O(\log\log n)$ bits per point. Each pointer from $C_i(u)$ to $C'_{\texttt{cont}(i)}(u)$ and from $D_{ij}(u)$ to $C'_{\texttt{down}(i,j,r)}(u)$ consumes $O(\log n)$ bits. We store $O(\log^{2\varepsilon} n)$ pointers per cell and there are $O(n/(\log n \log\log n))$ cells in all shallow cuttings of the range tree. Hence the total space used by all pointers is $O(n \log^{2\varepsilon} n)$ bits.

▶ **Lemma 3.** *For any interesting point p in a cell $C'_i(u)$, we can find the representation of p in $C'_i(u_f)$, where u_f is the child of u that contains p.*

Proof. First we identify the cell $D_{ij}(u)$ of $\mathcal{D}_i(u)$ that contains p and compute the x-rank of p in $D_{ij}(u)$. Since p is interesting, such a cell exists. Then we use the array $F''_{X,k}$ of this cell and find the x-rank of p in the cell $C'_{\texttt{down}(i,j,k)}$. ◀

For any point from $C_i(u)$ we can obtain its position in some cell $C'_{\texttt{cont}(i)}(u)$ in $O(1)$ time. Then we can move down and obtain its representation in a child of u in $O(1)$ time. We can access the original coordinates of p when a leaf node is reached. Thus we can "decode" a point p if we know its position in a cell $C_i(u)$ in $O(\log n/\log\log n)$ time.

We can reduce a three-dimensional query $(-\infty, a] \times (-\infty, b] \times (-\infty, c]$ to the rank space of a cell by binary search. Let $X(C_i)$ denote the list of points in a cell C_i sorted by x-coordinates. To compare a with the x-coordinate of $X(C_i)[g]$ for some index g, we decode the point $p = X(C_i)[g]$ as explained above. Hence we can find the predecessor of a in $X(C_i)$ by binary search in $O(\log \log n)$ time. We can find the predecessor of b in $Y(C_i)$ and the predecessor of c in $Z(C_i)$ using the same procedure, where $Y(C_i)$ and $Z(C_i)$ are the lists of points in C_i sorted by their y- and z-coordinates respectively.

Queries. Consider a four-dimensional 5-sided query $(-\infty, a] \times (-\infty, b] \times (-\infty, c] \times [z'_l, z'_r]$. We visit all canonical nodes that cover the range $[z'_l, z'_r]$. In every visited node we answer a three-dimensional query using the following procedure. We find a cell $C_i(u)$ that contains p. We transform $(-\infty, a] \times (-\infty, b] \times (-\infty, c]$ to the rank space of $C_i(u)$ and answer the transformed query on $C_i(u) \cap S(u)$. Every reported point is decoded using the procedure described above. If there is no cell $C_i(u)$ that contains p, then p dominates at least t_0 points from $S(u)$. In this case we visit all children of u and recursively answer three-dimensional dominance query in each child using the same procedure.

We need $O(\log \log n)$ time to find the cell $C_i(u)$ or determine that $C_i(u)$ does not exist. To answer a query on $C_i(u)$ we need $O(\log n)$ time (ignoring the time to report points, but taking into account the time that we need to transform a query to the rank space of $C_i(u)$). Thus the total time spent in a node u is $O(\log n)$. The time spent in descendants of u can be estimated as follows. Let T_u be the subtree of the range tree induced by u and its visited descendants. Let T'_u denote the subtree of T_u obtained by removing all leaves of T_u. Every leaf of T'_u is an internal node of T_u. Hence we report at least t_0 points for every leaf in T'_u. The height of T'_u is bounded by $O(\log n / \log \log n)$. Let l_u denote the number of leaves in T'_u. The total number of nodes in T'_u is bounded by $O(l_u \log n / \log \log n)$. Every node of T'_u has ρ children. Hence the total number of nodes in T_u does not exceed $O(l_u(\log^{1+\varepsilon} n / \log \log n))$. The time spent in all nodes of T_u can be bounded by $O(l_u \log^{2+\varepsilon} n)$ (again, ignoring the time to decode and report points). When we visit descendants of u we report at least $k_u = \Omega(l_u \cdot t_0)$ points and each point is decoded in $O(\log n / \log \log n)$ time. The total time spent in descendants of u is $O(l_u \log^{2+\varepsilon} n + k_u(\log n / \log \log n) = O(k_u(\log n / \log \log n))$. The time spent in all canonical nodes and their descendants is $O(\log^{2+\varepsilon} n + k(\log n / \log \log n))$.

Faster Decoding. We can speed-up the decoding procedure and thus the overall query time without increasing the asymptotic space usage. Our approach is very similar to the method used in compact two-dimensional range trees [13, 28, 11]. All nodes in the range tree are classified according to their depth. A node u is an i-node if the depth h_u of u divides ρ^i where $\rho = \log^{\varepsilon} n$, $h_u = x \cdot \rho^i$ for some $i \geq 0$, but h_u does not divide ρ^j for $j > i$. We keep an additional $4t_i$-shallow cutting $\mathcal{C}^i$ in every i-node u where $t_i = \rho^i \cdot \log^2 n$. As before for each cell C^j_i of $\mathcal{C}^j$ we construct a $2t_j$-shallow cutting $\mathcal{D}^j_i$. Let an i-descendant of a node u denote the highest i-node v that is a descendant of u. If a node u is an i-node, then it has ρ^i i-descendants. For every cell D^j_{ik} of each $\mathcal{D}^j_i$ and for every j-descendant u_l of u, we keep the index $r = \texttt{down}(j, i, l)$ of the cell $C^j_r(u_l)$ that contains $D^j_{ik}(u)$. For each point in $D^j_{ik}(u)$ we keep the index l of the i-descendant that contains p and the x-rank of p in $C^j_r(u_l)$ where $r = \texttt{down}(j, i, l)$.

Using these additional shallow cuttings, we can reduce the decoding time to $O(\log^{\varepsilon} n)$. To decode a point p in $S(u)$ we move down from a node u to its child $u_{0,1}$ and find a representation of p in $\mathcal{C}'(u_{0,1})$. Then we move to the child $u_{0,2}$ of $u_{0,1}$ and continue in the same manner until a 1-node $u_{1,1}$ is reached. Next we move from $u_{1,1}$ to its 1-descendant $u_{1,2}$,

then to a 1-descendant $u_{1,3}$ of $u_{1,2}$, and so on until a 2-node is reached. During the j-th iteration we move down along a sequence of j-nodes until a $(j+1)$-node or a leaf node is reached. During each iteration we visit $O(\log^{\varepsilon} n)$ nodes and spend $O(1)$ time in every node. There are at most $\log_{\rho} \log n = O(1/\varepsilon)$ iterations. Hence the decoding time for a point is $O(\log^{\varepsilon} n)$. The total query time is reduced to $O(\log^{1+2\varepsilon} n + k \log^{\varepsilon} n)$. If we replace ε with $\varepsilon/2$ in the above proof, we obtain our first result.

▶ **Theorem 4.** *There exists a linear-space data structure that answers four-dimensional 5-sided reporting queries in $O(\log^{1+\varepsilon} n + k \log^{\varepsilon} n)$ time and four-dimensional 5-sided emptiness queries in $O(\log^{1+\varepsilon} n)$ time.*

4 Faster Queries using More Space

In this section we will show how to reduce the decoding time to $O(1)$ per point by increasing the space usage. We make several modifications in the basic construction of Section 3.

For any i and j such that $1 \le i \le j \le \rho$ and for any internal node u of the range tree, we store the set $S(u,i,j)$. $S(u,i,j)$ is the union of sets $S(u_i)$, $S(u_{i+1})$, …, $S(u_j)$. For every set $S(u,i,j)$ we construct a t_0-shallow cutting $\mathcal{C}(u,i,j)$. For each cell C_l of $\mathcal{C}(u,i,j)$ we store a three-dimensional data structure that keeps points from $C_l \cap S(u,i,j)$ in the rank space and answers three-dimensional dominance queries in $O(k+1)$ time.

The decoding procedure is implemented in the same way as in Section 3, but with different parameter values. Recall that a node u is an i-node for some $i \ge 0$ if the depth of u divides ρ^i but does not divide ρ^{i+1}. We keep an additional $4t_{i+1}$-shallow cutting $\mathcal{C}^i(u,l,r)$ for every i-node u and every pair $1 \le l \le r \le \rho$. For every cell C_s of $\mathcal{C}^i(u,l,r)$ we keep a $2t_{i+1}$-shallow cutting $\mathcal{D}_s$. Consider a cell D_g of $\mathcal{D}_s$. For every $(i+1)$-descendant v of u and for every pair l, r satisfying $1 \le l \le r \le \rho$, we keep the index $x = \texttt{down}(D_g, v, l, r)$ such that the cell C_x of $\mathcal{C}^{i+1}(v,l,r)$ contains D_g. We also store a mapping from $\mathcal{C}(u,l,r)$ to $\mathcal{C}^i(u,l,r)$ for every i-node u. That is, for every cell C_f of $\mathcal{C}(u,l,r)$ we keep the index $g = \texttt{cont}(f)$, such that the cell $C_g \in \mathcal{C}^i(u,l,r)$ contains C_f; for every point $p \in S(u) \cap C_f$ we keep its identifier in $C_{\texttt{cont}(f)}$. For every cell C_g of $\mathcal{C}^i(u,l,r)$ we keep a mapping from C_g to $\mathcal{D}_g$. That is, for every point p in $C_g \cap S(u,l,r)$ we store the cell D_s of $\mathcal{D}_g$ that contains p and the identifier of p in D_s. Finally we also store a mapping from every cell D_s of each $\mathcal{D}_g$ to shallow cuttings in $(i+1)$-descendants of u. For every point $p \in D_s \cap S(u,l,r)$ we store (i) the i-descendant v of u such that $p \in S(v)$ and (ii) the identifier of p in C_x where $x = \texttt{down}(D_s, v, l, r)$.

Our modified data structure uses $O(n \log^{3\varepsilon} n)$ words of space. The representation of a point in $\mathcal{C}(u,i,j)$ takes $O(\log \log n)$ bits per point and every point is stored in $O(\log^{1+2\varepsilon} n / \log \log n)$ shallow cuttings $\mathcal{C}(u,i,j)$. The mapping from $\mathcal{C}(u,l,r)$ to $\mathcal{C}^i(u,l,r)$ in an i-node u takes $O(\log^{(i+1)\varepsilon} n)$ bits per point. We also need $O(\log^{(i+1)\varepsilon} n)$ bits per point to store the mapping from a cell C_g of $\mathcal{C}^i(u,l,r)$ to $\mathcal{D}_g$. The mapping from a cell D_s of $\mathcal{D}_g$ to shallow cuttings in $(i+1)$-descendants of u consumes the same space. The total number of points in all $S(u)$ where u is an i-node is $O(n \log^{1-i\varepsilon} n)$. The total number of points in all $S(u,l,r)$ where u is an i-node and $1 \le l \le r \le \rho$ is $O(n \log^{1+(2-i)\varepsilon} n)$. Hence the total space used by all mappings in all i-nodes is $O(n \log^{1+3\varepsilon} n)$ bits or $O(n \log^{3\varepsilon} n)$ words of $\log n$ bits.

Every point p in $C_i \cap S(u,l,r)$, where C_i is a cell of $\mathcal{C}(u,l,r)$, can be decoded in $O(1)$ time. Suppose that u is a j-node. Using the mapping from C_i to $\mathcal{C}^j(u,l,r)$, we can find the representation of p in $\mathcal{C}^j(u,l,r)$, i.e., a cell C_s that contains p and the identifier of p in C_s. If we know the identifier of p in C_s, we can find the representation of p in $\mathcal{D}_s$. Using the mapping from a cell of $\mathcal{D}_s$ to $(j+1)$-descendants of u, we can compute the representation of p in a cell C_v of $\mathcal{C}^{j+1}(v,l',r')$, where v is a direct $(j+1)$-descendant of u. Thus we moved

from a j-node to its $(j+1)$-descendant in $O(1)$ time. We continue in the same way and move to a $(j+2)$-descendant of u, then a $(j+3)$-descendant of u, and so on. After at most $(1/\varepsilon) = O(1)$ iterations, we reach a leaf node and obtain the original coordinates of p.

We can translate a query $(-\infty, a] \times (-\infty, b] \times (-\infty, c]$ into the rank space of a cell C_i in constant time. Let $X(C_i)$ denote the list of x-coordinates of $S(u, l, r) \cap C_i$. We keep $X(C_i)$ in the compact trie data structure of [20]. This data structure requires $O(\log \log n)$ bits per point. Elements of $X(C_i)$ are not stored in the compact trie; we only store some auxiliary information using $O(\log \log n)$ bits per element. Compact trie supports predecessor queries on $X(C_i)$ in $O(1)$ time, but the search procedure must access $O(1)$ elements of $X(C_i)$. Since we can decode a point from C_i in $O(1)$ time, we can also access an element of $X(C_i)$ in $O(1)$ time. Hence, we can compute the predecessor of a in $X(C_i)$ (and its rank) in $O(1)$ time. We can translate b and c to the rank space in the same way.

Queries. Consider a four-dimensional 5-sided query $(-\infty, a] \times (-\infty, b] \times (-\infty, c] \times [z'_1, z'_2]$. There are $O(\log n / \log \log n)$ canonical sets $S(u_i, l_i, r_i)$, such that $p.z \in [z'_1, z'_2]$ iff $p \in S(u_i, l_i, r_i)$ for some i. Canonical sets can be found as follows. Let ℓ_1 be the leaf that holds the largest $l_1 < z'_1$ and ℓ_2 be the leaf that holds the smallest $l_2 > z'_2$. Let v denote the lowest common ancestor of ℓ_1 and ℓ_2. Let π_1 denote the path from ℓ_1 to v (excluding v) and let π_2 denote the path from ℓ_2 to v (excluding v). For each node $u \in \pi_2$, we consider a canonical set $S(u, l, r)$ such that $u_l, \ldots, u_r$ are left siblings of some node $u_{r+1} \in \pi_2$. For each node $u \in \pi_1$, we consider a canonical set $S(u, l, r)$ such that $u_l, \ldots, u_r$ are right siblings of some node $u_{l-1} \in \pi_1$. Finally we consider the set $S(v, l, r)$ such that $v_l, \ldots, v_r$ have a left sibling on π_1 and a right sibling on π_2. The fourth coordinate of a point p is in the interval $[z'_1, z'_2]$ iff p is stored in one of the canonical sets described above. Hence we need to visit all canonical sets and answer a three-dimensional query $(-\infty, a] \times (-\infty, b] \times (-\infty, c]$ in each set.

There are $O(\log n / \log \log n)$ canonical sets $S(u, l, r)$. Each canonical set is processed as follows. We find the cell C_u of $\mathcal{C}(u, l, r)$ that contains $q_3 = (a, b, c)$. Then we translate q_3 into the rank space of $C_u \cap S(u, l, r)$ and answer the dominance query. Reported points are decoded in $O(1)$ time per point as explained above. We can also translate the query into the rank space of C_u in $O(1)$ time. If q_3 is not contained in any cell of $\mathcal{C}(u, l, r)$, then q_3 dominates at least $\log^2 n$ points of $S(u, l, r)$. We visit all children u_i of u for $l \le i \le r$ and recursively answer the dominance query in each child. Using the same arguments as in Section 3, we can show that the total number of visited nodes does not exceed $O(k/\log^{\varepsilon} n)$, where k is the number of reported points.

If we replace ε with $\varepsilon/3$ in the above proof, we obtain the following result.

▶ **Theorem 5.** *There exists an $O(n \log^{\varepsilon} n)$ space data structure that answers four-dimensional 5-sided reporting queries in $O(\log n + k)$ time and four-dimensional 5-sided emptiness queries in $O(\log n)$ time.*

We can extend our result to support dominance queries (or any $(2d-3)$-sided queries) in $d \ge 4$ dimensions using standard techniques.

▶ **Theorem 6.** *There exists an $O(n \log^{d-4+\varepsilon} n)$ space data structure that supports d-dimensional dominance range reporting queries in $O(\log^{d-3} n/(\log \log n)^{d-4} + k)$ time for any constant $d \ge 4$.*
There exists an $O(n \log^{d-4+\varepsilon} n)$ space data structure that supports d-dimensional $(2d-3)$-sided range reporting queries in $O(\log^{d-3} n/(\log \log n)^{d-4} + k)$ time for any constant $d \ge 4$.

5 Conclusions

In this paper we described data structures with linear and almost-linear space usage that answer four-dimensional range reporting queries in poly-logarithmic time provided that the query range is bounded on 5 sides. This scenario includes an important special case of dominance range reporting queries that was studied in a number of previous works [16, 35, 25, 1, 11, 10]; for instance, the offline variant of four-dimensional dominance reporting is used to solve the rectangle enclosure problem [11, 2]. Our result immediately leads to better data structures in $d \geq 4$ dimensions. E.g., we can answer d-dimensional dominance range reporting queries in $O(n \log^{d-4+\varepsilon} n)$ space and $O(\log \log n (\log n / \log \log n)^{d-3})$ time. We expect that the methods of this paper can be applied to other geometric problems, such as the offline rectangle enclosure problem.

Our result demonstrates that the space complexity of four-dimensional queries is determined by the number of sides, i.e., the number of inequalities that are needed to specify the query range. This raises the question about the space complexity of dominance range reporting in five dimensions. Is it possible to construct a linear-space data structure that supports five-dimensional dominance range reporting queries in poly-logarithmic time?

Compared to the fastest previous solution for the four-dimensional dominance range reporting problem [10], our method decreases the space usage by $O(\log n)$ factor without increasing the query time. However, there is still a small gap between the $O(\log n + k)$ query time, achieved by the fastest data structures, and the lower bound of $\Omega(\log n / \log \log n)$, proved in [33]. Closing this gap is another interesting open problem.

References

1 Peyman Afshani. On dominance reporting in 3d. In *Proc. 16th Annual European Symposium on Algorithms (ESA)*, pages 41–51, 2008.

2 Peyman Afshani, Timothy M. Chan, and Konstantinos Tsakalidis. Deterministic rectangle enclosure and offline dominance reporting on the RAM. In *Proceedings of 41st International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 77–88, 2014. doi:10.1007/978-3-662-43948-7_7.

3 Pankaj K. Agarwal. Range searching. In Jacob E. Goodman and Joseph O'Rourke, editors, *Handbook of Discrete and Computational Geometry, Second Edition.*, pages 809–837. Chapman and Hall/CRC, 2004. doi:10.1201/9781420035315.ch36.

4 Pankaj K. Agarwal and Jeff Erickson. Geometric range searching and its relatives. *Contemporary Mathematics*, 223:1–56, 1999.

5 Stephen Alstrup, Gerth Stølting Brodal, and Theis Rauhe. New data structures for orthogonal range searching. In *Proc. 41st Annual Symposium on Foundations of Computer Science, (FOCS)*, pages 198–207, 2000. doi:10.1109/SFCS.2000.892088.

6 Stephen Alstrup, Gerth Stølting Brodal, and Theis Rauhe. Optimal static range reporting in one dimension. In *Proc. 33rd Annual ACM Symposium on Theory of Computing (STOC)*, pages 476–482, 2001. doi:10.1145/380752.380842.

7 Jon Louis Bentley. Multidimensional divide-and-conquer. *Communications of the ACM*, 23(4):214–229, 1980. doi:10.1145/358841.358850.

8 Jon Louis Bentley and Hermann A. Maurer. Efficient worst-case data structures for range searching. *Acta Inf.*, 13:155–168, 1980. doi:10.1007/BF00263991.

9 Timothy M. Chan. Persistent predecessor search and orthogonal point location on the word RAM. In *Proc. 22nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1131–1145, 2011. doi:10.1137/1.9781611973082.85.

10 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. doi:10.1145/2483699.2483702.

11 Timothy M. Chan, Kasper Green Larsen, and Mihai Patrascu. Orthogonal range searching on the RAM, revisited. In *Proc. 27th ACM Symposium on Computational Geometry, (SoCG)*, pages 1–10, 2011.

12 Bernard Chazelle. Filtering search: a new approach to query-answering. *SIAM Journal on Computing*, 15(3):703–724, 1986. doi:10.1137/0215051.

13 Bernard Chazelle. A functional approach to data structures and its use in multidimensional searching. *SIAM Journal on Computing*, 17(3):427–462, 1988. Preliminary version in FOCS 1985.

14 Bernard Chazelle. Lower bounds for orthogonal range searching: I. the reporting case. *J. ACM*, 37(2):200–212, 1990. doi:10.1145/77600.77614.

15 Bernard Chazelle. Lower bounds for orthogonal range searching II. the arithmetic model. *J. ACM*, 37(3):439–463, 1990. doi:10.1145/79147.79149.

16 Bernard Chazelle and Herbert Edelsbrunner. Linear space data structures for two types of range search. *Discrete & Computational Geometry*, 2:113–126, 1987. Preliminary version in SoCG 1986. doi:10.1007/BF02187875.

17 Bernard Chazelle and Leonidas J. Guibas. Fractional cascading: I. A data structuring technique. *Algorithmica*, 1(2):133–162, 1986. doi:10.1007/BF01840440.

18 Bernard Chazelle and Leonidas J. Guibas. Fractional cascading: II. applications. *Algorithmica*, 1(2):163–191, 1986. doi:10.1007/BF01840441.

19 Arash Farzan, J. Ian Munro, and Rajeev Raman. Succinct indices for range queries with applications to orthogonal range maxima. In *Proc. 39th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 327–338, 2012. doi:10.1007/978-3-642-31594-7_28.

20 Roberto Grossi, Alessio Orlandi, Rajeev Raman, and S. Srinivasa Rao. More haste, less waste: Lowering the redundancy in fully indexable dictionaries. In *Proc. 26th International Symposium on Theoretical Aspects of Computer Science, (STACS)*, pages 517–528, 2009. doi:10.4230/LIPIcs.STACS.2009.1847.

21 Joseph JáJá, Christian Worm Mortensen, and Qingmin Shi. Space-efficient and fast algorithms for multidimensional dominance reporting and counting. In *Proc. 15th International Symposium on Algorithms and Computation (ISAAC)*, pages 558–568, 2004. doi:10.1007/978-3-540-30551-4_49.

22 Marek Karpinski and Yakov Nekrich. Space efficient multi-dimensional range reporting. In *Proc. 15th Annual International Conference on Computing and Combinatorics (COCOON)*, pages 215–224, 2009. doi:10.1007/978-3-642-02882-3_22.

23 George S. Lueker. A data structure for orthogonal range queries. In *Proc. 19th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 28–34, 1978. doi:10.1109/SFCS.1978.1.

24 Edward M. McCreight. Priority search trees. *SIAM Journal on Computing*, 14(2):257–276, 1985. doi:10.1137/0214021.

25 Yakov Nekrich. A data structure for multi-dimensional range reporting. In *Proc. 23rd ACM Symposium on Computational Geometry (SoCG)*, pages 344–353, 2007. doi:10.1145/1247069.1247130.

26 Yakov Nekrich. External memory range reporting on a grid. In *Proc. 18th International Symposium on Algorithms and Computation (ISAAC)*, pages 525–535, 2007. doi:10.1007/978-3-540-77120-3_46.

27 Yakov Nekrich. Space efficient dynamic orthogonal range reporting. *Algorithmica*, 49(2):94–108, 2007. doi:10.1007/s00453-007-9030-9.

28 Yakov Nekrich. Orthogonal range searching in linear and almost-linear space. *Computational Geometry: Theory & Applications*, 42(4):342–351, 2009. doi:10.1016/j.comgeo.2008.09.001.

29 Yakov Nekrich. Orthogonal range searching on discrete grids. In *Encyclopedia of Algorithms*, pages 1484–1489. Springer, New York, 2016. doi:10.1007/978-1-4939-2864-4_631.

30 Yakov Nekrich. Four-dimensional dominance range reporting in linear space. *CoRR*, abs/2003.06742, 2020. arXiv:2003.06742.

31 Yakov Nekrich and Gonzalo Navarro. Sorted range reporting. In *Proc. 13th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT)*, pages 271–282, 2012. doi:10.1007/978-3-642-31155-0_24.

32 Mark H. Overmars. Efficient data structures for range searching on a grid. *J. Algorithms*, 9(2):254–275, 1988. doi:10.1016/0196-6774(88)90041-7.

33 Mihai Patrascu. Unifying the landscape of cell-probe lower bounds. *SIAM J. Comput.*, 40(3):827–847, 2011. doi:10.1137/09075336X.

34 Sairam Subramanian and Sridhar Ramaswamy. The P-range tree: A new data structure for range searching in secondary memory. In *Proc. 6th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 378–387, 1995.

35 Darren Erik Vengroff and Jeffrey Scott Vitter. Efficient 3-d range searching in external memory. In *Proc. 28th Annual ACM Symposium on the Theory of Computing (STOC)*, pages 192–201, 1996.

Radon Numbers Grow Linearly

Dömötör Pálvölgyi
MTA-ELTE Lendület Combinatorial Geometry Research Group, Institute of Mathematics, Eötvös Loránd University (ELTE), Budapest, Hungary
http://domotorp.web.elte.hu
dom@cs.elte.hu

Abstract

Define the k-th Radon number r_k of a convexity space as the smallest number (if it exists) for which any set of r_k points can be partitioned into k parts whose convex hulls intersect. Combining the recent abstract fractional Helly theorem of Holmsen and Lee with earlier methods of Bukh, we prove that r_k grows linearly, i.e., $r_k \leq c(r_2) \cdot k$.

2012 ACM Subject Classification Mathematics of computing → Hypergraphs; Theory of computation → Computational geometry

Keywords and phrases discrete geometry, convexity space, Radon number

Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.60

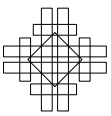
Funding *Dömötör Pálvölgyi*: Research supported by the Lendület program of the Hungarian Academy of Sciences (MTA), under grant number LP2017-19/2017.

Acknowledgements I would like to thank Boris Bukh and Narmada Varadarajan for discussions on [6], Andreas Holmsen for calling my attention to the difference between restricted and multiset Radon numbers, especially for confirming that Theorem 2 also holds for multisets, and Gábor Damásdi, Balázs Keszegh, Padmini Mukkamala and Géza Tóth for feedback on earlier versions of this manuscript, especially for fixing the computations in the proof of Lemma 3. I would also like to thank my anonymous referees for several valuable suggestions.

1 Introduction

Define a *convexity space* as a pair $(X, \mathcal{C})$, where X is any set of points and $\mathcal{C}$, the collection of convex sets, is any family over X that contains $\emptyset, X$, and is closed under (arbitrary) intersection and under (arbitrary) union of nested sets. The convex hull, $conv(S)$, of some point set $S \subset X$ is defined as the intersection of all convex sets containing S, i.e., $conv(S) = \cap\{C \in \mathcal{C} \mid S \subset C\}$; since $\mathcal{C}$ is closed under intersection, $conv(S)$ is the minimal convex set containing C. This generalization of convex sets includes several examples; for an overview, see the book by van de Vel [26] or for a more recent work, [20]. It is a natural question what properties of convex sets of $\mathbb{R}^d$ are preserved, or what the relationships are among them for general convexity spaces. A much investigated parameter is the *Radon number* r_k (sometimes also called *partition number* or *Tverberg number*), which is defined as the smallest number (if it exists) for which any set of r_k points can be partitioned into k parts whose convex hulls intersect. For $k = 2$, we simply write $r = r_2$.

In case of the convex sets of $\mathbb{R}^d$, it was shown by Radon [23] that $r = d + 2$ and by Tverberg [25] that $r_k = (d + 1)(k - 1) + 1$. Calder [7] and Eckhoff [11] raised the question whether $r_k \leq (r - 1)(k - 1) + 1$ also holds for general convexity spaces (when r exists), and this became known as Eckhoff's conjecture. It was shown by Jamison [16] that the conjecture is true if $r = 3$, and that the existence of r always implies that r_k exists and $r_k \leq r^{\lceil \log_2 k \rceil} \leq (2k)^{\log_2 r}$. His proof used the recursion $r_{k\ell} \leq r_k r_\ell$ which was later improved by Eckhoff [12] to $r_{2k+1} \leq (r - 1)(r_{k+1} - 1) + r_k + 1$, but this did not significantly change the growth rate of the upper bound. Recently Bukh [6] disproved the conjectured bound



$r_k \leq (r-1)(k-1)+1$ by showing an example where $r=4$, but $r_k \geq 3k-1$ (just one more than the conjectured value), and also improved the upper bound to $r_k = O(k^2 \log^2 k)$, where the hidden constant depends on r. We improve this to $r_k = O(k)$, which is optimal up to a constant factor and might lead to interesting applications.

▶ **Theorem 1.** *If a convexity space $(X, \mathcal{C})$ has Radon number r, then $r_k \leq c(r) \cdot k$.*

Our proof combines the methods of Bukh with recent results of Holmsen and Lee [15]. In particular, we will use the following version of the classical fractional Helly theorem [17].

▶ **Theorem 2** (Holmsen-Lee [15])**.** *For any $r \geq 3$ there is an f such that for any $\alpha > 0$ there is a $\beta > 0$ with the following property. If a convexity space $(X, \mathcal{C})$ has Radon number r, then for any finite family $\mathcal{F}$ of convex sets if at least an α fraction of the f-tuples of $\mathcal{F}$ are intersecting, then a β fraction of $\mathcal{F}$ intersects.*

There are several other connections between the parameters of a convexity space [26]; for example, earlier it was already shown [19] that in convexity spaces the Helly number is always strictly less than r (if r is finite), while in [15] it was also shown that the colorful Helly number [4] can be also bounded by some function of r (and this implied Theorem 2 combined with a combinatorial result from [14]).[1] It was also shown in [15] that it follows from the work of Alon et al. [2] that weak ε-nets [1] of size $c(\varepsilon, r)$ also exist and a (p,q)-theorem [3] also holds, so understanding these parameters better might lead to improved ε-net bounds. It remains an interesting challenge and a popular topic to find new connections among such theorems; for some recent papers studying the Radon numbers or Tverberg theorems of various convexity spaces, see [8, 9, 10, 13, 18, 22, 21, 24], while for a comprehensive survey, see Bárány and Soberón [5].

Restricted vs. multiset

In case of general convexity spaces, there are two, slightly different definitions of Radon numbers ([26]: 5.19). When we do not allow repetitions in the point set P to be partitioned, i.e., P consist of *different* points, the parameter is called the *restricted* Radon number, which we will denote by $r_k^{(1)}$. If repetitions are also allowed, i.e., we want to partition a multiset, the parameter is called the *unrestricted* or *multiset* Radon number, which we will denote by $r_k^{(m)}$. The obvious connection between these parameters is $r_k^{(1)} \leq r_k^{(m)} \leq (k-1)(r_k^{(1)}-1)+1$. In the earlier papers multiset Radon numbers were preferred, while later papers usually focused on restricted Radon numbers; we followed the spirit of the age, so the results in the Introduction were written using the definition of $r_k^{(1)}$, although some of the bounds (like Jamison's or Eckhoff's) are valid for both definitions. The proof of Theorem 1, however, also works for multisets, so we will in fact prove the stronger $r_k^{(m)} = O(k)$, and in the following simply use r_k for the multiset Radon number $r_k^{(m)}$.

A similar issue arises in Theorem 2; is $\mathcal{F}$ allowed to be a multifamily? Though not emphasized in [15], their proof also works in this case and we will use it for a multifamily. Note that this could be avoided with some cumbersome tricks, like adding more points to the convexity space without increasing the Radon number r to make all sets of a family different, but we do not go into details, as Theorem 2 anyhow holds for multifamilies.

[1] We would like to point out that a difficulty in proving these results is that the existence of a Carathéodory-type theorem is not implied by the existence of r.

2 Proof

Fix r, and a collection of points P with cardinality tk, where we allow repetitions and the cardinality is understood as the sum of the multiplicities. We will treat all points of P as if they were different even if they coincide in X, e.g., when taking subsets.

We need to show that if $t \geq c(r)$, then we can partition P into k sets whose convex hulls intersect. For a fixed constant s, define $\mathcal{F}$ to be the family of convex sets that are the convex hull of some s-element subset of P, i.e., $\mathcal{F} = \{conv(S) \mid S \subset P, |S| = s\}$. Since we treat all points of P as different, $\mathcal{F}$ will be a multifamily with $|\mathcal{F}| = \binom{tk}{s}$. We will refer to the point set S whose convex hull gave some $F = conv(S) \in \mathcal{F}$ as the *vertices* of F (despite that some of the points might be in the convex hull of the remaining ones). Note that for some $S \neq S'$, we might have $conv(S) = conv(S')$, but the vertices of $conv(S)$ and $conv(S')$ will still be S and S'; since P is a multiset, it is even possible that $S \cap S' = \emptyset$.

The constants t and s will be set to be large enough compared to some parameters that we get from Theorem 2 when we apply it to a fixed α. (Our arguments work for any $0 < \alpha < 1$.) First we set s to be large enough depending on α and r_f (where f is the fractional Helly number from Theorem 2; recall that $r_f \leq r^{\log f}$ is a constant [16]), then we set t to be large enough depending on s and β (which depends on our chosen α). In particular, we can set $s = \log(\frac{1}{1-\alpha_1}) r_f f^{fr_f}$ and $t = \max(\frac{s^2}{\beta}; \frac{(fs)^2}{k(1-\alpha_2)})$, where $0 < \alpha_1, \alpha_2 < 1$ are any two numbers such that $\alpha_1 \cdot \alpha_2 = \alpha$. Also, we note that the proof from [14, 15] gives $f \leq r^{r^{\log r}}$ and $\beta = \Omega(\alpha^{r^f})$ for Theorem 2. Combining all these to get the best bound, note that $r^f, f^{r_f} \leq R \approx r^{r^{r^{\log r}}}$. Set $\alpha = 1 - \frac{1}{R}$ with, e.g., $\alpha_1 = \alpha_2 \approx 1 - \frac{1}{2R}$. This keeps β constant, and both s and t around R, so we get an upper bound of approximately $r^{r^{r^{\log r}}}$ for t. (The simpler $\alpha_1 = \alpha_2 = \frac{1}{2}$ would give approximately $2^{r^{r^{r^{\log r}}}}$.)

Theorem 1 will be implied by the following lemma and Theorem 2.

▶ **Lemma 3.** *An α fraction of the f-tuples of $\mathcal{F}$ are intersecting.*

Proof. Since t is large enough, almost all f-tuples will be vertex-disjoint, thus it will be enough to deal with such f-tuples. More precisely, the probability of an f-tuple being vertex-disjoint is at least $(1 - \frac{fs}{tk})^{fs} \geq 1 - \frac{(fs)^2}{tk} \geq \alpha_2$ by the choice of t. We need to prove that at least an α_1 fraction of these vertex-disjoint f-tuples will be intersecting.

Partition the vertex-disjoint f-tuples into groups depending on which (fs)-element subset of P is the union of their vertices. We will show that for each group an α_1 fraction of them are intersecting. We do this by generating the f-tuples of a group uniformly at random and show that such a random f-tuple will be intersecting with probability at least α_1. For technical reasons, suppose that $m = \frac{s}{r_f}$ is an integer and partition the fs supporting points of the group randomly into m subsets of size fr_f, denoted by $V_1, \ldots, V_m$. Call an f-tuple *type* $(V_1, \ldots, V_m)$ if each set of the f-tuple intersects each V_i in r_f points. Since these V_i were picked randomly, it is enough to show that the probability that a $(V_1, \ldots, V_m)$-type f-tuple is intersecting is at least α_1.

The $(V_1, \ldots, V_m)$-type f-tuples can be uniformly generated by partitioning each V_i into f equal parts of size r_f. Therefore, it is enough to show that such a random f-tuple will be intersecting with probability at least α_1. Since $|V_i| \geq r_f$, there is at least one partition of the first r_f points of V_i to f parts whose convex hulls intersect. Since we can distribute the remaining $(f-1)r_f$ points of V_i to make all f parts equal, we get that when we partition V_i into f equal parts of size r_f, the convex hulls of these parts will intersect with probability at least $\binom{fr_f}{r_f, r_f, \ldots, r_f}^{-1} \geq f^{-fr_f}$. Since these events are independent for each i, we get that the final f-tuple will be intersecting with probability at least $1-(1-f^{-fr_f})^m \geq 1-e^{-mf^{-fr_f}} \geq \alpha_1$ by the choice of s. ◀

Therefore, if s is large enough, the conditions of Theorem 2 are met, so at least $\beta\binom{tk}{s}$ members of $\mathcal{F}$ intersect. In other words, these intersecting sets form an s-uniform hypergraph $\mathcal{H}$ on tk vertices that is β-dense. We need to show that $\mathcal{H}$ has k disjoint edges to obtain the desired partition of P into k parts with intersecting convex hulls. For a contradiction, suppose that $\mathcal{H}$ has only $k-1$ disjoint edges. Then every other edge meets one of their $(k-1)s$ vertices. There are at most $(k-1)s\binom{tk}{s-1}$ such edges, which is less than $\beta\binom{tk}{s}$ if $(k-1)s < \beta\frac{tk-s+1}{s}$, but this holds by the choice of t. This finishes the proof of Theorem 1. ◀

Concluding remarks

It is an interesting question to study how big f can be compared to r and the Helly number h of $(X, \mathcal{C})$. The current bound [15] gives $f \leq h^{r_h} \leq r^{r^{\log r}}$. We would like to point out that the first inequality, $f \leq h^{r_h}$, can be (almost) strict, as shown by the following example, similar to Example 3 (cylinders) of [20]. Let $X = \{1, \ldots, q\}^d$ be the points of a d-dimensional grid, and let $\mathcal{C}$ consist of the intersections of the axis-parallel affine subspaces with X. (Note that for $q = 2$, X will be the vertices of a d-dimensional cube, and $\mathcal{C}$ its faces.) It is easy to check that $h = 2$, $r = \lfloor \log(d+1) + 2 \rfloor$ and $f = d + 1$; the last equality follows from that for $\alpha = \frac{d!}{d^d}$ we need $\beta = \frac{1}{q}$ when $\mathcal{F}$ consists of all qd axis-parallel affine hyperplanes (if q is large enough).

It is tempting to assume that Theorem 1 would improve the second inequality, $h^{r_h} \leq r^{r^{\log r}}$, as instead of $r_h \leq r^{\log h}$ we can use $r_h = O(h)$. Unfortunately, recall that the hidden constant depended on r, in particular, it is around $r^{r^{r^{\log r}}}$. We have a suspicion that this might not be entirely sharp, so a natural question is whether this dependence could be removed to improve $r_k \leq r^{r^{r^{\log r}}} \cdot k$ to $r_k \leq c \cdot r \cdot k$. This would truly lead to an improvement of the upper bound on f in Theorem 2 and would lead to further applications [5].

References

1 N. Alon, I. Bárány, Z. Füredi, and D. J. Kleitman. Point selections and weak ε-nets for convex hulls. *Comb. Prob. Comput.*, 1:189–200, 1992.

2 N. Alon, G. Kalai, J. Matoušek, and R. Meshulam. Transversal numbers for hypergraphs arising in geometry. *Adv. in Appl. Math.*, 29:79–101, 2002.

3 N. Alon and D. J. Kleitman. Piercing convex sets and the Hadwiger-Debrunner (p, q)-problem. *Adv. Math.*, 96:103–112, 1992.

4 I. Bárány. A generalization of Carathéodory's theorem. *Disc. Math.*, 40:141–152, 1982.

5 I. Bárány and P. Soberón. Tverberg's theorem is 50 years old: a survey. *Bull. Amer. Math. Soc. (N.S.)*, 55(4):459–492, 2018.

6 B. Bukh. Radon partitions in convexity spaces. *arXiv*, 2010. `arXiv:1009.2384`.

7 J. R. Calder. Some elementary properties of interval convexities. *J. London Math. Soc.*, 2(3):422–428, 1971.

8 J. A. de Loera, R. N. La Haye, D. Rolnick, and P. Soberón. Quantitative combinatorial geometry for continuous parameters. *Discrete Comput. Geom.*, 57(2):318–334, 2017.

9 J. A. de Loera, R. N. La Haye, D. Rolnick, and P. Soberón. Quantitative Tverberg theorems over lattices and other discrete sets. *Discrete Comput. Geom.*, 58(2):435–448, 2017.

10 J. A. de Loera, T. A. Hogan, F. Meunier, and N. H. Mustafa. Tverberg theorems over discrete sets of points. *arXiv*, 2018. `arXiv:1803.01816`.

11 J. Eckhoff. Radon's theorem revisited. In *Contributions to geometry (Proc. Geom. Sympos., Siegen, 1978)*, pages 164–185, Basel, 1979. Birkhäuser.

12 J. Eckhoff. The partition conjecture. *Disc. Math.*, 221 (Selected papers in honor of Ludwig Danzer):61–78, 2000.

13 R. Fulek, B. Gärtner, A. Kupavskii, P. Valtr, and U. Wagner. The crossing Tverberg theorem. In *Symposium on Computational Geometry (SoCG 2019)*, pages 38:1–38:13, 2019.

14 A. F. Holmsen. Large cliques in hypergraphs with forbidden substructures. *Combinatorica*, 2019. accepted. `arXiv:1903.00245`.

15 A. F. Holmsen and D.-G. Lee. Radon numbers and the fractional Helly theorem. *Israel J. of Mathematics*, 2019. accepted. `arXiv:1903.01068`.

16 R. E. Jamison-Waldner. Partition numbers for trees and ordered sets. *Pacific J. Math.*, 96(1):115–140, 1981.

17 M. Katchalski and A. Liu. A problem of geometry in $\mathbb{R}^n$. *Proc. Amer. Math. Soc*, 75:284–288, 1979.

18 S. Letzter. Radon numbers for trees. *Disc. Math.*, 340:333–344, 2017.

19 F. W. Levi. On Helly's theorem and the axioms of convexity. *J. Indian Math. Soc.*, 15:65–76, 1951.

20 S. Moran and A. Yehudayoff. On weak ε-nets and the Radon number. In *Symposium on Computational Geometry (SoCG 2019)*, pages 51:1–51:14, 2019.

21 Z. Paták. Properties of closure operators in the plane. *arXiv*, 2019. `arXiv:1908.01677`.

22 P. Patáková. Bounding Radon's number via Betti numbers. *arXiv*, 2019. `arXiv:1909.08489`.

23 J. Radon. Mengen konvexer Körper, die einen gemeinsamen Punkt enthalte. *Math. Ann.*, 83:113–115, 1921.

24 P. Soberón. Tverberg partitions as weak ε-nets. *Combinatorica*, 39:447–458, 2019.

25 H. Tverberg. A generalization of Radon's theorem. *J. London Math. Soc.*, 41:123–128, 1966.

26 M. L. J. van de Vel. *Theory of convex structures*, volume 50. North-Holland Mathematical Library, Amsterdam, 1993.

Bounding Radon Number via Betti Numbers

Zuzana Patáková
Computer Science Institute, Charles University, Prague, Czech Republic
IST Austria, Klosterneuburg, Austria
zuzka@kam.mff.cuni.cz

Abstract

We prove general topological Radon-type theorems for sets in $\mathbb{R}^d$, smooth real manifolds or finite dimensional simplicial complexes. Combined with a recent result of Holmsen and Lee, it gives fractional Helly theorem, and consequently the existence of weak ε-nets as well as a (p, q)-theorem.

More precisely: Let X be either $\mathbb{R}^d$, smooth real d-manifold, or a finite d-dimensional simplicial complex. Then if $\mathcal{F}$ is a finite, intersection-closed family of sets in X such that the ith reduced Betti number (with $\mathbb{Z}_2$ coefficients) of any set in $\mathcal{F}$ is at most b for every non-negative integer i less or equal to k, then the Radon number of $\mathcal{F}$ is bounded in terms of b and X. Here k is the smallest integer larger or equal to $d/2 - 1$ if $X = \mathbb{R}^d$; $k = d - 1$ if X is a smooth real d-manifold and not a surface, $k = 0$ if X is a surface and $k = d$ if X is a d-dimensional simplicial complex.

Using the recent result of the author and Kalai, we manage to prove the following optimal bound on fractional Helly number for families of open sets in a surface: Let $\mathcal{F}$ be a finite family of open sets in a surface S such that the intersection of any subfamily of $\mathcal{F}$ is either empty, or path-connected. Then the fractional Helly number of $\mathcal{F}$ is at most three. This also settles a conjecture of Holmsen, Kim, and Lee about an existence of a (p, q)-theorem for open subsets of a surface.

2012 ACM Subject Classification Mathematics of computing → Geometric topology; Theory of computation → Computational geometry

Keywords and phrases Radon number, topological complexity, constrained chain maps, fractional Helly theorem, convexity spaces

Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.61

Related Version A full version of the paper is available at `https://arxiv.org/abs/1908.01677`.

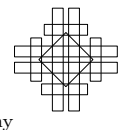
Funding The research stay of the author at IST Austria is funded by the project Improvement of internationalization in the field of research and development at Charles University, through the support of quality projects MSCA-IF (CZ.02.2.69/0.0/0.0/17_050/0008466).

Acknowledgements First and foremost, I am very grateful to Pavel Paták for numerous discussions, helpful suggestions and a proofreading. Many thanks to Xavier Goaoc for his feedback and comments, which have been very helpful in improving the overall presentation. I also thank Endre Makai for pointers to relevant literature, especially to the book [17]. Finally, many thanks to Natan Rubin for several discussions at the very beginning of the project.

1 Introduction

The classical Radon's theorem [15] states that it is possible to split any $d + 2$ points in $\mathbb{R}^d$ into two disjoint parts whose convex hulls intersect. It is natural to ask what happens to the statement, if one starts varying the notion of convexity.

Perhaps the most versatile generalization of the convex hull is the following. Let X be an underlying set and let $\mathcal{F}$ be a finite family of subsets of X. Let $S \subseteq X$ be a set. The *convex hull* $\mathrm{conv}_{\mathcal{F}}(S)$ *of* S *relative to* $\mathcal{F}$ is defined as the intersection of all sets from $\mathcal{F}$ that contain S. If there is no such set, the convex hull is, by definition, X. If $\mathrm{conv}_{\mathcal{F}} S = S$, the set S is called $\mathcal{F}$*-convex*.




This definition is closely related to so called *convexity spaces*[1], as defined for example in [18, 2, 17]. The only difference is that most authors require that in a convexity space $\operatorname{conv}\emptyset = \emptyset$, which is not needed in any of our considerations. Moreover, it can be easily forced by including $\emptyset$ to $\mathcal{F}$.

In our examples we are also going to use the definition of $\operatorname{conv}_{\mathcal{F}}$ for the family $\mathcal{F}$ of all (standard) convex sets in $\mathbb{R}^d$. We note that in this case $\operatorname{conv}_{\mathcal{F}}$ coincides with the standard convex hull.

We say that $\mathcal{F}$ has *Radon number* $r(\mathcal{F})$ if $r(\mathcal{F})$ is the smallest integer r such that any set $S \subseteq X$ of cardinality r can be split into two parts $S = P_1 \sqcup P_2$ satisfying $\operatorname{conv}_{\mathcal{F}}(P_1) \cap \operatorname{conv}_{\mathcal{F}}(P_2) \neq \emptyset$. If no such r exists, we put $r(\mathcal{F}) = \infty$. We note that Radon number is anti-monotone in the sense that $r(\mathcal{F}) \leq r(\mathcal{G})$ for $\mathcal{G} \subseteq \mathcal{F}$.

In this paper we show that very mild topological conditions are enough to force a bound on Radon number for sets in Euclidean space (Theorem 1). A simple trick allows us to give a version of the result for smooth manifolds or simplicial complexes, see Section 2.1. Furthermore, the proof technique also works for surfaces (Theorem 2). In Section 2.2 we list some important consequences, most notably a fractional Helly theorem (Theorem 3), which allows us to solve a conjecture of Holmsen, Kim, and Lee (a special case of Theorem 6).

2 New results

One can observe that bounded Radon number is not a property of a standard convexity since it is preserved by topological deformations of $\mathbb{R}^d$. In fact, we can even show that if the family $\mathcal{F}$ is *"not too topologically complicated"*, its Radon number is bounded. Let us first explain what "not too topologically complicated" means.

Topological complexity. Let $k \geq 1$ be an integer or ∞ and $\mathcal{F}$ a family of sets in a topological space X. We define the *k-level topological complexity* of $\mathcal{F}$ as:

$$\sup\left\{\widetilde{\beta}_i\left(\bigcap\mathcal{G};\mathbb{Z}_2\right) : \mathcal{G} \subseteq \mathcal{F}, 0 \leq i < k\right\}$$

and denote it by $TC_k(\mathcal{F})$. We call the number $TC_\infty(\mathcal{F})$ the *(full) topological complexity*.

Examples. Let us name few examples of families with bounded topological complexity: family of convex sets in $\mathbb{R}^d$, good covers[2], families of spheres and pseudospheres in $\mathbb{R}^d$, finite families of *semialgebraic sets* in $\mathbb{R}^d$ defined by a constant number of polynomial inequalities, where all polynomials have a constant degree, etc.

We can now state our main theorem.

▶ **Theorem 1** (Bounded mid-level topological complexity implies Radon)**.** *For every non-negative integers b and d there is a number $r(b,d)$ such that the following holds: If $\mathcal{F}$ is a finite family of sets in $\mathbb{R}^d$ with $TC_{\lceil d/2\rceil}(\mathcal{F}) \leq b$, then $r(\mathcal{F}) \leq r(b,d)$.*

Qualitatively, Theorem 1 is sharp in the sense that all (reduced) Betti numbers $\widetilde{\beta}_i$, $0 \leq i < \lceil d/2\rceil$, need to be bounded in order to obtain a bounded Radon number, see [7, Example 3].

---

[1] A pair $(X,\mathcal{C})$ is called a *convexity space* on X if $\mathcal{C} \subset 2^X$ is a family of subsets of X such that $\emptyset, X \in \mathcal{C}$ and $\mathcal{C}$ is closed under taking intersections; and unions of chains. The sets in $\mathcal{C}$ are called *convex*. Note that the last condition is trivially satisfied whenever $\mathcal{C}$ is finite.

[2] A family of sets in $\mathbb{R}^d$ where intersection of each subfamily is either empty or contractible.

2.1 Embeddability

We have seen that for a finite family of sets in $\mathbb{R}^d$, in order to have a bounded Radon number, it suffices to restrict the reduced Betti numbers up to $\lceil d/2 \rceil - 1$. Which Betti numbers do we need to restrict, if we replace $\mathbb{R}^d$ by some other topological space X? The following paragraphs provide some simple bounds if X is a simplicial complex or a smooth real manifold. The base for the statements is the following simple observation: Given a topological space X embeddable into $\mathbb{R}^d$, we may view any subset of X as a subset of $\mathbb{R}^d$ and use Theorem 1.

Since any (finite) k-dimensional simplicial complex embeds into $\mathbb{R}^{2k+1}$, we have:

> If K is a (finite) k-dimensional simplicial complex and $\mathcal{F}$ is a finite family of sets in K with $TC_{k+1}(\mathcal{F}) \leq b$, then $r(\mathcal{F}) \leq r(b, 2k+1)$.

Again, this bound is qualitatively sharp in the sense that all $\widetilde{\beta}_i$, $0 \leq i \leq k$, need to be bounded in order to have a bounded Radon number, see [7, Example 3].

Using the strong Whitney's embedding theorem [19], stating that any smooth real k-dimensional manifold embeds into $\mathbb{R}^{2k}$, we obtain the following:

> If M is a smooth k-dimensional real manifold and $\mathcal{F}$ is a finite family of sets in M with $TC_k(\mathcal{F}) \leq b$, then $r(\mathcal{F}) \leq r(b, 2k)$.

Unlike in the previous statements we do not know whether bounding all reduced Betti numbers $\widetilde{\beta}_i$, $0 \leq i \leq k-1$, is necessary. The following result about surfaces indicates that it possibly suffices to bound less. Let $\mathcal{F}$ be a finite family $\mathcal{F}$ of sets in a surface[3] S. In order to have a finite Radon number $r(\mathcal{F})$, it is enough to require that $TC_1(\mathcal{F})$ is bounded, that is, it only suffices to have a universal bound on the number of connected components.

▶ **Theorem 2.** *For each surface S and each integer $b \geq 0$ there is a number $r_S(b)$ such that each finite family $\mathcal{F}$ of sets in S satisfying $TC_1(\mathcal{F}) \leq b$ has $r(\mathcal{F}) \leq r_S(b)$.*

See Section 3.2 for the proof.

However, at the present time the author does not know how to generalize this result to higher dimensional manifolds. Given a d-dimensional manifold M, it is an open question whether $r(\mathcal{F})$ is bounded for all families $\mathcal{F} \subseteq M$ with bounded $TC_{\lceil d/2 \rceil}(\mathcal{F})$.

2.2 Consequences and related results

We say that $\mathcal{F}$ has *Helly number* $h(\mathcal{F})$, if $h(\mathcal{F})$ is the smallest integer h with the following property: If in a finite subfamily $\mathcal{S} \subseteq \mathcal{F}$ each h members of $\mathcal{S}$ have a point in common, then all the sets of $\mathcal{S}$ have a point in common. If no such h exists, we put $h(\mathcal{F}) = \infty$. By older results, bounded Radon number implies bounded Helly number [12] as well as bounded Tverberg numbers[4] [10, (6)]. From these consequences only the fact that for sets in $\mathbb{R}^d$ bounded $TC_{\lceil d/2 \rceil}$ implies bounded Helly number has been shown earlier [7].

Due to recent results by Holmsen and Lee, bounded Radon number implies colorful Helly theorem [9, Lemma 2.3] and bounded fractional Helly number [9, Theorem 1.1]. Thus, in combination with Theorem 1 and the results from the previous section, we have obtained the following fractional Helly theorem.

---

[3] By a *surface* we mean a compact two-dimensional real manifold.

[4] Given an integer $k \geq 3$, we say that $\mathcal{F}$ has k^{th} *Tverberg number* $r_k(\mathcal{F})$, if $r_k(\mathcal{F})$ is the smallest integer r such that any set $S \subseteq X$ of size r_k can be split into k parts $S = P_1 \sqcup P_2 \sqcup \ldots \sqcup P_k$ satisfying $\bigcap_{i=1}^{k} \mathrm{conv}_{\mathcal{F}} P_i \neq \emptyset$. We set $r_k(\mathcal{F}) = \infty$ if there is no such r_k.

▶ **Theorem 3.** *Let X be either $\mathbb{R}^d$, in which case we set $k = \lceil d/2 \rceil$, or a smooth real d-dimensional manifold, $d \geq 3$, in which case we set $k = d$, or a surface, in which case we set $k = 1$, or a (finite) d-dimensional simplicial complex, in which case we set $k = d + 1$. Then for every integer $b \geq 0$ there is a number $h_f = h_f(b, X)$ such that the following holds. For every $\alpha \in (0, 1]$ there exists $\beta = \beta(\alpha, b, X) > 0$ with the following property. Let $\mathcal{F}$ be a family of sets in X with $TC_k(\mathcal{F}) \leq b$ and $\mathcal{G}$ be a finite family of $\mathcal{F}$-convex sets, having at least an α fraction of the h_f-tuples with non-empty intersection, then there is a point contained in at least $\beta|\mathcal{G}|$ sets of $\mathcal{G}$.*

We note that Theorem 3 can be applied to many spaces X that are often encountered in geometry. Let us mention $\mathbb{R}^d$, Grassmanians, or flag manifolds.

We refer to the number h_f from the theorem as the *fractional Helly number*. Bounded fractional Helly number in turn provides a weak ε-net theorem [1] and a (p, q)-theorem [1]. The existence of a fractional Helly theorem for sets with bounded topological complexity might be seen as the most important application of Theorem 1, not only because it implies an existence of weak ε-nets and a (p, q)-theorem, but also in its own right. Its existence answers positively a question by Matoušek (personal communication), also mentioned in [3, Open Problem 3.6].

The bound on h_f we obtain from the proof is not optimal. So what is the optimal bound? The case of $(d-1)$-flats in $\mathbb{R}^d$ in general position shows that we cannot hope for anything better than $d + 1$. In Section 4 we establish a reasonably small bound for a large class of families $\mathcal{F}$ of open subsets of surfaces using a bootstrapping method based on the result of the author and Kalai [11]. In particular, for families $\mathcal{F}$ of open sets with $TC_1(\mathcal{F}) = 0$, we obtain the optimal bound.

▶ **Theorem 4** (Fractional Helly for surfaces)**.** *Let $b \geq 0$ be an integer. We set $k = 3$ for $b = 0$ and $k = 2b + 4$ for $b \geq 1$, respectively. Then for any surface S and $\alpha \in (0, 1)$ there exists $\beta = \beta(\alpha, b, S) > 0$ with the following property. Let $\mathcal{A}$ be a family of n open subsets of a surface S with $TC_1(\mathcal{A}) \leq b$. If at least $\alpha\binom{n}{k}$ of the k-tuples of $\mathcal{A}$ are intersecting, then there is intersecting subfamily of $\mathcal{A}$ of size at least βn.*

We note that the statement holds also for a family of open sets in $\mathbb{R}^2$, since the plane can be seen as an open subset of a 2-dimensional sphere.

The author conjectures that k in Theorem 4 is independent of b, more precisely, the conjectured value is three. The author also conjectures that the fractional Helly number for families in $\mathbb{R}^d$ is $d + 1$.

▶ **Conjecture 5.** *For any integers $b \geq 1, d \geq 2$ and $\alpha \in (0, 1)$ there exists $\beta = \beta(\alpha, b, d) > 0$ with the following property. Let $\mathcal{A}$ be a family of $n \geq d + 1$ sets in $\mathbb{R}^d$ with $TC_{\lceil d/2 \rceil}(\mathcal{A}) \leq b$. If at least $\alpha\binom{n}{d+1}$ of the $(d + 1)$-tuples of $\mathcal{A}$ intersect, then there is an intersecting subfamily of $\mathcal{A}$ of size at least βn.*

The proof of Theorem 4 is given in Section 4. By the results in [1], the fractional Helly theorem is the only ingredient needed to prove a (p, q)-theorem, hence combining Theorem 4 with results in [1] immediately gives Theorem 6. Let us recall that a family $\mathcal{F}$ of sets has the (p, q)-*property* if among every p sets of $\mathcal{F}$, some q have a point in common.

▶ **Theorem 6.** *Let $b \geq 0$ be an integer. Set $k = 3$ for $b = 0$ and $k = 2b + 4$ for $b \geq 1$, respectively. For any integers $p \geq q \geq k$ and a surface S, there exists an integer $C = C(p, q, S)$ such that the following holds. Let $\mathcal{F}$ be a finite family of open subsets of S with $TC_1(\mathcal{F}) \leq b$. If $\mathcal{F}$ has the (p, q)-property, then there is a set X that intersects all sets from $\mathcal{F}$ and has at most C elements.*

The case $b = 0$ in Theorem 6 settles a conjecture by Holmsen, Kim, and Lee [8, Conj. 5.3].

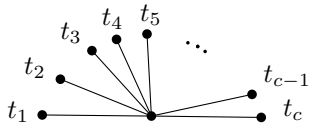
We have seen that bounded topological complexity has many interesting consequences. However, there is one parameter of $\mathcal{F}$ that cannot be bounded by the topological complexity alone. We say that $\mathcal{F}$ has *Carathéodory number* $c(\mathcal{F})$, if c is the smallest integer c with the following property: For any set $S \subseteq X$ and any point $x \in \mathrm{conv}_{\mathcal{F}}(S)$, there is a subset $S' \subseteq S$ of size at most c such that $x \in \mathrm{conv}_{\mathcal{F}}(S')$. If no such c exists, we put $c(\mathcal{F}) = \infty$.

It is easy to construct an example of a finite $\mathcal{F}$ of bounded full-level topological complexity with arbitrarily high Carathéodory's number.

▶ **Theorem 7** (Bounded topological complexity does not imply Carathéodory)**.** *For every positive integers $c \geq 2$ and $d \geq 2$ there is a finite family $\mathcal{F}$ of sets in $\mathbb{R}^d$ of full-level topological complexity zero, satisfying $c(\mathcal{F}) = c$.*

Proof. Indeed, consider a star with c spines $T_1, T_2, \ldots, T_c$ each containing a point t_i. Let $A_i := \bigcup_{j \neq i} T_j$ and $\mathcal{F} = \{A_1, A_2, \ldots, A_c\}$.

Then any intersection of the sets A_i is contractible, and hence topologically trivial. Let $S = \{t_1, \ldots, t_c\}$. Observe that $\mathrm{conv}_{\mathcal{F}} S = \mathbb{R}^d$. Let x be any point in $(\mathrm{conv}_{\mathcal{F}} S) \setminus \bigcup_{i=1}^{c} A_i$. Then $x \in \mathrm{conv}_F S$, and $x \notin \mathrm{conv}_F S'$ for any $S' \subsetneq S$. Thus $c(\mathcal{A}) = c$.



◀

3 Technique

The introduction of relative convex hulls allows us to strengthen and polish the techniques developed in [7]. Independently of these changes we also manage to separate the combinatorial and topological part of the proof, which improves the overall exposition. We start with the topological tools (Sections 3.1 and 3.2) including the proof of Theorem 1 modulo Proposition 13. We divide the proof of the main ingredient (Proposition 13) into two parts: Ramsey-type result (Section 3.3) and induction (Section 3.4).

Notation & convention. For an integer $n \geq 1$, let $[n] = \{1, \ldots, n\}$. If P is a set, we use the symbol 2^P to denote the set of all its subsets and $\binom{P}{n}$ to denote the family of all n-element subsets of P. We denote by Δ_n the standard n-dimensional simplex. If K is a simplicial complex, $V(K)$ stands for its set of vertices and $K^{(k)}$ stands for its k-dimensional skeleton, i.e. the subcomplex formed by all its faces of dimension up to k. Unless stated otherwise, we only work with abstract simplicial complexes.[5] All chain groups and chain complexes are considered with $\mathbb{Z}_2$-coefficients.

3.1 Homological almost embeddings

Homological almost embeddings are the first ingredient we need. Before defining them, let us first recall (standard) almost embeddings. Let $\mathbf{R}$ be a topological space.

[5] The definition of singular homology forces us to use the geometric standard simplex Δ_n on some places.

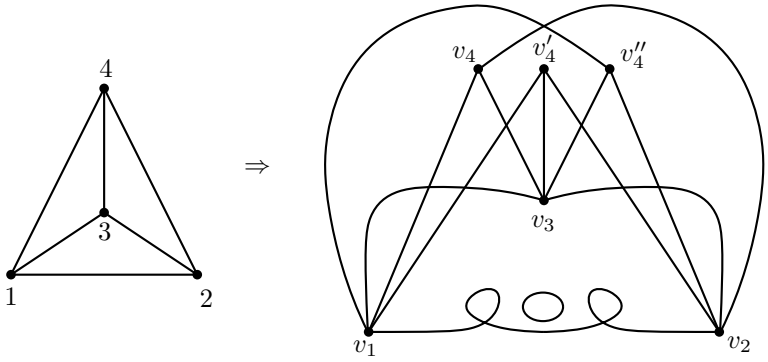


Figure 1 An example of a homological almost-embedding of K_4 into the plane.

▶ **Definition 8.** *Let K be an (abstract) simplicial complex with geometric realization $|K|$ and $\mathbf{R}$ a topological space. A continuous map $f\colon |K| \to \mathbf{R}$ is an* almost-embedding *of K into $\mathbf{R}$, if the images of disjoint simplices are disjoint.*

▶ **Definition 9.** *Let K be a simplicial complex, and consider a chain map $\gamma\colon C_*(K;\mathbb{Z}_2) \to C_*(\mathbf{R};\mathbb{Z}_2)$ from the simplicial chains in K to singular chains in $\mathbf{R}$.*

- **(i)** *The chain map γ is called* nontrivial[6] *if the image of every vertex of K is a finite set of points in $\mathbf{R}$ (a 0-chain) of* odd *cardinality.*
- **(ii)** *The chain map γ is called a* homological almost-embedding *of K in $\mathbf{R}$ if it is nontrivial and if, additionally, the following holds: whenever σ and τ are disjoint simplices of K, their image chains $\gamma(\sigma)$ and $\gamma(\tau)$ have disjoint supports, where the support of a chain is the union of (the images of) the singular simplices with nonzero coefficient in that chain.*

In analogy to almost-embeddings, there is no homological almost-embedding of the k-skeleton of $(2k+2)$-dimensional simplex into $\mathbb{R}^{2k}$:

▶ **Theorem 10** (Corollary 13 in [7])**.** *For any $k \geq 0$, the k-skeleton $\Delta^{(k)}_{2k+2}$ of the $(2k+2)$-dimensional simplex has no homological almost-embedding in $\mathbb{R}^{2k}$.*

Let us say a few words about the proof. It is based on the standard cohomological proof of the fact that $\Delta^{(k)}_{2k+2}$ does not "almost-embed" into $\mathbb{R}^{2k}$ and combined with the fact that cohomology "does not distinguish" between maps and non-trivial chain maps. For details see [7].

3.2 Constrained chain maps

We continue developing the machinery from [7] in order to capture our more general setting. To prove Theorem 1, we need one more definition (Definition 11). A curious reader may compare our definition of constrained chain map with the definition from [7]. Let us just remark that the definition presented here is more versatile. (Although it might not be obvious on the first sight.) Unlike the previous definition, the current form allows us to prove the bound on the Radon number. Nevertheless, both definitions are equivalent under some special circumstances.

[6] If we consider augmented chain complexes with chain groups also in dimension -1, then being nontrivial is equivalent to requiring that the generator of $C_{-1}(K) \cong \mathbb{Z}_2$ (this generator corresponds to the empty simplex in K) is mapped to the generator of $C_{-1}(\mathbf{R}) \cong \mathbb{Z}_2$.

Let $\mathbf{R}$ be a topological space, let K be a simplicial complex and let $\gamma : C_*(K) \to C_*(\mathbf{R})$ be a chain map from the simplicial chains of K to the singular chains of $\mathbf{R}$.

▶ **Definition 11** (Constrained chain map). *Let $\mathcal{F}$ be a finite family of sets in $\mathbf{R}$ and P be a (multi-)set*[7] *of points in $\mathbf{R}$. Let $\gamma : C_*(K) \to C_*(\mathbf{R})$ be the aforementioned chain map. We say that γ is* constrained by $(\mathcal{F}, \Phi)$ *if:*

- **(i)** *Φ is a map from K to 2^P such that $\Phi(\sigma \cap \tau) = \Phi(\sigma) \cap \Phi(\tau)$ for all $\sigma, \tau \in K$ and $\Phi(\emptyset) = \emptyset$.*
- **(ii)** *For any simplex $\sigma \in K$, the support of $\gamma(\sigma)$ is contained in $\operatorname{conv}_{\mathcal{F}} \Phi(\sigma)$.*

If there is some Φ such that a chain map γ from K is constrained by $(\mathcal{F}, \Phi)$, we say that γ is constrained by $(\mathcal{F}, P)$.

We can now prove an analogue of Lemma 26 from [7] and relate constrained maps and homological almost embeddings.

▶ **Lemma 12.** *Let $\gamma : C_*(K) \to C_*(\mathbf{R})$ be a nontrivial chain map constrained by $(\mathcal{F}, P)$. If $\operatorname{conv}_{\mathcal{F}} S \cap \operatorname{conv}_{\mathcal{F}} T = \emptyset$ whenever $S \subseteq P$ and $T \subseteq P$ are disjoint, then γ is a homological almost-embedding of K to $\mathbf{R}$.*

Proof. Let σ and τ be two disjoint simplices of K. The supports of $\gamma(\sigma)$ and $\gamma(\tau)$ are contained, respectively, in $\operatorname{conv}_{\mathcal{F}} \Phi(\sigma)$ and $\operatorname{conv}_{\mathcal{F}} \Phi(\tau)$. By the definition of Φ, $\Phi(\sigma)$ and $\Phi(\tau)$ are disjoint. Thus, by the assumption

$$\operatorname{conv}_{\mathcal{F}} \Phi(\sigma) \cap \operatorname{conv}_{\mathcal{F}} \Phi(\tau) = \emptyset.$$

Therefore, γ is a homological almost-embedding of K. ◀

The most important ingredient for the proof of Theorem 1 is the following proposition:

▶ **Proposition 13.** *For any finite simplicial complex K and a non-negative integer b there exists a constant $r_K(b)$ such that the following holds. For any finite family $\mathcal{F}$ in $\mathbf{R}$ with $TC_{\dim K}(\mathcal{F}) \leq b$ and a set P of at least $r_K(b)$ points in $\mathbf{R}$ there exists a nontrivial chain map $\gamma : C_*(K) \to C_*(\mathbf{R})$ that is constrained by $(\mathcal{F}, P)$.*

Furthermore, if $\dim K \leq 1$, one can even find such γ that is induced by some continuous map $f : |K| \to \mathbf{R}$ from the geometric realization $|K|$ of K to $\mathbf{R}$.

Before proving Theorems 1 and 2, let us relate Proposition 13 to the Radon number.

▶ **Proposition 14.** *Let $\mathbf{R}$ be a topological space and K a simplicial complex that does not homologically embed into $\mathbf{R}$. Then for each integer $b \geq 0$ and each finite family $\mathcal{F}$ of sets in $\mathbf{R}$ satisfying $TC_{\dim K}(\mathcal{F}) \leq b$, one has $r(\mathcal{F}) \leq r_K(b)$, where $r_K(b)$ is as in Proposition 13.*

Moreover, if $\dim K \leq 1$, it suffices to assume that K does not almost embed into $\mathbf{R}$.

Proof. If $r(\mathcal{F}) > r_K(b)$, then there is a set P of $r_K(b)$ points such that for any two disjoint subsets $P_1, P_2 \subseteq P$ we have $\operatorname{conv}_{\mathcal{F}}(P_1) \cap \operatorname{conv}_{\mathcal{F}}(P_2) = \emptyset$. Let $\gamma : C_*(K) \to C_*(\mathbf{R})$ be a nontrivial chain map constrained by $(\mathcal{F}, P)$ given by Proposition 13. By Lemma 12, γ is a homological almost-embedding of K, a contradiction.

[7] However, the switch to multisets requires some minor adjustments. If $P = \{p_i \mid i \in I\}$ is a multiset, one needs to replace the multiset P by the index set I in all definitions and proofs; and if $J \subseteq I$ consider $\operatorname{conv}_{\mathcal{F}}(J)$ as a shorthand notation for $\operatorname{conv}_{\mathcal{F}}(\{p_i \mid i \in J\})$. However, we have decided not to clutter the main exposition with such technical details.

If $\dim K \leq 1$, one can take γ to be induced by a continuous map $f\colon |K| \to \mathbf{R}$. However, one can easily check that in that case γ is a homological almost embedding if and only if f is an almost embedding. ◀

Theorems 1 and 2 are now immediate consequences of Proposition 14.

Proof of Theorem 1. Let $k = \lceil d/2 \rceil$. By Theorem 10, $\Delta^{(k)}_{2k+2}$ does not homologically almost embeds into $\mathbb{R}^d$, so Proposition 14 applies and yields Theorem 1. ◀

Proof of Theorem 2. By results in [6], for each surface S there is a finite graph G that does not almost embed[8] into S, so Proposition 14 applies. ◀

3.3 Combinatorial part of the proof

The classical Ramsey theorem [16] states that for all positive integers k, n and c there is a number $R_k(n;c)$ such that the following holds. For each set X satisfying $|X| \geq R_k(n;c)$ and each coloring[9] $\rho\colon \binom{X}{k} \to [c]$, there is a *monochromatic* subset $Y \subseteq X$ of size n, where a subset Y is monochromatic, if all k-tuples in Y have the same color. Note that the case $k = 1$ corresponds to the pigeon hole principle and $R_1(n;c) = n(c-1)+1$.

In order to perform the induction step in the proof of Proposition 13, we need the following Ramsey type theorem.

▶ **Proposition 15.** *For any positive integers k, m, n, c there is a constant $N_k = N_k(n;m;c)$ such that the following holds. Let X be a set and for every $V \subseteq X$ let $\rho_V\colon \binom{V}{k} \to [c]$ be a coloring[10] of the k-element subsets of V. If $|X| \geq N_k$, then there always exists an n-element subset $Y \subseteq X$ and a map $M_{(\cdot)}\colon \binom{Y}{m} \to 2^{X\setminus Y}$ such that all sets M_Z for $Z \in \binom{Y}{m}$ are disjoint, and each $Z \in \binom{Y}{m}$ is monochromatic in $\rho_{Z\cup M_Z}$.*

The fact that each k-tuple is colored by several different colorings ρ_V reflects the fact that we are going to color a cycle z by the singular homology of $\gamma(z)$ inside $\operatorname{conv}_{\mathcal{F}} \Phi(V)$ for various different sets V. There, it may easily happen that z and z' have the same color in V but different in V'.

Proof. Let $r = R_k(m;c)$. We claim that it is enough to take

$$N_k = R_r\left(n + \binom{n}{m}\cdot(r-m); \binom{r}{m}\right).$$

Suppose that $|X| \geq N_k$ and choose an arbitrary order of the elements of X.

By the choice of r, if $V \in \binom{X}{r}$, then there is a subset $A \subseteq V$ of size m such that ρ_V assigns the same color to all k-tuples in A. Let us introduce another coloring, $\eta\colon \binom{X}{r} \to \binom{[r]}{m}$, that colors each $V \in \binom{X}{r}$ by the relative[11] position of the first monochromatic A inside V (with respect to the lexicographic ordering).

By the definition of N_k and the fact that $|X| \geq N_k$, there is a subset U of size $n + \binom{n}{m}\cdot(r-m)$, such that all r-tuples in U have the same color in η, say color Ω.

---

[8] Compared to [6], recent works by Paták, Tancer [14], and Fulek, Kynčl [5] provide much smaller graphs which are not almost-embeddable into S.

[9] A coloring is just another name for a map. However, it is easier to say "the color of z", instead of "the image of z under ρ".

[10] If $|V| < k$, the coloring c_V is, by definition, the empty map.

[11] For illustration: If $V = \{2,4,6,8,\ldots,36\}$ and $A = \{2,4,34,36\}$ we assign V the "color" $\{1,2,17,18\}$, since the elements of A are on first, second, 17th and 18th position of V.

Consider the set $Y' = \{1, 2, \dots, n\}$. Since the rational numbers are dense, we can find an assignment

$$\begin{array}{rccc} N\colon & \binom{Y'}{m} & \to & \binom{\mathbb{Q}\setminus Y'}{r-m} \\ & Z' & \mapsto & N_{Z'} \end{array}$$

of mutually disjoint sets $N_{Z'}$ such that Z' is on the position Ω inside $Z' \cup N_{Z'}$.

The unique order-preserving isomorphism from $Y' \cup \bigcup N_{Z'}$ to U then carries Y' to the desired set Y and $N_{Z'}$ to the desired sets M_Z. ◀

3.4 The induction

Proof of Proposition 13. We proceed by induction on $\dim K$, similarly as in [7]. If the reader finds the current exposition too fast, we encourage him/her to consult [7] which goes slower and shows motivation and necessity of some ideas presented here. Note however, that our current setup is much more general.

Induction basis. If K is 0-dimensional with vertices $V(K) = \{v_1, \dots, v_m\}$, we set $r_K(b) = m$. If $P = \{x_1, \dots, x_n\}$ is a point set in $\mathbf{R}$ with $|P| \geq m$, we can take as Φ the map $\Phi(v_i) = \{x_i\}$. It remains to define γ. We want it to "map" v_i to x_i. However, γ should be a chain map from simplicial chains of K to singular chains in $\mathbb{R}^d$. Therefore for each vertex v_i we define $\gamma(v_i)$ as the unique map from[12] Δ_0 to x_i; and extend this definition linearly to the whole $C_0(K)$. By construction, γ is nontrivial and constrained by $(\mathcal{F}, \Phi)$.

Induction step. Let $\dim K = k \geq 1$. The aim is to find a chain map $\gamma\colon C_*(K^{(k-1)}) \to C_*(\mathbf{R})$ and a suitable map Φ such that γ is nontrivial, constrained by $(\mathcal{F}, \Phi)$ and $\gamma(\partial\sigma)$ has trivial homology inside $\operatorname{conv}_{\mathcal{F}} \Phi(\sigma)$ for each k-simplex $\sigma \in K$. Extending such γ to the whole complex K is then straightforward.

Let $s \geq 1$ be some integer depending on K which we determine later. To construct γ we will define three auxiliary chain maps

$$C_*\left(K^{(k-1)}\right) \xrightarrow{\alpha} C_*\left((\operatorname{sd} K)^{(k-1)}\right) \xrightarrow{\beta} C_*\left(\Delta_s^{(k-1)}\right) \xrightarrow{\gamma'} C_*(\mathbf{R}),$$

where $\operatorname{sd} K$ is the barycentric subdivision[13] of K.

Definition of α. We start with the easiest map, α. It maps each l-simplex σ from $K^{(k-1)}$ to the sum of the l-simplices in the barycentric subdivision of σ.

Definition of γ'. The map γ' is obtained from induction. Let the cardinality of P be large enough. Since $\dim \Delta_s^{(k-1)} = k-1$, by induction hypothesis, there is a nontrivial chain map $\gamma' : C_*(\Delta_s^{(k-1)}) \to C_*(\mathbf{R})$ and a map $\Psi\colon \Delta_s^{(k-1)} \to 2^P$ such that γ' is constrained by $(\mathcal{F}, \Psi)$.

In order to define Φ easily, we need to extend Ψ to Δ_s, hence for $\sigma \in \Delta_s$ we define

$$\Psi(\sigma) = \bigcup_{\tau \in \Delta_s^{(k-1)}, \tau \subseteq \sigma} \Psi(\tau). \tag{1}$$

[12] This is the only place where Δ_n is considered to be a geometric simplex.

[13] The *barycentric subdivision* $\operatorname{sd} K$ of an abstract simplicial complex K is the complex formed by all the chains contained in the partially ordered set $(K \setminus \{\emptyset\}, \subseteq)$, so called the *order complex* of $(K \setminus \{\emptyset\}, \subseteq)$.

If $\tau \subseteq \sigma \in \Delta_s^{(k-1)}$, then $\Psi(\tau) \cap \Psi(\sigma)$ was equal to $\Psi(\tau \cap \sigma) = \Psi(\tau)$. Thus the equality (1) does not change the value of $\Psi(\sigma)$ if $\sigma \in \Delta_s^{(k-1)}$ and it is indeed an extension of Ψ. Moreover, easy calculation shows that $\Psi(A) \cap \Psi(B) = \Psi(A \cap B)$ for any $A, B \in \Delta_s$.

Definition of β. With the help of Proposition 15 it is now easy to find the map β. Indeed, for each simplex $\tau \in \Delta_s$, let c_τ be the coloring that assigns to each k-simplex $\sigma \subseteq \tau$ the singular homology class of $\gamma'(\partial\sigma)$ inside $\operatorname{conv}_{\mathcal{F}}(\Psi(\tau))$. Let m be the number of vertices of $\operatorname{sd}\Delta_k$, n the number of vertices of $\operatorname{sd} K$ and c the maximal number of elements in $\widetilde{H}_k(\bigcap \mathcal{G}; \mathbb{Z}_2)$, where $\mathcal{G} \subseteq \mathcal{F}$. Clearly $c \leq 2^b$.

Thus if $s \geq N_{k+1}(n; m; c)$ from Proposition 15, the following holds.

(a) There is an inclusion j of $(\operatorname{sd} K)^{(k-1)}$ to a simplex $Y \subseteq \Delta_s$. We let $\varphi\colon K \to 2^{V(\Delta_s)}$ be the map that to each $\sigma \in K$ assigns the set $j(V(\operatorname{sd}\sigma))$.

(b) For each k-simplex μ in K there is a simplex M_μ in Δ_s with the following three properties:

- **(i)** For all k-simplices τ inside $\operatorname{sd}\mu$, the singular homology class of $\gamma'(j(\partial\tau))$ inside $\operatorname{conv}_{\mathcal{F}} \Psi(M_\mu \cup \varphi(\mu))$ is the same,
- **(ii)** each M_μ is disjoint from Y,
- **(iii)** all the simplices M_μ are mutually disjoint.

We define $M_\mu := \emptyset$ for $\mu \in K$ a simplex of dimension at most $k-1$. We set $\Phi(\mu) := \Psi(M_\mu \cup \varphi(\mu))$. Note that for a simplex $\sigma \in K^{(k-1)}$, $\Phi(\sigma)$ reduces to $\Psi(\varphi(\sigma))$.

Let β be the chain map induced by j. Observe that Φ satisfies $\Phi(\emptyset) = \emptyset$ and $\Phi(A \cap B) = \Phi(A) \cap \Phi(B)$, $A, B \in K$. Indeed, the first claim is obvious and for the second one let σ, τ be distinct simplices in K:

$$\begin{aligned}\Phi(\mu) \cap \Phi(\tau) &= \Psi\left(M_\mu \cup \varphi(\mu)\right) \cap \Psi\left(M_\tau \cup \varphi(\tau)\right) = \Psi\left([M_\mu \cup \varphi(\mu)] \cap [M_\tau \cup \varphi(\tau)]\right) \\ &= \Psi(\varphi(\mu) \cap \varphi(\tau)),\end{aligned}$$

where the the second equality express the fact that Ψ respects intersections and the last equality uses both (bii) and (biii). Then

$$\Phi(\mu) \cap \Phi(\tau) = \Psi(\varphi(\mu) \cap \varphi(\tau)) = \Psi(\varphi(\mu \cap \tau)) = \Phi(\mu \cap \tau)$$

since φ obviously respects intersections and $\dim(\mu \cap \tau) \leq k-1$.

We define γ on $K^{(k-1)}$ as the composition $\gamma' \circ \beta \circ \alpha$. Then, by the definition, γ is a nontrivial chain map constrained by $(\mathcal{F}, \Phi)$. It remains to extend it to the whole complex K.

If σ is a k-simplex of K, all the k-simplices ζ in $\operatorname{sd}\sigma$ have the same value of $\gamma'\beta(\partial\zeta)$ inside $\operatorname{conv}_{\mathcal{F}} \Phi(\sigma)$. Since there is an even number of them and we work with $\mathbb{Z}_2$-coefficients, $\gamma(\partial\sigma)$ has trivial homology inside $\operatorname{conv}_{\mathcal{F}} \Phi(\sigma)$. So for each such σ we may pick some $\gamma_\sigma \in C_k\left(\operatorname{conv}_{\mathcal{F}} \Phi(\sigma); \mathbb{Z}_2\right)$ such that $\partial\gamma_\sigma = \gamma(\partial\sigma)$ and extend γ by setting $\gamma(\sigma) := \gamma_\sigma$. Then, by definition, γ is a non-trivial chain map from $C_*(K; \mathbb{Z}_2)$ to $C_*(\mathbf{R}; \mathbb{Z}_2)$ constrained by $(\mathcal{F}, \Phi)$ and hence by $(\mathcal{F}, P)$.

It remains to show that if $\dim K \leq 1$, we can take γ that is induced by a continuous map $f\colon |K| \to \mathbf{R}$. If $\dim K = 0$, we map each point to a point, so the statement is obviously true.

If $\dim K = 1$, we inspect the composition $\gamma = \gamma' \circ \beta \circ \alpha$. It maps points of K to points in $\mathbf{R}$ in such a way that the homology class of $\gamma(\partial\tau)$ inside $\operatorname{conv}_{\mathcal{F}}(\Psi(\tau))$ is trivial for each edge τ of K. But this means that the endpoints of τ get mapped to points in the same path-component of $\operatorname{conv}_{\mathcal{F}}(\Psi(\tau))$ and can be connected by an actual path. ◀

4 A fractional Helly theorem on surfaces

The aim is to bring the constant h_f from Theorem 3 (applied to a surface S) down to three for $b = 0$ and to $2b + 4$ for $b \geq 1$, respectively. This will give Theorem 4. The presented method is based on the recent result of Kalai and the author [11] and allow us to significantly decrease h_f to a small value as soon as we have a finite upper bound on h_f.

Before we perform the bootstrapping, we need few definitions. Let $\mathcal{A} = \{A_1, \dots, A_n\}$ be subsets of a surface S. Set $A_I = \bigcap_{i \in I} A_i$ and let $N(\mathcal{A}) = \{I \in [n] \colon A_I \neq \emptyset\}$ be the nerve of $\mathcal{A}$. We put $f_k(\mathcal{A}) = |\{I \in N(\mathcal{A}) \colon |I| = k + 1\}|$. In words, f_k counts the number if intersecting $(k + 1)$-tuples from $\mathcal{A}$.

The main tool for the bootstrapping is the following proposition.

▶ **Proposition 16.** *Let $b \geq 0$ and $k \geq 2$ be integers satisfying that for $b = 0$, $k \geq 2$ and for $b \geq 1$, $k \geq 2b + 3$, respectively. Let S be a surface. Then for every $\alpha_1 \in (0, 1)$ there exists $\alpha_2 = \alpha_2(\alpha_1, b, k, S) > 0$ such that for any sufficiently large family $\mathcal{A}$ of n open sets in S with $TC_1(\mathcal{A}) \leq b$ the following holds:*

$$f_k(\mathcal{A}) \geq \alpha_1 \binom{n}{k+1} \quad \Rightarrow \quad f_{k+1}(\mathcal{A}) \geq \alpha_2 \binom{n}{k+2}.$$

Let $b \geq 0$ and let $k_0 = k_0(b)$ be an integer depending on b. Namely, we set $k_0(0) = 3$ and $k_0(b) = 2b + 4$ for $b \geq 1$. Let $k \geq k_0 + 1$. By a successive application of the proposition we get that if at least an α-fraction of all k_0-tuples intersect, then also some α'-fraction of all k-tuples intersect. By the (non-optimal) fractional Helly theorem (Theorem 3), we already know that if some α'-fraction of all h_f-tuples intersect, there is some β-fraction of all sets that have a point in common. Putting $k = h_f$ proves Theorem 4.

As mentioned, the proof of Proposition 16 heavily relies on [11, Theorem 4], which can be reformulated[14], in terms of bounded topological complexity, as follows:

▶ **Theorem 17** ([11]). *Let S be a surface, $b \geq 0$ an integer and let $k = k(b)$ be an integer depending on b, namely $k(0) \geq 2$ and $k(b) \geq 2b + 3$ for $b \geq 1$. Let $\mathcal{A}$ be a finite family of open sets in S with $TC_1(\mathcal{A}) \leq b$. Then*

$$f_{k+1}(\mathcal{A}) = 0 \quad \Rightarrow \quad f_k(\mathcal{A}) \leq c_1 f_{k-1}(\mathcal{A}) + c_2,$$

where $c_1 > 0, c_2 \geq 0$ are constants depending only on k, b and the surface S.

Hypergraphs. A hypergraph is *ℓ-uniform* if all its edges have size ℓ. A hypergraph is *ℓ-partite*, if its vertex set V can be partitioned into ℓ subsets $V_1, \dots, V_\ell$, called *classes*, so that each edge contains at most one point from each V_i. Let $K^\ell(t)$ denote the complete ℓ-partite ℓ-uniform hypergraph with t vertices in each of its ℓ vertex classes.

We need the following theorem of Erdős and Simonovits [4] about super-saturated hypergraphs (see also [13, Chapter 9.2]):

▶ **Theorem 18** ([4]). *For any positive integers ℓ and t and any $\varepsilon > 0$ there exists $\delta > 0$ with the following property: Let H be an ℓ-uniform hypergraph on n vertices and with at least $\varepsilon \binom{n}{\ell}$ edges. Then H contains at least $\lfloor \delta n^{\ell t} \rfloor$ copies (not necessarily induced) of $K^\ell(t)$.*

[14] We note that our reformulation is slightly weaker, however, we prefer a simpler exposition which is moreover adapted to our notion of topological complexity.

Proof of Proposition 16. Let $\mathcal{A} = \{A_1, \ldots, A_n\}$ be a family of sets in S satisfying the assumptions of the proposition. By Theorem 17, there exist constants $c_1 > 0, c_2 \geq 0$ depending on b, k and S such that $f_k(\mathcal{A}) \leq c_1 f_{k-1}(\mathcal{A}) + c_2$ provided $f_{k+1}(\mathcal{A}) = 0$. Since $f_{k-1}(\mathcal{A}) \leq \binom{n}{k}$, we have

$$f_{k+1}(\mathcal{A}) = 0 \quad \Rightarrow \quad f_k(\mathcal{A}) \leq (c_1 + c_2)\binom{n}{k}. \tag{2}$$

Let H be a $(k+1)$-uniform hypergraph whose vertices and edges correspond to the vertices and k-simplices of the nerve N of $\mathcal{A}$. Set

$$t := \left\lceil (c_1 + c_2) \cdot \frac{(k+1)^k}{k!} \right\rceil$$

By Erdős-Simonovits theorem ($\varepsilon = \alpha_1, \ell = k+1$), there is at least $\delta n^{(k+1)t}$ copies of $K^{k+1}(t)$ in H.

Since $K^{k+1}(t)$ has $(k+1)t$ vertices and t^{k+1} edges, it follows by (2) that for every copy of $K^{k+1}(t)$ in H there is an intersecting subfamily of size $k+2$ among the corresponding members of $\mathcal{A}$. Indeed, the implication (2) translates into checking that for $k \geq 2$,

$$t^{k+1} \quad > \quad (c_1 + c_2)\binom{(k+1)t}{k}.$$

On the other hand, each such intersecting $(k+2)$-tuple is contained in at most $n^{(k+1)t-(k+2)}$ distinct copies of $K^{k+1}(t)$ (this is the number of choices for the vertices not belonging to the considered $(k+2)$-tuple), and the result follows (i.e. $f_{k+1}(\mathcal{A}) \geq \delta n^{k+2} \geq \alpha_2 \binom{n}{k+2}$). ◀

References

1. N. Alon, G. Kalai, J. Matoušek, and R. Meshulam. Transversal numbers for hypergraphs arising in geometry. *Adv. Appl. Math.*, 29:79–101, 2002.
2. D. C. Kay and E. W. Womble. Axiomatic convexity theory and relationships between the Carathéodory, Helly, and Radon numbers. *Pacific Journal of Mathematics*, 38, August 1971. doi:10.2140/pjm.1971.38.471.
3. J. De Loera, X. Goaoc, F. Meunier, and N. Mustafa. The discrete yet ubiquitous theorems of Carathéodory, Helly, Sperner, Tucker, and Tverberg. *Bulletin of the American Mathematical Society*, June 2017. doi:10.1090/bull/1653.
4. P. Erdős and M. Simonovits. Supersaturated graphs and hypergraphs. *Combinatorica*, 3(2):181–192, 1983. doi:10.1007/BF02579292.
5. R. Fulek and J. Kynčl. The $\mathbb{Z}_2$-genus of Kuratowski minors. In *34th International Symposium on Computational Geometry*, volume 99 of *LIPIcs. Leibniz Int. Proc. Inform.* Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2018.
6. X. Goaoc, I. Mabillard, P. Paták, Z. Patáková, M. Tancer, and U. Wagner. On generalized Heawood inequalities for manifolds: a van Kampen–Flores-type nonembeddability result. *Israel J. Math.*, 222(2):841–866, 2017. doi:10.1007/s11856-017-1607-7.
7. X. Goaoc, P. Paták, Z. Patáková, M. Tancer, and U. Wagner. Bounding Helly numbers via Betti numbers. In *A journey through discrete mathematics*, pages 407–447. Springer, Cham, 2017.
8. A. Holmsen, M. Kim, and S. Lee. Nerves, minors, and piercing numbers. *Trans. Amer. Math. Soc.*, 371:8755–8779, 2019.
9. A. Holmsen and D. Lee. Radon numbers and the fractional Helly theorem, 2019. arXiv:1903.01068.
10. R. E. Jamison-Waldner. Partition numbers for trees and ordered sets. *Pacific J. Math.*, 96(1):115–140, 1981. URL: https://projecteuclid.org:443/euclid.pjm/1102734951.

11 G. Kalai and Z. Patáková. Intersection patterns of planar sets, 2019. arXiv:1907.00885.

12 F. W. Levi. On Helly's theorem and the axioms of convexity. *The Journal of the Indian Mathematical Society*, 15(0):65–76, 1951. URL: http://www.informaticsjournals.com/index.php/jims/article/view/17070.

13 J. Matoušek. *Lectures on discrete geometry*, volume 212 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 2002. doi:10.1007/978-1-4613-0039-7.

14 P. Paták and M. Tancer. Embeddings of k-complexes into $2k$-manifolds, 2019. arXiv:1904.02404.

15 J. Radon. Mengen konvexer Körper, die einen gemeinsamen Punkt enthalten. *Mathematische Annalen*, 83(1):113–115, March 1921. doi:10.1007/BF01464231.

16 F. P. Ramsey. On a problem in formal logic. *Proc. London Math. Soc.*, 30:264—-286, 1929.

17 V. P. Soltan. *Vvedenie v aksiomaticheskuyu teoriyu vypuklosti*. "Shtiintsa", Kishinev, 1984. In Russian, with English and French summaries.

18 M. L. J. van de Vel. *Theory of convex structures*, volume 50 of *North-Holland Mathematical Library*. North-Holland Publishing Co., Amsterdam, 1993.

19 H. Whitney. The self-intersections of a smooth n-manifold in $2n$-space. *Annals of Mathematics*, 45(2):220–246, 1944. URL: http://www.jstor.org/stable/1969265.

Barycentric Cuts Through a Convex Body

Zuzana Patáková
Computer Science Institute, Charles University, Prague, Czech Republic
IST Austria, Klosterneuburg, Austria
zuzka@kam.mff.cuni.cz

Martin Tancer
Department of Applied Mathematics, Charles University, Prague, Czech Republic
ReplaceWithMySurname@kam.mff.cuni.cz

Uli Wagner
IST Austria, Klosterneuburg, Austria
uli@ist.ac.at

Abstract

Let K be a convex body in $\mathbb{R}^n$ (i.e., a compact convex set with nonempty interior). Given a point p in the interior of K, a hyperplane h passing through p is called *barycentric* if p is the barycenter of $K \cap h$. In 1961, Grünbaum raised the question whether, for every K, there exists an interior point p through which there are at least $n+1$ distinct barycentric hyperplanes. Two years later, this was seemingly resolved affirmatively by showing that this is the case if $p = p_0$ is the point of maximal *depth* in K. However, while working on a related question, we noticed that one of the auxiliary claims in the proof is incorrect. Here, we provide a counterexample; this re-opens Grünbaum's question.

It follows from known results that for $n \geq 2$, there are always at least three distinct barycentric cuts through the point $p_0 \in K$ of maximal depth. Using tools related to Morse theory we are able to improve this bound: four distinct barycentric cuts through p_0 are guaranteed if $n \geq 3$.

2012 ACM Subject Classification Theory of computation → Computational geometry

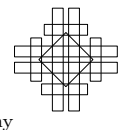
Keywords and phrases convex body, barycenter, Tukey depth, smooth manifold, critical points

Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.62

Related Version A full version of this paper is available at [19], `https://arxiv.org/abs/2003.13536`.

Funding *Zuzana Patáková*: The research stay at IST Austria is funded by the project Improvement of internationalization (CZ.02.2.69/0.0/0.0/17_050/0008466) in the field of research and development at Charles University, through the support of quality projects MSCA-IF.
Martin Tancer: Supported by the GAČR grant 19-04113Y and by the Charles University projects PRIMUS/17/SCI/3 and UNCE/SCI/004.

Acknowledgements We thank Stanislav Nagy for introducing us to Grünbaum's questions, for useful discussions on the topic, for providing us with many references, and for comments on a preliminary version of this paper. We thank Jan Kynčl and Pavel Valtr for letting us know about a more general counterexample they found, and Roman Karasev for pointing us to related work [15, 1] and for comments on a preliminary version of this paper. Finally, we thank an anonymous referee for many comments on a preliminary version of the paper which, in particular, yielded an important correction in Section 4.



1 Introduction

Grünbaum's questions. Let K be a convex body in $\mathbb{R}^n$ (i.e., compact convex set with nonempty interior). Given an interior point $p \in K$, a hyperplane h passing through p is called *barycentric* if p is the barycenter (also known as the centroid) of the intersection $K \cap h$. In 1961, Grünbaum [11] raised the following questions (see also [12, §6.1.4]):

▶ **Question 1.** *Does there always exist an interior point $p \in K$ through which there are at least $n+1$ distinct barycentric hyperplanes?*

▶ **Question 2.** *In particular, is this true if p is the barycenter of K?*

Seemingly, Question 1 was answered affirmatively by Grünbaum himself [12, §6.2] two years later, by using a variant of Helly's theorem to show that there are at least $n+1$ barycentric cuts through the point of K of maximal *depth* (we will recall the definition below). The assertion that Question 1 is resolved has also been reiterated in other geometric literature [6, A8]. However, when working on Question 2, which remains open, we identified a concrete problem in Grünbaum's argument for the affirmative answer for the point of the maximal depth. The first aim of this paper is to point out this problem, which re-opens Question 1.

Depth, depth-realizing hyperplanes, and the point of maximum depth. In order to describe the problem with Grünbaum's argument, we need a few definitions. Let p be a point in K. For a unit vector v in the unit sphere $S^{n-1} \subseteq \mathbb{R}^n$, let $h_v = h_v^p := \{x \in \mathbb{R}^n : \langle v, x-p\rangle = 0\}$ be the hyperplane orthogonal to v and passing through p, and let $H_v = H_v^p := \{x \in \mathbb{R}^n : \langle v, x-p\rangle \geq 0\}$ be the half-space bounded by h_v in the direction of v. Given p, we define the *depth function* $\delta^p : S^{n-1} \to [0,1]$ via $\delta^p(v) = \lambda(H_v \cap K)/\lambda(K)$, where λ is the Lebesgue measure (n-dimensional volume) in $\mathbb{R}^n$. The *depth* of a point p in K is defined as $\operatorname{depth}(p,K) := \inf_{v \in S^{n-1}} \delta^p(v)$. It is easy to see[1] that δ^p is a continuous function, therefore the infimum in the definition is attained at some $v \in S^{n-1}$. Any hyperplane h_v through p such that $\operatorname{depth}(p,K) = \delta^p(v)$ is said to *realize the depth* of p. Finally, a *point of maximal depth in K* is a point p_0 in the interior of K such that $\operatorname{depth}(p_0,K) := \max \operatorname{depth}(p,K)$ where the maximum is taken over all points in the interior of K.[2] The point of maximal depth always exists (by compactness of S^{n-1}) and it is unique (two such points would yield a point of larger depth on the segment between them).

Many depth-realizing hyperplanes? Grünbaum's argument has two ingredients. The first is the following result, known as Dupin's theorem [9], which dates back to 1822:

▶ **Theorem 3** (Dupin's Theorem)**.** *If a hyperplane h through p realizes the depth of p then it is barycentric with respect to p.*

---

[1] Given $v, v' \in S^{n-1}$, $\lambda(H_v \cap K)$ and $\lambda(H_{v'} \cap K)$ differ by at most $\lambda((H_v \Delta H_{v'}) \cap K)$ where Δ is the symmetric difference. For $\varepsilon > 0$ and v and v' sufficiently close, $\lambda((H_v \Delta H_{v'}) \cap K) < \varepsilon\lambda(K)$ as K is bounded.

[2] We remark that our depth function slightly differs from the function $f(H,p)$ used by Grünbaum [12, §6.2]. However, the point of maximal depth coincides with the "critical point" in [12] and hyperplanes realizing the depth for p_0 coincide with the 'hyperplanes through the critical point dividing the volume of K in the ratio $F_2(K)$'.

Grünbaum refers to Blaschke [2] for a proof; for a more recent reference, see [22, Lemma 2].[3] A stronger statement will be the content of Proposition 11 below.

The second ingredient in Grünbaum's argument is the following assertion (which in [12, §6.2] is deduced using a variant of Helly's theorem, without providing the details).

▶ **Postulate 4.** *If p_0 is the point of K of maximal depth, then there are at least $n+1$ distinct hyperplanes through p_0 that realize the depth.*

If correct, Postulate 4, in combination with Dupin's theorem, would immediately imply an affirmative answer to Question 1. However, it turns out that this step is problematic. Indeed, there is a counterexample to Postulate 4:

▶ **Proposition 5.** *Let $K = T \times I \subseteq \mathbb{R}^3$ where T is an equilateral triangle and I is a line segment (interval) orthogonal to T, and let $p_0 \in K$ be the point of maximal depth (which in this case coincides with the barycenter of K). Then there are only 3 hyperplanes realizing the depth of p_0.*

▶ Remark 6. We believe that Proposition 5 can be generalized to higher dimensions in the sense that, for every n, there are only n depth-realizing hyperplanes through the point of maximal depth in $\Delta \times I \subseteq \mathbb{R}^n$, where Δ is a regular $(n-1)$-simplex. However, we did not attempt to work out the details carefully, because Kynčl and Valtr [16] informed us about stronger counterexamples: For every n, there exists a convex body $K \in \mathbb{R}^n$ such that there are only 3 depth-realizing hyperplanes through the point of maximal depth in K. Therefore, we prefer to keep the proof of Proposition 5 as simple as possible and focus on dimension 3.

▶ Remark 7. We emphasize that Proposition 5 does not preclude an affirmative answer to Grünbaum's Question 1 (nor to Question 2), since $T \times I$ contains infinitely many distinct barycentric hyperplanes through p_0. Thus Grünbaum's questions remain open.

We also remark that a weakening of Postulate 4 is known to be true (see the 'Inverse Ray Basis Theorem [20], using the proof from [8]):[4,5]

▶ **Proposition 8.** *Let $U \subseteq S^{n-1}$ be the set of vectors u such that $\delta^{p_0}(u) = \operatorname{depth}(p_0, K)$. Then $0 \in \operatorname{conv} U$.*

In the special case that U is in general position, the cardinality of U is at least $n+1$ (otherwise $\dim \operatorname{conv} U < n$ and $\operatorname{conv} U$ would not contain the origin, by general position), which proves Postulate 4 in this special case. However, U need not be always in general position. For example, in the case $K = T \times I$ in $\mathbb{R}^3 = \mathbb{R}^2 \times \mathbb{R}$ of Proposition 5, the set U contains three vectors in the plane through the origin parallel with T. This is also the way we arrived at the counterexample from Proposition 5.

Inverse Ray Basis Theorem immediately implies that three barycentric hyperplanes are guaranteed in dimension at least 2.

---

[3] The idea of the proof is simple: For contradiction assume that h realizes the depth of p but that the barycenter b of $K \cap h$ differs from p. Let $v \in S^{n-1}$ be such that $h = h_v$ and $\operatorname{depth}(p, K) = \delta^p(v)$. Consider the affine $(d-2)$-space ρ in h passing through p and perpendicular to the segment bp. Then by a small rotation of h along ρ we can get $h_{v'}$ such that $\delta^p(v') < \delta^p(v)$ which contradicts that h realizes the depth of p. Of course, it remains to check the details.

[4] We remark that the second condition in the statement of the result in [20] is equivalent to the statement that $0 \in \operatorname{conv} U$, in our notation.

[5] Sketch of the inverse ray basis theorem: if there is a closed hemisphere $C \subseteq S^{n-1}$ which does not contain a point of U, let v be the center of C. Then a small shift of p_0 in the direction of v yields a point of larger depth, a contradiction.

▶ **Corollary 9.** *Let K be a convex body in $\mathbb{R}^n$ where $n \geq 2$ and p_0 be the point of maximal depth of K. Then there at least three distinct barycentric hyperplanes through p_0.*

Proof. Let U be the set from Proposition 8. Then, $0 \in \operatorname{conv} U$ and $U \subseteq S^{n-1}$ imply together $|U| \geq 2$. However, if $|U| = 2$, then $U = \{u, -u\}$ for some $u \in S^{n-1}$. This necessarily means $\operatorname{depth}(p_0, K) = \delta^{p_0}(u) = \delta^{p_0}(-u) = 1/2$ as $\delta^{p_0}(u) + \delta^{p_0}(-u) = 1$. Then for any other $v \in S^{n-1}$ we get $\min\{\delta^{p_0}(v), \delta^{p_0}(-v)\} \geq 1/2$ which implies $\delta^{p_0}(v) = \delta^{p_0}(-v) = 1/2$ as well. Therefore $v \in U$ contradicting $|U| = 2$.) ◀

Four barycentric cuts via critical points of C^1 functions. Using tools related to Morse theory, we are able to obtain one more barycentric hyperplane, provided that $n \geq 3$.

▶ **Theorem 10.** *Let K be a convex body in $\mathbb{R}^n$ where $n \geq 3$ and p_0 be the point of maximal depth of K. Then there are at least four distinct hyperplanes h such that p_0 is the barycenter of $K \cap h$.*

Here we should also mention related work of Blagojević and Karasev [15, Theorem 3.3] and [1, Theorem 1.13]. They show that there are at least $\mu(n)$ barycentric hyperplanes passing through *some* interior point of K (not necessarily the point of maximal depth), where $\mu(n) := \min_f \max_{p \in S^n} |f^{-1}(p)|$ is the minimum *multiplicity* of any continuous map $f\colon \mathbb{R}P^n \to S^n$ (here, $\mathbb{R}P^n$ is the n-dimensional real projective space). By calculations with Stiefel–Whitney classes, they obtain lower bounds for $\mu(n)$ that depend in a subtle (and non-monotone) way on n (see [15, Remark 1.3]). For example, $\mu(n) \geq \frac{n}{2} + 1$ if $n = 2^\ell - 2$, but for values of n of the form $n = 2^\ell - 1$ (e.g., for $n = 3$) their methods only give a lower bound of $\mu(n) \geq 2$.

Our argument in the proof of Theorem 10 is, in certain sense, tight. This is discussed in the full version [19, Section 5].

In what follows, we view S^{n-1} as a smooth manifold with its standard differential structure. A key tool in the proof of Theorem 10 is the following close connection between barycentric hyperplanes and the critical points of the depth function:

▶ **Proposition 11.** *Let $K \subseteq \mathbb{R}^n$ be a convex body and p be a point in the interior of K. Then the corresponding depth function $\delta^p\colon S^{n-1} \to \mathbb{R}$ is a C^1 function. In addition, $v \in S^{n-1}$ is a critical point of δ^p (that is, $D\delta^p(v) = 0$, where $Df(v)$ denotes the total derivative of a function f at v) if and only if h_v is barycentric.*

As mentioned earlier, Proposition 11 generalizes Dupin's theorem. Indeed, if $h = h_v$ realizes the depth, then v is a global minimum of δ^p, hence h is barycentric by Proposition 11.

In the proof, we closely follow computations by Hassairi and Regaieg [13] who stated an extension of Dupin's theorem to absolutely continuous probability measures. As explained in [18] (see Proposition 29, Example 7, and the surrounding text in [18]), the extension of Dupin's theorem does not hold in the full generality stated in [13], and it requires some additional assumptions. However, a careful check of the computations of Hassairi and Regiaeg [13] in the special case of uniform probability measures on convex bodies reveals not only Dupin's theorem but all items of Proposition 11.

Regarding the proof of Theorem 10, the Inverse Ray Basis Theorem (Proposition 8) and Corollary 9 imply that δ^{p_0} has at least three global minima. This gives three barycentric hyperplanes via Proposition 11. Furthermore, we also get three maxima of δ, as a maximum appears at v, if and only if a minimum appears at $-v$ (note that $h_v = h_{-v}$). However, it should not happen for a C^1 function on S^{n-1} that it has only such critical points. We will show that there is at least one more critical point, which yields another barycentric hyperplane via Proposition 11. Namely, we show the following proposition.

▶ **Proposition 12.** *Let $n \geq 2$ and let $f\colon S^n \to \mathbb{R}$ be a C^1 function. Let $m_1, \ldots, m_k$ be (not necessarily strict) local minima or maxima of f, where $k \geq 3$. Then there exists $u \in S^n$, different from $m_1, \ldots, m_k$, such that $Df(u) = 0$.*

This finishes the proof of Theorem 10 modulo Propositions 11 and 12. (Proposition 12 is applied with $k = 6$.) The main idea beyond the proof of Proposition 12 is that if we have at least three local minima or maxima, then we should also expect a saddle point (unless there are infinitely many local extrema). This would be an easy exercise for Morse functions (which are in particular C^2) via Morse theory (actually, the Morse inequalities would provide even more critical points). Working with C^1 functions adds a few difficulties, but all of them can be overcome.

Relation to probability and statistics. The depth function, as we define it above is a special case of the (Tukey) depth of a probability measure in $\mathbb{R}^d$, a well-known notion in statistics [23, 7, 8]. More precisely, given a probability measure $\mathbf{P}$ on $\mathbb{R}^d$ and $p \in \mathbb{R}^d$, we can define $\operatorname{depth}(p, \mathbf{P}) := \inf_{v \in S^{n-1}} \mathbf{P}(H_v)$. Then $\operatorname{depth}(p, K)$ is a special case of the uniform probability measure on a convex body K, i.e., $\mathbf{P}(A) := \lambda(A)/\lambda(K)$ for A Lebesgue-measurable. We refer to [18] for an extensive recent survey making many connections between the depth function in statistics and geometric questions.

There is a vast amount of literature, both in computational geometry and statistics, devoted to computing the depth function in various settings (which is not easy in general). We refer, for example, to [21, 4, 3, 5, 10, 17] and the references therein. From this point of view, understanding the minimal possible number of critical points of the depth function is a quite fundamental property of the depth function. Via Proposition 11, this is essentially equivalent to Grünbaum's questions.

Organization. Proposition 5 is proved in Section 2; Proposition 11 is proved in Section 3; and Proposition 12 is proved in Section 4.

2 Few hyperplanes realizing the depth

In this section we prove Proposition 5, assuming Proposition 11.

Preliminaries. Let us recall that given a bounded measurable set $Y \subseteq \mathbb{R}^n$ of positive measure, the *barycenter* of Y is defined as

$$\operatorname{cen} Y = \frac{\int_{\mathbb{R}^n} x\chi_Y(x)dx}{\int_{\mathbb{R}^n} \chi_Y(x)dx} = \frac{1}{\lambda(Y)} \int_Y x dx \tag{1}$$

where χ_Y is the characteristic function and the integral is considered as a vector in $\mathbb{R}^n$.

If Y splits as a disjoint union $Y = Y_1 \sqcup \cdots \sqcup Y_\ell$ of sets of positive measure then

$$\operatorname{cen} Y = \frac{1}{\lambda(Y)} \left(\sum_{i=1}^{\ell} \lambda(Y_i) \operatorname{cen} Y_i \right) \tag{2}$$

which easily follows from (1). If h is a hyperplane, and $Y \subseteq h$ has positive $(n-1)$-dimensional Lebesgue measure inside h, then the formula for the barycenter is analogous to (1):

$$\operatorname{cen} Y = \frac{\int_h x\chi_Y(x)d\lambda_{n-1}(x)}{\int_h \chi_Y(x)d\lambda_{n-1}(x)} = \frac{1}{\lambda_{n-1}(Y)} \int_Y x d\lambda_{n-1}(x) \tag{3}$$

where λ_{n-1} denotes the $(n-1)$-dimensional Lebesgue measure on h in this formula.

If $h \subseteq \mathbb{R}^n$ is a hyperplane whose orthogonal projection $\pi(h)$ onto $\mathbb{R}^{n-1} \times \{0\}$ (the first $n-1$ coordinates) equals $\mathbb{R}^{n-1} \times \{0\}$, then $\operatorname{cen} \pi(Y) = \pi(\operatorname{cen} Y)$.

Proof of Proposition 5. Let $T \subseteq \mathbb{R}^2$ be an equilateral triangle with $\operatorname{cen}(T) = 0$ and $I = [-1, 1]$. Then $\operatorname{cen}(K) = 0$. In addition, because the point of maximal depth p_0 is unique and invariant under isometries of K, we get $p_0 = 0$.

We will use the following notation: a, b, c are the vertices of T and α, β, and γ are lines perpendicular to T passing through a, b, and c respectively.

Now let h be a hyperplane passing through 0. We want to find out whether h realizes the depth. We will consider three cases:

- **(i)** h is perpendicular to T;
- **(ii)** h is not perpendicular to T and all intersection points of h with α, β, and γ belong to K;
- **(iii)** h is not perpendicular to T and at least one of the intersection points of h with α, β, and γ does not belong to K.

In case (i), we will find three candidates for hyperplanes realizing the depth. Then we show that there is no hyperplane realizing the depth in cases (ii) and (iii), which shows that only the three candidates from case (i) may realize the depth. They realize the depth because we have at least three hyperplanes realizing the depth by the discussion in the introduction above Theorem 10.

Let us focus on case (i). This is the same as considering the lines realizing the depth in an equilateral triangle. It is easy to check and well known (see e.g. [20, §5.3]) that the depth of the equilateral triangle is $4/9$ and it is realized by lines parallel with the sides of the triangle. It follows that we can reach depth $4/9$ in K by hyperplanes perpendicular to T and parallel with the three sides of T, and all other hyperplanes from case (i) bound a portion of K strictly larger than $4/9$ on each of their sides.

Case (ii) is very easy: It is easy to compute that each hyperplane of type (ii) splits K into two parts of equal volume $1/2$. Therefore, no such hyperplane realizes the depth.

Finally, we investigate case (iii). Here we show that no hyperplane h of case (iii) is barycentric. Therefore, by Theorem 3, it cannot realize the depth either.

We aim to show that 0 is not the barycenter of $h \cap K$. Let U be the orthogonal projection of $h \cap K$ to the triangle T. Equivalently, we want to show that 0 is not the barycenter of U. We also realize that $U = T \cap S$, where S is an infinite strip obtained as the orthogonal projection of $h \cap (\mathbb{R}^2 \times I)$ to $\mathbb{R}^2 \times \{0\}$; see Figure 1.

Let s be the center line of S. This is the line where h meets the plane of T. We remark that 0 belongs to s and in addition U is a proper subset of T (otherwise we would be in case (ii)). We again distinguish three cases:

- **(a)** none of the vertices a, b, c belongs to U,
- **(b)** one of the vertices a, b, c belongs to U,
- **(c)** two of the vertices a, b, c belong to U.

In all the cases we will show $\operatorname{cen} U \neq \operatorname{cen} T$. In case (a), s splits one of the vertices of T from the other two. Without loss of generality, a is on one side of s and b and c are on the other side. The center line s also splits U into two parts. Let W' be the (closed) part on the side of a, W'' be the mirror image of W' along S and $W := W' \cup W''$. Note that W is a proper subset of U; indeed, since $\operatorname{cen} T = 0$ and T is equilateral, the line s splits the segment ab closer to b and the segment ac closer to c. By the symmetry of W, the barycenter $\operatorname{cen} W$ belongs to the line s. However, this means that the barycenter of U is not on s; it is on the bc side of s. Formally, this follows from (2) for the decomposition $U = W \sqcup (U \setminus W)$.

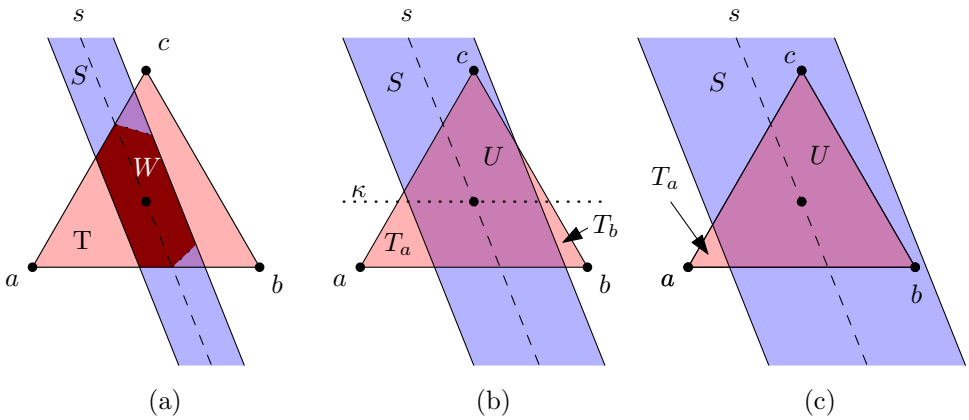


Figure 1 Three cases for the intersection $U = T \cap S$.

In case (b), without loss of generality, U contains c. Then $T \setminus U$ is the union of two triangles T_a and T_b. Let κ be the line parallel with ab passing through 0. Without loss of generality, up to rotating T, κ is the x-axis. From (2), we get $0 = \operatorname{cen} T = \frac{1}{\lambda(T)}(\lambda(U)\operatorname{cen} U + \lambda(T_a)\operatorname{cen} T_a + \lambda(T_b)\operatorname{cen} T_b)$. The barycenters $\operatorname{cen} T_a$ and $\operatorname{cen} T_b$ are below the line κ or on it. At least one of these barycenters is strictly below ($\operatorname{cen} T_a$ is on κ if and only if c belongs to the closure of T_a, and similarly with T_b). Therefore, $\operatorname{cen} U$ must be strictly above κ if the above equality is supposed to hold.

In case (c), it is even more obvious that $\operatorname{cen} U \neq \operatorname{cen} T$. Without loss of generality U contains b and c. Then $T \setminus U$ is a triangle T_a. Since both T and T_a are convex and T_a does not contain $\operatorname{cen} T$, we have $\operatorname{cen} T_a \neq \operatorname{cen} T$. Therefore $\operatorname{cen} T \neq \operatorname{cen} U$ follows from (2) for the decomposition $T = U \sqcup T_a$. ◀

Bipyramid over a triangle. In $\mathbb{R}^3$, we have a candidate example of a convex body, namely the regular bipyramid B over an equilateral triangle T, such that there are exactly four barycentric hyperplanes (with respect to the barycenter of B, which coincides with the point of maximal depth in this case). On the one hand, this is not surprising, because this is $n + 1$ hyperplanes, where $n = 3$ is the dimension of the ambient space. On the other hand, if this is true, then it answers negatively, in dimension 3, a question from [6, A8], whether $2^n - 1$ barycentric hyperplanes always exist. More concretely, we conjecture that the only barycentric hyperplanes are the following: three planes perpendicular to T which meet T in lines realizing the depth of T (these would be the hyperplanes realizing the depth), and the plane of T (this is the one extra plane). Unfortunately, in this case, it is not so easy to analyze the depth function as in the case of $T \times I$.

3 Critical points of the depth function

Here we prove Proposition 11. We follow [13] with a slightly adjusted notation and adding a few more details here and there.

Proof of Proposition 11. Without loss of generality, we can assume that the point p coincides with the origin and we suppress it from the notation. That is, we write δ for the depth function instead of δ^p.

Let $e_1, \ldots, e_n$ be the canonical basis of $\mathbb{R}^n$ and let

$$S^{n-1}_{j+} = \{u = \sum_{i=1}^{n} u_i e_i \in S^{n-1}; u_j > 0\} \qquad \text{and} \qquad S^{n-1}_{j-} = \{u = \sum_{i=1}^{n} u_i e_i \in S^{n-1}; u_j < 0\}$$

be the relatively open hemispheres of S^{n-1} with poles at e_j and $-e_j$, for $j \in [n]$. These sets form an atlas on S^{n-1}.

Let us consider $j \in [n]$. Given $x \in \mathbb{R}^n$ and $i \in [n]$, x_i denotes the ith coordinate of x, that is $x = \sum_{i=1}^{n} x_i e_i$. With a slight abuse of the notation, we identify $\mathbb{R}^{n-1}$ with the subspace of $\mathbb{R}^n$ spanned by $e_1, \ldots, e_{j-1}, e_{j+1}, \ldots, e_n$. Let $\hat{x} := \sum_{i=1, i\neq j}^{n} x_i e_i \in \mathbb{R}^{n-1}$. Following [13] we consider the diffeomorphisms $u \mapsto \beta(u) = -\frac{\hat{u}}{u_j}$ between S^{n-1}_{j+} and $\mathbb{R}^{n-1}$ or between S^{n-1}_{j-} and $\mathbb{R}^{n-1}$. We will check the required properties of δ locally at each of the $2n$ hemispheres S^{n-1}_{j+} or S^{n-1}_{j-} (with respect to the aforementioned diffeomorphisms). Given that all cases are symmetric, it is sufficient to focus only on the S^{n-1}_{n+} case. That is, from now on, we assume that $j = n$ and $\mathbb{R}^{n-1}$ is spanned by the first $(n-1)$ coordinates in the convention above. Given a point $x \in \mathbb{R}^n$, we also write it as $x = (\hat{x}; x_n)$.

Now, for $y \in \mathbb{R}^{n-1}$ we consider the hyperplane h'_y in $\mathbb{R}^n$ containing the origin and defined by $h'_y = \{(\hat{x}; x_n) \in \mathbb{R}^n \colon x_n = \langle y, \hat{x}\rangle\}$. Note that if $u \in S^{n-1}_{j+}$, then $h'_{\beta(u)} = \{x \in \mathbb{R}^n \colon \langle x, u\rangle = 0\}$. In particular, since p is the origin, $h'_{\beta(u)}$ coincides with h_u used in the introduction for definition of the depth function. This also means that the map $y \mapsto h'_y$ provides a parametrization of a family of those hyperplanes containing the origin which do not contain e_n. We also set H'_y to be the positive halfspace bounded by h'_y: $H'_y = \{(\hat{x}; x_n) \in \mathbb{R}^n \colon x_n \geq \langle y, \hat{x}\rangle\}$. Again, if $u \in S^{n-1}_{j+}$, then $H'_{\beta(u)}$ coincides with H_u from the introduction (here we use $u_n > 0$).

Now, we consider the map $f \colon \mathbb{R}^{n-1} \to \mathbb{R}$ defined by

$$f(y) = \lambda(H'_y \cap K) = \int_{\mathbb{R}^{n-1}} \int_{\langle y,\hat{x}\rangle}^{\infty} \chi_K(\hat{x}; x_n) dx_n d\hat{x}, \tag{4}$$

where χ_K is the characteristic function of K. When $y = \beta(u)$ for some $u \in S^{n-1}_{j+}$, then $f(\beta(u)) = \delta(u)$. Therefore, given that the map $u \to \beta(u)$ is a diffeomorphism, it is sufficient to prove that f is a C^1 function and that $\beta(v) \in \mathbb{R}^{n-1}$ is a critical point of f if and only if $h'_{\beta(v)} = h_v$ is barycentric.

The aim now is to differentiate $f(y)$ with respect to y. We will show that the total derivative equals

$$Df(y) = -\int_{\mathbb{R}^{n-1}} \hat{x} \cdot \chi_K(\hat{x}; \langle y, \hat{x}\rangle) d\hat{x} \tag{5}$$

considering the integral on the right-hand side as a vector. Deducing (5) is a quite routine computation skipped in [13].[6] However, this is the step in the proof of Theorem 3.1 in [13] which reveals that some extra assumptions in [13] are necessary. Thus we carefully deduce (5) at the end of this proof for completeness.

We will also see that all partial derivatives of f are continuous which means that f is a C^1 function which is one of our required conditions. Now we want to show that $Df(\beta(v)) = 0$ if and only if h_v is barycentric.

[6] When compared with formula (3.1) in [13], we obtain a different sign in front of the integral. This is caused by integration over the opposite halfspace.

First, assume that $Df(\beta(v)) = 0$. This gives

$$0 = \frac{\int_{\mathbb{R}^{n-1}} \hat{x} \cdot \chi_K(\hat{x}; \langle \beta(v), \hat{x} \rangle) d\hat{x}}{\int_{\mathbb{R}^{n-1}} \chi_K(\hat{x}; \langle \beta(v), \hat{x} \rangle) d\hat{x}} \tag{6}$$

which means that 0 is the barycenter of $K \cap h'_{\beta(v)}$ from the definition of $h'_{\beta(v)}$. On the other hand, if 0 is the barycenter of $K \cap h'_{\beta(v)}$, then we deduce (6) which implies $Df(\beta(v)) = 0$.

It remains to show (5). For this purpose, we compute partial derivatives $\frac{\partial}{\partial y_k} f(y)$, $1 \leq k \leq n-1$. In the following computations, recall that e_k stands for the standard basis vector for the kth coordinate and let $\int_a^b := -\int_b^a$ if $a > b$. We get

$$\begin{aligned}\frac{\partial}{\partial y_k} f(y) &= \lim_{t \to 0} \frac{1}{t} \int_{\mathbb{R}^{n-1}} \left(\int_{\langle y + te_k, \hat{x} \rangle}^{\infty} \chi_K(\hat{x}; x_n) dx_n - \int_{\langle y, \hat{x} \rangle}^{\infty} \chi_K(\hat{x}; x_n) dx_n \right) d\hat{x} \\ &= \lim_{t \to 0} \int_{\mathbb{R}^{n-1}} \frac{1}{t} \int_{\langle y, \hat{x} \rangle + tx_k}^{\langle y, \hat{x} \rangle} \chi_K(\hat{x}; x_n) dx_n d\hat{x}.\end{aligned}$$

Let $y, \hat{x} \in \mathbb{R}^{d-1}$ be such that $(\hat{x}; \langle y, \hat{x} \rangle) \notin \partial K$. Then we get

$$\lim_{t \to 0} \frac{1}{t} \int_{\langle y, \hat{x} \rangle + tx_k}^{\langle y, \hat{x} \rangle} \chi_K(\hat{x}; x_n) dx_n = -x_k \chi_K(\hat{x}; \langle y, \hat{x} \rangle),$$

because $(\hat{x}; \langle y, \hat{x} \rangle) \notin \partial K$ implies that the function $\chi_K(\hat{x}; x_n)$ as a function of x_n is constant on the interval $(\langle y, \hat{x} \rangle - |tx_k|, \langle y, \hat{x} \rangle + |tx_k|)$ for small enough $|t|$. Therefore, by the dominated convergence theorem,

$$\frac{\partial}{\partial y_k} f(y) = \int_{\mathbb{R}^{n-1}} -x_k \chi_K(\hat{x}; \langle y, \hat{x} \rangle) d\hat{x}. \tag{7}$$

For fixed y, the condition $(\hat{x}; \langle y, \hat{x} \rangle) \notin \partial K$ holds for almost every $\hat{x}$ because $(\hat{x}; \langle y, \hat{x} \rangle) \in h_y$ and h_y passes through the interior of K (through the origin). By another application of dominated convergence theorem, we realize that the right hand side of (7) is continuous in y (this time, we consider a sequence $y^i \to y$ and we observe that $\chi_K(\hat{x}; \langle y^i, \hat{x} \rangle) \to \chi_K(\hat{x}; \langle y, \hat{x} \rangle)$ for almost every $\hat{x}$). Therefore the total derivative of f at any y exists and (7) gives the formula (5). ◀

▶ Remark 13. In the last paragraph of the proof above we crucially use the convexity of K. Without convexity, there is a compact nonconvex polygon $K' \subseteq \mathbb{R}^2$, with 0 in the interior, such that there is y with the property that the set of those $\hat{x}$ for which $(\hat{x}; \langle y, \hat{x} \rangle) \in \partial K'$ has positive measure; see Figure 2. In fact, even (5) does not hold for K'. Here we took K' to be the polygon from Example 7 of [18], and we refer the reader to that paper for more details.

4 One more critical point

In this section, we prove Proposition 12. Given a manifold M and a continuous function $f\colon M \to \mathbb{R}$ and $s \in \mathbb{R}$ we define the *level set* $L_s := \{w \in M\colon f(w) = s\}$. In the proof of Proposition 12 we will need that the level sets are well-behaved in the neighborhoods of points u for which the total derivative $Df(u)$ is nonzero.

▶ **Proposition 14.** *Let $n \geq 1$, $f\colon \mathbb{R}^n \to \mathbb{R}$ be a C^1 function and $u \in \mathbb{R}^n$ be such that $Df(u) \neq 0$. Then there is a neighborhood $N(u)$ of u such that for every $v, w \in N(u)$ if $f(v) = f(w)$, then v and w can be connected with a path within the level set $L_{f(v)}$. (It is allowed that this path leaves $N(u)$ provided that it stays in $L_{f(v)}$.)*

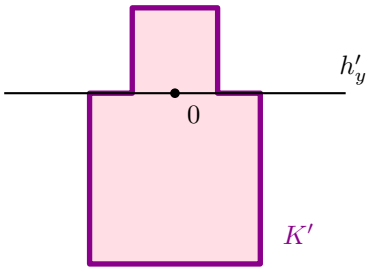


Figure 2 A nonconvex polygon K' and y such that the total derivative of f does not exist at y.

Proof. Without loss of generality assume that $\frac{\partial f}{\partial x_n}(u) > 0$, otherwise we permute the coordinates and/or swap x_n and $-x_n$. Consistently with the previous section, given $x \in \mathbb{R}^n$, we write $x = (\hat{x}, x_n)$ where $\hat{x} \in \mathbb{R}^{n-1}$ and $x_n \in \mathbb{R}$. Now we consider the C^1 function $F\colon \mathbb{R}^{n-1} \times \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ defined as $F(\hat{x}, t, x_n) := f(\hat{x}, x_n) - t$. Note that $\frac{\partial F}{\partial x_n} = \frac{\partial f}{\partial x_n}$. We also observe that $F(\hat{u}, f(u), u_n) = 0$. Therefore, by the implicit function theorem, there is an open neighborhood N' of $(\hat{u}, f(u))$ in $\mathbb{R}^{n-1} \times \mathbb{R}$ such that there is a C^1 function $g\colon N' \to \mathbb{R}$ with $g(\hat{u}, f(u)) = u_n$ and that $F(\hat{v}, t, g(\hat{v}, t)) = 0$ for any $(\hat{v}, t) \in N'$. From the definition of F this gives

$$f(\hat{v}, g(\hat{v}, t)) = t. \tag{8}$$

By possibly restricting the neighborhood to a smaller set, we can assume that N' is the Cartesian product of a neighborhood $N'(\hat{u})$ of $\hat{u}$ in $\mathbb{R}^{n-1}$ and $N'(f(u))$ of $f(u)$ in $\mathbb{R}$, and that both $N'(\hat{u})$ and $N'(f(u))$ are open balls. Moreover, we can assume that $\frac{\partial F}{\partial x_n}(\hat{v}, t, v_n) > 0$ for any $(\hat{v}, t, v_n) \in N' \times N''(u_n)$ where $N''(u_n)$ is some neighborhood of u_n in $\mathbb{R}$, again a ball. Now we possibly further restrict $N'(\hat{u})$ and $N'(f(u))$ so that $g(\hat{v}, t)$ belongs to $N''(u_n)$ for any $(\hat{v}, t) \in N'$.

The condition on the partial derivative of F implies that for every $(\hat{v}, t) \in N'$ the equation $F(\hat{v}, t, x_n) = 0$ has at most one solution $x_n \in N''(u_n)$. Therefore it has a unique solution $x_n = g(\hat{v}, t)$. In other words we get:

$$\text{If } f(\hat{v}, x_n) = t, \text{ then } x_n = g(\hat{v}, t). \tag{9}$$

Now, we define $N(u) := \Psi^{-1}(N')$ where $\Psi\colon \mathbb{R}^{n-1} \times \mathbb{R} \to \mathbb{R}^{n-1} \times \mathbb{R}$ is defined as $\Psi(v) = (\hat{v}, f(v))$ for any $v \in \mathbb{R}^{n-1} \times \mathbb{R}$. In particular $(\hat{v}, f(v))$ belongs to N' for any $v \in N(u)$.

Let $t := f(v) = f(w)$. From (9) we get $v_n = g(\hat{v}, t)$ and $w_n = g(\hat{w}, t)$. Let us consider an arbitrary path $P\colon [0,1] \to N'(\hat{u})$ connecting $\hat{v}$ and $\hat{w}$. Let us "lift" P to a path $P_t\colon [0,1] \to \mathbb{R}^{n-1} \times \mathbb{R}$ given by $P_t(s) := (P(s), g(P(s), t))$. This is a path connecting v and w. We will be done once we show $P_t([0,1]) \subseteq L_t$. This means that we are supposed to show that $f(P(s), g(P(s), t)) = t$ for every $s \in [0,1]$ which follows from (8). ◀

Let $x \in \mathbb{R}^n$ and $\rho > 0$, by $B(x, \rho) \subseteq \mathbb{R}^n$ we denote the compact ball of radius ρ centered in x with respect to the standard Euclidean metric.

▶ **Lemma 15.** *Let $f\colon \mathbb{R}^n \to \mathbb{R}$ be a C^1 function, let $x \in \mathbb{R}^n$ and let $\zeta, \rho > 0$. Assume that $\|Df(u)\| \geq \zeta$ for every $u \in B(x, \rho)$. Then there is $v \in B(x, \rho)$ such that $f(v) \geq f(x) + \frac{\zeta\rho}{2}$.*

The proof is given in the full version [19]; intuitively, we follow the gradient to find v.

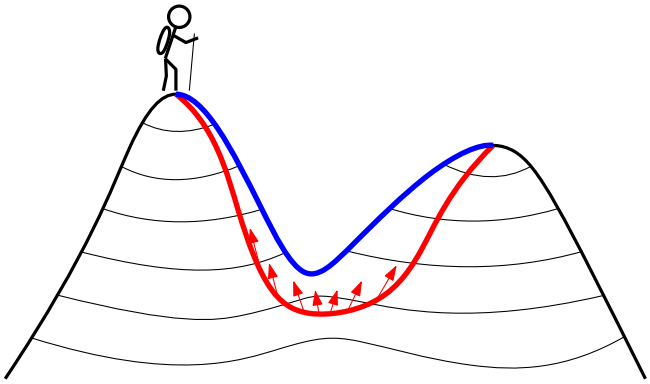


Figure 3 If we are in mountains and we want to hike from one peak to another without losing too much altitude, then the best way is to pass through a saddle point (see the upper path in blue). If we do not pass very close to a saddle point, then the positive gradient allows us to improve the path (see the lower path in red).

Proof of Proposition 12. First, we can assume that all local extrema $m_1, \dots, m_k$ are strict. Indeed, if some of them is not strict, say m_1, then we can find $u \neq m_1, \dots, m_k$ with $Df(u) = 0$ in a neighborhood of m_1.

Next, because $k \geq 3$, there are at least two local maxima or two local minima among $m_1, \dots, m_k$. Without loss of generality, m_1 and m_2 are local maxima.

Now, let us consider a path $\gamma\colon [0,1] \to S^n$ such that $\gamma(0) = m_1$ and $\gamma(1) = m_2$. Let $\min_f(\gamma) := \min\{f(\gamma(t))\colon t \in [0,1]\}$ (the minimum exists by compactness) and let $s := \sup(\min_f(\gamma))$ where the supremum is taken over all γ as above.

Before we proceed with the formal proof, let us sketch the main idea of the proof; see also Figure 3. For contradiction assume that $Df(u) \neq 0$ for every $u \in S^n \setminus \{m_1, \dots, m_k\}$. Consider γ such that $\min_f(\gamma)$ is very close to s. We will be able to argue that we can assume that such γ is not close to any of the other extrema $m_3, \dots, m_k$. This guarantees that $\|Df(\gamma(t))\|$ is bounded from 0 for every $t \in [0,1]$ except the cases when $\gamma(t)$ is close to m_1 or m_2. Using Lemma 15, we will be able to modify γ to γ' with $\min_f(\gamma') > s$ obtaining a contradiction with the definition of s.

In further consideration, we consider the standard metric on S^n obtained by the standard embedding of S^n into $\mathbb{R}^{n+1}$ and restricting the Euclidean metric on $\mathbb{R}^{n+1}$ to a metric on S^n. For every $i \in [k]$, we pick two closed metric[7] balls B_i and B_i' centered in m_i. Namely, B_i is chosen so that m_i is a global extreme on B_i. We also assume that the balls B_i are pairwise disjoint. Next, we distinguish whether m_i is a local maximum or minimum. If m_i is a local maximum, let us define $a_i := \max\{f(x)\colon x \in \partial B_i\}$. Note that $f(m_i) > a_i$ as m_i is a global maximum on B_i. Then we pick a closed ball B_i' centered in m_i inside B_i so that $f(x) > a_i$ for every $x \in B_i'$. If m_i is a local minimum, we proceed analogously. We set $a_i := \min\{f(x)\colon x \in \partial B_i\}$ and we pick B_i' so that $f(x) < a_i$ for every $x \in B_i'$. For later use, we also define $a_i' := \min\{f(x)\colon x \in B_i'\}$ for $i \in \{1,2\}$. Note that $a_i' > a_i$.

Given a path γ connecting m_1 and m_2, we say that γ is *avoiding* if it does not pass through the interior of any of the balls $B_3', \dots, B_k'$.

[7] By a metric ball we mean a ball with a given center and radius. This way, we distinguish a metric ball from a general topological ball.

▷ Claim 16. Let γ be a path connecting m_1 and m_2. Then there is an avoiding path $\bar{\gamma}$ connecting m_1 and m_2 such that $\min_f(\bar{\gamma}) \geq \min_f(\gamma)$.

Proof. Assume that γ enters a ball B_i' for $i \in \{3, \dots, k\}$. Let us distinguish whether m_i is a local maximum or minimum.

First assume that m_i is a local maximum. Then $\min_f(\gamma) \leq a_i$ because γ has to pass through ∂B_i. By a homotopy, fixed outside the interior of B_i' we can assume that γ avoids m_i (here we use $n \geq 2$); see, e.g., the proof of Proposition 1.14 in [14] how to perform this step.[8] In addition, by further homotopy fixed outside the interior of B_i' we can modify γ so that it avoids the interior of B_i' (the second homotopy pushes γ in direction away from m_i). This does not affect $\min_f(\gamma)$ because $f(x) > a_i$ for every $x \in B_i'$.

Next let us assume that m_i is a local minimum. Then $\min_f(\gamma) < a_i$ because γ has to pass through $\partial B_i'$ (this is not a symmetric argument when compared with the previous case). Modify γ by analogous homotopies as above; however, this time with respect to B_i (so that γ completely avoids the interior of B_i). Because $\min_f(\gamma) < a_i$ and $f(x) \geq a_i$ for $x \in \partial B_i$, the minimum of γ cannot decrease by these modifications. By performing these modifications for all B_i' when necessary, we get the required $\bar{\gamma}$. ◁

Now, let us consider a diffeomorphism $\psi\colon S^n \setminus \{m_k\} \to \mathbb{R}^n$ given by the stereographic projection (in particular, it maps closed balls avoiding m_k to closed balls). Let $g\colon \mathbb{R}^n \to \mathbb{R}$ be defined as $g := f \circ \psi^{-1}$. Let $n_i := \psi(m_i)$ for $i \in [k-1]$. Once we find $v \in \mathbb{R}^n$, $v \neq n_1, \dots, n_{k-1}$ such that $Dg(v) = 0$, then $u := \psi^{-1}(v)$ is the required point with $Df(u) = 0$. Note that n_1, n_2 are still local maxima of g and $n_3, \dots, n_{k-1}$ are local maxima or minima. We also set $D_i := \psi(B_i)$ and $D_i' := \psi(B_i')$ for $i \in [k-1]$ and $C_k := \psi(B_k \setminus \{m_k\})$, $C_k' := \psi(B_k' \setminus \{m_k\})$. The sets D_i and D_i' are closed (metric) balls centered in n_i whereas C_k and C_k' are complements of open (metric) balls in $\mathbb{R}^n$. Let K be the compact set obtained from $\mathbb{R}^n$ by removing the interiors of $D_1', \dots, D_{k-1}', C_k'$. Let us fix small enough $\eta > 0$ such that the closed η-neighborhood K_η of K avoids $n_1, \dots, n_{k-1}$. We will also use the notation $K_{\eta/3}$ for the closed $\frac{\eta}{3}$-neighborhood of K. See Figure 4.

Assume, for contradiction, that K_η does not contain v with $Dg(v) = 0$. Because K_η is compact and g is C^1, there is $\zeta > 0$ such that $\|Dg(w)\| \geq \zeta$ for every $w \in K_\eta$.

For every $w \in K_{\eta/3}$ let $N(w)$ be the neighborhood given by Proposition 14 (the neighborhood is considered in the whole $\mathbb{R}^n$ not only in $K_{\eta/3}$). By possibly restricting $N(w)$ to smaller sets, we can assume that each $N(w)$ is open and fits into a ball of radius $\frac{2}{3}\eta$. (In particular, if $w \in K_{\eta/3}$, then $N(w) \subseteq K_\eta$.)

▷ Claim 17. There is $\varepsilon > 0$ such that for every $x \in K_{\eta/3}$ the metric ball $B(x, \varepsilon) \subseteq \mathbb{R}^n$ centered in x of radius ε fits into $N(w)$ for some $w \in K_{\eta/3}$.

This is a variant of the Lebesgue number lemma; see the full version [19] for a proof.

Let ε be the value obtained from Claim 17. Because some ball $B(x, \varepsilon)$ fits into some $N(w)$ which fits into a ball of radius $\frac{2}{3}\eta$, we get $\varepsilon \leq \frac{2}{3}\eta$.

Let γ be a path in S^n such that

(s1) $s - \min_f(\gamma) < a_1' - a_1$;
(s2) $s - \min_f(\gamma) < a_2' - a_2$; and
(s3) $s - \min_f(\gamma) < \frac{\zeta\varepsilon}{4}$.

[8] We point out that the current online version of [14] contains a different proof of Proposition 1.14. Therefore, here we refer to the printed version of the book.

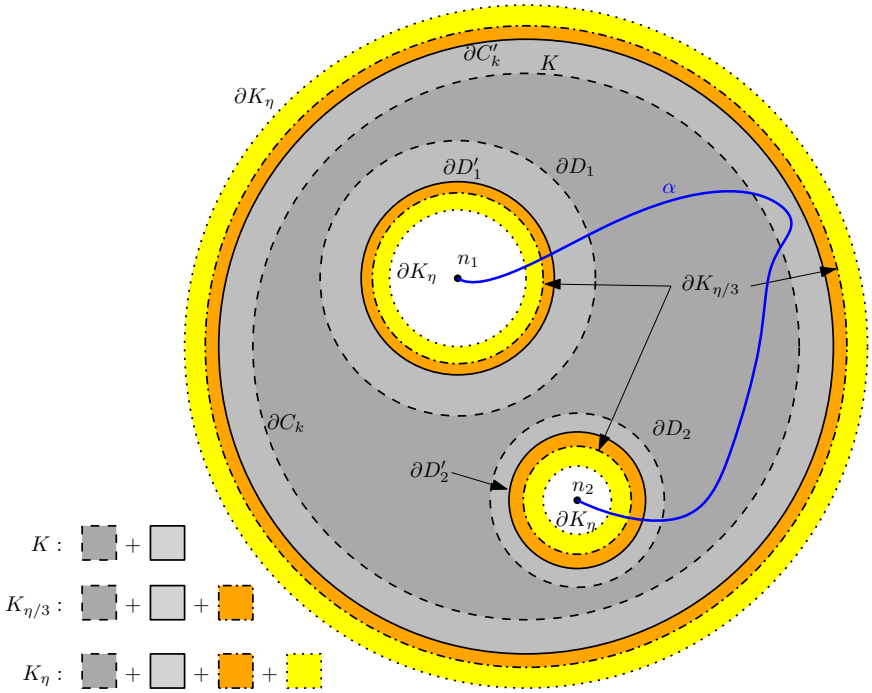


Figure 4 The sets K, $K_{\eta/3}$ and K_η and some path α connecting n_1 and n_2 of the form $\alpha = \psi \circ \gamma$ where γ is avoiding. In the picture, $k = 3$.

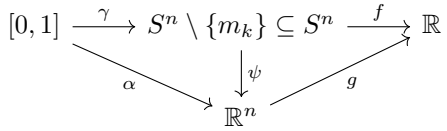


Figure 5 The maps α, γ, ψ, f and g. The two triangles are commutative.

By Claim 16, we can assume that γ is avoiding. We will start modifying γ to γ' with $\min_f(\gamma') > s$, which will be the required contradiction. Let $\alpha := \psi \circ \gamma$; see the diagram at Figure 5. Then α connects n_1 and n_2, and α avoids the interiors of $D'_3, \dots, D'_{k-1}$ and C'_k; see Figure 4.

Because, α is a continuous function on the compact interval $[0,1]$, we get, by the Heine-Cantor theorem, that α is uniformly continuous. In particular, there is $\delta > 0$ such that if $t_1, t_2 \in [0,1]$ with $|t_1 - t_2| \leq \delta$, then $\|\alpha(t_1) - \alpha(t_2)\| \leq \frac{\varepsilon}{3}$. Let us consider a positive integer $\ell > \frac{1}{\delta}$. We will be modifying α in two steps. First, we get α'' such that $\alpha''(t) > s$ if $t = \frac{j}{\ell}$ for some $j \in \{0, \dots, \ell\}$. Then we modify α'' individually on the intervals $(\frac{j}{\ell}, \frac{j+1}{\ell})$ for $j \in \{0, \dots, \ell - 1\}$ obtaining α' with $\min_g(\alpha') > s$. (Given a path $\beta\colon [0,1] \to \mathbb{R}^n$ connecting n_1 and n_2, we define $\min_g(\beta) := \min\{g(\beta(t))\colon t \in [0,1]\} = \min_f(\psi^{-1} \circ \beta)$.) The required γ' will be obtained as $\psi^{-1} \circ \alpha'$.

For the first step, let us first say that an interval $I_j = [\frac{j}{\ell}, \frac{j+1}{\ell}]$ where $j \in \{0, \dots, \ell - 1\}$ *requires a modification* if $g(\alpha(t)) \leq s$ for some $t \in I_j$. This in particular means that $\alpha(t) \in K$ for this t: Indeed, this follows from (s1) and (s2). We already know that α avoids the interiors of $D'_3, \dots, D'_{k-1}$ and C'_k. It remains to check that $\alpha(t)$ does not belong to the interiors of D'_1 and D'_2 as well. Because α has to meet ∂D_1 and ∂D_2, we get that $\min_f(\gamma) = \min_g(\alpha) \leq a_1, a_2$ from the definition of a_1 and a_2. By (s1) and (s2), we get $s < a'_1, a'_2$. Therefore, from the definition of a'_1 and a'_2, we get that $\alpha(t)$ cannot belong neither to D'_1 nor to D'_2 as required.

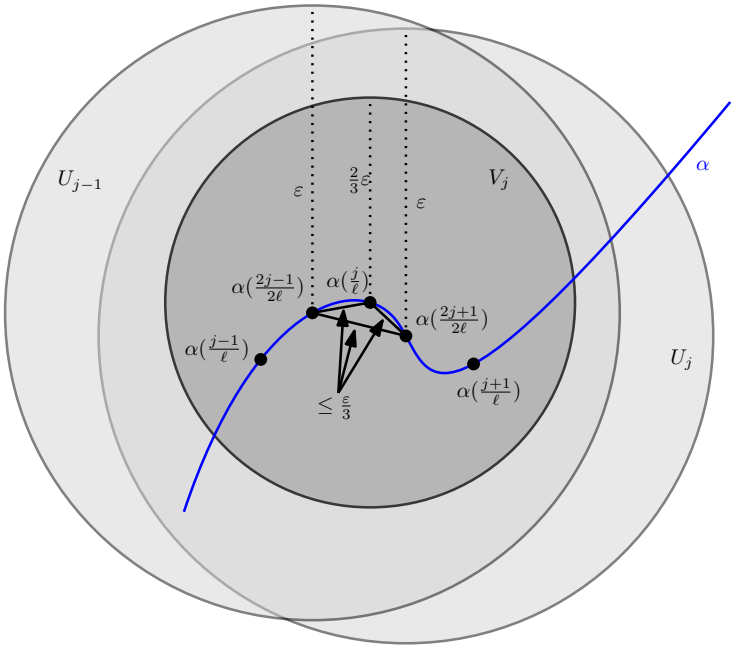


Figure 6 The sets U_{j-1}, U_j and V_j in the case that $g(\alpha(\frac{j}{\ell})) \leq s$.

By the uniform continuity, the fact that $g(\alpha(t)) \leq s$ for some $t \in I_j$ implies that $\alpha(I_j)$ belongs to the closed $\frac{\varepsilon}{3}$-neighborhood of K. In particular, $\alpha(I_j)$ belongs to $K_{\eta/3}$ as $\varepsilon \leq \frac{2}{3}\eta < \eta$.

Now, for each I_j which requires a modification, consider the open ε-ball $U_j \subseteq \mathbb{R}^n$ centered in $\alpha(\frac{2j+1}{2\ell})$. (Note that, $\frac{2j+1}{2\ell}$ is the midpoint of I_j.) From the previous considerations, the center of each U_j belongs to $K_{\eta/3}$ and the whole U_j is a subset of K_η.

Now we perform the first step. Consider $t = \frac{j}{\ell}$ for some $j \in \{0, \dots, \ell\}$. If $g(\alpha(t)) > s$, then we do nothing. Note that this includes the cases $j = 0$ or $j = \ell$. If $g(\alpha(t)) \leq s$, then both intervals I_{j-1} and I_j require a modification. By the uniform continuity, the open ball $V_j \subseteq \mathbb{R}^n$ centered in $\alpha(t)$ of radius $\frac{2\varepsilon}{3}$ is a subset of both U_{j-1} and U_j; see Figure 6. We observe that V_j is a subset of K_η as $V_j \subseteq U_j$. In particular, by the definition of ζ, we get that $\|Dg(w)\| \geq \zeta$ for every $w \in V_j$. By Lemma 15, used on a closed ball of a slightly smaller radius $\frac{\varepsilon}{2}$, there is a point v in V_j such that

$$g(v) \geq g(\alpha(t)) + \frac{\zeta\varepsilon}{4} \geq \min_g(\alpha) + \frac{\zeta\varepsilon}{4} = \min_f(\gamma) + \frac{\zeta\varepsilon}{4}.$$

Using (s3), we get $g(v) > s$. Now, by a homotopy, we modify α to α'' so that it stays fixed outside the interval $(t - \frac{1}{4\ell}, t + \frac{1}{4\ell})$, the modification of α occurs only in V_j and $\alpha''(t) = v$; see Figure 7. We perform these modifications simultaneously for every $t = \frac{j}{\ell}$ with $g(\alpha(t)) \leq s$. This is possible as the intervals $[t - \frac{1}{4\ell}, t + \frac{1}{4\ell}]$ are pairwise disjoint. This way, we obtain the required α''.

Finally, we perform the second step of the modification. Let $I_j = [\frac{j}{\ell}, \frac{j+1}{\ell}]$ be an interval requiring a modification. We already know that $g(\alpha''(\frac{j}{\ell})) > s$ and $g(\alpha''(\frac{j+1}{\ell})) > s$. In addition, we know that both $\alpha''(\frac{j}{\ell})$ and $\alpha''(\frac{j+1}{\ell})$ belong to U_j as they belong to V_j or V_{j+1}. We set $\alpha'(\frac{j}{\ell}) := \alpha''(\frac{j}{\ell})$ and $\alpha'(\frac{j+1}{\ell}) := \alpha''(\frac{j+1}{\ell})$. Next, we aim to define α' on $(\frac{j}{\ell}, \frac{j+1}{\ell})$, which is the interior of I_j, so that $\min(g(\alpha'(I_j))) > s$. By Claim 17, U_j fits into some $N(w)$ for some $w \in K_{\eta/3}$. (Here we use that the center of U_j belongs to $K_{\eta/3}$.) Now,

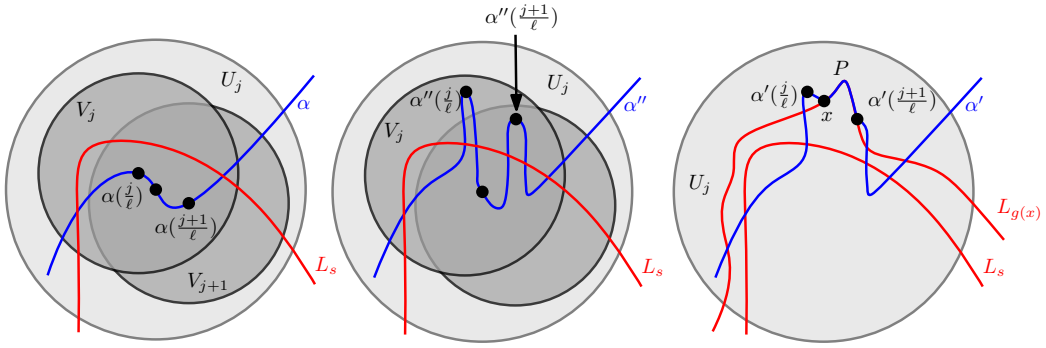


Figure 7 The first and the second step of modifications of α on an interval I_j requiring a modification (the modification is shown only on this interval).

Proposition 14 implies that $\alpha'(\frac{j}{\ell})$ and $\alpha'(\frac{j+1}{\ell})$ may be connected by a path $P\colon [0,1] \to \mathbb{R}^n$ such that $g(P(t)) > s$ for every $t \in [0,1]$: Indeed, let us assume that, without loss of generality, $g(\alpha'(\frac{j}{\ell})) \geq g(\alpha'(\frac{j+1}{\ell})) > s$. First, draw P as a straight line from $\alpha'(\frac{j}{\ell})$ towards $\alpha'(\frac{j+1}{\ell})$ until we reach a (first) point $x \in U_j \subseteq N(w)$ with $g(x) = g(\alpha'(\frac{j+1}{\ell}))$; of course, it may happen that $x = \alpha'(\frac{j+1}{\ell})$. Then by Proposition 14, x and $\alpha'(\frac{j}{\ell})$ can be connected within the level set $L_{g(x)}$; see Figure 7. (This may mean that P leaves $N(w)$, or even K_η, but this is not problem for the argument.) Altogether, we set α' on I_j so that it follows the path P, and this we do independently on each interval requiring a modification. Other intervals remain unmodified.

From the construction, we get $\min_g(\alpha') > s$; therefore the path $\gamma' := \psi^{-1} \circ \alpha'$ satisfies $\min_f(\gamma') = \min_g(\alpha') > s$ which contradicts the definition of s. ◀

References

1. P. Blagojević and R. Karasev. Local multiplicity of continuous maps between manifolds, 2016. Preprint. arXiv:1603.06723.
2. W. Blaschke. Über affine Geometrie IX: Verschiedene Bemerkungen und Aufgaben. *Ber. Verh. Sächs. Akad. Wiss. Leipzig. Math.-Nat. Kl.*, 69:412–420, 1917.
3. D. Bremner, D. Chen, J. Iacono, S. Langerman, and P. Morin. Output-sensitive algorithms for Tukey depth and related problems. *Stat. Comput.*, 18(3):259–266, 2008. doi:10.1007/s11222-008-9054-2.
4. T. M. Chan. An optimal randomized algorithm for maximum Tukey depth. In *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 430–436. ACM, New York, 2004.
5. D. Chen, P. Morin, and U. Wagner. Absolute approximation of Tukey depth: theory and experiments. *Comput. Geom.*, 46(5):566–573, 2013. doi:10.1016/j.comgeo.2012.03.001.
6. H. T. Croft, K. J. Falconer, and R. K. Guy. *Unsolved problems in geometry.* Problem Books in Mathematics. Springer-Verlag, New York, 1994. Corrected reprint of the 1991 original, Unsolved Problems in Intuitive Mathematics, II.
7. D. L. Donoho. Breakdown properties of multivariate location estimators, 1982. Unpublished qualifying paper, Harvard University.
8. D. L. Donoho and M. Gasko. Breakdown properties of location estimates based on halfspace depth and projected outlyingness. *Ann. Statist.*, 20(4):1803–1827, 1992. doi:10.1214/aos/1176348890.
9. C. Dupin. *Applications de géométrie et de méchanique, a la marine, aux ponts et chaussées, etc., pour faire suite aux Développements de géométrie, par Charles Dupin.* Bachelier, successeur de Mme. Ve. Courcier, libraire, 1822.

10 R. Dyckerhoff and P. Mozharovskyi. Exact computation of the halfspace depth. *Comput. Statist. Data Anal.*, 98:19–30, 2016. doi:10.1016/j.csda.2015.12.011.

11 B. Grünbaum. On some properties of convex sets. *Colloq. Math.*, 8:39–42, 1961. doi:10.4064/cm-8-1-39-42.

12 B. Grünbaum. Measures of symmetry for convex sets. In *Proc. Sympos. Pure Math., Vol. VII*, pages 233–270. Amer. Math. Soc., Providence, R.I., 1963.

13 A. Hassairi and O. Regaieg. On the Tukey depth of a continuous probability distribution. *Statist. Probab. Lett.*, 78(15):2308–2313, 2008.

14 A. Hatcher. *Algebraic topology*. Cambridge University Press, Cambridge, 2002.

15 R. Karasev. Geometric coincidence results from multiplicity of continuous maps, 2011. Preprint. arXiv:1106.6176.

16 J. Kynčl and P. Valtr, 2019. Personal communication.

17 X. Liu, K. Mosler, and P. Mozharovskyi. Fast computation of Tukey trimmed regions and median in dimension $p > 2$. *J. Comput. Graph. Statist.*, 28(3):682–697, 2019. doi:10.1080/10618600.2018.1546595.

18 S. Nagy, C. Schütt, and E. M. Werner. Halfspace depth and floating body. *Stat. Surv.*, 13:52–118, 2019.

19 Z. Patáková, M. Tancer, and U. Wagner. Barycentric cuts through a convex body, 2020. Preprint. arXiv:2003.13536.

20 P. J. Rousseeuw and I. Ruts. The depth function of a population distribution. *Metrika*, 49(3):213–244, 1999.

21 P. J. Rousseeuw and A. Struyf. Computing location depth and regression depth in higher dimensions. *Statistics and Computing*, 8(3):193–203, 1998.

22 C. Schütt and E. Werner. Homothetic floating bodies. *Geom. Dedicata*, 49(3):335–348, 1994.

23 J. Tukey. Mathematics and the picturing of data. In *Proceedings of the International Congress of Mathematicians (Vancouver, B. C., 1974), Vol. 2*, pages 523–531, 1975.

Sketched MinDist

Jeff M. Phillips
School of Computing, University of Utah, Salt Lake City, UT, USA
http://www.cs.utah.edu/~jeffp/
jeffp@cs.utah.edu

Pingfan Tang
School of Computing, University of Utah, Salt Lake City, UT, USA
https://my.eng.utah.edu/~pingfant/
tang1984@cs.utah.edu

Abstract

We sketch geometric objects J as vectors through the MinDist function, setting the ith coordinate

$$v_i(J) = \inf_{p \in J} \|p - q_i\|$$

for $q_i \in Q$ from a point set Q. Building a vector from these coordinate values induces a simple, effective, and powerful distance: the Euclidean distance between these sketch vectors. This paper shows how large this set Q needs to be under a variety of shapes and scenarios. For hyperplanes we provide direct connection to the sensitivity sampling framework, so relative error can be preserved in d dimensions using $|Q| = O(d/\varepsilon^2)$. However, for other shapes, we show we need to enforce a minimum distance parameter ρ, and a domain size L. For $d = 2$ the sample size Q then can be $\tilde{O}((L/\rho) \cdot 1/\varepsilon^2)$. For objects (e.g., trajectories) with at most k pieces this can provide stronger *for all* approximations with $\tilde{O}((L/\rho) \cdot k^3/\varepsilon^2)$ points. Moreover, with similar size bounds and restrictions, such trajectories can be reconstructed exactly using only these sketch vectors. Cumulatively, these results demonstrate that these MinDist sketch vectors provide an effective and efficient shape model, a compact representation, and a precise representation of geometric objects.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases curve similarity, sensitivity sampling, sketching

Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.63

Related Version available at https://arxiv.org/abs/1907.02171.

Funding *Jeff M. Phillips*: Thanks to NSF CCF-1350888, ACI-1443046, CNS- 1514520, CNS-1564287, and IIS-1816149.

1 Introduction

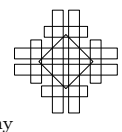
In this paper we analyze a new sketch for geometric objects, which we introduced in a recent more empirically-focused paper [23]. For an object $J \in \mathcal{J}$, where $J \subset \mathbb{R}^d$, this depends on a set of *landmarks* $Q \subset \mathbb{R}^d$; for now let $n = |Q|$. These landmarks induce a *sketched representation* $v_Q(J) \in \mathbb{R}^n$ where the ith coordinate $v_i(J)$ is defined via a MinDist operation

$$v_i(J) = \mathsf{dist}(q_i, J) = \inf_{p \in J} \|p - q_i\|,$$

using the ith landmark $q_i \in Q$. When the object J is implicit, we simply use v_i. The most useful implication of this sketch is a simple new distance $\mathtt{d}_Q$ between two objects $J_1, J_2 \in \mathcal{J}$; the Euclidean distance between the (normalized as $\bar{v}_Q = \frac{1}{\sqrt{|Q|}} v_Q$) sketched representations

$$\mathtt{d}_Q(J_1, J_2) = \left\|\bar{v}_Q(J_1) - \bar{v}_Q(J_2)\right\|.$$

A second implication we will show, is that shapes J can often be recovered exactly from the sketch $v_Q(J)$ – demonstrating the richness of information it captures.



Our recent paper [23] introduces other variants of this distance (using other norms or using the $\arg\min_{p \in J}$ points on each $J \in \mathcal{J}$). We focus on this version as it is the simplest, cleanest, easiest to use, and was the best or competitive with the best on all empirical tasks. Indeed, for the pressing case of measuring a distance between trajectories, this new distance measure dominates a dozen other distance measures (including dynamic time warping, discrete Frechet distance, edit distance for real sequences) in terms of classification performance. In practice we find we only need $|Q| = 20$ landmarks to achieve high classification accuracy. It is also considerably more efficient in clustering and nearest neighbor tasks [23]; since it uses Euclidean distance, Lloyds algorithms works for k-means clustering and extremely efficient nearest neighbor packages [1, 25] automatically work with no extra engineering.

The goal of this paper is to formally understand how many landmarks in Q are needed for various error guarantees, and how to chose the locations of these points Q.

Our aims in the choice of Q are two-fold: first, we would like to approximate $\mathtt{d}_Q$ with $\mathtt{d}_{\tilde{Q}}$, and second we would like to recover $J \in \mathcal{J}$ exactly only using $v_Q(J)$. The specific results vary depending on the initial set Q and the object class $\mathcal{J}$. More precisely, the approximation goal aims to preserve d_Q for all objects J in some class $\mathcal{J}$ with a subset $\tilde{Q} \subset Q$ of landmarks. Or possibly a weighted set of landmarks $W, \tilde{Q}$ with $|\tilde{Q}| = N$, so each q_i is associated with a weight w_i and the weighted distance is defined

$$\mathtt{d}_{\tilde{Q},W}(J_1, J_2) = \sqrt{\sum_{i=1}^{N} w_i \cdot (v_i(J_1) - v_i(J_2))^2} = \left\| \tilde{v}_{\tilde{Q}}(J_1) - \tilde{v}_{\tilde{Q}}(J_2) \right\|$$

where $\tilde{v}_{\tilde{Q}} = (\tilde{v}_1, \cdots, \tilde{v}_N)$ with $\tilde{v}_i = \sqrt{w_i} v_i$. The set Q could also represent a continuous measure ω, which replaces w, and an integral on domain Ω replaces the sum. Specifically, our aim is an $(\rho, \varepsilon, \delta)$-*approximation of* Q *over* $\mathcal{J}$ so when $W, \tilde{Q}$ is selected by a random process that succeeds with probability at least $1 - \delta$, then for a *pair* $J_1, J_2 \in \mathcal{J}$ with $\mathtt{d}_Q(J_1, J_2) \geq \rho$

$$(1 - \varepsilon)\mathtt{d}_Q(J_1, J_2) \leq \mathtt{d}_{\tilde{Q},W}(J_1, J_2) \leq (1 + \varepsilon)\mathtt{d}_Q(J_1, J_2).$$

When this holds *for all* pairs in $\mathcal{J}$, we say it is a *strong* $(\rho, \varepsilon, \delta)$-*approximation of* Q *over* $\mathcal{J}$. In some cases we set to 0 either δ (the process is deterministic) or ρ (this preserves arbitrarily small distances), and may be able to use uniform weights $w_i = \frac{1}{|\tilde{Q}|}$ for all selected points.

1.1 Our Results

We begin with a special signed variant of the distance associated with the class $\mathcal{J}$ of $(d-1)$-dimensional hyperplanes (which for instance could model linear separators or linear regression models). This has $v_i(J)$ as negative on one side of the separator. In this variant, we show that if Q is full rank, then we can recover J from $v_Q(J)$, and a variant of sensitivity sampling can be used to select $O(d/(\delta\varepsilon^2))$ points to provide a $(0, \varepsilon, \delta)$-approximation $W, \tilde{Q}$. Or by selecting $O(\frac{d}{\varepsilon^2}(d \log d + \log \frac{1}{\delta}))$ results in a strong $(0, \varepsilon, \delta)$-approximation (Theorem 2).

Next we consider the more general case where the objects are bounded geometric objects $\mathcal{S}$. For such objects it is useful to consider a bounded domain $\Omega_L = [0, L]^d$ (for d a fixed constant), and consider the case where each $S \in \mathcal{S}$ and landmarks satisfy $S, Q \subset \Omega_L$. In this case, the number of samples required for a $(\rho, \varepsilon, \delta)$-approximation is $\mathfrak{S}_Q \frac{1}{\varepsilon^2 \delta}$ where

$$\mathfrak{S}_Q = O\left(\left(\frac{L}{\rho}\right)^{\frac{2d}{2+d}} \min\left(\log \frac{L}{\eta}, \log n, \left(\frac{L}{\rho}\right)^2 \right)^{\frac{2}{2+d}} \right), \tag{1}$$

where $\eta = \min_{q,q' \in Q} \|q - q'\|_\infty$. A few special cases are worth expanding upon. When Q is continuous and uniform over Ω_L then $\mathfrak{S}_Q = O((L/\rho)^{\frac{2d}{2+d}})$, and this is tight in $\mathbb{R}^2$ at $\mathfrak{S}_Q = \Theta(L/\rho)$. That is, we can show that $\mathfrak{S}_Q = \Theta(L/\rho)$ may be needed in general. When $d = 2$ but not necessarily uniform on Ω_L, then $\mathfrak{S}_Q = O(\frac{L}{\rho} \min\{\sqrt{\log n}, L/\rho\})$. And when Q is on a grid over Ω_L in $\mathbb{R}^2$ of resolution $\Theta(\rho)$, then $\mathfrak{S}_Q = O(\frac{L}{\rho}\sqrt{\log \frac{L}{\rho}})$, just a $\sqrt{\log L/\rho}$ factor more than the lower bound.

We conclude with some specific results for trajectories, represented as piecewise-linear curves. When considering the class $\mathcal{T}_k$ with at most k segments, then $O(\frac{1}{\varepsilon^2}\mathfrak{S}_Q(k^3 \log \mathfrak{S}_Q + \log \frac{1}{\delta}))$ samples is sufficient for a *strong* $(\rho, \varepsilon, \delta)$-approximation.

Also when considering trajectories $\mathcal{T}_\tau$ where the critical points are at distance at least τ apart from any non-adjacent part of the curve, we can *exactly reconstruct* the trajectory from v_Q as long as Q is a grid of side length $\Omega(\tau)$. It is much cleaner to describe the results for trajectories and Q precisely on a grid, but these results should extend for any object with k piecewise-linear boundaries, and critical points sufficiently separated, or Q as having any point in each sufficiently dense grid cell, as opposed to being exactly on the grid lattice.

1.2 Connections to other Domains, and Core Challenges

Before deriving these results, it is useful to lay out the connection to related techniques, including ones that our results will build on, and the challenges in applying them.

Sensitivity sampling. Sensitivity sampling [20, 16, 18, 26] is an important technique for our results. This typically considers a dataset X (a subset of a metric space), endowed with a measure $\mu : X \to \mathbb{R}^+$, and a family of cost functions F. These cost functions are usually related to the fitting of a data model or a shape S to X, and for instance on a single point $x \in X$, for $f \in F$, where

$$f(x) = \mathsf{dist}(x, S)^2 = \inf_{p \in S} \|x - p\|^2$$

is the squared distance from x to the closest point p on the shape S. And then $\bar{f} = \int_X f(x)\mathsf{d}\mu(x)$. The *sensitivity* [20] of $x \in X$ w.r.t. (F, X, μ) is defined as $\sigma_{F,X,\mu}(x) := \sup_{f \in F} \frac{f(x)}{\bar{f}}$, and the *total sensitivity* of F is defined as $\mathfrak{S}(F) = \int_X \sigma_{F,X,\mu}(x)\mathsf{d}\mu(x)$. This concept is quite general, and has been widely used in applications ranging from various forms of clustering [16, 18] to dimensionality reduction [17] to shape-fitting [26]. In particular, this will allow us to draw N samples $\tilde{X}$ iid from X proportional to $\sigma_{F,X,\mu}(x)$, and weighted $\tilde{w}(\tilde{x}) = \frac{\mathfrak{S}(F)}{N \cdot \sigma_{F,X,\mu}(\tilde{x})}$; we call this $\sigma_{F,X,\mu}$*-sensitive sampling*. Then $\tilde{X}$ is a $(0, \varepsilon, \delta)$-coreset; that is, with probability $1 - \delta$ for *each* $f \in F$

$$(1 - \varepsilon)\bar{f} \leq \int_{\tilde{X}} f(\tilde{x})\mathsf{d}\tilde{w}(\tilde{x}) \leq (1 + \varepsilon)\bar{f},$$

using $N = O(\frac{\mathfrak{S}(F)}{\varepsilon^2 \delta})$ [20]. The same error bound holds for *all* $f \in F$ (then it is called a $(0, \varepsilon, \delta)$*-strong coreset*) with $N = O(\frac{\mathfrak{S}(F)}{\varepsilon^2}(\mathfrak{s}_F \log \mathfrak{S}(F) + \log \frac{1}{\delta}))$ where $\mathfrak{s}_F$ is the shattering dimension of the range space $(X, \mathsf{ranges}(F))$ [5]. Each range $r \in (X, \mathsf{ranges}(F))$ is defined as points in a sublevel set of a cost function $r = \{x \in X \mid \frac{\mu(x)}{\mathfrak{S}(F)} \frac{f(x)}{\bar{f}} \leq \xi\}$ for some $f \in F$, $\xi \in \mathbb{R}$.

It seems natural that a form of our results would follow directly from these approaches. However, two significant and intertwined challenges remain. First, our goal is to approximate the distance between a pair of sketches $\|v_Q(J_1) - v_Q(J_2)\|$, whereas these results effectively only preserve the norm of a single sketch $\|v_Q(J_1)\|$; this prohibits many of the geometric

arguments in the prior work on this subject. Second, the total sensitivity $\mathfrak{S}(F)$ associated with unrestricted Q and pairs $J_1, J_2 \in \mathcal{J}$ is in general unbounded (as we prove in Section 3.1). Indeed, if the total sensitivity was bounded, it would imply a mapping to bounded vector space [20], wherein the subtraction of the two sketches $v_Q(J_1) - v_Q(J_2)$ would still be an element of this space, and the norm bound would be sufficient.

We circumvent these challenges in two ways. First, we identify a special case in Section 2 (with negative distances, for hyperplanes) under which there is a mapping of the sketch $v_Q(J_1)$ to metric space independent of the size and structure of Q. This induces a bound for total sensitivity related to a single object, and allows the subtraction of two sketches to be handled within the same framework.

Second, we enforce a lower bound on the distance $\mathtt{d}_Q(J_1, J_2) > \rho$ and an upper bound on the domain $\Omega_L = [0, L]^d$. This induces a restricted class of pairs $\mathcal{J}_{L/\rho}$ where L/ρ is a scaleless parameter, and it shows up in bounds we are then able to produce for the total sensitivity with respect to $\mathcal{J}_{L/\rho}$ and $Q \subset \Omega_L$.

Leverage scores, and large scales. The *leverage score* [14] of the ith column a_i of matrix A is defined as $\tau_i(A) := a_i^T (AA^T)^+ a_i$, where $(\cdot)^+$ is the Moore-Penrose pseudoinverse. This definition is more specific and linear-algebraic than sensitivity, but has received more attention for scalable algorithm development and approximation [14, 4, 13, 10, 22, 11].

However, the full version shows that if F is the collection of some functions defined on a set Q of n points ($\mu(q_i) = \frac{1}{n}$ for all $q_i \in Q$), where each $f \in F$ is the square of some function v in a finite dimensional space V spanned by a basis $\{v^{(1)}, \cdots, v^{(\kappa)}\}$, then we can build a $\kappa \times n$ matrix A where the ith column is $\frac{1}{\sqrt{n}}\big(v^{(1)}(q_i), \cdots, v^{(\kappa)}(q_i)\big)^T$, and then $\frac{1}{n} \cdot \sigma_{F,Q,\mu}(q_i)$ is precisely the leverage score of the ith column of the matrix A. A similar observation has been made by Varadarajan and Xiao [26].

A concrete implication of this connection is that we can invoke an online row sampling algorithm of Cohen *et al.* [11]. In our context, this algorithm would stream over Q, maintaining (ridge) estimates of the sensitivity of each q_i from a sample $\tilde{Q}_{i-1}$, and retaining each q_i in that sample based on this estimate. This provides a streaming approximation bound not much weaker than the sampling or gridding bounds we present; see full version.

MinDist and shape reconstruction. The fields of computational topology and surface modeling have extensively explored [6, 24, 8] the distance function to a compact set $J \subset \mathbb{R}^d$

$$\mathtt{d}_J(x) = \mathsf{dist}(x, J) = \inf_{p \in J} \|x - p\|,$$

their approximations, and the offsets $J^r = \mathtt{d}_J^{-1}([0, r])$. For instance the Hausdorff distance between two compact sets J, J' is $\mathtt{d}_{\mathsf{H}}(J, J') = \|\mathtt{d}_J - \mathtt{d}_{J'}\|_\infty$. The gradient of $\mathtt{d}_J$ implies stability properties about the medial axis [9]. And most notably, this stability of $\mathtt{d}_J$ with respect to a sample $P \sim J$ or $P \sim \partial J$ is closely tied to the development of shape reconstruction (aka geometric and topological inference) through α-shapes [15], power crust [2], and the like. The intuitive formulation through $\mathtt{d}_J$ (as opposed to Voronoi diagrams of P) has led to more statistically robust variants [8, 24] which also provide guarantees in shape recovery up to small feature size [7], essentially depending on the maximum curvature of ∂J.

Our formulation flips this around. Instead of considering samples P from J (or ∂J) we consider samples Q from some domain $\Omega \subset \mathbb{R}^d$. This leads to new but similar sampling theory, still depending on some feature size (represented by various scale parameters ρ, τ, and η), and still allowing recovery properties of the underlying objects. While the samples P

from J can be used to estimate Hausdorff distance via an all-pairs $O(|P|^2)$-time comparison, our formulation requires only a $O(|Q|)$-time comparison to compute $\mathtt{d}_Q$. We leave as open questions the recovering of topological information about an object $J \in \mathcal{J}$ from $v_Q(J)$.

2 The Distance Between Two Hyperplanes using Signed Sketches

A more detailed derivation of the results in this section are presented in the full version where proofs and a few technical details require more careful notation to navigate.

Let $\mathcal{H} = \{h \mid h \text{ is a hyperplane in } \mathbb{R}^d\}$ represent the space of all hyperplanes. Each hyperplane h can be represented by a vector $u \in \mathbb{R}^{d+1}$ composed as a normal vector $\bar{u} = (u_1, \ldots, u_d) \in \mathbb{R}^d$ with $\|\bar{u}\| = 1$ and offset u_{d+1}. Then the ith coordinate of a sketch vector can be derived as a *signed* distance from q_i as $v_i(h) = u_{d+1} + \langle \bar{u}, q_i \rangle$.

Recovery. Our recent paper [23] showed that if Q is full rank (there exist $d+1$ points in Q not on a common hyperplane) then $\mathtt{d}_Q(h_1, h_2) \neq 0$ if $h_1 \neq h_2$, and thus $\mathtt{d}_Q$ is a metric. This full rank condition on Q is also sufficient to recover h from $v_Q(h)$; e.g., using PCA.

Distance Preservation. Next we show that we can use a σ-sensitive sample $\tilde{Q}, W$ as a $(0, \varepsilon, \delta)$-coreset for this formulation; that is $\mathtt{d}_{\tilde{Q},W}$ preserves relative error with respect to $\mathtt{d}_Q$. We assume Q is full rank, and has uniform weight $\mu = \frac{1}{n}$ on each point. Using $X = Q$ we need to define the family of functions F to complete the tuple (F, Q, μ). To this end, let V be a $(d+1)$-dimensional function space with each element v_u is a linear function defined $v_u(q_i) = v_i(h) = u_{d+1} + \langle \bar{u}, q_i \rangle$. Now each $f \in F$ is defined as $f(q) = v(q)^2$, and through its representation u, each $h \in \mathcal{H}$ maps to a unique element of F.

However, we are interested in preserving $\mathtt{d}_Q$ which requires a pair $h_1, h_2 \in \mathcal{H}$, with corresponding normals $u^{(1)}, u^{(2)}$. Since V is a linear function space, then using $u = u^{(1)} - u^{(2)}$, then $f_{h_1,h_2}(q) = v_u(q)^2$, and $\mathtt{d}_Q(h_1, h_2) = (\frac{1}{n} \sum_{q \in Q} f_{h_1,h_2}(q))^{\frac{1}{2}}$. Note that u will likely not correspond to a single halfspace since the first d coordinates may not be a unit vector, but that is not an issue for this framework. Using Langberg and Schulman [20], the total sensitivity is $d+1$, and the sensitivities can be calculated (e.g., via leverage scores).

▶ **Theorem 1.** *Consider full rank $Q \subset \mathbb{R}^d$ and halfspaces $\mathcal{H}$ with $\varepsilon, \delta \in (0,1)$. A σ-sensitive sample $\tilde{Q}$ of (Q, F) of size $|\tilde{Q}| \geq \frac{d+1}{\delta \varepsilon^2}$ results in a $(0, \varepsilon, \delta)$-coreset. It is an $(0, \varepsilon, \delta)$-approximation so with probability at least $1 - \delta$, for each pair $h_1, h_2 \in \mathcal{H}$*

$$(1 - \varepsilon)\mathtt{d}_Q(h_1, h_2) \leq \mathtt{d}_{\tilde{Q},W}(h_1, h_2) \leq (1 + \varepsilon)\mathtt{d}_Q(h_1, h_2).$$

We can also achieve a strong coreset for this variant using results from Braverman *et al.* [5]. For this we need to provide an additional bound about the shattering dimension $\mathfrak{s} = \dim(Q, \mathcal{X})$ associated with each $f \in F$ and a weight $w : Q \to \mathbb{R}_+$. The range in the range space associated with f_{h_1,h_2} is defined for some η as

$$X_{h_1,h_2,\eta} = \{q \in Q \mid w(q) f_{h_1,h_2}(q) \leq \eta\}.$$

For $f \in F$ defined by a single halfspace, this is classically known to be $O(d)$. For the more general functions $f_{h_1,h_2} \in F$ defined by two halfspaces h_1, h_2, the same asymptotic bound can be shown using straight-forward decomposition properties of range spaces (see full version for proof). Then we can obtain the following result.

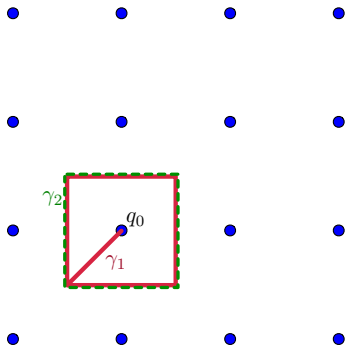


Figure 1 Q is the set of blue points, γ_1 is the red curve, γ_2 is the green (dashed) curve, and they coincide with each other on the boundary of the square.

▶ **Theorem 2.** *Consider full rank $Q \subset \mathbb{R}^d$ and halfspaces $\mathcal{H}$ with $\varepsilon, \delta \in (0,1)$. A σ-sensitive sample $\tilde{Q}$ of (Q,F) of size $|\tilde{Q}| = O(\frac{d}{\varepsilon^2}(d \log d + \log \frac{1}{\delta}))$ results in a strong $(0,\varepsilon,\delta)$-coreset. And thus a strong $(0,\varepsilon,\delta)$-approximation so with probability at least $1-\delta$, for all $h_1, h_2 \in \mathcal{H}$*

$$(1-\varepsilon)\mathtt{d}_Q(h_1,h_2) \leq \mathtt{d}_{\tilde{Q},W}(h_1,h_2) \leq (1+\varepsilon)\mathtt{d}_Q(h_1,h_2).$$

3 Sketched MinDist for Two Geometric Objects

In this section, we mildly restrict $\mathtt{d}_Q$ to the distance between any two geometric objects, in particular, bounded closed sets. Let $\mathcal{S} = \{S \subset \mathbb{R}^d \mid S \text{ is a bounded closed set}\}$ be the space of objects $\mathcal{J}$ we consider.

As before define $v_i(S) = \inf_{p \in S} \|p - q_i\|$, and then for $S_1, S_2 \in \mathcal{S}$ define $f_{S_1,S_2}(q_i) = (v_i(S_1) - v_i(S_2))^2$. The associated function space is $F(\mathcal{S}) = \{f_{S_1,S_2} \mid S_1, S_2 \in \mathcal{S}\}$. Setting $\mu(q) = \frac{1}{n}$ for all $q \in Q$, then $(\mathtt{d}_Q(S_1,S_2))^2 = \bar{f}_{S_1,S_2} := \sum_{i=1}^n \mu(q_i) f_{S_1,S_2}(q_i)$. Using sensitivity sampling to estimate $\mathtt{d}_Q(S_1,S_2)$ requires a bound on the total sensitivity of $F(\mathcal{S})$.

We show that while the total sensitivity $\mathfrak{S}(F(\mathcal{S}))$ is unbounded in general, it is tied to the ratio L/ρ between the diameter of the domain L, and the minimum allowed $\mathtt{d}_Q$ distance between objects ρ. In particular, it can be at least proportional to this, and in $\mathbb{R}^2$ in most cases (e.g., for near-uniform Q) is at most proportional to L/ρ or not much larger for any Q.

3.1 Lower Bound on Total Sensitivity

Suppose Q is a set of n points in $\mathbb{R}^2$ and no two points are at the same location, then for any $q_0 \in Q$ we can draw two curves γ_1, γ_2 as shown in Figure 1, where γ_1 is composed by five line segments and γ_2 is composed by four line segments. The four line segments of the γ_2 forms a square, on its boundary γ_1 and γ_2 coincide with each other, and inside this square, q_0 is the endpoint of γ_1. We can make this square small enough, such that all points $q \neq q_0$ are outside this square. So, we have $\mathsf{dist}(q_0,\gamma_1) = 0$ and $\mathsf{dist}(q_0,\gamma_2) \neq 0$, and $\mathsf{dist}(q,\gamma_1) = \mathsf{dist}(q,\gamma_2) = 0$ for all $q \neq q_0$. Thus, we have $f_{\gamma_1,\gamma_2}(q_0) > 0$ and $f_{\gamma_1,\gamma_2}(q) = 0$ for all $q \neq q_0$, which implies

$$\sigma_{F(\mathcal{S}),Q,\mu}(q_0) \geq \frac{f_{\gamma_1,\gamma_2}(q_0)}{\bar{f}_{\gamma_1,\gamma_2}} = \frac{f_{\gamma_1,\gamma_2}(q_0)}{\frac{1}{n}\sum_{q \in Q} f_{\gamma_1,\gamma_2}(q)} = \frac{n f_{\gamma_1,\gamma_2}(q_0)}{f_{\gamma_1,\gamma_2}(q_0)} = n.$$

Since this construction of two curves γ_1, γ_2 can be repeated around any point $q \in Q$,

$$\mathfrak{S}(F(\mathcal{S})) = \sum_{q \in Q} \mu(q)\sigma_{F(\mathcal{S}),Q,\mu}(q) \geq \sum_{q \in Q} \frac{1}{n} n = n.$$

We can refine this bound by introducing two parameters L, ρ for $\mathcal{S}$. Given $L > \rho > 0$ and a set $Q \subset \mathbb{R}^d$ of n points, we define $\mathcal{S}(L) = \{S \in \mathcal{S} \mid S \subset [0, L]^d\}$ and $F(\mathcal{S}(L), \rho) = \{f_{S_1,S_2} \in F(\mathcal{S}) \mid S_1, S_2 \in \mathcal{S}(L),\ \mathtt{d}_Q(S_1, S_2) \geq \rho\}$. The following lowerbounds the total sensitivity of $F(\mathcal{S}(L), \rho)$ for $d = 2$; it holds for any $d \geq 2$ using the construction in a 2d subspace.

▶ **Lemma 3.** *For $d = 2$ we can construct a set $Q \subset [0, L]^2$ such that $\mathfrak{S}(F(\mathcal{S}(L), \rho)) = \Omega(\frac{L}{\rho})$.*

Proof. We uniformly partition $[0, L]^2$ into n grid cells, such that $C_1 \frac{L}{\rho} \leq n \leq C_2 \frac{L}{\rho}$ for constants $C_1, C_2 \in (0, 1)$. The side length of each grid is $\eta = \frac{L}{\sqrt{n}}$. We take Q as the n grid points, and for each point $q \in Q$ we can choose two curves γ_1 and γ_2 (similar to curves in Figure 1) such that $\mathsf{dist}(q, \gamma_1) = 0$, $\mathsf{dist}(q, \gamma_2) \geq C_2\eta$, and $\mathsf{dist}(q', \gamma_1) = \mathsf{dist}(q', \gamma_2) = 0$ for all $q' \in Q \setminus \{q\}$. Thus, $\mathtt{d}_Q(\gamma_1, \gamma_2) \geq C_2 \frac{\eta}{\sqrt{n}} = C_2 \frac{L}{n} \geq \rho$. So, $f_{\gamma_1,\gamma_2} \in F(\mathcal{S}(L), \rho))$ and $\sigma(q) \geq n$ for all $q \in Q$ and $\mathfrak{S}(F(\mathcal{S}(L), \rho)) \geq n \geq C_1 \frac{L}{\rho}$, which implies $\mathfrak{S}(F(\mathcal{S}(L), \rho)) = \Omega(\frac{L}{\rho})$. ◀

3.2 Upper Bound on the Total Sensitivity

A simple upper bound of $\mathfrak{S}(F(\mathcal{S}(L), \rho)$ is $O(\frac{L^2}{\rho^2})$; it follows from the L/ρ constraint. The sensitivity of each point $q \in Q$ is defined as $\sup_{f_{S_1,S_2} \in F(\mathcal{S}(L),\rho)} \frac{f_{S_1,S_2}(q)}{\bar{f}_{S_1,S_2}}$, where $f_{S_1,S_2}(q) = O(L^2)$ for all $S_1, S_2 \in \mathcal{S}(L)$ and $q \in Q \subset [0, L]^d$, and the denominator $\bar{f}_{S_1,S_2} \geq \rho^2$ by assumption for all $f_{S_1,S_2} \in F(\mathcal{S}(L), \rho)$. Hence, the sensitivity of each point in Q is $O(\frac{L^2}{\rho^2})$, and thus their average, the total sensitivity is $O(\frac{L^2}{\rho^2})$. In this section we will improve and refine this bound.

We introduce two variables that only depends on $Q = \{q_1, \cdots, q_n\} \subset [0, L]^d$:

$$C_q := \max_{0<r\leq L} \frac{r^d}{L^d} \frac{n}{|Q \cap B_\infty(q, r)|} \quad \text{for } q \in Q, \text{ and } C_Q := \frac{1}{n} \sum_{q \in Q} C_q^{\frac{2}{2+d}}. \tag{2}$$

where $B_\infty(q, r) := \{x \in \mathbb{R}^d \mid \|x - q\|_\infty \leq r\}$. Intuitively, $\frac{|Q \cap B_\infty(q,r)|}{r^d}$ is proportional to the point density in region $B_\infty(q, r)$, and the value of $\frac{r^d}{L^d} \frac{n}{|Q \cap B_\infty(q,r)|}$ can be maximized, when the region $B_\infty(q, r)$ has smallest point density, which means r should be as large as possible but the number of points contained in $B_\infty(q, r)$ should be as small as possible. A trivial bound of C_q is n, but if we make $C_{q_0} = n$ for one point q_0, then it implies the value of C_q for other points will be small, so for C_Q it is possible to obtain a bound better than $n^{\frac{2}{d+2}}$.

Importantly, these quantities C_q and C_Q will be directly related to the sensitivity of a single point $\sigma(q)$ and the total sensitivity of the point set $\mathfrak{S}_Q$, respectively. We formalize this in two technical lemmas: First (in Lemma 7) $\sigma(q) \leq O((C_q(L/\rho)^d)^{\frac{2}{2+d}})$ and hence $\mathfrak{S}_Q = O(C_Q \cdot (L/\rho)^{\frac{2d}{2+d}})$; and second (in Lemma 8) we show $C_Q \leq O((\min\{\log \frac{L}{\eta}, \log n\})^{\frac{2}{2+d}})$ for Q of size n and $\eta = \min_{q,q' \in Q,\ q \neq q'} \|q - q'\|_\infty$.

Since $f_{S_1,S_2} \in F(\mathcal{S}(L), \rho)$, we know $f_{S_1,S_2}(q) \leq dL^2$ for all $q \in Q$ and $\frac{1}{n} \sum_{q' \in Q} f_{S_1,S_2}(q') \geq \rho^2$, so $\sigma(q) \leq \frac{dL^2}{\rho^2}$ for all $q \in Q$. Thus, we can expand $\frac{1}{|Q|} \sum_{q \in Q} \sigma(q)$ using Lemma 7 and factor out C_Q using Lemma 8 to immediately obtain the following theorem.

▶ **Theorem 4.** *Suppose $L > \rho > 0$, $Q = \{q_1, \cdots, q_n\} \subset [0, L]^d$ and $\eta = \min_{q,q' \in Q,\ q \neq q'} \|q - q'\|_\infty$. Then, we have*

$$\mathfrak{S}(F(\mathcal{S}(L), \rho)) \leq \mathfrak{S}_Q = O\left(\left(\frac{L}{\rho}\right)^{\frac{2d}{2+d}} \min\left(\log \frac{L}{\eta}, \log n, \left(\frac{L}{\rho}\right)^2\right)^{\frac{2}{2+d}}\right).$$

From Lemma 7 and Theorem 4, using [20][Lemma 2.1] we can obtain the following.

▶ **Theorem 5.** *Let $L > \rho > 0$, $Q = \{q_1, \cdots, q_n\} \subset [0, L]^d$, $S_1, S_2 \in \mathcal{S}(L)$ and $\mathtt{d}_Q(S_1, S_2) \geq \rho$. Then for $\delta, \varepsilon \in (0, 1)$ a σ-sensitive sampling of size $N \geq \frac{\mathfrak{S}_Q}{\delta\varepsilon^2}$ provides $\tilde{Q}$, a $(\rho, \varepsilon, \delta)$-coreset; that is with probability at least $1 - \delta$, we have*

$$(1 - \varepsilon)\mathtt{d}_Q(S_1, S_2) \leq \mathtt{d}_{\tilde{Q},W}(S_1, S_2) \leq (1 + \varepsilon)\mathtt{d}_Q(S_1, S_2).$$

If Q describes a continuous uniform distribution in $[0, L]^d$ (or sufficiently close to one, like points on a grid), then there exists an absolute constant $C > 0$ such that $C_q \leq C$ for all $q \in Q$, then in Lemma 7 $\sigma(q) \leq C_d\big(\frac{L}{\rho}\big)^{\frac{2d}{2+d}}$ for all $q \in Q$, and in Theorem 4 $\mathfrak{S}_Q \leq C_d\big(\frac{L}{\rho}\big)^{\frac{2d}{2+d}}$. So, for uniform distribution, the sample size of Q in Theorem 5 is independent from the size of Q, and for $d = 2$ the bound $\mathfrak{S}_Q = O(L/\rho)$ matches the lower bound in Lemma 3.

▶ **Corollary 6.** *If Q describes the continuous uniform distribution over $[0, L]^d$, then the sample size in Theorem 5 can be reduced to $N = O\Big(\big(\frac{L}{\rho}\big)^{\frac{2d}{2+d}} \frac{1}{\delta\varepsilon^2}\Big)$.*

Technical lemmas bounding $\sigma(q)$ and C_Q.

▶ **Lemma 7.** *For function family $F(\mathcal{S}(L), \rho)$ the sensitivity for any $q \in Q \subset [0, L]^d$ is bounded*

$$\sigma(q) \leq C_d C_q^{\frac{2}{2+d}} \left(\frac{L}{\rho}\right)^{\frac{2d}{2+d}},$$

where $C_d = 4^{\frac{2}{2+d}}(8\sqrt{d})^{\frac{2d}{2+d}}$ and C_q given by (2).

Proof. Recall $\sigma(q) = \sup_{f_{S_1,S_2} \in F(\mathcal{S}(L),\rho)} \frac{f_{S_1,S_2}(q)}{\frac{1}{n}\sum_{q' \in Q} f_{S_1,S_2}(q')}$. For any fixed $q \in Q$, for now suppose $f_{S_1,S_2} \in F(\mathcal{S}(L), \rho)$ satisfies this supremum $\sigma(q) = \frac{f_{S_1,S_2}(q)}{\frac{1}{n}\sum_{q' \in Q} f_{S_1,S_2}(q')}$. We define $\mathsf{dist}(q, S) = \inf_{p \in S} \|q - p\|$ (so for $q_i \in Q$ then $\mathsf{dist}(q_i, S) = v_i(S)$), and then use the parameter $M := |\mathsf{dist}(q, S_1) - \mathsf{dist}(q, S_2)|$, where $M^2 = f_{S_1,S_2}(q)$. If $M = 0$, then obviously $f_{S_1,S_2}(q) = M^2 = 0$, and $\sigma(q) = 0$. So, without loss of generality, we assume $M > 0$ and $\mathsf{dist}(q, S_1) = \tau$ and $\mathsf{dist}(q, S_2) = \tau + M$. We first prove $\sigma(q) \leq C_d C_q \frac{L^d}{M^d}$. There are two cases for the relationship between τ and M, as shown in Figure 2.

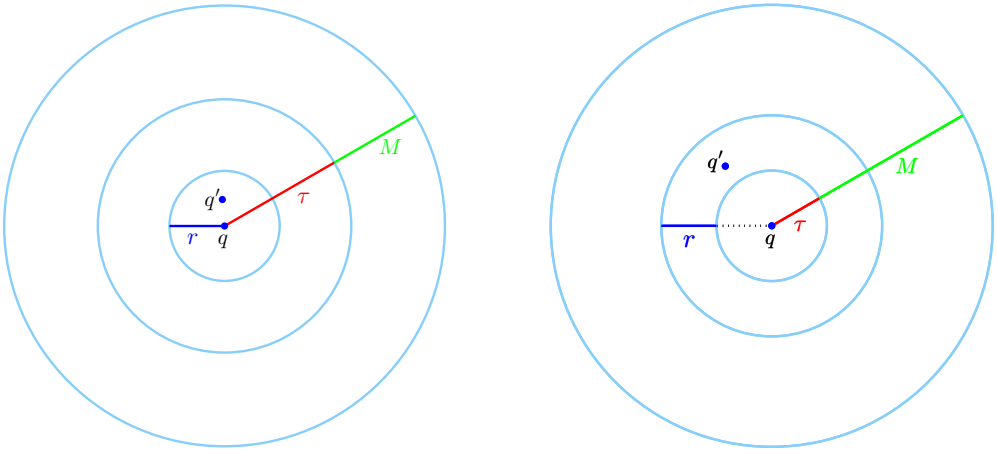


Figure 2 Left: Case 1, $r = \frac{M}{8} \leq \tau$, and $q' \in B(q, r)$. Right: Case 2, $r = \frac{M}{8} > \tau$, and $q' \in B(q, \tau + r)$.

Case 1: $\tau \geq \frac{M}{8}$. For any $q' \in B(q, \frac{M}{8}) := \{q' \in \mathbb{R}^d \mid \|q' - q_i\| \leq \frac{M}{8}\}$, we have $\tau + M = \mathsf{dist}(q, S_2) \leq \mathsf{dist}(q, q') + \mathsf{dist}(q', S_2) \leq \frac{M}{8} + \mathsf{dist}(q', S_2)$, which implies for all $q' \in B(q, \frac{M}{8})$

$$\mathsf{dist}(q', S_2) \geq \tau + M - \frac{M}{8} = \tau + \frac{7}{8}M.$$

Similarly $\mathsf{dist}(q', S_1) \leq \mathsf{dist}(q', q) + \mathsf{dist}(q, S_1) \leq \frac{M}{8} + \tau$ for all $q' \in B(q, \frac{M}{8})$. Thus for all $q' \in B(q, \frac{M}{8})$

$$|\mathsf{dist}(q', S_2) - \mathsf{dist}(q', S_1)| \geq \mathsf{dist}(q', S_2) - \mathsf{dist}(q', S_1) \geq \tau + \frac{7}{8}M - (\tau + \frac{M}{8}) = \frac{3}{4}M.$$

Case 2: $0 \leq \tau < \frac{M}{8}$. For any $q' \in B(q, \tau + \frac{M}{8}) := \{q' \in \mathbb{R}^d \mid \mathsf{dist}(q', q) \leq \tau + \frac{M}{8}\}$, we have $\tau + M = \mathsf{dist}(q', S_2) \leq \mathsf{dist}(q, q') + \mathsf{dist}(q', S_2) \leq \tau + \frac{M}{8} + \mathsf{dist}(q', S_2)$, which implies for all $q' \in B(q, \tau + \frac{M}{8})$

$$\mathsf{dist}(q', S_2) \geq \frac{7}{8}M.$$

Combined with $\tau < \frac{M}{8}$ and $\mathsf{dist}(q', S_1) \leq \mathsf{dist}(q', q) + \mathsf{dist}(q, S_1) \leq \tau + \frac{M}{8} + \tau = \frac{M}{8} + \frac{M}{8} + \frac{M}{8} \leq \frac{3}{8}M$ for all $q' \in B(q, \tau + \frac{M}{8})$, we have

$$|\mathsf{dist}(q', S_2) - \mathsf{dist}(q', S_1)| \geq \mathsf{dist}(q', S_2) - \mathsf{dist}(q', S_1) \geq \frac{7}{8}M - \frac{3}{8}M = \frac{M}{2}.$$

Combining these two cases on τ, for all $q' \in B(q, \frac{M}{8})$ $|\mathsf{dist}(q', S_2) - \mathsf{dist}(q', S_1)| \geq \frac{M}{2}$. Then since $B_\infty(q, \frac{r}{\sqrt{d}}) \subset B(q, r)$ for all $r \geq 0$, from

$$C_q = \max_{0<r\leq L} \frac{r^d}{L^d} \frac{n}{|Q \cap B_\infty(q, r)|} \geq (\frac{1}{8\sqrt{d}})^d \frac{M^d}{L^d} \frac{n}{|Q \cap B_\infty(q, \frac{M}{8\sqrt{d}})|},$$

we can bound the denominator in $\sigma(q)$ as

$$\begin{aligned}\frac{1}{n}\sum_{q'\in Q} f_{S_1,S_2}(q') &\geq \frac{1}{n} \sum_{q'\in Q\cap B_\infty(q,\frac{M}{8\sqrt{d}})} f_{S_1,S_2}(q') = \frac{1}{n} \sum_{q'\in Q\cap B_\infty(q,\frac{M}{8\sqrt{d}})} (\mathsf{dist}(q', S_1) - \mathsf{dist}(q', S_2))^2 \\ &\geq \frac{1}{4}\frac{1}{n}M^2 \left|Q \cap B_\infty(q, \frac{M}{8\sqrt{d}})\right| \geq \frac{1}{4}(\frac{1}{8\sqrt{d}})^d \frac{M^2}{C_q}\frac{M^d}{L^d} = \frac{1}{4}(\frac{1}{8\sqrt{d}})^d \frac{1}{C_q}\frac{M^{2+d}}{L^d},\end{aligned}$$

which implies

$$\sigma(q) = \frac{M^2}{\frac{1}{n}\sum_{q'\in Q} f_{S_1,S_2}(q')} \leq 4(8\sqrt{d})^d M^2 C_q \frac{L^d}{M^{2+d}} = 4(8\sqrt{d})^d C_q \frac{L^d}{M^d}.$$

Combining this with $\sigma(q) \leq \frac{M^2}{\rho^2}$, we have $\sigma(q) \leq \min\left(\frac{M^2}{\rho^2}, 4(8\sqrt{d})^d C_q \frac{L^d}{M^d}\right)$. If $M^{2+d} \leq 4(8\sqrt{d})^d C_q \rho^2 L^d$, then $\frac{M^2}{\rho^2} \leq 4(8\sqrt{d})^d C_q \frac{L^d}{M^d}$, which means $\sigma(q) \leq \min\left(\frac{M^2}{\rho^2}, 4(8\sqrt{d})^d C_q \frac{L^d}{M^d}\right) = \frac{M^2}{\rho^2} \leq 4^{\frac{2}{2+d}}(8\sqrt{d})^{\frac{2d}{2+d}} C_q^{\frac{2}{2+d}} \left(\frac{L}{\rho}\right)^{\frac{2d}{2+d}}$. If $M^{2+d} \geq 4(8\sqrt{d})^d C_q \rho^2 L^d$, then $4(8\sqrt{d})^d C_q \frac{L^d}{M^d} \leq \frac{M^2}{\rho^2}$, so $\sigma(q) \leq \min\left(\frac{M^2}{\rho^2}, 4(8\sqrt{d})^d C_q \frac{L^d}{M^d}\right) = 4(8\sqrt{d})^d C_q \frac{L^d}{M^d} \leq 4^{\frac{2}{2+d}}(8\sqrt{d})^{\frac{2d}{2+d}} C_q^{\frac{2}{2+d}} \left(\frac{L}{\rho}\right)^{\frac{2d}{2+d}}$. ◀

Hence, to bound the total sensitivity of $F(\mathcal{S}(L), \rho)$, we need a bound of $C_Q = \frac{1}{n}\sum_{q\in Q} C_q^{\frac{2}{2+d}}$.

▶ **Lemma 8.** *Suppose $Q \subset [0,L]^d$ of size n, $\eta = \min_{q,q' \in Q,\ q \neq q'} \|q - q'\|_\infty$, and C_Q is given by (2). Then using $C_d = 2^{d+1}$ we have*

$$C_Q \leq C_d \min\Big(\big(\log_2 \frac{L}{\eta}\big)^{\frac{2}{2+d}}, \big(\frac{1}{d}\log_2 n\big)^{\frac{2}{2+d}} \Big).$$

Proof. We define $\widetilde{C}_Q := \frac{1}{n}\sum_{q\in Q} C_q$, and using Hölder inequality we have

$$C_Q = \frac{1}{n}\sum_{q\in Q} C_q^{\frac{2}{2+d}} \leq \frac{1}{n}\Big(\sum_{q\in Q} C_q\Big)^{\frac{2}{2+d}} n^{\frac{d}{2+d}} = \Big(\frac{1}{n}\sum_{q\in Q} C_q\Big)^{\frac{2}{2+d}} = (\widetilde{C}_Q)^{\frac{2}{2+d}}.$$

So, we only need to bound $\widetilde{C}_Q$.

We define $r_q := \arg\max_{0<r\leq L} \frac{r^d}{L^d} \frac{n}{|Q \cap B_\infty(q,r)|}$ for all $q \in Q$, $Q_i := \{q \in Q \mid \frac{L}{2^{i+1}} < r_q \leq \frac{L}{2^i}\}$, and $A := \{i \geq 0 \mid i \text{ is an integer and } |Q_i| > 0\}$.

For any fixed $i \in A$, we use $l_i := \frac{L}{2^{i+1}}$ as the side length of grid cell to partition the region $[0,L]^d$ into $s_i = (\frac{L}{l_i})^d = 2^{(i+1)d}$ grid cells: $\Omega_1 . \cdots, \Omega_{s_i}$ where each Ω_j is a closed set, and define $Q_{i,j} := Q_i \cap \Omega_j$. Then, $|Q_i \cap \bar{B}_\infty(q,l_i)| \geq |Q_{i,j}|$ for all $q \in Q_{i,j}$ where $\bar{B}_\infty(q,l_i) := \{q' \in \mathbb{R}^d \mid \|q' - q\|_\infty \leq l_i\}$, and we have

$$\begin{aligned}
\sum_{q\in Q_i} \frac{r_q^d}{L^d} \frac{1}{|Q_i \cap B_\infty(q,r_q)|} &\leq \sum_{q\in Q} \frac{L^d}{2^{id}L^d} \frac{1}{|Q_i \cap B_\infty(q,r_q)|} \leq \frac{1}{2^{id}} \sum_{q\in Q_i} \frac{1}{|Q_i \cap \bar{B}_\infty(q,l_i)|} \\
&\leq \frac{1}{2^{id}} \sum_{j\in[s_i],|Q_{i,j}|>0} \sum_{q\in Q_{i,j}} \frac{1}{|Q_i \cap \bar{B}_\infty(q,l_i)|} \leq \frac{1}{2^{id}} \sum_{j\in[s_i],|Q_{i,j}|>0} \sum_{q\in Q_{i,j}} \frac{1}{|Q_{i,j}|} \\
&= \frac{1}{2^{id}} \sum_{j\in[s_i],|Q_{i,j}|>0} \frac{|Q_{i,j}|}{|Q_{i,j}|} \leq \frac{s_i}{2^{id}} = \frac{2^{(i+1)d}}{2^{id}} = 2^d.
\end{aligned}$$

Then using the definitions of $\widetilde{C}_Q$ and r_q we have

$$\begin{aligned}
\widetilde{C}_Q &= \sum_{q\in Q} \max_{0<r\leq L} \frac{r^d}{L^d} \frac{1}{|Q \cap B_\infty(q,r)|} = \sum_{q\in Q} \frac{r_q^d}{L^d} \frac{1}{|Q \cap B_\infty(q,r_q)|} = \sum_{i\in A}\sum_{q\in Q_i} \frac{r_q^d}{L^d} \frac{1}{|Q \cap B_\infty(q,r)|} \\
&\leq \sum_{i\in A}\sum_{q\in Q_i} \frac{r_q^d}{L^d} \frac{1}{|Q_i \cap B_\infty(q,r)|} \leq \sum_{i\in A} 2^d = 2^d|A|.
\end{aligned}$$

We assert $r_q \geq Ln^{-\frac{1}{d}}$ for all $q \in Q$. This is because for any $r \in (0, Ln^{-\frac{1}{d}})$ we have

$$\frac{r^d}{L^d} \frac{n}{|Q \cap B_\infty(q,r)|} \leq \frac{L^d}{nL^d} \frac{n}{1} = 1 \leq \frac{L^d}{L^d} \frac{n}{|Q \cap B_\infty(q,L)|},$$

which implies the optimal $r_q \in [Ln^{-\frac{1}{d}}, L]$. Moreover, since $r_q \geq \min_{q'\in Q,\ q'\neq q} \|q - q'\|_\infty \geq \eta$, we have $r_q \geq \max(Ln^{-\frac{1}{d}}, \eta)$ for all $q \in Q$. If $i > \min\big(\log_2 \frac{L}{\eta}, \frac{1}{d}\log_2 n\big)$, then $\frac{L}{2^i} < \max(Ln^{-\frac{1}{d}}, \eta) \leq r_q$, and from the definition of Q_i and A we know $i \notin A$, which implies $|A| \leq 1 + \min\big(\log_2 \frac{L}{\eta}, \frac{1}{d}\log_2 n\big)$. Hence we obtain $\widetilde{C}_Q \leq 2^{d+1} \min\big(\log_2 \frac{L}{\eta}, \frac{1}{d}\log_2 n\big)$ and using $C_Q = (\widetilde{C}_Q)^{\frac{2}{2+d}}$ we prove the lemma. ◀

4 Strong Coresets for the Distance Between PL Curves

In this section, we study the distance $\mathtt{d}_Q$ defined on a subset of $\mathcal{S}(L)$: the collection of k-piecewise linear curves, and use the framework in [5] to construct a strong approximation for Q. This requires a bound on the shattering dimension, not possible for unrestricted

objects as in Section 3. We assume the multiset Q contains m distinct points $q_1, \cdots, q_m$, where each point q_i appears m_i times and $\sum_{i=1}^{m} m_i = n$. So, in this section Q will be viewed as a set $\{q_1, \cdots, q_m\}$ (not a multiset) and each point $q \in Q$ has a weight $w(q_i) = \frac{m_i}{n}$.

Suppose $\mathcal{T}_k := \{\gamma = \langle c_0, \cdots, c_k\rangle \mid c_i \in \mathbb{R}^d\}$ is the collection of all piecewise-linear curves with k line segments in $\mathbb{R}^d$. For $\gamma = \langle c_0, \cdots, c_k\rangle \in \mathcal{T}_k$, $\langle c_0, \cdots, c_k\rangle$ is the sequence of $k+1$ critical points of γ. The value $\mathsf{dist}(q, \gamma) = \inf_{p\in\gamma} \|p-q\|$, and function $f_{\gamma_1,\gamma_2}(q) = (\mathsf{dist}(q,\gamma_1) - \mathsf{dist}(q,\gamma_2))^2$ are defined as before. We now use weights $w(q_i) = \frac{m_i}{n}$ $\left(\sum_{q\in Q} w(q) = 1\right)$ and the resulting distance is $\mathtt{d}_Q(\gamma_1,\gamma_2) = \left(\sum_{q\in Q} w(q) f_{\gamma_1,\gamma_2}(q)\right)^{\frac{1}{2}}$.

For $L > \rho > 0$, $Q = \{q_1, \cdots, q_m\} \subset \mathbb{R}^d$, we define

$$\mathcal{X}_k^d(L,\rho) := \{(\gamma_1,\gamma_2) \in \mathcal{T}_k \times \mathcal{T}_k \mid \gamma_1,\gamma_2 \in \mathcal{S}(L),\ \mathtt{d}_Q(\gamma_1,\gamma_2) \geq \rho\}\,.$$

We next consider the sensitivity adjusted weights $w'(q) = \frac{\sigma(q)}{\mathfrak{S}_Q} w(q)$ and cost function $g_{\gamma_1,\gamma_2}(q) = \frac{1}{\sigma(q)} \frac{f_{\gamma_1,\gamma_2}(q)}{\bar{f}_{\gamma_1,\gamma_2}}$. These use the general bounds for sensitivity in Lemma 7 and Theorem 4, with as usual $\bar{f}_{\gamma_1,\gamma_2} = \sum_{q\in Q} w(q) f_{\gamma_1,\gamma_2}(q)$. These induce an adjusted range space $(Q, \mathcal{T}'_{k,d})$ where each element is defined

$$T_{\gamma_1,\gamma_2,\eta} = \{q \in Q \mid w'(q) g_{\gamma_1,\gamma_2}(q) \leq \eta,\ \gamma_1,\gamma_2 \in \mathcal{X}_k^d(L,\rho)\}.$$

Now to apply the strong coreset construction of Braverman *et al.* [5][Theorem 5.5] we only need to bound the shattering dimension of $(Q, \mathcal{T}'_{k,d})$.

Two recent results provide bounds on the VC dimension of range spaces related to trajectories. Given a range space $(X, \mathcal{R})$ with VC dimension ν and shattering dimension $\mathfrak{s}$, it is known that $\mathfrak{s} = O(\nu \log \nu)$ and $\nu = O(\mathfrak{s})$. So up to logarithmic factors these terms are bounded by each other. First Driemel *et al.* [12] shows VC dimension for a ground set of curves $\mathbb{X}_m$ of length m, with respect to metric balls around curves of length k, for various distance between curves. The most relevant case is where $m = 1$ (so the ground set are points like Q), and the Hausdorff distance is considered, where the VC dimension is bounded $O(d^2k^2 \log(km)) = O(k^2 \log k)$ for $d = 2$, and is at least $\Omega(\max\{k, \log m\}) = \Omega(k)$. Second, Matheny *et al.* [21] considered ground sets $\mathbb{X}_k$ of trajectories of length k, and ranges defined by geometric shapes which may intersect those trajectories anywhere to include them in a subset, but this result is also not directly relevant. Neither of these cases directly imply the results for our intended range space, since ours involves a pair of trajectories.

▶ **Lemma 9.** *The shattering dimension of range space $(Q, \mathcal{T}'_{k,d})$ is $O(k^3)$, for constant d.*

Proof. Suppose $(\gamma_1,\gamma_2) \in \mathcal{X}_k^d(L,\rho)$ and $\eta \geq 0$, where $\gamma_1 = \langle c_{1,0}, \cdots, c_{1,k}\rangle$ and $\gamma_2 = \langle c_{2,0}, \ldots, c_{2,k}\rangle$, then we can define the range $T_{\gamma_1,\gamma_2,\eta}$ as

$$\begin{aligned} T_{\gamma_1,\gamma_2,\eta} &:= \{q \in Q \mid w'(q) g_{\gamma_1,\gamma_2}(q) \leq \eta\} \\ &= \{q \in Q \mid w(q) f_{\gamma_1,\gamma_2}(q) \leq \mathfrak{S}_Q \bar{f}_{\gamma_1,\gamma_2} \eta\} \\ &= \{q \in Q \mid w(q)(\mathsf{dist}(q,\gamma_1) - \mathsf{dist}(q,\gamma_2))^2 \leq \mathfrak{S}_Q \bar{f}_{\gamma_1,\gamma_2} \eta\}. \end{aligned}$$

For a trajectory γ defined by critical points $c_0, c_1, \ldots, c_k$ for $j \in [k]$ define s_j as the segment between c_{j-1}, c_j and ℓ_j as the line extension of that segment. The distance between q and a segment s_j is illustrated in Figure 3 and defined

$$\xi_j := \mathsf{dist}(q, s_j) = \begin{cases} \mathsf{dist}(q, c_{j-1}), & \text{if } \langle c_j - c_{j-1},\ q - c_{j-1}\rangle \leq 0 \\ \mathsf{dist}(q, c_j), & \text{if } \langle c_{j-1} - c_j,\ q - c_j\rangle \leq 0 \\ \mathsf{dist}(q, \ell_j), & \text{otherwise} \end{cases}.$$

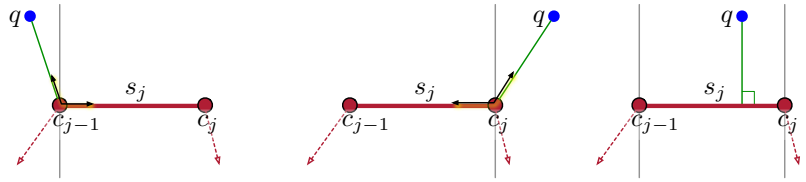


Figure 3 Illustration of the $\mathsf{dist}(q, s_j)$ from point q to segment s_j.

Then $\mathsf{dist}(q,\gamma) = \min_{j\in[k]} \xi_j$. For trajectories γ_1 and γ_2, specify these segment distances as $\xi_i^{(1)}$ and $\xi_i^{(2)}$, respectively. Then the expression for $T_{\gamma_1,\gamma_2,\eta}$ can be rewritten as

$$\begin{aligned}
T_{\gamma_1,\gamma_2,\eta} &= \{q \in Q \mid w'(q) g_{\gamma_1,\gamma_2}(q) \le \eta\} \\
&= \{q \in Q \mid w(q)(\min_{j\in[k]} \xi_j^{(1)} - \min_{j\in[k]} \xi_j^{(2)})^2 \le \mathfrak{S}_Q \bar{f}_{\gamma_1,\gamma_2}\eta\} \\
&= \cup_{j_1,j_2\in[k]} \{q \in Q \mid \xi_{j_1}^{(1)} \le \xi_j^{(1)}, \xi_{j_2}^{(2)} \le \xi_j^{(2)} \forall j \in [k], w(q)(\xi_{j_1}^{(1)} - \xi_{j_2}^{(2)})^2 \le \mathfrak{S}_Q \bar{f}_{\gamma_1,\gamma_2}\eta\} \\
&= \bigcup_{j_1,j_2\in[k]} \left(\begin{array}{l} \left(\cap_{j\in[k],j\neq j_1} \{q \in Q \mid \xi_{j_1}^{(1)} \le \xi_j^{(1)}\}\right) \\ \cap \left(\cap_{j\in[k],j\neq j_2} \{q \in Q \mid \xi_{j_2}^{(2)} \le \xi_j^{(2)}\}\right) \\ \cap \{q \in Q \mid \sqrt{w(q)}(\xi_{j_1}^{(1)} - \xi_{j_2}^{(2)}) \le (\mathfrak{S}_Q \bar{f}_{\gamma_1,\gamma_2}\eta)^{\frac{1}{2}}\} \\ \cap \{q \in Q \mid \sqrt{w(q)}(\xi_{j_2}^{(2)} - \xi_{j_1}^{(1)}) \le (\mathfrak{S}_Q \bar{f}_{\gamma_1,\gamma_2}\eta)^{\frac{1}{2}}\} \end{array} \right).
\end{aligned}$$

This means set $T_{\gamma_1,\gamma_2,\eta}$ can be decomposed as the union and intersection of $O(k^3)$ simply-defined subsets of Q. Specifically looking at the last line, this can be seen as the union over $O(k^2)$ sets (the outer union), and the first two lines are the intersection of $O(k)$ sets, and the last two lines inside the union are the intersection with one set each.

Next we argue that each of these $O(k^3)$ simply defined subsets of Q can be characterized as an element of a range space. By standard combinatorics [19, 3], the bound of the shattering dimension of the entire range space is $O(k^3)$ times the shattering dimension of any of these simple ranges spaces.

To get this simple range space shattering dimension bound, we can use a similar linearization method (see full version). For any simple range space $\mathcal{R}$ determined by the set decomposition of $T_{\gamma_1,\gamma_2,\eta}$, we can introduce new variables $c_0 \in \mathbb{R}, z, c \in \mathbb{R}^{d'}$, where z depends only on q, and c_0, c_i depend only on γ_1, γ_2 and r, and d' only depends on d. Here, Q is a fixed set and thus $\mathfrak{S}_Q$ is a constant. By introducing new variables we can construct an injective map $\varphi : Q \mapsto \mathbb{R}^{d'}$, s.t. $\varphi(q) = z$. There is also an injective map from $\mathcal{R}$ to $\{\{z \in \varphi(Q) \mid c_0 + z^T c \le 0\} \mid c_0 \in \mathbb{R}, c \in \mathbb{R}^{d'}\}$. Since the shattering dimension of the range space $(\mathbb{R}^{d'}, \mathcal{H}^{d'})$, where $\mathcal{H}^{d'} = \{h \text{ is a halfspace in } \mathbb{R}^{d'}\}$, is $O(d')$, we have the shattering dimension of$(Q, \mathcal{R})$ is $O(d') \le C_d$ where C_d is a positive constant depending only on d. Piecing this all together we obtain $C_d k^3$ bound for the shattering dimension of $(Q, \mathcal{T}'_{k,d})$. ◀

Now, we invoke Lemma 9 and [5][Theorem 5.5] to get a $(\rho, \varepsilon, \delta)$-strong coreset for $\mathcal{X}_k^d(L, \rho)$.

▶ **Theorem 10.** *Let $L > \rho > 0$, $Q \subset [0, L]^d$, and consider trajectory pairs $\mathcal{X}_k^d(L, \rho)$. Suppose $\sigma(q)$ and $\mathfrak{S}_Q$ are defined in Lemma 7 and Theorem 4 respectively. Then for $\delta, \varepsilon \in (0, 1)$ a σ-sensitive sampling of size $N = O(\frac{\mathfrak{S}_Q}{\varepsilon^2}(k^3 \log \mathfrak{S} + \log \frac{1}{\delta}))$ provides $\tilde{Q}$, a strong $(\rho, \varepsilon, \delta)$-coreset; that is with probability at least $1 - \delta$, for all pairs $\gamma_1, \gamma_2 \in \mathcal{X}_k^d(L, \rho)$ we have*

$$(1 - \varepsilon)\mathtt{d}_Q(\gamma_1, \gamma_2) \le \mathtt{d}_{\tilde{Q},W}(\gamma_1, \gamma_2) \le (1 + \varepsilon)\mathtt{d}_Q(\gamma_1, \gamma_2).$$

5 Trajectory Reconstruction

Let $\mathcal{T} := \{\gamma = \langle c_0, \cdots, c_k\rangle \mid c_i \in \mathbb{R}^2, k \geq 1\}$ be the set of all piecewise-linear curves in $\mathbb{R}^2$. Each curve in $\mathcal{T}$ is specified by a series of critical points $\langle c_0, c_1, \ldots, c_k\rangle$, and k line segments $s_1, s_2, \ldots, s_k$, where s_i is the line segment $\overline{c_{i-1}c_i}$. In this section we study how to recover γ from Q and $v_Q(\gamma)$ for $\gamma \in \mathcal{T}$.

Restrictions on curves and Q. For $\tau > 0$ we define a family of curves $\mathcal{T}_\tau \subset \mathcal{T}$ s.t. each $\gamma \in \mathcal{T}_\tau$ has two restrictions:

- (R1): Angles $\angle_{[c_{i-1},c_i,c_{i+1}]}$ at an internal critical point c_i is non-zero (i.e., in $(0, \pi)$).
- (R2): Each critical point c_i is τ-*separated*, that is the disk $B(c_i, \tau) = \{x \in \mathbb{R}^2 \mid \|x - c_i\| \leq \tau\}$ only intersects the two adjacent segments s_{i-1} and s_i of γ, or one adjacent segment for endpoints (i.e., only the s_1 for c_0 and s_k for c_k).

We next restrict that all curves (and Q) lie in region $\Omega \subset \mathbb{R}^2$. Let $\mathcal{T}_\tau(\Omega)$ be the subset of $\mathcal{T}_\tau$ where all curves γ have all critical points within Ω, and in particular, no $c_i \in \gamma$ is within a distance τ of the boundary of Ω.

To define Q, for $\eta > 0$, define an infinite grid $G_\eta = \{g_v \in \mathbb{R}^2 \mid g_\eta = \eta v \text{ for } v = (v_1, v_2) \in \mathbb{Z}^2\}$, where $\mathbb{Z}$ is all integers. Suppose $\eta \leq \frac{\tau}{32}$, then $Q = G_\eta \cap \Omega = \{q_1, \cdots, q_n\}$, $\gamma \in \mathcal{T}_\tau(\Omega)$, $v_i = \min_{p\in\gamma} \|q_i - p\|$ and $v_Q(\gamma) = (v_1, \ldots, v_n)$. We define some notations that are used in this section for the implied circle $C_i := \{x \in \mathbb{R}^2 | \ \|x - q_i\| = v_i\}$, the closed disk $B_i := \{x \in \mathbb{R}^2 | \ \|x - q_i\| \leq v_i\}$, and the open disk $\dot{B}_i := \{x \in \mathbb{R}^2 | \ \|x - q_i\| < v_i\}$ around each q_i or radius v_i. When the radius is specified as r (with perhaps $r \neq v_i$), then we, as follows, denote the associated circle $C_{i,r}$, closed disk $B_{i,r}$, and open disk $\dot{B}_{i,r}$ around q_i.

For Q, $\gamma \in \mathcal{T}_\tau(\Omega)$ and $v_Q(\gamma)$ we have the following three observations.

- (O1): In any disk with radius less than τ, there is at most one critical point of γ; by (R2).
- (O2): If a point moves along γ, it can only stop or change direction at critical points of γ.
- (O3): For any $q_i \in Q$, γ cannot go into $\dot{B}_i$. Moreover, C_i must contain at least one point of γ, and if this point is not a critical point, then γ must be tangent to C_i at this point.

The restriction (R2) only implies if there is a critical point of γ, then in its neighborhood γ has at most two line segments. However, if there is no critical point in a region, then the shape of γ can be very complicated in this region, so we need to first identify the regions that contain a critical point.

These observations form the basis for the algorithm and its proof of correctness. We next describe the algorithm, state the main results, and provide intuition for the proofs. For space, some pseudocode and full proofs which rely heavily on case analysis are in the full version.

Recovery algorithm. The entire algorithm is overviewed in Algorithm 1. For each critical point $c \in \gamma$, there exists $q \in Q$ such that $\mathsf{dist}(q, c) < \eta$. So to recover γ, we first traverse $\{q_i \in Q \mid v_i < \eta\}$ and use $\textsc{isCritical}(q_i)$ to solve the decision problem of if there is a critical point in $B_{i,3\eta}$. Whenever there is a critical point in $B_{i,3\eta}$, we then use $\textsc{FindCritical}(q_i)$ to find it – collectively, this finds all critical points of γ. Finally, we use $\textsc{DetermineOrder}$ (Algorithm 2) to determine the order of all critical points of γ, which recovers γ.

Existence of critical points. In $\textsc{isCritical}(q_i)$ we consider the common tangent line of C_i and C_j for all q_j in a neighborhood of q_i. If no common tangent line can go through $B_{i,3\eta}$ without going into the interior of any other circle centered in $B_{i,3\eta}$, unlike Figure 4(Left), then it implies there is a critical point of γ in $B_{i,3\eta}$. This can be confirmed checking the possible tangent lines for circles centered at grid points in $B_{i,3\eta}$ and around q_i; cases for endpoints and internal points are illustrated in Figure 4(Center,Right).

Algorithm 1 Recover $\gamma \in \mathcal{T}_\tau(L)$ from Q and $v_Q(\gamma)$.

Initialize $Q_\eta := \{q_i \in Q \mid v_i < \eta\}$, close set $Q_{\mathrm{r}} := \emptyset$, endpoints $E = \emptyset$ and critical points $A := \emptyset$.
for each $q_i \in Q_\eta$ **do**
  if $q_i \in Q_{\mathrm{r}}$ or IsCritical(q_i)=False **then**
    continue
  Let $(c, S) :=$ FindCritical(q_i).
  if $|S| = 1$ **then**
    $E := E \cup \{(c, S)\}$. // c is an endpoint of γ
  Let $A := A \cup \{(c, S)\}$ and $Q_{\mathrm{r}} := Q_{\mathrm{r}} \cup \big(Q_\eta \cap B_{c,16\eta}\big)$. // aggregate critical points
return $\gamma :=$ DetermineOrder(E, A)

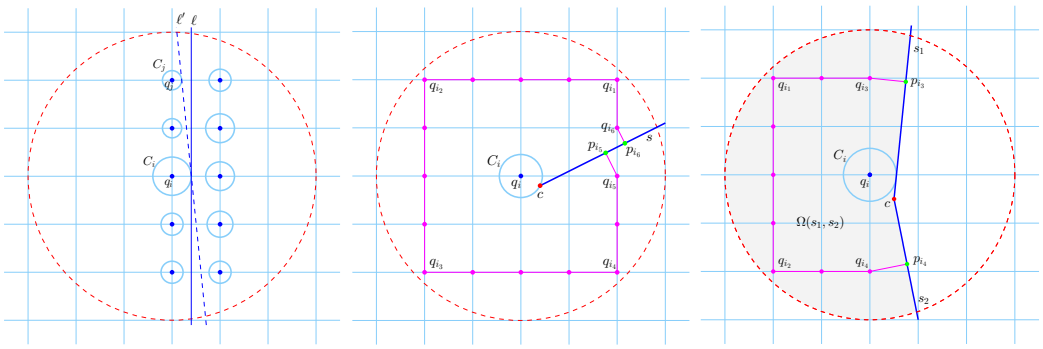


Figure 4 Determining no critical point (Left), endpoint (Center), or internal critical point (Right).

▶ **Lemma 11.** *Suppose $q_i \in Q$ and $v_i < \eta$. If* IsCritical(q_i) *returns* True*, then there must be a critical point of γ in $B_{i,3\eta}$. Moreover, for any critical point $c \in \gamma$ there exists some $q_i \in Q$ such that $v_i < \eta$ and* IsCritical(q_i) *returns* True *for the input q_i.*

Finding a critical point. If there is a critical point c in $B_{i,3\eta}$, then using (R2) we know in the neighborhood of c, γ has a particular pattern: it either has one line segment, or two line segments. We will need two straightforward subfunctions:

- FCT (*Find Common Tangents*) takes in three grid points q_i, q_j, q_k, and returns the all common tangent lines of C_j and C_k which do not intersect the interior of disks $\dot{B}_l$ of a disk associated with a point $q_l \in Q_{i,8\eta}$. This generates a feasible superset of possible nearby line segments which may be part of γ.
- MOS (*Merge-Overlapping-Segments*) takes a set of line segments, and returns a smaller set, merging overlapping segments. This combines the just generated potential line segments of γ.

Now in FindCritical(q_i) for each pair $q_j, q_k \in B_{i,8\eta}$, we first use FCS to find the common tangent line of C_j, C_k that could be segments of γ, and then use MOS to reduce this set down to a minimal set of possibilities S_m. By definition, there must be a critical point c, and thus can be at most 2 actual segments of γ within $B_{i,8\eta}$, so we can then refine S_m. We first check if c is an endpoint, in which case there must be only one valid segment. If not, then there must be 2, and we need to consider all pairs in S_m. This check can be done by verifying that *every* C_k for $q_k \in Q_{i,8\eta}$ is tangent to the associated ray $\mathsf{ray}(s)$ (for an endpoint) or for the associated rays $\mathsf{ray}(s)$ and $\mathsf{ray}(s')$ for their associated segment pairs (for an internal critical point). Some of these cases are illustrated in Figure 5.

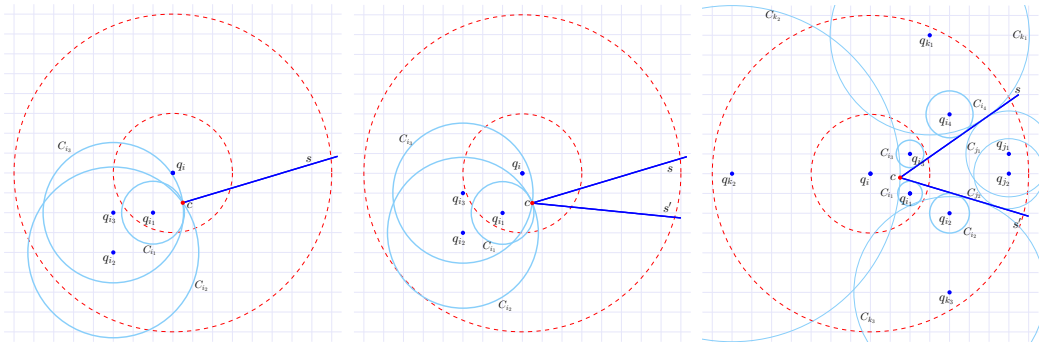


Figure 5 Illustration of how $Q \cap B_{i,8\eta}$ is sufficient to recover a critical point c in $B_{i,3\eta}$ for the c and endpoint (Left), or an internal point with small angle (Center) or large angle (Right).

▶ **Lemma 12.** *Suppose $c' \in B_{i,3\eta}$ is a critical point of γ, and (c, S) is the output of* FindCritical(q_i)*, then $c = c'$. Moreover, $|S| = 1$ if and only if c is an endpoint of γ.*

Using isCritical and FindCritical we can find all critical points (E, A) with associated line segments of γ, so the final step is to use function DetermineOrder(E, A) (Algorithm 2) to determine their order, as we argue it will completely recover γ.

Algorithm 2 DetermineOrder(E, A): Determine the order of critical points.

Choose any $(c_0, S_0) \in E$, let $k = |A| - 1$, $A := A \setminus \{(c_0, S_0)\}$, $s_1 \in S_0$ and $\gamma := \langle c_0 \rangle$.
for $i = 1$ **to** k **do**
  Find closest c from $(c, S) \in A$ to c_{i-1} so c is on $\mathsf{ray}(s_i)$, and let $A := A \setminus \{(c, S)\}$.
  Append $c_i = c$ to γ, and if $i < k$ then let $s_{i+1} = s$ where $s \in S$ is not parallel with s_i.
return γ

▶ **Theorem 13.** *Suppose $Q = G_\eta \cap \Omega$, $\eta \leq \frac{\tau}{32}$, and $v_Q(\gamma)$ is generated by Q and $\gamma \in \mathcal{T}_\tau(\Omega)$ with k segments, then Algorithm 1 can recover γ from $v_Q(\gamma)$ in $O(|Q| + k^2)$ time.*

References

1 Ann Arbor Algorithms. K-graph. Technical report, `https://github.com/aaalgo/kgraph`, 2018.
2 Nina Amenta, Sunghee Choi, and Ravi Krishna Kolluri. The power crust. In *Proceedings of the sixth ACM symposium on Solid modeling and applications*, 2001.
3 Martin Anthony and Peter L. Bartlett. *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, 1999.
4 Christos Boutsidis, Michael W Mahoney, and Petros Drineas. An improved approximation algorithm for the column subset selection problem. In *Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms*, 2009.
5 Vladimir Braverman, Dan Feldman, and Harry Lang. New frameworks for offline and streaming coreset constructions, 2016. `arXiv:1612.00889`.
6 Frederic Chazal and David Cohen-Steiner. Geometric inference. URL: `https://geometrica.saclay.inria.fr/team/Fred.Chazal/papers/GeomInference5.pdf`.
7 Frederic Chazal, David Cohen-Steiner, and Andre Lieutier. A sampling theory for compact sets in euclidean space. *DCG*, 41:461–479, 2009.
8 Frédéric Chazal, David Cohen-Steiner, and Quentin Mérigot. Geometric inference for probability measures. *Foundations of Computational Mathematics*, pages 1–19, 2010.

9 Frédéric Chazal and Andre Lieutier. The "λ-medial axis". *Graphical Models*, 67:304–331, 2005.

10 Michael B. Cohen, Cameron Musco, and Christopher Musco. Input sparsity time low-rank approximation via ridge leverage score sampling. In *ACM-SIAM Symposium on Discrete Algorithms*, 2017.

11 Michael B. Cohen, Cameron Musco, and Jakub Pachocki. Online row sampling. In *International Workshop on Approximation, Randomization, and Combinatorial Optimization*, 2016.

12 Anne Driemel, Jeff M. Phillips, and Ioannis Psarros. On the vc dimension of metric balls under frechet and hausdorff distances. In *International Symposium on Computational Geometry*, 2019.

13 Petros Drineas, Malik Magdon-Ismail, Michael W. Mahoney, and David P. Woodruff. Fast approximation of statistical leverage. *Journal of Machine Learning Research*, 13:3475–3506, 2012.

14 Petros Drineas, Michael W. Mahoney, and S. Muthukrishnan. Relative-error CUR matrix decompositions. *SIAM Journal of MAtrix Analysis and Applications*, 30:844–881, 2008.

15 Herbert Edelsbrunner and Ernst P. Mücke. Three-dimensional alpha shapes. *ACM Transactions on Graphics*, 13:43–72, 1994.

16 Dan Feldman and Michael Langberg. A unified framework for approximating and clustering data. In *Proceedings ACM Symposium on Theory of Computing*, 2011.

17 Dan Feldman, Melanie Schmidt, and Christian Sohler. Turning big data into tiny data: Constant-size coresets for k-means, PCA, and projective clustering. In *Proceedings 24th ACM-SIAM Symposium on Discrete Algorithms*, 2013.

18 Dan Feldman and Lenard J. Schulman. Data reduction for weighted and outlier-resistant clustering. In *Proc. ACM-SIAM Symposium on Discrete Algorithms*, 2012.

19 S. Har-Peled. *Geometric Approximation Algorithms*. Mathematical Surveys and Monographs. American Mathematical Society, 2011.

20 Michael Langberg and Leonard J. Schulman. Universal ε-approximators for integrals. In *SODA*, pages 598–607, 2010.

21 Michael Matheny, Dong Xie, and Jeff M. Phillips. Scalable spatial scan statistics for trajectories, 2019. `arXiv:1906.01693`.

22 Cameron Musco and Christopher Musco. Recursive sampling for the Nyström method. In *NIPS*, 2017.

23 Jeff M. Phillips and Pingfan Tang. Simple distances for trajectories via landmarks. In *SIGSPATIAL*. (long version: `arXiv:1804.11284`), 2019.

24 Jeff M. Phillips, Bei Wang, and Yan Zheng. Geomtric inference on kernel density estimates. In *SOCG*, 2015.

25 Ilya Razenshteyn and Ludwig Schmidt. Falconn-fast lookups of cosine and other nearest neighbors. `https://falconn-lib.org`, 2018.

26 Kasturi Varadarajan and Xin Xiao. On the sensitivity of shape fitting problems. In Deepak D'Souza, Telikepalli Kavitha, and Jaikumar Radhakrishnan, editors, *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2012)*, pages 486–497, Dagstuhl, Germany, 2012. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. `doi:10.4230/LIPIcs.FSTTCS.2012.486`.

Fast Algorithms for Minimum Cycle Basis and Minimum Homology Basis

Abhishek Rathod
Department of Mathematics, Technical University of Munich (TUM),
Boltzmannstr. 3, 85748 Garching b. München, Germany
abhishek.rathod@tum.de

Abstract

We study the problem of finding a minimum homology basis, that is, a shortest set of cycles that generates the 1-dimensional homology classes with $\mathbb{Z}_2$ coefficients in a given simplicial complex K. This problem has been extensively studied in the last few years. For general complexes, the current best deterministic algorithm, by Dey et al. [8], runs in $O(N^\omega + N^2 g)$ time, where N denotes the number of simplices in K, g denotes the rank of the 1-homology group of K, and ω denotes the exponent of matrix multiplication. In this paper, we present two conceptually simple randomized algorithms that compute a minimum homology basis of a general simplicial complex K. The first algorithm runs in $\tilde{O}(m^\omega)$ time, where m denotes the number of edges in K, whereas the second algorithm runs in $O(m^\omega + Nm^{\omega-1})$ time.

We also study the problem of finding a minimum cycle basis in an undirected graph G with n vertices and m edges. The best known algorithm for this problem runs in $O(m^\omega)$ time. Our algorithm, which has a simpler high-level description, but is slightly more expensive, runs in $\tilde{O}(m^\omega)$ time.

2012 ACM Subject Classification Theory of computation → Computational geometry; Mathematics of computing → Algebraic topology

Keywords and phrases Computational topology, Minimum homology basis, Minimum cycle basis, Simplicial complexes, Matrix computations

Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.64

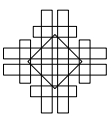
Funding This research has been supported by the DFG Collaborative Research Center SFB/TRR 109 "Discretization in Geometry and Dynamics".

Acknowledgements The author would like to thank Ulrich Bauer and Michael Lesnick for valuable discussions, and anonymous reviewers for their useful comments.

1 Introduction

Minimum cycle bases in graphs have several applications, for instance, in analysis of electrical networks, analysis of chemical and biological pathways, periodic scheduling, surface reconstruction and graph drawing. Also, algorithms from diverse application domains like electrical circuit theory and structural engineering require cycle basis computation as a preprocessing step. Cycle bases of small size offer a compact description that is advantageous from a mathematical as well as from an application viewpoint. For this reason, the problem of computing a minimum cycle basis has received a lot of attention, both in its general setting as well as in special classes of graphs such as planar graphs, sparse graphs, dense graphs, network graphs, and so on. We refer the reader to [15] for a comprehensive survey.

In topological data analysis, "holes" of different dimensions in a geometric dataset constitute "features" of the data. Algebraic topology offers a rigorous language to formalize our intuitive picture of holes in these geometric objects. More precisely, a basis for the first homology group H_1 can be taken as a representative of the one-dimensional holes in the geometric object. The advantages of using minimum homology bases are twofold: firstly, one can bring geometry in picture by assigning appropriate weights to edges, and secondly,



smaller cycles are easier to understand and analyze, especially visually. We focus solely on the bases of the first homology group since the problem of computing a shortest basis for higher homology groups with $\mathbb{Z}_2$ coefficients was shown to be NP-hard by Chen and Freedman [5].

2 Background and Preliminaries

2.1 Cycle Basis

Let $G = (V, E)$ be a connected graph. A subgraph of G which has even degree for each vertex is called a *cycle* of G. A cycle is called *elementary* if the set of edges form a connected subgraph in which each vertex has degree 2. We associate an incidence vector C, indexed on E, to each cycle, so that $C_e = 1$ if e is an edge of the cycle, and $C_e = 0$ otherwise. The set of incidence vectors of cycles forms a vector space over $\mathbb{Z}_2$, called the *cycle space* of G. It is a well-known fact that for a connected graph G, the cycle space is of dimension $|E| - |V| + 1$. Throughout, we use ν to denote the dimension of the cycle space of a graph. A basis of the cycle space, that is, a maximal linearly independent set of cycles is called a *cycle basis*.

Suppose that the edges of G have non-negative weights. Then, the weight of a cycle is the sum of the weights of its edges, and the weight of a cycle basis is the sum of the weights of the basis elements. The problem of computing a cycle basis of minimum weight is called the *minimum cycle basis* problem. Since we assume all edge weights to be non-negative, there always exists a minimum cycle basis of elementary cycles, allowing us to focus on minimum cycle basis comprising entirely of elementary cycles.

A simple cycle C is *tight* if it contains a shortest path between every pair of points in C. We denote the set of all tight cycles in the graph by $\mathcal{T}$. Tight cycles are sometimes also referred to as *isometric* cycles [1, 15]. Tight cycles play an important role in designing algorithms for minimum cycle basis, owing to the following theorem by Horton.

▶ **Theorem 1** (Horton [13]). *A minimum cycle basis $\mathcal{M}$ consists only of tight cycles.*

A key structural property about minimum cycle bases was proved by de Pina.

▶ **Theorem 2** (de Pina [7]). *Cycles $C_1 \ldots, C_\nu$ form a minimum cycle basis if there are vectors $S_1, \ldots, S_\nu$ such that for all i, $1 \leq i \leq \nu$, the following hold:*
Prefix Orthogonality: *$\langle C_j, S_i \rangle = 0$ for all $1 \leq j \leq i$.*
Non-Orthogonality: *$\langle C_i, S_i \rangle = 1$.*
Shortness: *C_i is a minimum weight cycle in $\mathcal{T}$ with $\langle C_i, S_i \rangle = 1$.*

The vectors $S_1, \ldots, S_\nu$ in Theorem 2 are called *support vectors*. The recent line of algorithmic work [1, 7, 16, 17, 18] on the minimum cycle basis problem rely on Theorem 2. In fact, these algorithms may all be seen as refinements of the algorithm by de Pina, see Algorithm 1.

Algorithm 1 De Pina's Algorithm for computing a minimum cycle basis.

```
Initialize S_i to the i-th unit vector e_i for 1 ≤ i ≤ ν
for i ← 1, ..., ν do
    Compute a minimum weight cycle C_i with ⟨C_i, S_i⟩ = 1.
    for j ← i + 1, ..., ν do
        S_j = S_j + ⟨C_i, S_j⟩ S_i
    end for
end for
Return {C_1, ..., C_ν}.
```

Algorithm 1 works by inductively maintaining a set of support vectors $\{S_i\}$ so that the conditions of Theorem 2 are satisfied when the algorithm terminates. In particular, Lines 4 and 5 of the algorithm ensure that the set of vectors S_j for $j > i$ are orthogonal to vectors $C_1, \ldots, C_i$. Updating the vectors S_j as outlined in Lines 4 and 5 of Algorithm 1 takes time $O(m^3)$ time in total. Using a divide and conquer procedure for maintaining S_j, Kavitha et al. [17] improved the cost of maintaining the support vectors to $O(m^\omega)$. See Algorithm 2.

Algorithm 2 Divide and conquer procedure for fast computation of support vectors by Kavitha et al. [17].

1: Initialize S_i to the i-th unit vector e_i for $1 \leq i \leq \nu$.
2: MinCycleBasis$(1, \nu)$.

3: **procedure** MinCycleBasis(ℓ, u)
4: **if** $\ell = u$ **then**
5: Compute a minimum weight cycle C_ℓ with $\langle C_\ell, S_\ell \rangle = 1$.
6: **else**
7: $q \leftarrow \lfloor (\ell+u)/2 \rfloor$.
8: MinCycleBasis(ℓ, q).
9: $\mathbf{C} \leftarrow [C_\ell, \ldots, C_q]$.
10: $\mathbf{W} \leftarrow (\mathbf{C}^T[S_\ell, \ldots, S_q])^{-1}\mathbf{C}^T[S_{q+1}, \ldots, S_u]$.
11: $[S_{q+1}, \ldots, S_u] \leftarrow [S_{q+1}, \ldots, S_u] + [S_\ell, \ldots, S_q]\mathbf{W}$.
12: MinCycleBasis$(q+1, u)$.
13: **end if**
14: **end procedure**
15: Return $\{C_1, \ldots, C_\nu\}$.

▶ **Lemma 3** (Lemma 5.6 in [15]). *The total number of arithmetic operations performed in lines 9 to 11 of Algorithm 2 is $O(m^\omega)$. That is, the support vectors satisfying conditions of Theorem 2 can be maintained in $O(m^\omega)$ time.*

Finally, in [1], Amaldi et al. designed an $O(m^\omega)$ time algorithm for computing a minimum cycle basis by improving the complexity of Line 5 of Algorithm 2 to $o(m^\omega)$ (from $O(m^2 n)$ in [17]), while using the $O(m^\omega)$ time divide-and-conquer template for maintaining the support vectors as presented in Algorithm 2. The $o(m^\omega)$ procedure for Line 3 is achieved by performing a Monte Carlo binary search on the set of tight cycles (sorted by weight) to find a minimum weight cycle C_i that satisfies $\langle C_i, S_i \rangle = 1$. An efficient binary search is made possible on account of the following key structural property about tight cycles.

▶ **Theorem 4** (Amaldi et al. [1]). *The total length of the tight cycles is at most $n\nu$.*

Amaldi et al. [1] also show that there exists an $O(nm)$ algorithm to compute the set of all the tight cycles of an undirected graph G. See Sections 2 and 3 of [1] for details about Amaldi et al.'s algorithm.

2.2 Matrix operations

The *column rank profile* (respectively *row rank profile*) of an $m \times n$ matrix A with rank r, is the lexicographically smallest sequence of r indices $[i_1, i_2, \ldots, i_r]$ (respectively $[j_1, j_2, \ldots, j_r]$) of linearly independent columns (respectively rows) of A. Suppose that $\{a_1, a_2, \ldots, a_n\}$ represent the columns of A. Then, following Busaryev et al. [3], we define the *earliest basis* of A as the set of columns $\mathcal{E}(A) = \{a_{i_1}, a_{i_2}, \ldots, a_{i_r}\}$.

It is well-known that classical Gaussian elimination can be used to compute rank profile in $O(nmr)$ time. The current state-of-the-art deterministic matrix rank profile algorithms run in $O(mnr^{\omega-2})$ time.

▶ **Theorem 5** ([10, 14]). *There is a deterministic $O(mnr^{\omega-2})$ time algorithm to compute the column rank profile of an $m \times n$ matrix A.*

In case of randomized algorithms, Cheung, Kwok and Lau [6] presented a breakthrough Monte Carlo algorithm for rank computation that runs in $(\mathsf{nnz}(A) + r^\omega)^{1+o(1)}$ time, where $o(1)$ in the exponent captures some missing multiplicative $\log n$ and $\log m$ factors, and $\mathsf{nnz}(A)$ denotes the number of nonzero entries in A. Equivalently, the complexity for Cheung et al.'s algorithm can also be written as $\tilde{O}(\mathsf{nnz}(A) + r^\omega)$. The notation $\tilde{O}(\cdot)$ is often used in literature to hide small polylogarithmic factors in time bounds. While the algorithm by Cheung et al. also computes r linearly independent columns of A, the columns may not correspond to the column rank profile. However, building upon Cheung et al.'s work, Storjohann and Yang established the following result.

▶ **Theorem 6** (Storjohann and Yang [19, 20, 21]). *There exists a Monte Carlo algorithm for computing row (resp. column) rank profile of a matrix A that runs in $(\mathsf{nnz}(A) + r^\omega)^{1+o(1)}$ time. The failure probability of this algorithm is $1/2$.*

Once again, the $o(1)$ in the exponent captures some missing multiplicative $\log n$ and $\log m$ factors, see [19], and hence the complexity can also be written as $\tilde{O}(\mathsf{nnz}(A) + r^\omega)$.

2.3 Homology

In this work, we restrict our attention to simplicial homology with $\mathbb{Z}_2$ coefficients. For a general introduction to algebraic topology, we refer the reader to [12]. Below we give a brief description of homology over $\mathbb{Z}_2$.

Let K be a connected simplicial complex. We will denote by $K^{(p)}$ the set of p-dimensional simplices in K, and n_p the number of p-dimensional simplices in K. Also, the p-dimensional skeleton of K will be denoted by K_p. In particular, the 1-skeleton of K (which is an undirected graph) will be denoted by K_1.

We consider formal sums of simplices with $\mathbb{Z}_2$ coefficients, that is, sums of the form $\sum_{\sigma \in K^{(p)}} a_\sigma \sigma$, where each $a_\sigma \in \{0, 1\}$. The expression $\sum_{\sigma \in K^{(p)}} a_\sigma \sigma$ is called a *p-chain*. Since chains can be added to each other, they form an Abelian group, denoted by $\mathsf{C}_p(K)$. Since we consider formal sums with coefficients coming from $\mathbb{Z}_2$, which is a field, $\mathsf{C}_p(K)$, in this case, is a vector space of dimension n_p over $\mathbb{Z}_2$. The p-simplices in K form a (natural) basis for $\mathsf{C}_p(K)$. This establishes a natural one-to-one correspondence between elements of $\mathsf{C}_p(K)$ and subsets of $K^{(p)}$. Thus, associated with each chain is an incidence vector v, indexed on $K^{(p)}$, where $v_\sigma = 1$ if σ is a simplex of v, and $v_\sigma = 0$ otherwise. The *boundary* of a p-simplex is a $(p-1)$-chain that corresponds to the set of its $(p-1)$-faces. This map can be linearly extended from p-simplices to p-chains, where the boundary of a chain is the $\mathbb{Z}_2$-sum of the boundaries of its elements. Such an extension is known as the *boundary homomorphism*, and denoted by $\partial_p : \mathsf{C}_p(K) \to \mathsf{C}_{p-1}(K)$. A chain $\zeta \in \mathsf{C}_p(K)$ is called a *p-cycle* if $\partial_p \zeta = 0$, that is, $\zeta \in \ker \partial_p$. The group of p-dimensional cycles is denoted by $\mathsf{Z}_p(K)$. As before, since we are working with $\mathbb{Z}_2$ coefficients, $\mathsf{Z}_p(K)$ is a vector space over $\mathbb{Z}_2$. A chain $\eta \in \mathsf{C}_p(K)$ is said to be a p-boundary if $\eta = \partial_{p+1} c$ for some chain $c \in \mathsf{C}_{p+1}(K)$, that is, $\eta \in \operatorname{im} \partial_{p+1}$. The group of p-dimensional boundaries is denoted by $\mathsf{B}_p(K)$. In our case, $\mathsf{B}_p(K)$ is also a vector space, and in fact a subspace of $\mathsf{C}_p(K)$.

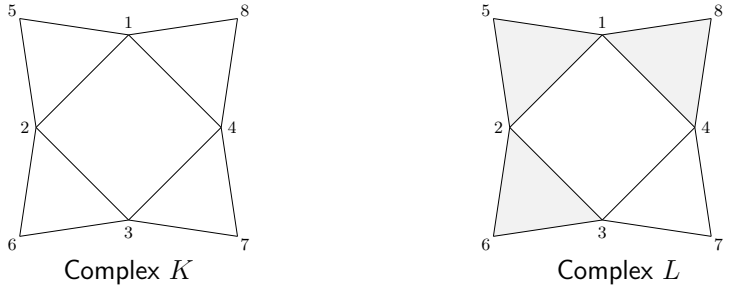


Figure 1 Consider complexes K and L in the figure above with unit weights on the edges. Since K has no 2-simplices, its 1-skeleton K_1 is identical to K itself. The set of cycles $\mathcal{C} = \{\{1,2,5\},\{1,4,8\},\{3,4,7\},\{2,3,6\},\{1,2,3,4\}\}$ constitutes a minimum cycle basis for the respective 1-skeletons K_1 and L_1 (viewed as graphs). The set $\mathcal{C}$ also constitutes a minimum homology basis for K. The set $\mathcal{C}' = \{\{1,2,3,4\},\{3,4,7\}\}$ constitutes a minimum homology basis for L.

Thus, we can consider the quotient space $\mathsf{H}_p(K) = \mathsf{Z}_p(K)/\mathsf{B}_p(K)$. The elements of the vector space $\mathsf{H}_p(K)$, known as the p-th *homology group* of K, are equivalence classes of p-cycles, where p-cycles are equivalent if their $\mathbb{Z}_2$-difference is a p-boundary. Equivalent cycles are said to be *homologous*. For a p-cycle ζ, its corresponding homology class is denoted by $[\zeta]$. Bases of $\mathsf{B}_p(K)$, $\mathsf{Z}_p(K)$ and $\mathsf{H}_p(K)$ are called *boundary bases*, *cycle bases*, and *homology bases* respectively. The dimension of the p-th homology group of K is called the p-th *Betti number* of K, denoted by $\beta_p(K)$. We are primarily interested in the first Betti number $\beta_1(K)$. For notational convenience, let $g = \beta_1(K)$, and denote the dimension of $\mathsf{B}_1(K)$ by b.

Using the natural bases for $\mathsf{C}_p(K)$ and $\mathsf{C}_{p-1}(K)$, the matrix $[\partial_p\sigma_1\ \partial_p\sigma_2 \cdots \partial_p\sigma_{n_p}]$ whose column vectors are boundaries of p-simplices is called the p-th boundary matrix. Abusing notation, we denote the p-th boundary matrix by ∂_p. For the rest of the paper, we use n, m and N to denote the number of vertices, edges and simplices in the complex respectively.

A set of p-cycles $\{\zeta_1, \ldots, \zeta_g\}$ is called a *homology cycle basis* if the set of classes $\{[\zeta_1], \ldots, [\zeta_g]\}$ forms a homology basis. For brevity, we abuse notation by using the term "homology basis" for $\{\zeta_1, \ldots, \zeta_g\}$. Assigning non-negative weights to the edges of K, the weight of a cycle is the sum of the weights of its edges, and the weight of a homology basis is the sum of the weights of the basis elements. The problem of computing a minimum weight basis of $\mathsf{H}_1(K)$ is called the *minimum homology basis* problem. Note that, when the input simplicial complex is a graph, the notions of homology basis and cycle basis coincide. Please refer to Figure 1 for an example.

For the special case when the input complex is a surface, Erickson and Whittlesey [11] gave a $O(N^2 \log N + gN^2 + g^3N)$-time algorithm. Recently, Borradaile et al. [2] gave an improved deterministic algorithm that runs in $O((h+c)^3 n \log n + m)$ where c denotes the number of boundary components, and h denotes the genus of the surface. For small values of c and h, the algorithm runs in nearly linear time.

For general complexes, Dey et al. [9] and Chen and Freedman [4] generalized the results by Erickson and Whittlesey [11] to arbitrary complexes. Subsequently, introducing the technique of annotations, Busaryev et al. [3] improved the complexity to $O(N^\omega + N^2 g^{\omega-1})$. More recently, Dey et al. [8] designed an $O(N^\omega + N^2 g)$ time algorithm by adapting the divide and conquer algorithm for computing a minimum cycle basis of Kavitha et al. [17] for the purpose of computing a minimum homology basis. Dey et al. also designed a randomized 2-approximation algorithm for the same problem that runs in $O(N^\omega \sqrt{N \log N})$ expected time.

3 An algorithm for computing minimum cycle basis

Given a graph $G = (V, E)$, let $\{C_1, \ldots, C_{|\mathcal{T}|}\}$ be the list of tight cycles in G sorted by weight, and let $\mathbf{M}_{\mathcal{T}}(G) = [C_1\, C_2 \ldots\, C_{|\mathcal{T}|}]$ be the matrix formed with cycles C_i as its columns. Using Theorem 4, since the total length of tight cycles is at most $n\nu$, and since each tight cycle consists of at least three edges, we have that $|\mathcal{T}| \leq \frac{n\nu}{3}$. Also, the rank of $\mathbf{M}_{\mathcal{T}}(G)$ is ν and $\mathbf{M}_{\mathcal{T}}(G)$ is a sparse matrix with $\mathsf{nnz}(\mathbf{M}_{\mathcal{T}}(G))$ bounded by $n\nu$. This sparsity is implicitly used in the design of the Monte Carlo binary search algorithm for computing minimum cycle basis, as described in [1]. We now present a simple and fast algorithm for minimum cycle basis that exploits the sparsity and the low rank of $\mathbf{M}_{\mathcal{T}}(G)$ more directly.

Algorithm 3 Algorithm for minimum cycle basis.

1: Compute the sorted list of tight cycles in G, and assemble the matrix $\mathbf{M}_{\mathcal{T}}(G)$.
2: Compute the column rank profile $[i_1, i_2, \ldots, i_\nu]$ of $\mathbf{M}_{\mathcal{T}}(G)$ using Storjohann and Yang's algorithm described in [20].
3: RETURN $\mathcal{E}(\mathbf{M}_{\mathcal{T}}(G))$.

▶ **Theorem 7.** *There is a Monte Carlo algorithm that computes the minimum cycle basis in $\tilde{O}(m^\omega)$ time, with failure probability at most $1/2$.*

Proof. The correctness of the algorithm follows immediately from Theorem 1. For instance, if $\mathcal{E}(\mathbf{M}_{\mathcal{T}}(G))$ is not a minimum cycle basis, then let k be the smallest integer such that the k-th smallest cycle in a minimum cycle basis contained in $\mathbf{M}_{\mathcal{T}}(G)$ is smaller than the k-th smallest cycle in $\mathcal{E}(\mathbf{M}_{\mathcal{T}}(G))$. Since the columns in $\mathbf{M}_{\mathcal{T}}(G)$ are sorted by weight, the existence of such a k contradicts the fact that $\mathcal{E}(\mathbf{M}_{\mathcal{T}}(G))$ is the earliest basis of $\mathbf{M}_{\mathcal{T}}(G)$.

The list of tight cycles in G can be computed in $O(nm)$ time using the algorithm described in Section 2 of [1]. Hence, Step 1 of Algorithm 3 takes $O(nm \log(nm))$ time (which in turn is same as $O(nm \log n)$ time). Moreover, using Theorem 6, the complexity of Step 2 is bounded by $\tilde{O}(n\nu + \nu^\omega)$. Since $n, \nu < m$, the complexity of Algorithm 3 is bounded by $\tilde{O}(m^\omega)$. Using Theorem 6, the failure probability of the algorithm is at most $1/2$. ◀

4 Minimum homology basis, minimum cycle basis and tight cycles

To begin with, note that since every graph is a 1-dimensional simplicial complex, the minimum cycle basis problem is a restriction of the minimum homology basis problem to instances (simplicial complexes) that have no 2-simplices. In this section, we refine this observation by deriving a closer relation between the two problems.

We assume that we are provided a complex K in which all edges are assigned non-negative weights. Given a non-negative weight $w(\sigma)$ for each edge σ, we define the weight of a cycle z as the sum of the weights of the edges, $w(z) = \sum_{\sigma \in z} w(\sigma)$. Let $\mathcal{B} = \{\eta_1, \ldots, \eta_b\}$ be a basis for the boundary vector space $\mathsf{B}_1(K)$ indexed so that $w(\eta_i) \leq w(\eta_{i+1})$, $1 \leq i < b$ (with ties broken arbitrarily). Also, let $\mathcal{H} = \{\zeta_1, \ldots, \zeta_g\}$ be a minimum homology basis of K indexed so that $w(\zeta_i) \leq w(\zeta_{i+1})$, $1 \leq i < g$ (with ties broken arbitrarily). Then, the set $\mathcal{C} = \{\eta_1, \ldots, \eta_b, \zeta_1, \ldots, \zeta_g\}$ is a cycle basis for K_1. Let $\mathcal{M}$ be a minimum cycle basis of K_1. Every element $C \in \mathcal{M}$ is homologous to a cycle $\sum_{i=1}^{g} a_i \zeta_i$ where $a_i \in \{0, 1\}$ for each i. Then, for some fixed integers p and q, $\mathcal{M} = \{B_1, \ldots, B_q, C_1, \ldots, C_p\}$ is indexed so that the elements $B_1, \ldots, B_q$ are null-homologous and the elements $C_1, \ldots, C_p$ are non-bounding cycles. Also, we have $w(B_j) \leq w(B_{j+1})$ for $1 \leq j < q$ (with ties broken arbitrarily), and $w(C_j) \leq w(C_{j+1})$ for $1 \leq j < p$ (with ties broken arbitrarily).

▶ **Lemma 8.**
1. *For every minimum homology basis,* $w(\zeta_1) = w(C_1)$.
2. *There exists a minimum homology basis* $\overline{\mathcal{H}}$ *with* ζ_1 *homologous to* C_1.

Proof. Suppose there exists a minimum homology basis with $w(\zeta_1) < w(C_1)$. Let $\zeta_1 = \sum_{i=1}^{p} a_i C_i + \sum_{j=1}^{q} b_j B_j$, where $a_i \in \{0,1\}$ for each i and $b_j \in \{0,1\}$ for each j. Since ζ_1 is a non-bounding cycle, there exists at least one i with $a_i = 1$. Let $\ell \in [1,p]$ be the largest index in the above equation with $a_\ell = 1$. Rewriting the equation, we obtain $C_\ell = \sum_{i=1}^{\ell-1} a_i C_i + \sum_{j=1}^{q} b_j B_j + \zeta_1$. Since $w(\zeta_1) < w(C_1)$ by assumption, we have $w(\zeta_1) < w(C_\ell)$ because $w(C_\ell) \geq w(C_1)$ by indexing of $\mathcal{M}$. It follows that the basis obtained by exchanging C_ℓ for ζ_1, that is, $\{B_1, \ldots, B_q, \zeta_1, C_1, \ldots, C_{\ell-1}, C_{\ell+1}, \ldots, C_p\}$ gives a smaller cycle basis than the minimum one, a contradiction.

Now, suppose there exists a minimum homology basis with $w(\zeta_1) > w(C_1)$. Let $C_1 = \sum_{i=1}^{g} a_i \zeta_i + \sum_{j=1}^{b} b_j \eta_j$. As before, since C_1 is not null-homologous, there exists at least one i with $a_i = 1$. Let $\ell \in [1,g]$ be the largest index in the above equation with $a_\ell = 1$. Then, $\zeta_\ell = \sum_{i=1}^{\ell-1} a_i \zeta_i + \sum_{j=1}^{b} b_j \eta_j + C_1$. Note that $w(\zeta_\ell) \geq w(\zeta_1)$ because of the indexing, and $w(\zeta_1) > w(C_1)$ by assumption. Therefore, the set $\{C_1, \zeta_1, \ldots, \zeta_{\ell-1}, \zeta_{\ell+1}, \ldots, \zeta_p\}$ obtained by exchanging ζ_ℓ for C_1 gives a smaller homology basis than the minimum one, a contradiction. This proves the first part of the lemma.

From the first part of the lemma, we have $w(\zeta_1) = w(C_1)$ for every minimum homology basis. Let $\mathcal{H}$ be an arbitrary minimum homology basis. Then, if C_1 is not homologous to $\zeta_1 \in \mathcal{H}$, by using basis exchange we can obtain $\overline{\mathcal{H}} = \{C_1, \zeta_1, \ldots, \zeta_{\ell-1}, \zeta_{\ell+1}, \ldots, \zeta_p\}$, which is the minimum homology basis with its first element homologous to C_1, and having the same weight as $w(C_1)$, proving the claim. ◀

We now prove a theorem which allows us to harness fast algorithms for minimum cycle basis in service of improving time complexity of algorithms for minimum homology basis.

▶ **Theorem 9.** *Given a simplicial complex* K*, and a minimum cycle basis* $\mathcal{M} = \{B_1, \ldots, B_q, C_1, \ldots, C_p\}$ *of* K_1*, there exists a minimum homology basis* $\overline{\mathcal{H}}$ *of* K*, and a set* $\{C_{i_1}, \ldots, C_{i_g}\} \subset \{C_1, \ldots, C_p\} \subset \mathcal{M}$ *such that, for every* $k \in [1,g]$*, we have* C_{i_k} *homologous to a cycle spanned by* $\zeta_1, \ldots, \zeta_k$*, and* $w(C_{i_k}) = w(\zeta_k)$*. Moreover,* $i_1 = 1$*, and* i_k *for* $k > 1$ *is the smallest index for which* C_{i_k} *is not homologous to any cycle spanned by* $\{C_{i_1}, \ldots, C_{i_{k-1}}\}$*. In particular, the set* $\{C_{i_1}, \ldots, C_{i_g}\} \subset \mathcal{M}$ *constitutes a minimum homology basis of* K*.*

Proof. The key argument is essentially the same as for the proof of Lemma 8. Nonetheless, we present it here for the sake of completeness. We shall prove the claim by induction. Lemma 8 covers the base case. By induction hypothesis, there is an integer k, and a minimum homology basis $\mathcal{H} = \{\zeta_1, \ldots, \zeta_g\}$, for which, vectors $\{C_{i_1}, \ldots, C_{i_k}\} \subseteq \{C_1, \ldots, C_p\}$ are such that, for every $j \in [1,k]$, we have C_{i_j} homologous to a cycle spanned by $\zeta_1, \ldots, \zeta_j$, and $w(C_{i_j}) = w(\zeta_j)$. Let i_{k+1} be the smallest index for which $C_{i_{k+1}} \in \{C_1, \ldots, C_p\}$ is not homologous to any cycle spanned by $\{C_{i_1}, \ldots, C_{i_k}\}$.

Suppose that $w(\zeta_{k+1}) < w(C_{i_{k+1}})$. Let $\zeta_{k+1} = \sum_{i=1}^{p} a_i C_i + \sum_{j=1}^{q} b_j B_j$. Let $\ell \in [1,p]$ be the largest index in the above equation with $a_\ell = 1$. Then, $C_\ell = \sum_{i=1}^{\ell-1} a_i C_i + \sum_{j=1}^{q} b_j B_j + \zeta_{k+1}$. From the induction hypothesis, we can infer that $\ell \geq i_{k+1}$, and hence $w(C_\ell) \geq w(C_{i_{k+1}})$ by indexing of $\mathcal{M}$. Thus, if $w(\zeta_{k+1}) < w(C_{i_{k+1}})$, then we have $w(\zeta_{k+1}) < w(C_\ell)$. It follows that, $\{B_1, \ldots, B_q, \zeta_{k+1}, C_1, \ldots, C_{\ell-1}, C_{\ell+1}, \ldots, C_p\}$ obtained by exchanging C_ℓ for ζ_{k+1} gives a smaller cycle basis than the minimum one, contradicting the minimality of $\mathcal{H}$.

Now, suppose that $w(\zeta_{k+1}) > w(C_{i_{k+1}})$. Let $C_{i_{k+1}} = \sum_{i=1}^{g} a_i \zeta_i + \sum_{j=1}^{b} b_j \eta_j$. Let $\ell \in [1,g]$ be the largest index in the above equation with $a_\ell = 1$. Rewriting the equation, we obtain $\zeta_\ell = \sum_{i=1}^{\ell-1} a_i \zeta_i + \sum_{j=1}^{b} b_j \eta_j + C_{i_{k+1}}$. Again, using the induction hypothesis, $\ell \geq k+1$, and

hence, $w(\zeta_\ell) \geq w(\zeta_{k+1})$ because of the indexing. Since we have assumed $w(\zeta_{k+1}) > w(C_{i_{k+1}})$, this gives us $w(\zeta_\ell) > w(C_{i_{k+1}})$. Hence, the set $\{C_{i_{k+1}}, \zeta_1, \ldots, \zeta_{\ell-1}, \zeta_{\ell+1}, \ldots, \zeta_p\}$ obtained by exchanging ζ_ℓ for $C_{i_{k+1}}$ gives a smaller homology basis than the minimum one, contradicting the minimality of $\mathcal{H}$.

From the first part of the proof, we have established that $w(C_{i_{k+1}}) = w(\zeta_{k+1})$. So, if $C_{i_{k+1}}$ is not homologous to $\zeta_{k+1} \in \mathcal{H}$ and $w(\zeta_{k+1}) = w(C_{i_{k+1}})$, then $\overline{\mathcal{H}} = \{C_{i_{k+1}}, \zeta_1, \ldots, \zeta_{\ell-1}, \zeta_{\ell+1}, \ldots, \zeta_p\}$ obtained by exchanging ζ_ℓ for $C_{i_{k+1}}$ is the desired minimum homology basis, proving the induction claim. ◀

Previously, it was known from Erickson and Whittlesey [11] that $\mathcal{H}$ is contained in $\mathcal{T}$.

▶ **Theorem 10** (Erickson and Whittlesey [11])**.** *With non-negative weights, every cycle in a shortest basis of $\mathsf{H}_1(K)$ is tight. That is, if $\mathcal{H}$ is any minimum homology basis of K, then $\mathcal{H} \subset \mathcal{T}$.*

Using Theorems 1 and 9, we can refine the above observation.

▶ **Corollary 11.** *Let $\mathcal{T}$ denote the set of tight cycles of K_1, and let $\mathcal{M}$ be a minimum cycle basis of K_1. Then, there exists a minimum homology basis $\mathcal{H}$ of K such that $\mathcal{H} \subset \mathcal{M} \subset \mathcal{T}$.*

5 Algorithms for minimum homology basis

To begin with, note that since $\mathsf{C}_p(K), \mathsf{Z}_p(K), \mathsf{B}_p(K)$ and $\mathsf{H}_p(K)$ are vector spaces, the problem of computing a minimum homology basis can be couched in terms of matrix operations.

Given a complex K, let $\{C_1, \ldots, C_{|\mathcal{T}|}\}$ be the list of tight cycles in K_1 sorted by weight, and let $\mathbf{M}_{\mathcal{T}}(K_1) = [C_1\, C_2 \ldots\, C_{|\mathcal{T}|}]$ be the matrix formed with cycles C_i as its columns. Then, the matrix $\hat{\mathbf{Z}} = [\partial_2 \mid \mathbf{M}_{\mathcal{T}}(K_1)]$ has $O(N + n\nu)$ columns and $O(N + n\nu)$ non-zero entries since $\mathbf{M}_{\mathcal{T}}(K_1)$ has $O(n\nu)$ columns and $O(n\nu)$ non-zero entries by Theorem 4, and ∂_2 has $O(N)$ columns and $O(N)$ non-zero entries. Since $\hat{\mathbf{Z}}$ has m rows, the rank of $\hat{\mathbf{Z}}$ is bounded by m. This immediately suggests an algorithm for computing minimum homology basis analogous to Algorithm 3.

Algorithm 4 Algorithm for minimum homology basis.

1: Compute the sorted list of tight cycles in $\mathbf{M}_{\mathcal{T}}(K_1)$, and assemble matrix $\hat{\mathbf{Z}}$.
2: Compute the column rank profile $[j_1, j_2, \ldots, j_b, i_1, i_2, \ldots, i_g]$ of $\hat{\mathbf{Z}}$ using Storjohann and Yang's algorithm [20], where columns $\{\hat{\mathbf{Z}}_{j_k}\}$ and $\{\hat{\mathbf{Z}}_{i_\ell}\}$ are linearly independent columns of ∂_2 and $\mathbf{M}_{\mathcal{T}}(K_1)$ respectively.
3: Return Columns $\{\hat{\mathbf{Z}}_{i_1}, \hat{\mathbf{Z}}_{i_2}, \ldots, \hat{\mathbf{Z}}_{i_g}\}$.

▶ **Theorem 12.** *Algorithm 4 is a Monte Carlo algorithm for computing a minimum homology basis that runs in $\tilde{O}(m^\omega)$ time with failure probability at most $\frac{1}{2}$.*

Proof. The correctness of the algorithm is an immediate consequence of Theorem 9 since, by definition, i_k is the smallest index for which $\hat{\mathbf{Z}}_{i_k}$ is not homologous to any cycle spanned by $\{\hat{\mathbf{Z}}_{i_1}, \ldots, \hat{\mathbf{Z}}_{i_{k-1}}\}$.

The list of tight cycles in G can be computed in $O(nm)$ time using the algorithm described in Section 2 of [1]. Hence, Step 1 of Algorithm 4 takes $O(nm \log n)$ time. Moreover, using Theorem 6, the complexity of Step 2 is bounded by $\tilde{O}(N + n\nu + m^\omega)$, which is the same as $\tilde{O}(m^\omega)$ since N and $n\nu$ are both in $\tilde{O}(m^\omega)$, and the failure probability is at most $1/2$. ◀

When the number of 2-simplices in complex K is significantly smaller than the number of edges, the complexity for minimum homology can be slightly improved by decoupling the minimum homology basis computation from the minimum cycle basis computation, as illustrated in Algorithm 5.

Algorithm 5 Algorithm for minimum homology basis.

1: Compute a minimum cycle basis $\mathcal{M}$ of K_1 using the Monte Carlo algorithm by Amaldi et al. [1]. Let $\mathbf{B}_{\mathcal{M}}$ be the matrix whose columns are cycle vectors in $\mathcal{M}$ sorted by weight.
2: Assemble the matrix $\tilde{\mathbf{Z}} = [\partial_2 \mid \mathbf{B}_{\mathcal{M}}]$.
3: Compute the column rank profile $[j_1, j_2, \ldots, j_b, i_1, i_2, \ldots, i_g]$ of $\tilde{\mathbf{Z}}$ using the deterministic algorithm by Jeannerod et al. [14], where columns $\{\tilde{\mathbf{Z}}_{j_k}\}$ and $\{\tilde{\mathbf{Z}}_{i_\ell}\}$ are linearly independent columns of ∂_2 and $\mathbf{B}_{\mathcal{M}}$ respectively.
4: RETURN Columns $\{\tilde{\mathbf{Z}}_{i_1}, \tilde{\mathbf{Z}}_{i_2}, \ldots, \tilde{\mathbf{Z}}_{i_g}\}$.

▶ **Theorem 13.** *Minimum homology basis can be computed in $O(m^\omega + Nm^{\omega-1})$ time using the Monte Carlo algorithm described in Algorithm 5. The algorithm fails with probability at most $\nu \log(nm)\, 2^{-k}$, where $k = m^{0.1}$.*

Proof. As in Theorem 12, the correctness of the algorithm is an immediate consequence of Theorem 9. The algorithm fails only when Step 1 returns an incorrect answer, the probability of which is as low as $\nu \log(nm)\, 2^{-k}$, where $k = m^{0.1}$, see Theorem 3.2 of [1].

The minimum cycle basis algorithm by Amaldi et al. [1] runs in $O(m^\omega)$ time (assuming the current exponent of matrix multiplication $\omega > 2$). Furthermore, using Theorem 5, the complexity of Line 3 is bounded by $O(Nm^{\omega-1})$. So, the overall complexity of the algorithm is $O(m^\omega + Nm^{\omega-1})$. ◀

Note that in Line 3 of Algorithm 5, it is possible to replace the deterministic algorithm by Jeannerod et al. [14] with the Monte Carlo algorithm by Storjohann and Yang's algorithm [20]. In that case, the complexity of the algorithm will once again be $\tilde{O}(m^\omega)$, and the failure probability will be at most $1 - \frac{1}{2}(1 - \nu \log(nm) 2^{-k})$.

We would like to point out that the complexities of Algorithm 4 and Algorithm 5 are, *in general*, not comparable. For instance, for families of complexes with $N^{1-\epsilon} = \omega(m)$, for some $\epsilon > 0$, Algorithm 4 is faster than Algorithm 5. However, for families of complexes with $N = o(m)$, Algorithm 5 is faster than Algorithm 4. Moreover, for families of complexes with $g = \Theta(N)$, where, as before, g denotes the rank of $\mathsf{H}_1(K)$, Algorithms 4 and 5 are both faster than Dey et al.'s algorithm [8] (which runs in $O(N^\omega + N^2 g)$ time).

6 Discussion

In this paper, we show that questions about minimum cycle basis and minimum homology basis can be naturally recast into the problem of computing rank profiles of matrices, leading to fast algorithms with simple and elegant high-level descriptions. The column rank profile (or the earliest basis) of a matrix has previously been used to compute the minimum homology basis of a simplicial complex [3, 8]. Such a greedy approach that picks, at each step, an independent cycle of the smallest index, works because of the matroid structure of homology bases and cycle bases. What's novel about our approach is that we point out that, for both problems, independence can be efficiently checked owing to the sparsity of the matrices comprising of candidate cycles.

It is also worth noting that for the algorithms presented in this paper, the simplicity of high-level description doesn't translate to simple algorithms that can be easily implemented because the black-box subroutines employed by these algorithms are fairly complex.

Maintenance of support vectors has served as a key ingredient in designing algorithms for minimum cycle basis since de Pina. Our algorithm, however, does not explicitly maintain support vectors, and in that sense, is somewhat conceptually different from the recent algorithms for computing minimum cycle bases.

References

1 Edoardo Amaldi, Claudio Iuliano, Tomasz Jurkiewicz, Kurt Mehlhorn, and Romeo Rizzi. Breaking the $O(m^2n)$ barrier for minimum cycle bases. In Amos Fiat and Peter Sanders, editors, *Algorithms - ESA 2009*, pages 301–312, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

2 Glencora Borradaile, Erin Wolf Chambers, Kyle Fox, and Amir Nayyeri. Minimum cycle and homology bases of surface-embedded graphs. *JoCG*, 8(2):58–79, 2017. doi:10.20382/jocg.v8i2a4.

3 Oleksiy Busaryev, Sergio Cabello, Chao Chen, Tamal K. Dey, and Yusu Wang. Annotating simplices with a homology basis and its applications. In Fedor V. Fomin and Petteri Kaski, editors, *Algorithm Theory – SWAT 2012*, pages 189–200, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

4 Chao Chen and Daniel Freedman. Measuring and computing natural generators for homology groups. *Comput. Geom. Theory Appl.*, 43(2):169–181, February 2010. doi:10.1016/j.comgeo.2009.06.004.

5 Chao Chen and Daniel Freedman. Hardness results for homology localization. *Discrete & Computational Geometry*, 45(3):425–448, April 2011. doi:10.1007/s00454-010-9322-8.

6 Ho Yee Cheung, Tsz Chiu Kwok, and Lap Chi Lau. Fast matrix rank algorithms and applications. *J. ACM*, 60(5):31:1–31:25, October 2013. doi:10.1145/2528404.

7 José Coelho de Pina. *Applications of shortest path methods.* PhD thesis, Universiteit van Amsterdam, 1995.

8 Tamal K. Dey, Tianqi Li, and Yusu Wang. Efficient algorithms for computing a minimal homology basis. In Michael A. Bender, Martín Farach-Colton, and Miguel A. Mosteiro, editors, *LATIN 2018: Theoretical Informatics*, pages 376–398, Cham, 2018. Springer International Publishing.

9 Tamal K. Dey, Jian Sun, and Yusu Wang. Approximating loops in a shortest homology basis from point data. In *Proceedings of the Twenty-sixth Annual Symposium on Computational Geometry*, SoCG '10, pages 166–175, New York, NY, USA, 2010. ACM. doi:10.1145/1810959.1810989.

10 Jean-Guillaume Dumas, Clément Pernet, and Ziad Sultan. Simultaneous computation of the row and column rank profiles. In *Proceedings of the 38th International Symposium on Symbolic and Algebraic Computation*, ISSAC '13, pages 181–188, New York, NY, USA, 2013. ACM. doi:10.1145/2465506.2465517.

11 Jeff Erickson and Kim Whittlesey. Greedy optimal homotopy and homology generators. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '05, pages 1038–1046, Philadelphia, PA, USA, 2005. Society for Industrial and Applied Mathematics.

12 Allen Hatcher. *Algebraic topology.* Cambridge University Press, Cambridge, 2002.

13 J. D. Horton. A polynomial-time algorithm to find the shortest cycle basis of a graph. *SIAM J. Comput.*, 16(2):358–366, April 1987. doi:10.1137/0216026.

14 Claude-Pierre Jeannerod, Clément Pernet, and Arne Storjohann. Rank-profile revealing gaussian elimination and the cup matrix decomposition. *Journal of Symbolic Computation*, 56:46–68, 2013. doi:10.1016/j.jsc.2013.04.004.

15 Telikepalli Kavitha, Christian Liebchen, Kurt Mehlhorn, Dimitrios Michail, Romeo Rizzi, Torsten Ueckerdt, and Katharina A. Zweig. Cycle bases in graphs characterization, algorithms, complexity, and applications. *Computer Science Review*, 3(4):199–243, 2009. doi:10.1016/j.cosrev.2009.08.001.

16 Telikepalli Kavitha, Kurt Mehlhorn, and Dimitrios Michail. New approximation algorithms for minimum cycle bases of graphs. *Algorithmica*, 59(4):471–488, April 2011. doi:10.1007/s00453-009-9313-4.

17 Telikepalli Kavitha, Kurt Mehlhorn, Dimitrios Michail, and Katarzyna Paluch. A faster algorithm for minimum cycle basis of graphs. In Josep Díaz, Juhani Karhumäki, Arto Lepistö, and Donald Sannella, editors, *Automata, Languages and Programming*, pages 846–857, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

18 Kurt Mehlhorn and Dimitrios Michail. Minimum cycle bases: Faster and simpler. *ACM Trans. Algorithms*, 6(1):8:1–8:13, December 2009. doi:10.1145/1644015.1644023.

19 Arne Storjohann and Shiyun Yang. Linear independence oracles and applications to rectangular and low rank linear systems. In *Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation*, ISSAC '14, pages 381–388, New York, NY, USA, 2014. ACM. doi:10.1145/2608628.2608673.

20 Arne Storjohann and Shiyun Yang. A relaxed algorithm for online matrix inversion. In *Proceedings of the 2015 ACM on International Symposium on Symbolic and Algebraic Computation*, ISSAC '15, pages 339–346, New York, NY, USA, 2015. ACM. doi:10.1145/2755996.2756672.

21 Shiyun Yang. Algorithms for fast linear system solving and rank profile computation. Master's thesis, University of Waterloo, 2014.

Dense Graphs Have Rigid Parts

Orit E. Raz
Einstein Institute of Mathematics, The Hebrew University of Jerusalem, Israel
oritraz@mail.huji.ac.il

József Solymosi
Department of Mathematics, University of British Columbia, Vancouver, B.C., Canada
solymosi@math.ubc.ca

Abstract

While the problem of determining whether an embedding of a graph G in $\mathbb{R}^2$ is *infinitesimally rigid* is well understood, specifying whether a given embedding of G is *rigid* or not is still a hard task that usually requires ad hoc arguments. In this paper, we show that *every* embedding (not necessarily generic) of a dense enough graph (concretely, a graph with at least $C_0 n^{3/2}(\log n)^\beta$ edges, for some absolute constants $C_0 > 0$ and β), which satisfies some very mild general position requirements (no three vertices of G are embedded to a common line), must have a subframework of size at least three which is rigid. For the proof we use a connection, established in Raz [*Discrete Comput. Geom.*, 2017], between the notion of graph rigidity and configurations of lines in $\mathbb{R}^3$. This connection allows us to use properties of line configurations established in Guth and Katz [*Annals Math.*, 2015]. In fact, our proof requires an extended version of Guth and Katz result; the extension we need is proved by János Kollár in an Appendix to our paper.

We do not know whether our assumption on the number of edges being $\Omega(n^{3/2}\log n)$ is tight, and we provide a construction that shows that requiring $\Omega(n \log n)$ edges is necessary.

2012 ACM Subject Classification Mathematics of computing → Combinatoric problems; Mathematics of computing → Graph theory

Keywords and phrases Graph rigidity, line configurations in 3D

Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.65

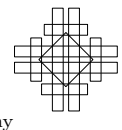
Funding *József Solymosi*: The work of the second author has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 741420, 617747, 648017). His research is also supported by NSERC and OTKA (K 119528) grants.

Acknowledgements The authors also thank Omer Angel and Ching Wong for several useful comments regarding the paper.

1 Introduction

Let $G = ([n], E)$ be a graph on n vertices and m edges, and let $\mathbf{p} = (p_1, \ldots, p_n)$ be an embedding of the vertices of G in $\mathbb{R}^2$. A pair $(G, \mathbf{p})$ of a graph and an embedding is called a *framework*. A pair of frameworks $(G, \mathbf{p})$ and $(G, \mathbf{q})$ are *equivalent* if for every edge $\{i, j\} \in E(G)$ we have $\|p_i - p_j\| = \|q_i - q_j\|$, where $\|\cdot\|$ stands for the standard Euclidean norm in $\mathbb{R}^2$. Two frameworks are *congruent* if there is a rigid motion of $\mathbb{R}^2$ that maps p_i to q_i for every i; equivalently, if $\|p_i - p_j\| = \|q_i - q_j\|$ for every pair i, j (not necessarily in $E(G)$). We say a framework $(G, \mathbf{p})$ is rigid if there exists a neighborhood B of $\mathbf{p}$ (in $(\mathbb{R}^2)^n$), such that, for every equivalent framework $(G, \mathbf{p}')$, with $\mathbf{p}' \in B$, we have that the two frameworks are in fact congruent.

For a given G, if there exists an embedding $\mathbf{p}_0$ of its vertices, such that the framework $(G, \mathbf{p}_0)$ is rigid, then it is known that in fact for every *generic* embedding $\mathbf{p}$ the framework $(G, \mathbf{p})$ is rigid (see [1]). In this sense one can define the notion of rigidity of an abstract graph



G in $\mathbb{R}^2$, without specifying an embedding. That is, a graph G is *rigid* in $\mathbb{R}^2$ if a generic embedding $\mathbf{p}$ of its vertices in $\mathbb{R}^2$ yields a rigid framework $(G, \mathbf{p})$. A graph G is *minimally rigid* if it is rigid and removing any of its edges results in a non-rigid graph. Graphs that are minimally rigid in $\mathbb{R}^2$ have a simple combinatorial characterization, described by Geiringer [7] and (later) by Laman [6]. Namely, a graph G with n vertices is minimally rigid in $\mathbb{R}^2$ if and only if G has exactly $2n-3$ edges and every subgraph of G with k vertices has at most $2k-3$ edges. Every rigid graph has a minimally rigid subgraph.

To see that rigidity is indeed a generic notion, one defines the stricter notion of *infinitesimal rigidity*. Given a graph G as above, consider the map $f_G : (\mathbb{R}^2)^n \to \mathbb{R}^{|E|}$, given by

$$\mathbf{p} \mapsto (\|p_i - p_j\|)_{\{i,j\}\in E},$$

for some arbitrary (but fixed) ordering of the edges of G. Let M_G be the Jacobian matrix of f_G (which is an $|E| \times 2n$ matrix). A framework $(G, \mathbf{p})$ is called *infinitesimally rigid* if the rank of M_G at $\mathbf{p}$ is exactly $2n-3$. It is not hard to see that the rank of M_G is always at most $2n-3$. Combining this with the fact that a generic embedding $\mathbf{p}$ achieves the maximal rank of M_G, one concludes that being infinitesimally rigid is a generic property. As it turns out (and not hard to prove), infinitesimal rigidity of $(G, \mathbf{p})$ implies rigidity of $(G, \mathbf{p})$, and therefore it follows that rigidity is a generic notion too. Moreover, for rigid graphs G, it is straightforward to describe a (measure zero) subset X of $\mathbb{R}^{2n}$ where the rank of M_G is strictly smaller than $2n-3$, and thus for such embeddings $\mathbf{p}$ the framework $(G, \mathbf{p})$ is not infinitesimally rigid. However, for $\mathbf{p} \in X$, $(G, \mathbf{p})$ might be rigid or not. To tell whether a given $\mathbf{p} \in X$ is rigid or not is a non-trivial task, and we are not aware of any general method to test it, rather than ad-hoc arguments specific to the given graph.

Our results

In this paper, we show that *every* embedding (not necessarily generic) of a dense enough graph, that satisfies some very mild general position requirements, must have a subframework of size at least three which is rigid. Concretely, we prove the following theorem.

▶ **Theorem 1.** *There exists an absolute constant C_0 such that the following holds. Let G be a graph on n vertices and $C_0 n^{3/2}(\log n)^{\beta}$ edges. Let $\boldsymbol{p} = (p_1, \ldots, p_n)$ be an (injective) embedding of the vertices of G in $\mathbb{R}^2$ such that no three of the vertices are embedded to a common line. Then there exists a subset $S \subset [n]$ of size at least three, such that the framework $(G[S], P_S)$, where $P_S := \{p_i \mid i \in S\}$, is rigid.*

We do not know whether the assumption that G has $\Omega(n^{3/2}\log n)$ edges in Theorem 1 is necessary, and in fact we believe an analogue statement should hold for graphs with less edges. The following theorem yields a lower bound on the number of edges, namely, $\Omega(n \log n)$, needed for the conclusion in Theorem 1 to hold.

▶ **Theorem 2.** *For every $d \geq 2$, there exists a graph H_d, with $n = 2^d$ vertices and $\frac{1}{2}n\log n$ edges, and an embedding $\boldsymbol{p}$ of H_d in $\mathbb{R}^2$, such that no three vertices of H_d are embedded to a common line in $\mathbb{R}^2$ and every subframework of $(H_d, \boldsymbol{p})$ of size at least three is non-rigid.*

The paper is organized as follows. In Section 2, we review a connection established in [8] between rigidity questions and certain line configurations in $\mathbb{R}^3$. In Section 3, we establish some properties regarding embeddings of complete bipartite graphs in $\mathbb{R}^2$. In Section 4, we review results from Guth and Katz [4] regarding point-line incidences in $\mathbb{R}^3$ and state a refined incidence result (proved in an Appendix to our paper by János Kollár). In Section 5, we give the proof of Theorem 1. In Section 6, we provide a construction that proves Theorem 2.

2 Rigidity in the plane and line configurations in $\mathbb{R}^3$

In this section we review some known facts that we need for our analysis. We review a reduction, introduced first in Raz [8], to connect the notion of graph rigidity of planar structures with line configurations in $\mathbb{R}^3$. The reduction uses the so called Elekes–Sharir framework, see [3, 4]. Specifically, we represent each orientation-preserving rigid motion of the plane (called a *rotation* in [3, 4]) as a point $(c, \cot(\theta/2))$ in $\mathbb{R}^3$, where c is the center of rotation, and θ is the (counterclockwise) angle of rotation. (Note that pure translations are mapped in this manner to points at infinity.) Given a pair of distinct points $a, b \in \mathbb{R}^2$, the locus of all rotations that map a to b is a line $\ell_{a,b}$ in the above parametric 3-space, given by the parametric equation

$$\ell_{a,b} = \{(u_{a,b} + tv_{a,b},\ t) \mid t \in \mathbb{R}\}, \tag{1}$$

where $u_{a,b} = \frac{1}{2}(a+b)$ is the midpoint of ab, and $v_{a,b} = \frac{1}{2}(a-b)^{\perp}$ is a vector orthogonal to $\vec{ab}$ of length $\frac{1}{2}\|a-b\|$, with $\vec{ab}$, $v_{a,b}$ positively oriented (i.e., $v_{a,b}$ is obtained by turning $\vec{ab}$ counterclockwise by $\pi/2$).

Note that every non-horizontal line ℓ in $\mathbb{R}^3$ can be written as $\ell_{a,b}$, for a unique (ordered) pair $a, b \in \mathbb{R}^2$. More precisely, if ℓ is also non-vertical, the resulting a and b are distinct. If ℓ is vertical, then a and b coincide, at the intersection of ℓ with the xy-plane, and ℓ represents all rotations of the plane about this point.

A simple yet crucial property of this transformation is that, for any pair of pairs (a, b) and (c, d) of points in the plane, $\|a-c\| = \|b-d\|$ if and only if $\ell_{a,b}$ and $\ell_{c,d}$ intersect, at (the point representing) the unique rotation τ that maps a to b and c to d. This also includes the special case where $\ell_{a,b}$ and $\ell_{c,d}$ are parallel, corresponding to the situation where the transformation that maps a to b and c to d is a pure translation (this is the case when $\vec{ac}$ and $\vec{bd}$ are parallel and of equal length).

Note that no pair of lines $\ell_{a,b}$, $\ell_{a,c}$ with $b \neq c$ can intersect (or be parallel), because such an intersection would represent a rotation that maps a both to b and to c, which is impossible.

▶ **Lemma 3** (Raz [8, Lemma 6.1]). *Let $L = \{\ell_{a_i,b_i} \mid a_i, b_i \in \mathbb{R}^2,\ i = 1, \ldots, r\}$ be a collection of $r \geq 3$ (non-horizontal) lines in $\mathbb{R}^3$.*

(a) *If all the lines of L are concurrent, at some common point τ, then the sequences $A = (a_1, \ldots, a_r)$ and $B = (b_1, \ldots, b_r)$ are congruent, with equal orientations, and τ (corresponds to a rotation that) maps a_i to b_i, for each $i = 1, \ldots, r$.*

(b) *If all the lines of L are coplanar, within some common plane h, then the sequences $A = (a_1, \ldots, a_r)$ and $B = (b_1, \ldots, b_r)$ are congruent, with opposite orientations, and h defines, in a unique manner, an orientation-reversing rigid motion h^* that maps a_i to b_i, for each $i = 1, \ldots, r$.*

(c) *If all the lines of L are both concurrent and coplanar, then the points of A are collinear, the points of B are collinear, and A and B are congruent.*

The following corollary is now straightforward.

▶ **Corollary 4.** *Let G be a graph, over n vertices, and let $\boldsymbol{p}$ be an embedding of G in the plane. Assume that there exists an open neighborhood B of $\boldsymbol{p}$ (in $(\mathbb{R}^2)^n$) with the following property: For every $\boldsymbol{p}' \in B$, if $(G, \boldsymbol{p}')$ is equivalent to $(G, \boldsymbol{p})$, then the lines $\ell_i := \ell_{p_i,p_i'}$, for $i = 1, \ldots, n$, are necessarily concurrent. Then the framework $(G, \boldsymbol{p})$ is rigid.*

Proof. This follows from Lemma 3(a) and the definition of rigidity of a framework. ◀

3 Embeddings of complete bipartite graphs in $\mathbb{R}^2$

We first recall a lemma and some notation introduced in Raz [9]. For completeness, we give all the details here. For $\mathbf{p} = (p_1, \ldots, p_{d+1}), \mathbf{p}' = (p'_1, \ldots, p'_{d+1}) \in (\mathbb{R}^d)^{d+1}$, we define

$$\Sigma_{\mathbf{p},\mathbf{p}'} := \{(q, q') \in \mathbb{R}^d \times \mathbb{R}^d \mid \|p_i - q\| = \|p'_i - q'\| \;\; i = 1, \ldots, d+1\},$$

and let $\sigma_{\mathbf{p},\mathbf{p}'}$ (resp., $\sigma'_{\mathbf{p},\mathbf{p}'}$) denote the projection of $\Sigma_{\mathbf{p},\mathbf{p}'}$ onto the first d (resp., last d) coordinates of $\mathbb{R}^d \times \mathbb{R}^d$.

We have the following lemma.

▶ **Lemma 5.** *Let $\boldsymbol{p}, \boldsymbol{p}'$ be in general position. Then $\sigma_{\boldsymbol{p},\boldsymbol{p}'}$ is a quadric surface, and there exists an invertible affine transformation $T : \mathbb{R}^d \to \mathbb{R}^d$, such that $T(\sigma_{\boldsymbol{p},\boldsymbol{p}'}) = \sigma'_{\boldsymbol{p},\boldsymbol{p}'}$ and $(q, q') \in \Sigma_{\boldsymbol{p},\boldsymbol{p}'}$ if and only if $q \in \sigma_{\boldsymbol{p},\boldsymbol{p}'}$ and $q' = T(q)$.*

Proof. By definition, for $(q, q') \in \Sigma_{\mathbf{p},\mathbf{p}'}$ we have

$$\begin{aligned} \|p_i - q\|^2 &= \|p'_i - q'\|^2, && i = 1, \ldots, d+1, \text{ or} \\ \|p_i\|^2 - 2p_i \cdot q + \|q\|^2 &= \|p'_i\|^2 - 2p'_i \cdot q' + \|q'\|^2, && i = 1, \ldots, d+1. \end{aligned}$$

Subtracting the $(d+1)$th equation from each of the other equations, we get the system

$$\begin{aligned} \|p_i\|^2 - \|p_{d+1}\|^2 - 2(p_i - p_{d+1}) \cdot q &= \|p'_i\|^2 - \|p'_{d+1}\|^2 - 2(p'_i - p'_{d+1}) \cdot q', \quad i = 1, \ldots, d \\ \|p_{d+1}\|^2 - 2p_{d+1} \cdot q + \|q\|^2 &= \|p'_{d+1}\|^2 - 2p'_{d+1} \cdot q' + \|q'\|^2. \end{aligned}$$

The system can be rewritten as

$$\begin{aligned} \tfrac{1}{2}u - Aq &= \tfrac{1}{2}v - Bq', \\ \|p_{d+1}\|^2 - 2p_{d+1} \cdot q + \|q\|^2 &= \|p'_{d+1}\|^2 - 2p'_{d+1} \cdot q' + \|q'\|^2, \end{aligned}$$

where A (resp., B) is a $d \times d$ matrix whose ith row equals $p_i - p_{d+1}$ (resp., $p'_i - p'_{d+1}$), and

$$\begin{aligned} u &= \left(\|p_1\|^2 - \|p_{d+1}\|^2, \|p_2\|^2 - \|p_{d+1}\|^2, \ldots, \|p_d\|^2 - \|p_{d+1}\|^2\right) \\ v &= \left(\|p'_1\|^2 - \|p'_{d+1}\|^2, \|p'_2\|^2 - \|p'_{d+1}\|^2, \ldots, \|p'_d\|^2 - \|p'_{d+1}\|^2\right) \end{aligned}$$

are vectors in $\mathbb{R}^d$. Our assumption that each of $\mathbf{p}, \mathbf{p}'$ is in general position implies that each of A, B is invertible. Hence we have

$$q' = B^{-1}Aq + w,$$

for $w = \frac{1}{2}B^{-1}(v - u) \in \mathbb{R}^d$. Let $T(q) := B^{-1}Aq + w$. So $(q, q') \in \Sigma_{\mathbf{p},\mathbf{p}'}$ if and only if $q' = T(q)$ and

$$\|p_{d+1}\|^2 - 2p_{d+1} \cdot q + \|q\|^2 = \|p'_{d+1}\|^2 - 2p'_{d+1} \cdot T(q) + \|T(q)\|^2, \tag{2}$$

where the latter constraint comes from considering the $(d+1)$st equation, using $q' = T(q)$. We conclude that $\sigma_{\mathbf{p},\mathbf{p}'}$ is the quadric given by (2). Moreover, $(q, q') \in \Sigma_{\mathbf{p},\mathbf{p}'}$ if and only if $q \in \sigma_{\mathbf{p},\mathbf{p}'}$ and $q' = T(q)$. Hence, T maps $\sigma_{\mathbf{p},\mathbf{p}'}$ into $\sigma'_{\mathbf{p},\mathbf{p}'}$. This completes the proof. ◀

We now apply Lemma 5 to describe the non-rigid frameworks of $K_{3,m}$ embedded in $\mathbb{R}^2$.

▶ **Lemma 6.** *Let $K_{3,m}$ denote the $3 \times m$ complete bipartite graph and let $\boldsymbol{p} : [3] \to \mathbb{R}^2$ and $\boldsymbol{q} : [m] \to \mathbb{R}^2$ be an embedding of the vertices of $K_{3,m}$ in the plane. Suppose $m \geq 5$. Then the framework $(K_{3,m}, \boldsymbol{p} \cup \boldsymbol{q})$ is rigid, unless $\boldsymbol{p} \cup \boldsymbol{q}$ embeds the vertices of the graph to a pair of two lines in $\mathbb{R}^2$.*

Proof. By Bolker and Roth [2], a framework $(K_{3,m}, \mathbf{p}, \mathbf{q})$ is infinitesimally rigid in $\mathbb{R}^2$ if and only if $\mathbf{p} \cup \mathbf{q}$ embeds the vertices of the graph to a conic section in $\mathbb{R}^2$. (In fact, we only need the property that if the embedding is not on a conic section, then the framework is rigid.) Since infinitesimal rigidity implies rigidity, we only need to consider the case where the image of $\mathbf{p} \cup \mathbf{q}$ is a conic section.

Assume first that the points $\mathbf{p} = (p_1, p_2, p_3)$ lie on a common line in $\mathbb{R}^2$. In this case, the conic section supporting $\mathbf{p} \cup \mathbf{q}$ is necessarily a pair of two lines. So in this case we are done.

Assume next that $\mathbf{p} = (p_1, p_2, p_3)$ are not collinear, and that $\mathbf{p} \cup \mathbf{q}$ is irreducible. Let B be a neighborhood of $\mathbf{p} \cup \mathbf{q}$ and let $(\mathbf{p}', \mathbf{q}') \in B$ be an embedding of the vertices of $K_{3,m}$ to this neighborhood. Taking B sufficiently small, we may assume that also $\mathbf{p}' = (p_1', p_2', p_3')$ are not collinear.

We apply Lemma 5 to the pair $(\mathbf{p}, \mathbf{p}')$. Then there exists an affine transformation $T : \mathbb{R}^2 \to \mathbb{R}^2$, and a quadric surface $\sigma_{\mathbf{p},\mathbf{p}'}$ such that each of $\mathbf{q}, \mathbf{q}'$ lies on a conic section in $\mathbb{R}^2$ (namely, the points of $\mathbf{q}$ lie on $\sigma_{\mathbf{p},\mathbf{p}'}$ and the points of $\mathbf{q}'$ lie on $\sigma'_{\mathbf{p},\mathbf{p}'} = T(\sigma_{\mathbf{p},\mathbf{p}'})$, and we have $q_j' = T(q_j)$ for every $j = 1, \ldots, m$.

Recall that $\mathbf{p} \cup \mathbf{q}$ also lies on a conic section. Since two distinct conic sections can share at most four points, and using $m \geq 5$, we conclude that $\sigma_{\mathbf{p},\mathbf{p}'}$ and the conic section supporting $\mathbf{p} \cup \mathbf{q}$ have a common irreducible component. But $\mathbf{p} \cup \mathbf{q}$ is supported by an irreducible conic section, and therefore $\mathbf{p} \cup \mathbf{q} \subset \sigma_{\mathbf{p},\mathbf{p}'}$.

By the properties of $\sigma_{\mathbf{p},\mathbf{p}'}$ given by Lemma 5, we must have $T(p_i) = p_i'$, for each $i = 1, 2, 3$, since $0 = \|p_i - p_i\| = \|T(p_i) - p_i'\|$. This implies that $\|p_i - p_j\| = \|p_i' - p_j'\|$, for every $i, j = 1, 2, 3$. That is, $\mathbf{p}, \mathbf{p}'$ are congruent configurations, and $T(\mathbf{p} \cup \mathbf{q}) = \mathbf{p}' \cup \mathbf{q}'$. We conclude that T is a rigid motion of $\mathbb{R}^2$ and that $\mathbf{p} \cup \mathbf{q}$, $\mathbf{p}' \cup \mathbf{q}'$ are congruent.

We showed that for some neighborhood B of $(\mathbf{p}, \mathbf{q})$, and for every $(\mathbf{p}', \mathbf{q}') \in B$, if the frameworks $(K_{3,m}, (\mathbf{p}, \mathbf{q})$ and $(K_{3,m}, (\mathbf{p}', \mathbf{q}')$ are equivalent, then they are also congruent. So in this case, the framework $(K_{3,m}, (\mathbf{p}, \mathbf{q})$ is rigd, by definition. This completes the proof of the lemma. ◀

▶ **Corollary 7.** *Let $(\boldsymbol{p}, \boldsymbol{q})$ be an embedding of some $3 + m$ vertices in $\mathbb{R}^2$, with $m \geq 5$, $\boldsymbol{p} = (p_1, p_2, p_3)$, $\boldsymbol{q} = (q_1, \ldots, q_m)$. Suppose that for every neighborhood B of $(\boldsymbol{p}, \boldsymbol{q})$ (in $(\mathbb{R}^2)^{3+m}$), there exists $(\boldsymbol{p}', \boldsymbol{q}') \in B$ such that the following holds: The lines $L_{\boldsymbol{p},\boldsymbol{p}'} := \{\ell_{p_i,p_i'} \mid i = 1, 2, 3\}$ and $L_{\boldsymbol{q},\boldsymbol{q}'} := \{\ell_{q_i,q_i'} \mid i = 1, \ldots, m\}$ lie on a (common) doubly ruled surface Q in $\mathbb{R}^3$. Assume further that the lines of $L_{\boldsymbol{p},\boldsymbol{p}'}$ lie on one ruling of the surface Q and the lines of $L_{\boldsymbol{q},\boldsymbol{q}'}$ on the other ruling of Q. Then the embedding $(\boldsymbol{p}, \boldsymbol{q})$ is supported by a pair of lines in $\mathbb{R}^2$.*

Proof. Let $(\mathbf{p}, \mathbf{q})$ be an embedding of some $3+m$ vertices as in the statement. By assumption, for every neighborhood B of $(\mathbf{p}, \mathbf{q})$ there exists $(\mathbf{p}', \mathbf{q}') \in B$ and a doubly ruled surface Q, such that the lines of $L_{\mathbf{p},\mathbf{p}'}$ lie on one ruling of Q, and the lines of $L_{\mathbf{q},\mathbf{q}'}$ on the other ruling of Q. In particular, $\ell_{p_i,p_i'} \cap \ell_{q_j,q_j'} \neq \emptyset$, for every $i \in [3]$, $j \in [m]$.

By the definition of the lines $\ell_{p_i,p_i'}, \ell_{q_j,q_j'}$, this implies that $\|p_i - q_j\| = \|p_i' - p_j'\|$ for every $i \in [3]$, $j \in [m]$. In other words, regarding $(\mathbf{p}, \mathbf{q})$ and $(\mathbf{p}', \mathbf{q}')$ as embeddings of the graph $K_{3,m}$, we see that the frameworks $(K_{3,m}, (\mathbf{p}, \mathbf{q}))$ and $(K_{3,m}, (\mathbf{p}', \mathbf{q}'))$ are equivalent. Note that these frameworks are not congruent, since the lines $L_{\mathbf{p},\mathbf{p}'} \cup L_{\mathbf{q},\mathbf{q}'}$ are neither concurrent nor coplanar.

Since such an embedding $(\mathbf{p}', \mathbf{q}')$ exists in every neighborhood B of $(\mathbf{p}, \mathbf{q})$, we conclude that the framework $(K_{3,m}, (\mathbf{p}, \mathbf{q}))$ is not rigid. By Lemma 6, $(\mathbf{p}, \mathbf{q})$ is supported by a pair of lines in $\mathbb{R}^2$. This completes the proof. ◀

4 Point-line incidences in $\mathbb{R}^3$

We recall the following theorem of Guth and Katz [4].

▶ **Theorem 8** (Guth and Katz [4, Theorem 2.10]). *Let L be a set of n lines in $\mathbb{R}^3$, such that at most $\sqrt{n}$ lines lie in any plane or any regulus. Then the number of 2-rich points in L is at most $O(n^{3/2})$.*

▶ **Theorem 9** (Guth and Katz [4, Theorem 4.5]). *Let L be a set of n lines in $\mathbb{R}^3$, such that at most $\sqrt{n}$ lines lie in any plane. Let $k \geq 3$. Then the number of points in $\mathbb{R}^3$ incident to at least k lines of L is at most $O\left(n^{3/2}k^{-2} + nk^{-1}\right)$.*

We need a slightly refined version of Theorem 8. We thank János Kollár for providing us with a detailed proof of the required statement; his proof (of, in fact, a slightly stronger statement) is given in the Appendix.

▶ **Theorem 10.** *Let L be a set of n lines in $\mathbb{R}^3$, such that:*
(i) *Every plane in $\mathbb{R}^3$ contains at most $\lceil n^{1/2} \rceil$ lines of L.*
(ii) *Every regulus in $\mathbb{R}^3$ contains at most $2n$ pairs of intersecting lines.*
Then the number of 2-rich points in L is at most $O(n^{3/2})$. ◀

Combining Theorem 9 and Theorem 10, we conclude:

▶ **Theorem 11.** *Let L be a set of n lines in $\mathbb{R}^3$, such that:*
(i) *Every plane in $\mathbb{R}^3$ contains at most $\lceil n^{1/2} \rceil$ lines of L.*
(ii) *Every regulus in $\mathbb{R}^3$ contains at most $2n$ pairs of intersecting lines.*
Let $2 \leq k \leq n$. Then the number of points in $\mathbb{R}^3$ incident to at least k lines of L is at most $O\left(n^{3/2}k^{-2} + nk^{-1}\right)$.

5 Proof of Theorem 1

Consider an embedding $\mathbf{p} = (p_1, \ldots, p_n)$ of the vertices of G in the plane, such that no three of the points are collinear. We prove the theorem by induction on the number, n, of vertices in G. We assume that G has $C_n n^{3/2}$ edges, and later optimize C_n, and get $C_n = C_0 \log n$, for some absolute constant C_0, as in the statement of the theorem. For the induction's base cases, we take $C_3 \leq \cdots \leq C_{n_0}$ to be large enough so that for every $3 \leq k \leq n_0$ we will have $C_k k^{3/2} \geq \binom{k}{2}$. This means that a graph G with k vertices and $C_k k^{3/2}$ edges, for $3 \leq k \leq n_0$, is necessarily the complete graph on k vertices. Since every framework of the complete graph is rigid, this proves the base case.

Assume that the statement is true for every n' with $3 \leq n' < n$ and we prove it for n.

An associated line configuration in $\mathbb{R}^3$

Let $\mathbf{p}' = (p'_1, \ldots, p'_n)$ be another embedding of the vertices of G, taken from a neighborhood B of $\mathbf{p}$, with the property that for every edge $\{i, j\}$ of G, we have $\|p_i - p_j\| = \|p'_i - p'_j\|$. That is, we take $\mathbf{p}'$ such that the frameworks $(G, \mathbf{p})$ and $(G, \mathbf{p}')$ are equivalent. Assume further that each p'_i is taken from a small neighborhood of p_i so that in particular no three points of $\mathbf{p}'$ are collinear. Moreover, we may assume that no triple p'_i, p'_j, p'_k is the reflection of p_i, p_j, p_k. Indeed, taking the neighborhoods of the points p_i sufficiently small we can ensure that the orientation (sign of the determinant of the vectors $\overrightarrow{p_i p_j}, \overrightarrow{p_i p_k}$) is the same in $\mathbf{p}$ and in $\mathbf{p}'$ for every triple i, j, k.

For each $i = 1, \ldots, n$ put $\ell_i := \ell_{p_i, p'_i}$ and consider the set of lines $L = \{\ell_1, \ldots, \ell_n\}$. Note that for every edge $\{i, j\}$ in G, the corresponding lines ℓ_i, ℓ_j necessarily intersect. The other direction is not true; that is, the lines ℓ_i, ℓ_j may intersect even if $\{i, j\}$ is not an edge in G.

Our assumptions on $\mathbf{p}$ and $\mathbf{p}'$, combined with Lemma 3, imply that no three lines of L lie on a common plane.

We claim that taking the neighborhood B of $\mathbf{p}$ to be sufficiently small, and taking $\mathbf{p}' \in B$, we can guarantee that no eight lines of L lie on a common regulus R with at least three lines on each of the rulings of R (note that this means in particular that, for any subset $L' \subset L$ of size $k \geq 8$, no regulus in $\mathbb{R}^3$ contains more than $2k$ pairs of intersecting lines of L'). Indeed, fix any ordered 8-tuple $\pi = (p_{i_1}, \ldots, p_{i_8})$ (a subset of the points of $\mathbf{p}$). Applying Corollary 7 (with $m = 5$), and using our assumption that no three points of $\mathbf{p}$ are collinear, we get that for some neighborhood B_π of π, and for every $\pi' = (p'_{i_1}, \ldots, p'_{i_8}) \in B_\pi$, the lines $\{\ell_{p_{i_1}, p'_{i_1}}, \ldots, \ell_{p_{i_8}, p'_{i_8}}\}$ do not lie on a common regulus such that $\{\ell_{p_{i_1}, p'_{i_1}}, \ell_{p_{i_2}, p'_{i_2}}, \ell_{p_{i_3}, p'_{i_3}}\}$ lie on one ruling of the regulus and $\{\ell_{p_{i_4}, p'_{i_4}}, \ldots, \ell_{p_{i_8}, p'_{i_8}}\}$ on the other ruling of the regulus. Repeating this for each ordered 8-tuples of $\mathbf{p}$, we see that there exists a neighborhood B of $\mathbf{p}$ such that the claim follows.

Note in addition that, by Corollary 4, if for every choice of $\mathbf{p}'$, in any arbitrarily small neighborhood of $\mathbf{p}$, the lines of L are concurrent, this means that the framework $(G, \mathbf{p})$ is rigid, and we are done. We therefore assume that the lines of L are not concurrent.

No dense subgraphs of G

Note that, by our induction hypothesis, if G contains a subgraph with $3 \leq n' < n$ vertices and $C_{n'}(n')^{3/2}$ edges, we are done. Therefore we assume that every subgraph of G with $3 \leq n' < n$ vertices has less than $C_{n'}(n')^{3/2}$ edges.

We call a point in $\mathbb{R}^3$ *k-rich* if it is incident to exactly k lines of L. Such a point is the intersection point of exactly $\binom{k}{2}$ pairs of lines, but possibly only a subset of those pairs correspond to edges of G. Our assumption that G has no dense subgraphs implies in particular, that for every k-rich point, with $3 \leq k < n$, the number of pairs of lines meeting at that point that also form an edge in G is at most $C_k k^{3/2}$.

Clearly, every 2-rich point, is the intersection of exactly one pair of lines and hence corresponds to at most one edge of G. We set C_2 to satisfy $C_2 2^{3/2} \geq 1$.

For $t = 2, \ldots, \log n$, let $E_t \subset E$ be the subset of edges that meet at a k-rich point for $2^{t-1} \leq k < 2^t$. Clearly, we have $E = \bigcup_{t=2}^{\log n} E_t$. We apply Theorem 11 to upper bound $\sum_{t=1}^{\log(n/d)} |E_t|$, for some parameter d, which we choose later. We split the sum into two separate sums, according to which additive term in the bound from Theorem 11 dominates.

Edges meeting at a k-rich point, for $2 \leq k \leq n^{1/2}$

For $2 \leq t < \frac{1}{2} \log n$, we have, by Theorem 11, that

$$|E_t| \leq \frac{\rho n^{3/2}}{2^{2(t-1)}} \cdot C_{2^t} (2^t)^{3/2} = 4\rho C_{2^t} n^{3/2} \frac{1}{2^{t/2}},$$

where ρ is some absolute constant (given implicitly in Theorem 11). Thus

$$\begin{aligned}\sum_{t=2}^{\lfloor \frac{1}{2}\log n\rfloor} |E_t| &\le 4\rho C_{n^{1/2}} n^{3/2} \sum_{t=2}^{\lfloor \frac{1}{2}\log n\rfloor} \frac{1}{2^{t/2}} \\ &\le 4\rho C_{n^{1/2}} n^{3/2} \cdot \frac{\frac{1}{2}(1-\frac{2^{1/2}}{n^{1/4}})}{1-2^{-1/2}} \\ &\le \rho' C_{n^{1/2}} n^{3/2},\end{aligned}$$

for some absolute constant ρ'.

Edges meeting at a k-rich point, for $n^{1/2} \le k \le n/d$

Similarly, for $\frac{1}{2}\log n \le t \le \log(n/d)$, where $d > 2$ is a parameter, we have

$$|E_t| \le \frac{\rho n}{2^{t-1}} \cdot C_{2^t}(2^t)^{3/2} = 2\rho C_{2^t} n 2^{t/2},$$

for some absolute constant ρ. Thus

$$\begin{aligned}\sum_{t=\lceil \frac{1}{2}\log n\rceil}^{\lfloor \log(n/d)\rfloor} |E_t| &\le 2\rho C_{n/d} n \sum_{t=\lceil \frac{1}{2}\log n\rceil}^{\lfloor \log(n/d)\rfloor} 2^{t/2} \\ &\le 2\rho C_{n/d} n \cdot \frac{2^{1/2} n^{1/4}}{2^{1/2}-1} \left(\left(\frac{n^{1/2}}{d} \right)^{1/2} - 1 \right) \\ &\le \frac{\rho''}{\sqrt{d}} C_{n/d} n^{3/2},\end{aligned}$$

for some absolute constant ρ''.

Combining the two inequalities above, we get

$$\sum_{t=2}^{\lfloor \log(n/d)\rfloor} |E_t| \le B \left(C_{n^{1/2}} + \frac{C_{n/d}}{\sqrt{d}} \right) n^{3/2}, \tag{3}$$

where $B := \max\{\rho', \rho''\}$ is an absolute constant. That is, (3) gives an upper bound on the number of edges of G that correspond to pairs of lines meeting at a k-rich point, with $2 \le k \le n/d$.

Recall our assumption that G has at least $C_n n^{3/2}$ edges (and each edge corresponds to a pair of meeting lines of L). We take C_n so that

$$C_n \ge 2B \left(C_{n^{1/2}} + \frac{C_{n/d}}{\sqrt{d}} \right).$$

With this choice, and in view of (3), we get that

$$\sum_{t=2}^{\lfloor \log(n/d)\rfloor} |E_t| \le \frac{1}{2} C_n n^{3/2}.$$

We conclude that at least half of the edges of G meet at a k-rich point, for $k > n/d$. In particular, there exists a point which is k-rich, with $k > n/d$.

αn-rich point

Assume first that there exists a point which is αn-rich, with $1/d \leq \alpha \leq 2/3$. Let L_1 denote the subset of αn lines going through this point. If the number of edges meeting at that point (i.e., the number of pairs of lines of L_1 that correspond to an edge in G) is at least $C_{\alpha n}(\alpha n)^{3/2}$, then we are done by induction. Consider the subset of lines $L_2 := L \setminus L_1$ that do not go through this αn-rich point. If the number of edges induced by L_2 is at least $C_{(1-\alpha)n}((1-\alpha)n)^{3/2}$, we are again done by induction. Finally, note that every line of L_2 intersects at most one line of L_1. Otherwise, we would have three coplanar lines, contradicting our assumption. Therefore, the total number of edges we have is at most

$$C_{\alpha n}(\alpha n)^{3/2} + C_{(1-\alpha)n}((1-\alpha)n)^{3/2} + (1-\alpha)n,$$

which must be at least $C_n n^{3/2}$, by our assumption on the number of edges in G. Thus

$$C_{\alpha n}\alpha^{3/2} + C_{(1-\alpha)n}(1-\alpha)^{3/2} + (1-\alpha)n^{-1/2} \geq C_n.$$

Using $C_{\alpha n}, C_{(1-\alpha)n} \leq C_n$ (by monotonicity of the sequence C_n), this implies

$$C_n(\alpha^{3/2} + (1-\alpha)^{3/2}) + (1-\alpha)n^{-1/2} \geq C_n$$

or

$$\frac{1-\alpha}{C_n n^{1/2}} \geq 1 - \alpha^{3/2} - (1-\alpha)^{3/2}. \tag{4}$$

Using $1/d \leq \alpha \leq 2/3$, we have

$$\frac{1-\alpha}{C_n n^{1/2}} \leq \frac{d-1}{dC_n n^{1/2}}.$$

Combined with (4), the last inequality implies

$$1 - \alpha^{3/2} - (1-\alpha)^{3/2} \leq \frac{d-1}{dC_n n^{1/2}}. \tag{5}$$

Note that for every $0 < \alpha < 1$, the left-hand side of (5) is positive. Moreover, for every closed interval $[a, b] \subset [0, 1]$, with $0 < a < b < 1$, the function $f(\alpha) = 1 - \alpha^{3/2} - (1-\alpha)^{3/2}$ attains a minimum which is a positive number. Let $\delta_0 > 0$ denote the minimum of f over $[1/d, 2/3]$. Taking n_0 large enough (and recalling that $n \geq n_0$), the right-hand side of (5) can be guaranteed to be smaller than δ_0 (for any positive δ_0). This yields a contradiction to (5).

k-rich point, with $k > 2n/3$

Assume next that there exists a k-rich point with $k > 2n/3$. Fix such a point, and denote by m the number of lines not incident to this point. That is, we fix a $(n-m)$-rich point, with $m < n/3$. Note that $m \geq 1$, by our assumption that not all the lines of L are concurrent.

Similar to the analysis in the previous case above, if the number of edges meeting at the given $n - m$ rich point is at least $C_{n-m}(n-m)^3/2$, then we are done by induction. Thus, we assume this is not the case. Note that in this case, and if $m = 2$, we get that in this case the total number of edges in G is at most

$$C_{n-2}(n-2)^{3/2} + 1 + 2,$$

where here we used our assumption that no three lines of L lie on a common plane. So we must have

$$C_{n-2}(n-2)^{3/2} + 1 + 2 \geq C_n n^{3/2}$$

which implies

$$3 \geq C_n(n^{3/2} - (n-2)^{3/2}),$$

which yields a contradiction, taking C_n larger than some absolute constant. So we must have $m \geq 3$.

Next, if the number of edges among the m lines not incident to our $(n-m)$-rich point is at least $C_m m^{3/2}$, we are again done by induction. Otherwise, we have that the total number of edges is at most

$$C_{n-m}(n-m)^{3/2} + C_m m^{3/2} + m,$$

which, on the other hand, must be at least $C_n n^{3/2}$, since this is the total number of edges in G, by assumption. Using $C_m, C_{n-m} \leq C_n$, this implies

$$\begin{aligned} C_n(n-m)^{3/2} + C_n m^{3/2} + m &\geq C_n n^{3/2} && \text{or} \\ (n-m)^{3/2} + m^{3/2} + \frac{1}{C_n} m &\geq n^{3/2} && \text{or} \\ \frac{1}{C_n m^{1/2}} &\geq \left(\frac{n}{m}\right)^{3/2} - 1 - \left(\frac{n}{m} - 1\right)^{3/2}, \end{aligned}$$

which implies

$$\frac{1}{C_n} \geq \left(\frac{n}{m}\right)^{3/2} - 1 - \left(\frac{n}{m} - 1\right)^{3/2}. \tag{6}$$

Consider the function $f(x) = x^{3/2} - 1 - (x-1)^{3/2}$. Note that f is monotone increasing in x, for $x \in [1, \infty)$. Thus, the inequality (6) implies

$$\frac{1}{C_n} \geq \min\left\{ f\left(\frac{n}{m}\right) \mid 3 \leq m \leq n/3 \right\} = f(3),$$

which yields a contradiction if C_n is larger than some absolute constant.

To summarize, in at least one of the two cases analyzed above it must be possible to apply the induction hypothesis; otherwise, in each of the two cases, we get a contradiction. This completes the proof of the theorem, for any monotone increasing function C_n satisfying

$$C_n \geq 2B\left(C_{n^{1/2}} + \frac{C_{n/d}}{\sqrt{d}}\right).$$

Solving the recurrence relation, one can take $C_n = C_0(\log n)^\beta$, for $\beta = \log_2(4B)$ and some absolute value $C_0 > 0$. Indeed, since we may choose $d \geq 4$ arbitrarily (but independently of n), we may assume that $\frac{2B}{\sqrt{d}} \leq \frac{1}{2}$. Thus, any choice of C_n monotone increasing in n, will satisfy

$$\frac{1}{2} C_n \geq \frac{2B}{\sqrt{d}} C_{n/d}.$$

So we need to show that

$$\frac{1}{2} C_n \geq 2B C_{n^{1/2}}.$$

That is, we need to show

$$\frac{1}{2}C_0(\log n)^\beta \geq 2BC_0(\log n^{1/2})^\beta$$
$$= 2BC_0\frac{1}{2^\beta}(\log n)^\beta,$$

which is equivalent to requiring $2^\beta \geq 4B$ or $\beta \geq \log_2(4B)$, as claimed.

This completes the proof of the theorem. ◀

6 Proof of Theorem 2

Let H_d be the graph induced by a hypercube in $\mathbb{R}^d$. That is, each vertex corresponds to a d-tuple in $\{0,1\}^d$, and a pair of vertices are connected by an edge if and only if the corresponding d-tuples are different by exactly one entry. So H_d has 2^d vertices and $d2^{d-1}$ edges.

We now describe an embedding $\mathbf{p}$ of the vertices of H_d in $\mathbb{R}^2$. For this, we start with an embedding $\bar{\mathbf{p}}$ of H in $\mathbb{R}^d$. We take the standard embedding of the hypercube, namely, we map a vertex with corresponding d-tuple $(b_1,\ldots,b_d)$, to the point $(b_1,\ldots,b_d)$ in $\mathbb{R}^d$.

▷ Claim 12. No three vertices of H_d are embedded by $\bar{\mathbf{p}}$ to a common line in $\mathbb{R}^d$.

Proof. Consider two distinct d-tuples $(b_1,\ldots,b_d)$ and $(b'_1,\ldots,b'_d)$. Assume without loss of generality that $b_1 \neq b'_1$. Then, for every $t \in \mathbb{R}\setminus\{0,1\}$, we have $tb_1+(1-t)b'_1 \notin \{0,1\}$. Thus no other point on the line connecting $(b_1,\ldots,b_d)$ and $(b'_1,\ldots,b'_d)$ is a vertex of H_d. ◁

Identify a point in $\mathbb{R}^{2d}$ with a $2\times d$ matrix, regarded as a linear transformation from $\mathbb{R}^d$ to $\mathbb{R}^2$. We define $\mathbf{p} := T\circ\bar{\mathbf{p}}$, where $T:\mathbb{R}^d\to\mathbb{R}^2$ is a linear transformation. We choose $T\in\mathbb{R}^{2d}$ so that with this choice no three distinct vertices of H_d are embedded by $\mathbf{p}$ to a common line. To prove the existence of such T we need the following claim.

▷ Claim 13. Let $q_1,q_2,q_3\in\mathbb{R}^d$ be three distinct non-collinear points. Then there exists an algebraic subvariety $Z\subset\mathbb{R}^{2d}$, of codimension at least one, such that for every $T\in\mathbb{R}^{2d}\setminus Z$, the points Tq_1,Tq_2,Tq_3 are not collinear.

Proof. There exists a polynomial, P, over 6 variables and with rational coefficients, such that, for every $p_1,p_2,p_3\in\mathbb{R}^2$, $P(p_1,p_2,p_3)=0$ if and only if the points p_1,p_2,p_3 are collinear. Namely, P is just the determinant of the 2×2 matrix with columns p_2-p_1 and p_3-p_1. Consider the equation

$$P(Tq_1,Tq_2,Tq_3)=0. \tag{7}$$

Since q_1,q_2,q_3 are given, this is an equation in the entries of T, which defines a subvariety of $\mathbb{R}^{2d}$.

It is easy to see that (7) is not identically zero. Indeed, consider a linear transformation T which maps the plane spanned by the vectors q_2-q_1, q_3-q_1 (this is a plane through the origin) to $\mathbb{R}^2$ injectively. Such T does not satisfy (7). Thus (7) defines a subvariety Z of $\mathbb{R}^{2d}$ of codimension at least one. This proves the claim. ◁

For every triple u_1,u_2,u_3 of vertices of H_d, we apply Claim 13 to the points $q_i := \bar{\mathbf{p}}(u_i)$ for $i=1,2,3$. Let $\mathcal{Z}$ be the family of algebraic subvariety of $\mathbb{R}^{2d}$ of "bad" choices of T, given by applying Claim 13 to each triple of vertices. Since each element of $\mathcal{Z}$ is of codimension at least one, and $\mathcal{Z}$ is finite, the union of the elements of $\mathcal{Z}$ does not cover $\mathbb{R}^{2d}$. Therefore,

there exists a choice of T that does not lie on any of the elements of $\mathcal{Z}$. Using such T in the definition of $\mathbf{p}$, we get that no three distinct vertices of H_d are embedded by $\mathbf{p}$ to a common line.

Finally, we claim that the framework $(H_d, \mathbf{p})$ does not have a rigid subframework of size larger than two. In fact, we prove the following stronger property.

▷ Claim 14. Let x, y be any pair of distinct vertices of H_d, such that $\{x, y\}$ is not an edge of H_d. Consider a neighborhood, B, of $\mathbf{p}$ in $\mathbb{R}^2$ arbitrarily small. Then there exists an embedding $\mathbf{p}' \in B$, such that $\mathbf{p}$ and $\mathbf{p}'$ are equivalent, but $\|\mathbf{p}(x) - \mathbf{p}(y)\| \neq \|\mathbf{p}'(x) - \mathbf{p}'(y)\|$.

Proof. We prove the claim by induction on d. The base case $d = 2$ is easy to see. Consider $d > 2$. The vertices of H_d can be regarded as a disjoint union of two copies $H_{d-1}^{(1)}$, $H_{d-1}^{(2)}$of H_{d-1}. Note that each vertex $u \in H_{d-1}^{(1)}$ can be associated with a vertex $u' \in H_{d-1}^{(2)}$, such that $\{u, u'\}$ is an edge in H_d. Moreover, note that by the definition of our embedding $\mathbf{p}$, all the edges of this form (edges between a vertex of $H_{d-1}^{(1)}$ and a vertex of $H_{d-1}^{(2)}$) have the same length ℓ.

Let x, y be a pair of distinct vertices of H_d such that $\{x, y\}$ is not an edge in H_d. Assume first that the pair x, y is in one of the copies of H_{d-1}, say in $H_{d-1}^{(1)}$. Let $\mathbf{q} := \mathbf{p}_{|_{H_{d-1}^{(1)}}}$ be the embedding $\mathbf{p}$ of H, restricted the subgraph $H_{d-1}^{(1)}$. By the induction hypothesis, for every arbitrarily small neighborhood of $\mathbf{q}$, there exists an embedding $\mathbf{q}'$ in this neighborhood, such that $\mathbf{q}, \mathbf{q}'$ are equivalent, but $\|\mathbf{q}(x) - \mathbf{q}(y)\| \neq \|\mathbf{q}'(x) - \mathbf{q}'(y)\|$. By the symmetry of $H_{d-1}^{(1)}$ and $H_{d-1}^{(2)}$ it is easy to see that this can be extended to an embedding $\mathbf{p}'$ of H_d which is congruent to $\mathbf{p}$. This proves the claim in this case.

Assume next that, say, $x \in H_{d-1}^{(1)}$, $y \in H_{d-1}^{(2)}$, and recall that $\{x, y\}$ is not an edge in H_d. Consider a neighborhood of $\mathbf{p}$, arbitrarily small. For each vertex $u \in H_{d-1}^{(1)}$, take a rotation r_u of the plane centered at u, with angle of rotation ε. We apply this rotation only to the (unique) vertex $u' \in H_{d-1}^{(2)}$ with the property that $\{u, u'\}$ is an edge in H_d. This induces a new embedding $\mathbf{p}'$ of H_d. Clearly, taking $\varepsilon > 0$ sufficiently small, $\mathbf{p}'$ is in the given neighborhood of $\mathbf{p}$. Moreover, since $\mathbf{p}'$ applied to the vertices of $H_{d-1}^{(2)}$ is a translation of $\mathbf{p}'$ applied to $H_{d-1}^{(1)}$, it is clear that by construction that $\mathbf{p}$ and $\mathbf{p}'$ are equivalent. Finally, we claim that for ε sufficiently small, we have $\|\mathbf{p}(x) - \mathbf{p}(y)\| \neq \|\mathbf{p}'(x) - \mathbf{p}'(y)\|$. To see this it is sufficient to restrict our attention to the vertices $x, y' \in H_{d-1}^{(1)}$ and $x', y \in H_{d-1}^{(2)}$, where $\{x, x'\}$ and $\{y', y\}$ are edges in H_d. Note that since $\{x, y\}$ is not an edge, x, x', y, y' are distinct. Also, by construction, $\|\mathbf{p}(x) - \mathbf{p}(y')\| = \|\mathbf{p}'(x') - \mathbf{p}'(y)\|$ and $\|\mathbf{p}(x) - \mathbf{p}(x')\| = \|\mathbf{p}'(y') - \mathbf{p}'(y)\|$. It is now easy to see, again by the construction of $\mathbf{p}'$ that $\|\mathbf{p}(x) - \mathbf{p}(y)\| \neq \|\mathbf{p}'(x) - \mathbf{p}'(y)\|$, as claimed. ◁

References

1 L. Asimow and B. Roth. The rigidity of graphs. *Trans. Amer. Math. Soc.*, 245:279–289, 1978.

2 E. D. Bolker and B. Roth. When is a bipartite graph a rigid framework? *Pacific J. Math.*, 90:27–44, 1980.

3 Gy. Elekes and M. Sharir. Incidences in three dimensions and distinct distances in the plane. *Combinat. Probab. Comput.*, 20:571–608, 2011.

4 L. Guth and N. H. Katz. On the Erdős distinct distances problem in the plane. *Annals Math.*, 18:155–190, 2015.

5 J. Kollár. Szemerédi-trotter-type theorems in dimension 3. *Adv. Math.*, 271:30–61, 2015.

6 G. Laman. On graphs and rigidity of plane skeletal structures. *J. Engrg. Math.*, 4:333–338, 1970.

7 H. Pollaczek-Geiringer. über die gliederung ebener fachwerke, zamm. *Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik*, 7.1:58–72, 1927.

8 O. E. Raz. Configurations of lines in space and combinatorial rigidity. *Discrete Comput. Geom. (special issue)*, 58:986–1009, 2017.

9 O. E. Raz. Distinct distances for points lying on curves in $\mathbb{R}^d$ – the bipartite case. *manuscript*, 2020.

A Appendix for "Dense graphs have rigid parts" by János Kollár*

Let $\mathcal{L}$ be a set of m distinct lines in $\mathbb{C}^3$. A weighted number of their intersection points is

$$I(\mathcal{L}) := \sum_{p\in\mathbb{C}^3} \big(r(p)-1\big),$$

where $r(p)$ denotes the number of lines passing through a point p. Our aim is to outline the proof of the following variant of [5, Theorem 6]. The difference is that, unlike in [5, Theorem 6], we allow more than $2c\sqrt{m}$ lines on a regulus (that is, a smooth quadric surface), but we restrict the number of intersections between them.

▶ **Proposition 15.** *Let $\mathcal{L}$ be a set of m distinct lines in $\mathbb{C}^3$. Let c be a constant such that every plane contains at most $c\sqrt{m}$ of the lines and, for every regulus, the lines on it have at most c^2m intersection points with each other. Then*

$$I(\mathcal{L}) \le \big(29.1+\tfrac{c}{2}\big)\cdot m^{3/2}.$$

Proof. Following the method of [4], there is an algebraic surface S of degree $\le \sqrt{6m}-2$ that contains all the lines in $\mathcal{L}$. We decompose S into its irreducible components $S=\cup_j S_j$.

Now we follow the count as in [5, Paragraph 24]. The bound for external intersections (when a line not on S_j meets a line on S_j) is the same as in [5, Paragraph 18]. The remaining internal intersections (when a line on S_j meets a line on the same S_j) is done one surface at a time. The only change is with the count on a regulus, which is done in [5, Paragraph 19].

Thus let Q_j be a regulus that contains n_j lines. If $n_j \le 2c\sqrt{m}$ then we use the formula on the bottom of p. 38: $I(\mathcal{L}_j) \le \frac{c}{2} n_j\sqrt{m}$. If $n_j \ge 2c\sqrt{m}$ then we use that, by assumption

$$I(\mathcal{L}_j) \le c^2m = 2c\sqrt{m}\tfrac{c}{2}\sqrt{m} \le n_j\tfrac{c}{2}\sqrt{m}.$$

So $I(\mathcal{L}_j) \le \frac{c}{2} n_j\sqrt{m}$ always holds for every regulus and this is the only information about lines on a regulus that the proof in [5, Paragraph 24] uses. The rest of the proof is unchanged. ◀

*Department of Mathematics, Princeton University, kollar@math.princeton.edu

Incidences Between Points and Curves with Almost Two Degrees of Freedom

Micha Sharir
School of Computer Science, Tel Aviv University, Israel
michas@tau.ac.il

Oleg Zlydenko
School of Computer Science, Tel Aviv University, Israel
zlydenko@gmail.com

Abstract

We study incidences between points and (constant-degree algebraic) curves in three dimensions, taken from a family $\mathcal{C}$ of curves that have *almost two degrees of freedom*, meaning that (i) every pair of curves of $\mathcal{C}$ intersect in $O(1)$ points, (ii) for any pair of points p, q, there are only $O(1)$ curves of $\mathcal{C}$ that pass through both points, and (iii) a pair p, q of points admit a curve of $\mathcal{C}$ that passes through both of them if and only if $F(p,q) = 0$ for some polynomial F of constant degree associated with the problem. (As an example, the family of unit circles in $\mathbb{R}^3$ that pass through some fixed point is such a family.)

We begin by studying two specific instances of this scenario. The first instance deals with the case of unit circles in $\mathbb{R}^3$ that pass through some fixed point (so called *anchored* unit circles). In the second case we consider tangencies between *directed points* and circles in the plane, where a directed point is a pair (p,u), where p is a point in the plane and u is a direction, and (p,u) is tangent to a circle γ if $p \in \gamma$ and u is the direction of the tangent to γ at p. A lifting transformation due to Ellenberg et al. maps these tangencies to incidences between points and curves ("lifted circles") in three dimensions. In both instances we have a family of curves in $\mathbb{R}^3$ with almost two degrees of freedom.

We show that the number of incidences between m points and n anchored unit circles in $\mathbb{R}^3$, as well as the number of tangencies between m directed points and n arbitrary circles in the plane, is $O(m^{3/5}n^{3/5} + m + n)$ in both cases.

We then derive a similar incidence bound, with a few additional terms, for more general families of curves in $\mathbb{R}^3$ with almost two degrees of freedom, under a few additional natural assumptions.

The proofs follow standard techniques, based on polynomial partitioning, but they face a critical novel issue involving the analysis of surfaces that are infinitely ruled by the respective family of curves, as well as of surfaces in a dual three-dimensional space that are infinitely ruled by the respective family of suitably defined dual curves. We either show that no such surfaces exist, or develop and adapt techniques for handling incidences on such surfaces.

The general bound that we obtain is $O(m^{3/5}n^{3/5} + m + n)$ plus additional terms that depend on how many curves or dual curves can lie on an infinitely-ruled surface.

2012 ACM Subject Classification Mathematics of computing → Combinatorics

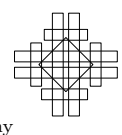
Keywords and phrases Incidences, Polynomial partition, Degrees of freedom, Infinitely-ruled surfaces, Three dimensions

Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.66

Related Version A full version of this paper is available at `https://arxiv.org/abs/2003.02190`.

Funding *Micha Sharir*: Partially supported by ISF Grant 260/18, by grant 1367/2016 from the German-Israeli Science Foundation (GIF), and by Blavatnik Research Fund in Computer Science at Tel Aviv University.

Acknowledgements The authors would like to thank Noam Solomon for many helpful discussions and for providing us with a copy of his ongoing work with Guth. Thanks are also due to the anonymous reviewers for their helpful comments.



1 Introduction

Our results: An overview. In this paper we study several incidence problems involving points and curves in three dimensions, where the curves are 3-parameterizable (each of them can be defined by three real parameters) and have *almost two degrees of freedom*, a notion that we discuss in detail below. We begin by deriving improved incidence bounds for two specific classes of such curves, one of which (studied in Section 2) is the class of *anchored unit circles* (unit circles that pass through some fixed point), and the other (studied in Section 3) is a class of "lifted circles" that arise in the context of tangencies between so-called *directed points* and circles in the plane. In both cases, the incidence bound, for m points and n curves, is $O(m^{3/5}n^{3/5} + m + n)$. We then study the problem for general curves that satisfy the above properties (and a few other natural assumptions), and derive the same bound as above, with additional terms that depend on various parameters associated with the problem. See Section 4 and the full version [12] for full details.

We begin with a review of the setup and of several basic features that arise in the analysis.

Incidence problems. Let P be a set of m points, and let C be a set of n algebraic curves of some bounded degree in $\mathbb{R}^3$. Let $I(P, C)$ denote the number of incidences between the points of P and the curves of C, i.e., $I(P, C) = |\{(p, c) \mid p \in P, c \in C, p \in c\}|$. The *incidence problem* for P and C is to bound $I(P, C)$. More precisely, we want to estimate $I(m, n) := \max_{|P|=m,|C|=n} I(P, C)$, where the maximum is over all sets P of m points and C of n curves from some specific family of curves in $\mathbb{R}^3$ (such as lines, circles, etc.).

The simplest formulation of the incidence problem involves incidences between points and lines in the plane, where we have

▶ **Theorem 1** (Szemerédi and Trotter [13]). *For sets P of m points and L of n lines in the plane, we have $I(P, L) = O(m^{2/3}n^{2/3} + m + n)$, and the bound is tight in the worst case.*

The same asymptotic upper bound can be proven for unit circles as well, except that the matching lower bound is not known to hold, and is strongly suspected to be only close to linear. For general circles, of arbitrary radii, we have

▶ **Theorem 2** (Agarwal et al. [1] and Marcus and Tardos [8]). *For sets P of m points and C of n (arbitrary) circles in the plane we have $I(P, C) = O(m^{2/3}n^{2/3} + m^{6/11}n^{9/11}\log^{2/11}(m^3/n) + m + n)$.*

Another variant of the incidence problem, which has recently been studied in Ellenberg et al. [3], and which is relevant to the study in this paper, is bounding the number of tangencies between lines and circles in the plane. In more detail, let a *directed point* in the plane be a pair (p, u), where $p \in \mathbb{R}^2$ and u is a direction (parameterized by its slope). A tangency occurs between a circle c and a directed point (p, u) when $p \in c$ and u is the direction of the tangent to c at p; see Figure 1. Unlike the standard case of point-circle incidences, there can be at most one circle that is tangent to a given pair of directed points (and in general there is no such circle). Ellenberg et al. [3] showed:

▶ **Theorem 3** (Ellenberg et al. [3]). *For a set P of m directed points and a set C of n (arbitrary) circles in the plane, there are $O(n^{3/2})$ tangencies between the circles in C and the directed points in P, assuming that each point of P is incident (i.e., tangent) to at least two circles.*

In fact, the bound in [3] also holds for more general sets of curves, and over fields other than $\mathbb{R}$. An immediate corollary of Theorem 3 is that the number of incidences between m directed points and n circles is $O(n^{3/2}+m)$. We will discuss this problem further in Section 3, where we obtain the improved bound $O(m^{3/5}n^{3/5}+m+n)$ mentioned above.

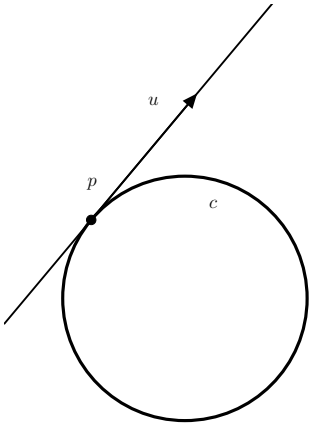


Figure 1 The tangent to circle c at point p has direction u. We then say that the directed point (p,u) is tangent (or incident) to c.

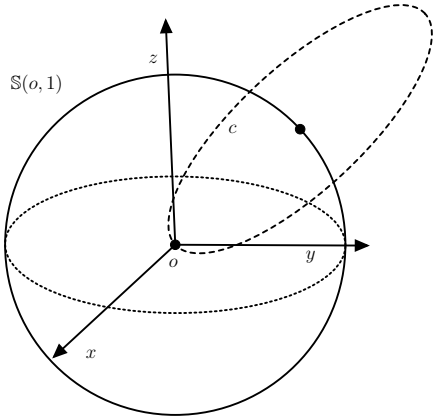


Figure 2 An anchored circle c. Its center is on the unit sphere $\mathbb{S}(o,1)$, and it passes through the origin o.

As has been observed, time and again, the result of Theorem 1, including both the upper and the lower bound, is applicable to point-line incidences $\mathbb{R}^3$ as well (and, in fact, in any higher-dimensional space $\mathbb{R}^d$), unless we impose some additional constraint on the number of coplanar input lines. The following celebrated theorem of Guth and Katz [5] gives such an improved bound[1].

▶ **Theorem 4** (Guth and Katz [5]). *Let P be a set of m points and L be a set of n lines in $\mathbb{R}^3$. Assume further that no plane in $\mathbb{R}^3$ contains more than q lines of L, for some parameter $q \le n$. Then $I(P,L) = O\left(m^{1/2}n^{3/4} + m^{2/3}n^{1/3}q^{1/3} + m + n\right)$. Moreover, the bound is tight in the worst case.*

A similar argument can be made for point-circle incidences in $\mathbb{R}^3$ (or again in any dimension ≥ 3) – here we need to constrain the number of input circles that can lie in any common plane or sphere. The best known upper bound, due to Sharir and Solomon [11], is (see also Sharir et al. [10] for an earlier, weaker bound).

▶ **Theorem 5** (Sharir and Solomon [11]). *Let P be a set of m points and let C be a set of n circles in $\mathbb{R}^3$, and let $q < n$ be an integer. If no sphere or plane contains more than q circles of C, then*

$$I(P,C) = O\left(m^{3/7}n^{6/7} + m^{2/3}n^{1/3}q^{1/3} + m^{6/11}n^{5/11}q^{4/11}\log^{2/11}(m^3/q) + m + n\right).$$

Polynomial partitioning. The polynomial partitioning technique is the most recently developed method for deriving incidence bounds (and many other results too), and it is due to Guth and Katz [5], with an extended version given later by Guth [4]. We use the following version (specialized to our needs), where $Z(f)$ denotes the zero set $\{z \in \mathbb{R}^3 \mid f(z) = 0\}$ of a real (trivariate) polynomial f.

[1] The theorem is not stated explicitly in [5], but it is an immediate consequence of the analysis in [5].

▶ **Theorem 6** (Polynomial partitioning [4, 5]). *Let P be a set of m points and C be a set of n algebraic curves of some constant degree in $\mathbb{R}^3$. Then, for any $1 < D$ such that $D^3 < m$ and $D^2 < n$, there is a polynomial f of degree at most D such that each of the $O(D^3)$ (open) connected components of $\mathbb{R}^3 \setminus Z(f)$ contains at most $O(m/D^3)$ points of P, and is crossed by at most $O(n/D^2)$ curves of C.*

Note that the theorem has no guarantee regarding the number of points of P on $Z(f)$, or the number of curves of C that are contained in $Z(f)$.

One of the main techniques for proving incidence bounds via polynomial partitioning proceeds as follows. We first establish a simple (and weak) incidence bound (usually referred to as a *bootstrapping* bound) by some other method. Then we apply Theorem 6, and use the bootstrapping bound in every connected component (cell) of $\mathbb{R}^3 \setminus Z(f)$. Incidences between curves in C and points on $Z(f)$ must be treated separately, using a different set of tools and techniques, typically taken from algebraic geometry.

Degrees of freedom. We say that a family $\mathcal{C}$ of constant-degree irreducible algebraic curves in $\mathbb{R}^3$ has s *degrees of freedom* (of multiplicity μ) if:

1. each pair of curves of $\mathcal{C}$ intersect in at most μ points; and
2. for each s-tuple $p_1, \ldots, p_s$ of distinct points in $\mathbb{R}^3$ there are at most μ curves of $\mathcal{C}$ that pass through all these points.

The definition extends, verbatim, to curves in any other dimension or in the plane.

The notion of degrees of freedom can be defined for arbitrary families of curves (not necessarily algebraic). However, for various technical reasons, mainly to be able to apply Theorem 6, we confine ourselves to the case of constant-degree algebraic curves.

Many natural families of curves have a small number of degrees of freedom:

- Lines have two degrees of freedom with multiplicity one (in any space $\mathbb{R}^d$). Indeed, each pair of lines intersect in at most one point, and through any pair of points only a single line can be drawn.
- Similarly, unit circles in the plane have two degrees of freedom as well, with multiplicity two. (Note that unit circles in $\mathbb{R}^3$, or in any higher-dimensional space, do not have two degrees of freedom, but they have three degrees of freedom, as follows from the next example.)
- Circles of arbitrary radii, in any space $\mathbb{R}^d$, have three degrees of freedom.

The following theorem is a generalization of Theorem 1, and is due to Pach and Sharir [9]. The original bound applies to more general families of curves, but we stick to the algebraic setup.

▶ **Theorem 7** (Pach and Sharir [9]). *Let P be a set of m points in the plane, and let C be a set of n irreducible algebraic curves in the plane of degree at most k and with s degrees of freedom (with multiplicity μ); here k, s and μ are assumed to be constants. Then:*

$$I(P,C) = O\left(m^{\frac{s}{2s-1}} n^{\frac{2s-2}{2s-1}} + m + n\right),$$

where the constant of proportionality depends on k, s and μ.

Note that this is the Szemerédi-Trotter bound for lines (for which $s = 2$), and also for unit circles in the plane.

▶ Remark. If we apply Theorem 7 to the family of circles of arbitrary radii, in any dimension (for which $s = 3$), we get the bound $I(P,C) = O(m^{3/5}n^{4/5} + m + n)$, which is weaker than the bound in Theorem 2.

Infinitely ruled surfaces. Extending the constraint that the parameter q imposes in Theorem 4, we use the following concept, studied by Sharir and Solomon in [11], adapting a similar reasoning from Guth and Zahl [6]. An algebraic surface V in $\mathbb{R}^3$ is *infinitely ruled* by a family $\mathcal{C}$ of curves, if each point $q \in V$ is incident to infinitely many curves of $\mathcal{C}$ that are fully contained in V. For example, the only surfaces that are infinitely ruled by lines are planes, and the only surfaces that are infinitely ruled by circles are planes and spheres; see Lubbes [7]. Sharir and Solomon have considered this notion in [11] to show:

▶ **Theorem 8** (Sharir and Solomon [11]). *Let P be a set of m points and C a set of n irreducible algebraic curves in $\mathbb{R}^3$, taken from a family $\mathcal{C}$, so that the curves of $\mathcal{C}$ are algebraic of constant degree, and with s degrees of freedom (of some multiplicity μ). If no surface that is infinitely ruled by curves of $\mathcal{C}$ contains more than q curves of C, for a parameter $q < n$, then $I(P,C) = O\left(m^{\frac{s}{3s-2}} n^{\frac{3s-3}{3s-2}} + m^{\frac{s}{2s-1}} n^{\frac{s-1}{2s-1}} q^{\frac{s-1}{2s-1}} + m + n\right)$, where the constant of proportionality depends on s, μ, and the degree of the curves in $\mathcal{C}$.*

Note that Theorem 4 is a special case of this result, with $s = 2$, where the infinitely ruled surfaces are planes.

An additional tool that we rely on is also due to Sharir and Solomon [11]. It is the following theorem, which is part of Theorem 1.13 in [11], and is a generalization of a result of Guth and Zahl [6] (that was stated there only for doubly ruled surfaces).

▶ **Theorem 9** (Sharir and Solomon [11]). *Let $\mathcal{C}$ be a family of algebraic curves in $\mathbb{R}^3$ of constant degree E. Let f be a complex irreducible polynomial of degree $D \gg E$. If $Z(f)$ is not infinitely ruled by curves from $\mathcal{C}$ then there exist absolute constants c, t, such that, except for at most cD^2 exceptional curves, every curve in $\mathcal{C}$ that is fully contained in $Z(f)$ is incident to at most cD t-rich points, namely points that are incident to at least t curves in $\mathcal{C}$ that are also fully contained in $Z(f)$.*

Almost two degrees of freedom. We introduce the following notion. A family $\mathcal{C}$ of algebraic irreducible curves in $\mathbb{R}^3$ has *almost s degrees of freedom* (of multiplicity μ) if:

1. each pair of curves of $\mathcal{C}$ intersect in at most μ points;
2. for each s-tuple $p_1, \ldots, p_s$ of distinct points in $\mathbb{R}^3$ there are at most μ curves of $\mathcal{C}$ that pass through all these points; and
3. there exists a curve of $\mathcal{C}$ that passes through $p_1, \ldots, p_s$, if and only if $F(p_1, \ldots, p_s) = 0$, where F is some $3s$-variate real polynomial of constant degree associated with $\mathcal{C}$.

With this definition we want to capture families $\mathcal{C}$ of curves that have some s degrees of freedom, but are such that for most s-tuples of points there is no curve of $\mathcal{C}$ that passes through all of them. As we demonstrate in this work, this additional restriction helps us improve the upper bound for incidences between points and curves from such a family.

As with the case of standard degrees of freedom, there are natural examples that fall under this definition. One such example is the family of unit circles in $\mathbb{R}^3$ (or in any $\mathbb{R}^d$, for $d \geq 3$), which, as is easily checked, has almost three degrees of freedom, with multiplicity two.

Our results. Although the above definition applies for general values of s and d, in this paper we focus on the special case $s = 2$ and $d = 3$.

In Section 2, we study the incidence problem between points and unit circles in three dimensions that pass through a fixed point (so-called *anchored* unit circles). With this additional constraint, this family has almost two degrees of freedom. We use this property to

prove the bootstrapping bound $I(m,n) = O(m^{3/2} + n)$, which improves the naive bootstrapping bound $I(m,n) = O(m^2 + n)$ for general families of curves with two degrees of freedom. We then prove that no surface is infinitely ruled by this family of curves. Combining this with some additional arguments, most notably an argument that establishes the absence of infinitely ruled surfaces in a suitably defined dual context (needed to establish our improved bootstrapping bound), gives us the following incidence bound:

$$I(m,n) = O(m^{3/5}n^{3/5} + m + n).$$

We remark that Sharir et al. [10] have obtained the bound

$$I(m,n) = O^*(m^{5/11}n^{9/11} + m^{2/3}n^{1/2}q^{1/6} + m + n) \tag{1}$$

for m points and n *non-anchored* unit circles in $\mathbb{R}^3$ (where $O^*(\cdot)$ hides small sub-polynomial factors). While this bound applies to general families of unit circles, it does not imply our bound for anchored circles (and it depends on the threshold parameter q, of which our bound is independent).

In Section 3, we bound the number of tangencies between circles and directed points in the plane. We transform this problem to an incidence problem between points and curves with almost two degrees of freedom in $\mathbb{R}^3$, resulting from lifting the given circles to three dimensions, using a method of Ellenberg et al.In this case as well, we prove the bootstrapping bound[2] $I(m,n) = O(m^{3/2} + n)$, show that no surface is infinitely ruled by this family of curves, and combine these statements (with some other considerations) to get the same asymptotic bound $I(m,n) = O(m^{3/5}n^{3/5} + m + n)$.

In Section 4, we extend the proofs from Sections 2 and 3, for more general families of curves with almost two degrees of freedom in three dimensions. A large part of the analysis can be generalized directly, but in general, there may exist surfaces that are infinitely ruled by these families of curves. Additionally, as already noted, our analysis in Sections 2 and 3 also involves a stage where it studies the problem in a dual setting, and the existence of infinitely ruled surfaces is an issue that has to be dealt with in this setting too. As in Theorem 4, the bound depends on the maximum number of curves that can lie on a surface that is infinitely ruled by the given family of curves, and on a similar threshold parameter in the dual space. We also need to impose a few additional natural conditions on the family of curves to obtain our result.

The bound that we obtain is $O(m^{3/5}n^{3/5} + m + n)$ plus additional terms that depend on the threshold parameters for infinitely ruled surfaces, both in the primal and in the dual setups. These terms are subsumed in the bound just stated when the relevant parameters are sufficiently small. See Section 4 and the full version [12] for the precise bound.

We exemplify (in [12]) the general bound for families of lines in $\mathbb{R}^3$ that have almost two degrees of freedom, a problem that has also been looked at by Guth and Solomon (work in progress).

We conclude the paper in Section 5 by listing some open problems and suggesting directions for further research.

[2] Note the difference between this bound and the bound in Ellenberg et al. noted earlier. It is this stronger version that allows us to derive our bound, mentioned below.

2 Anchored unit circles in space

The setup. As stated in Section 1, unit circles in space have almost three degrees of freedom. We reduce the setup to one with almost two degrees of freedom, by considering only circles that pass through a fixed point, say the origin. We call such circles *anchored (unit) circles*. An anchored circle c has radius 1 and center on the unit sphere $\mathbb{S}(o,1)$ centered at o (see Figure 2). The main result of this section is

▶ **Theorem 10.** *The number of incidences between m points and n anchored circles in $\mathbb{R}^3$ is*

$$I(P,C) = O(m^{3/5}n^{3/5} + m + n).$$

2.1 Proof of Theorem 10

We obtain the desired bound by following the general approach in [11]. Using special properties of the underlying setup, we obtain the following improved bootstrapping bound (over the simple "naive" bound $O(m^2+n)$ used in [11]).

▶ **Lemma 11.** *The number of incidences between a set P of m points and a set C of n anchored unit circles in $\mathbb{R}^3$ is $I(P,C) = O(m^{3/2}+n)$.*

The proof of the lemma is given in Section 2.2 below. Assuming for now that the lemma holds, we apply the technique of [11], with suitable modifications, to derive the incidence bound in Theorem 10. We show, by induction on n, that $I(P,C) \le A\left(m^{3/5}n^{3/5}+m+n\right)$, for a suitable constant A. It is trivial to verify that this bound holds for n smaller than some constant threshold n_0, by choosing A sufficiently large, so we focus on the induction step.

We first construct, using Theorem 6, a partitioning polynomial f in $\mathbb{R}^3$, of some specified (maximum) degree D, so that each cell (connected component) of $\mathbb{R}^3 \setminus Z(f)$ contains at most $O(m/D^3)$ points of P, and is crossed by at most $O(n/D^2)$ circles of C.

For each (open) cell τ of the partition, let P_τ denote the set of points of P inside τ, and let C_τ denote the set of circles of C that cross τ; we have $m_\tau := |P_\tau| = O(m/D^3)$, and $n_\tau := |C_\tau| = O(n/D^2)$. We apply the bootstrapping bound of Lemma 11 within each cell τ, to obtain

$$I(P_\tau, C_\tau) = O\left(m_\tau^{3/2} + n_\tau\right) = O\left((m/D^3)^{3/2} + (n/D^2)\right) = O\left(m^{3/2}/D^{9/2} + n/D^2\right).$$

Multiplying by the number of cells, we get that the number of incidences within the cells is

$$\sum_\tau I(P_\tau, C_\tau) = O\left(D^3 \cdot \left(m^{3/2}/D^{9/2} + n/D^2\right)\right) = O\left(m^{3/2}/D^{3/2} + nD\right).$$

We choose $D = am^{3/5}/n^{2/5}$, for a sufficiently small constant a. For this to make sense, we require that $1 \le D \le a' \min\{m^{1/3}, n^{1/2}\}$, for another sufficiently small constant $a' > 0$, which holds when $b_1 n^{2/3} \le m \le b_2 n^{3/2}$, for suitable constants b_1, b_2 that depend on a and a'. For m in this range, the incidence bound is $O(m^{3/5}n^{3/5})$. As we detail in the full version [12], (i) when $m < b_1 n^{2/3}$, we apply Lemma 11 to the entire sets P and C, and get the bound $O(n)$, and (ii) when $m > b_2 n^{3/2}$, we choose $D = a' n^{1/2}$, for the a' used above, and get the bound $O(m)$. Combining all three cases, we obtain the overall within-cells bound

$$O\left(m^{3/5}n^{3/5} + m + n\right). \tag{2}$$

Consider next incidences involving points that lie on $Z(f)$. A circle γ that is not fully contained in $Z(f)$ crosses it in at most $O(D)$ points, which follows from Bézout's theorem (see, e.g., [2]). This yields a total of $O(nD) = O(m^{3/5}n^{3/5} + m)$ incidences, within the asymptotic bound in (2). It therefore remains to bound the number of incidences between the points of P on $Z(f)$ and the anchored circles that are fully contained in $Z(f)$.

We follow the proof of Theorem 1.4 in [11], which considers each irreducible component of $Z(f)$ separately, and distinguishes between components that are *infinitely ruled* by anchored circles, and components that are not. Let $\mathcal{C}$ denote the infinite family of all possible anchored (unit) circles. Fortunately for us, we have:

▶ **Lemma 12.** *No algebraic surface is infinitely ruled by anchored unit circles.*

Proof. Briefly, the only surfaces to consider are planes and spheres, and neither can be infinitely ruled by anchored unit circles. See the full version [12] for details. ◀

Write $m^* = |P \cap Z(f)|$ and $m_0 = |P \setminus Z(f)|$, so $m = m_0 + m^*$. The analysis in [11], which we follow here, handles each irreducible component of $Z(f)$ separately. Enumerate these components as $Z(f_1), \ldots, Z(f_k)$, for suitable irreducible polynomials $f_1, \ldots, f_k$, of respective degrees $D_1, \ldots, D_k$, where $\sum_{i=1}^{k} D_i \leq D$. By Lemma 12, none of these components is infinitely ruled by anchored circles.

Let P_i (resp., C_i) denote the set of all points of P (resp., anchored circles of C) that are contained (resp., fully contained) in $Z(f_i)$, assigning each point and circle to the first such component (in the above order), when it is contained in more than one component. The "cross-incidences", between points and circles assigned to different components, occur at crossing points between circles and components that do not contain them, and their number is therefore $O(nD)$, which satisfies our asymptotic bound. It therefore suffices to bound the number of incidences between points and circles assigned to the same component.

By Theorem 9, there exist absolute constants c, t, such that there are at most cD_i^2 'exceptional' anchored circles in C_i, namely, anchored circles that contain more than cD_i *t-rich* points of $P \cap Z(f_i)$, namely points that are incident to at least t circles from C_i. Denote the number of t-rich points (resp., *t-poor* points, namely points that are not t-rich) as m_{rich} (resp., m_{poor}), so $m_{rich} + m_{poor} = m^*$. By choosing a and a' (in the definition of D) sufficiently small, we can ensure, as is easily checked, that $\sum_i D_i^2 \leq (\sum_i D_i)^2 \leq D^2 \leq n/(2c)$.

The number of incidences on the non-exceptional circles, summed over all components $Z(f_i)$, is $O(m_{poor} + nD)$. Indeed, each non-exceptional circle contains at most cD_i t-rich points, for a total of $O(nD_i)$ incidences, and the sum of these bounds is $O(nD)$. Any t-poor point lies on at most t circles of C_i, for a total of $tm_{poor} = O(m_{poor})$ incidences (over all sets C_i).

For the exceptional circles, we apply the induction hypothesis, as their overall number is at most $c\sum_i D_i^2 \leq cD^2 \leq n/2$. Note that in this inductive step we only need to consider the t-rich points, as the t-poor points have already been taken care of. By the induction hypothesis, the corresponding incidence bound between the points and circles that were assigned to (the same) f_i is at most

$$A\left(m_i^{3/5}(cD_i^2)^{3/5} + m_i + cD_i^2\right),$$

where m_i is the number of t-rich points assigned to f_i. We now sum over i. Clearly, $\sum_i m_i = m_{rich}$. We also have $\sum_i cD_i^2 \leq n/2$. As for the first term, we use Hölder's inequality:

$$\sum_i m_i^{3/5}(cD_i^2)^{3/5} = c^{3/5}\sum_i m_i^{3/5}D_i^{6/5} \le c^{3/5}\left(\sum_i m_i\right)^{3/5}\left(\sum_i D_i^3\right)^{2/5} \le c^{3/5}m^{3/5}\left(\sum_i D_i^3\right)^{2/5}.$$

Finally, using the fact that $\sum_i D_i^3 \le D^3$, we get the overall bound:

$$A\left(c^{3/5}m^{3/5}D^{6/5} + m_{rich} + n/2\right) \le A\left(\frac{m^{3/5}n^{3/5}}{2^{3/5}} + m_{rich} + n/2\right),$$

since $c^{3/5}D^{6/5} \le (n/2)^{3/5}$, by construction.

We now add to this quantity the bound for incidences within the cells, as well as the various other bounds involving points on $Z(f)$. Together, we can upper bound these bounds by $B\left(m^{3/5}n^{3/5} + n + m_0 + m_{poor}\right)$, for a suitable absolute constant B. By choosing A sufficiently large, the sum of all the bounds encountered in the analysis is at most $A\left(m^{3/5}n^{3/5} + m + n\right)$. This establishes the induction step, and thereby completes the proof of Theorem 10, modulo the still missing proof of Lemma 11, presented next.

2.2 Proof of Lemma 11

The lemma improves upon the naive (and standard) bootstrapping bound, used in [11], which is $O(m^2 + n)$, for m points and n anchored circles. We dualize the setup, exploiting the underlying geometry, mapping each circle $\gamma \in \mathcal{C}$ to a suitable algebraic representation of the point $q_\gamma = (\alpha_\gamma, \beta_\gamma, \phi_\gamma)$ in 3-space, where $(\alpha_\gamma, \beta_\gamma)$ represents the center of γ as a point on $\mathbb{S}(o, 1)$, and ϕ_γ represents the angle by which the circle is rotated around the line connecting o to its center. We denote by $\mathcal{C}^*$ the infinite family of all these dual points q_γ (over all possible anchored unit circles γ).

We also map each point $p \in P$ to the locus h_p of all dual points q_γ that represent anchored circles γ that are incident to p, and argue (in [12]) that h_p is a one-dimensional curve.

Let $\mathcal{H}$ denote the family of all dual curves h_p for points $p \in \mathbb{R}^3$ (actually, only points in the ball of radius 2 around o, with o excluded, are relevant). We show (see [12]) that $\mathcal{H}$ has (almost) two degrees of freedom. Let $C^* \subset \mathcal{C}^*$ be the set of points q_γ dual to the anchored circles $\gamma \in C$, and let $H \subset \mathcal{H}$ be the set of curves h_p dual to the points $p \in P$. We have thus reduced our problem to that of bounding the number of incidences between C^* and H, to which we can apply Theorem 8, using the fact that the curves of $\mathcal{H}$ have two degrees of freedom, to get the bound

$$I(P, C) = I(C^*, H) = O\left(n^{1/2}m^{3/4} + n^{2/3}m^{1/3}q^{1/3} + n + m\right), \tag{3}$$

where q is the maximum number of curves from H that lie on a common surface that is infinitely ruled by $\mathcal{H}$. Fortunately again for us, we have:

▶ **Lemma 13.** *No algebraic surface is infinitely ruled by $\mathcal{H}$.*

Proof. See the full version [12]. ◀

It thus follows that $I(P, C) = I(C^*, H) = O\left(n^{1/2}m^{3/4} + n + m\right)$, which is upper bounded by $O(n + m^{3/2})$. This completes the proof of Lemma 11, and, consequently, also of Theorem 10.

3 Point-circle tangencies in the plane

The setup. Let C be a set of n circles in the plane. A *directed point* in the plane is a pair (p, u), where $p \in \mathbb{R}^2$ and u is a direction, which we parameterize by its slope. A circle c is said to be *incident* (or *tangent*) to a directed point (p, u) if c passes through p, and c is tangent to the line emanating from p in direction u. See Figure 1.

As stated in Theorem 3, Ellenberg et al. [3] (using a somewhat different notation) have shown that the number of directed points that are incident to at least two circles of C is $O(n^{3/2})$. Using the main technical idea in [3], we represent directions by their slopes[3], and regard each directed point (p, u) as a point in $\mathbb{R}^3$, where the z-coordinate is the slope; from now on, we let the parameter u denote the slope. We map each circle c in $\mathbb{R}^2$ to the curve $c^* = \{(p, u) \mid c \text{ is incident to } (p, u)\}$, to which we refer as a *lifted circle*, or the *lifted image* of c. As is easily checked, c^* is an algebraic curve of degree 4.

Denote by $\mathcal{C}$ the infinite family of all possible lifted circles. It is easy to show that the curves of $\mathcal{C}$ have almost two degrees of freedom; see the full version [12] for details.

The setup then becomes similar to what we have seen in Section 2, and we have

▶ **Theorem 14.** *The number of incidences between m directed points and n circles in the plane is $O(m^{3/5}n^{3/5} + m + n)$.*

3.1 Proof of Theorem 14

We only give a brief sketch of the proof. Details can be found in [12]. The high-level approach in the proof of the theorem is very similar to the one presented in the previous section. We establish the improved bootstrapping bound.

▶ **Lemma 15.** *The number of incidences between m directed points and n circles in the plane is $O(m^{3/2} + n)$.*

Assuming that the lemma holds, we prove, by induction on n that, for $|P| = m$ and $|C| = n$, $I(P, C) \leq A(m^{3/5}n^{3/5} + m + n)$, for a suitable absolute constant A. Again, the case of small n is trivial, with a suitable choice of A, and we only focus on the induction step.

As before, we first construct a partitioning polynomial f in $\mathbb{R}^3$, of the same maximum degree as in the previous section, and bound the number of incidences within the partition cells by $O(m^{3/5}n^{3/5} + m + n)$. Consider then incidences involving points that lie on $Z(f)$. A lifted circle c^* that is not fully contained in $Z(f)$ crosses it in at most $O(D)$ points, for an overall $O(nD)$ bound, which, as before, is asymptotically subsumed by the bound within the cells. It therefore remains to bound the number of incidences between the points of P on $Z(f)$ and the lifted circles that are fully contained in $Z(f)$.

Again, we handle each irreducible component of $Z(f)$ separately, and make use of the fortunate property in the following lemma. Its proof is given in the full version [12], and it strongly exploits the geometry of this setup.

▶ **Lemma 16.** *No algebraic surface is infinitely ruled by lifted circles.*

We can now continue exactly as in Section 2 and establish the asserted bound. See [12] for full details.

[3] This excludes y-vertical directions from the analysis. We assume, without loss of generality, that no input directed point has vertical direction (i.e., slope $\pm\infty$).

3.2 Proof of Lemma 15

We dualize the setup, exploiting the underlying geometry in the plane, by mapping each circle $c \in C$, with center (ξ, η) and radius r, to the point $q_c = (\xi, \eta, \zeta)$, where $\zeta = r^2 - \xi^2 - \eta^2$, and by mapping each directed point (p, u) to the locus $h_{p,u}$ of all dual points that represent circles that are incident to (p, u). It is easily seen that $h_{p,u}$ is the intersection of two planes, and is therefore a line in $\mathbb{R}^3$. We have thus reduced the problem to that of incidences between n points (those dual to the circles of C) and m lines (the lines $h_{p,u}$, for $(p, u) \in P$) in three dimensions. We can apply the result of Guth and Katz [5] (see Theorem 4) for estimating the number of these incidences, and obtain

$$I(P, C) = I(C^*, H) = O\left(n^{1/2}m^{3/4} + n^{2/3}m^{1/3}q^{1/3} + n + m\right), \tag{4}$$

where H is the set of the dual lines $h_{p,u}$, and q is the maximum number of lines of H that can lie in a common plane. This is a notable difference with the analysis in Section 2: There we showed that no surface is infinitely ruled by the dual curves, whereas here every plane is such a surface. Handling incidences on planes requires extra work, presented in the next subsection.

3.3 Coplanar lines

The gist of the analysis in this subsection is to control the value of q. For this, we distinguish between planes that contain at most q lines of H, for a suitable threshold value q that we will set later, and those that contain more than q lines. We handle the latter type of planes using a different technique that strongly exploits the geometry of the problem, and are then left with a subproblem in which (4) can be used.

Recall that if a circle c has center q and radius r then the *power* of a point w with respect to c is $|wq|^2 - r^2$. As is well known, and easy to see, the duality transform that we have used has the property that for each non-vertical plane π in $\mathbb{R}^3$ there exist a point w in $\mathbb{R}^2$ and a power ρ, such that the point dual to a circle c lies on π if and only if w has power ρ with respect to c.

Let π be any fixed non-vertical plane in $\mathbb{R}^3$, and let w and ρ be the corresponding point and power (in $\mathbb{R}^2$). We show in the full version [12] that a line $h_{p,u}$ is fully contained in π if and only if (a) p lies on the fixed circle $\gamma(w, \sqrt{\rho})$, with center w and radius $\sqrt{\rho}$, and (b) u is the direction of the line connecting p and w. See Figure 3.

Let $P^+ = \{(p, u)^+ = (p, -1/u) \mid (p, u) \in P\}$, and let $\mathcal{W}$ denote the set of all possible "power circles" $\gamma(w, \sqrt{\rho})$ as just defined. By construction, $h_{p,u}$ lies in a plane $\pi(w, \sqrt{\rho})$ if and only if $(p, u)^+$ is incident to the corresponding circle $\gamma(w, \sqrt{\rho})$.

We fix a threshold value q, to be determined shortly, and partition $\mathcal{W}$ into two subsets $\mathcal{W}^+$, $\mathcal{W}^-$, where $\mathcal{W}^+$ (resp., $\mathcal{W}^-$) consists of those circles in $\mathcal{W}$ that are incident to more than (resp., at most) q directed points of P^+. We refer to circles in $\mathcal{W}^+$ (resp., in $\mathcal{W}^-$) as being *q-rich* (resp., *q-poor*). The same notation carries over to the corresponding power planes in 3-space.

The analysis, whose full details are given in the full version [12], then shows that the number of incidences involving circles that are incident to at least one *q-rich point*, namely a point that is incident to at least one q-rich circle, is $O(m|\mathcal{W}^+| + n)$, and that $|\mathcal{W}^+| = O\left(\frac{m^2}{q^3} + \frac{m}{q}\right)$. Thus the number of these incidences is

$$O(m|\mathcal{W}^+| + n) = O\left(\frac{m^3}{q^3} + \frac{m^2}{q} + n\right).$$

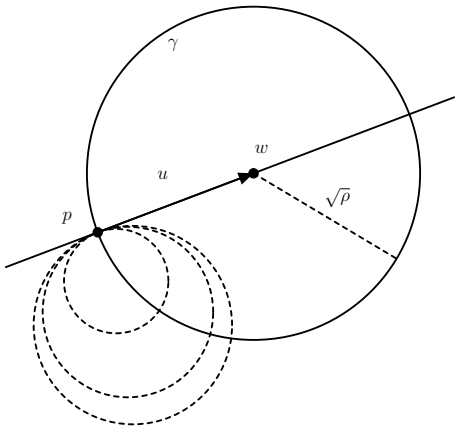


Figure 3 A circle $\gamma = \gamma(w, \sqrt{\rho})$ and a point p on γ with direction u to the center w of γ. Any circle incident to (p, u) (such as the dashed circles) has power ρ with respect to w.

The same (in fact, a smaller) bound applies for vertical planes π.

The remaining incidences only involve the surviving circles and the q-poor points. By construction, we now have that, in the dual 3-space, no plane contains more than q lines $h_{p,u}$, so by Guth and Katz's bound [5], the number of surviving incidences is $O\left(n^{1/2}m^{3/4} + n^{2/3}m^{1/3}q^{1/3} + n + m\right)$, for a total number of incidences $O\left(n^{1/2}m^{3/4} + n^{2/3}m^{1/3}q^{1/3} + n + \frac{m^3}{q^3} + \frac{m^2}{q}\right)$. By choosing the appropriate q (details in the full version), this bound becomes $O\left(n^{1/2}m^{3/4} + n\right) = O\left(m^{3/2} + n\right)$, thus completing the proof of the lemma.

4 Generalizations

Due to lack of space, we only state the main result, and leave all details to the full version [12].

▶ **Theorem 17.** *Let C be a set of n curves in $\mathbb{R}^3$ that are taken from a 3-parameterizable family $\mathcal{C}$ with almost two degrees of freedom, and let P be a set of m points in $\mathbb{R}^3$ whose duals (as defined in the previous sections) are all curves.*[4] *Assume that no surface that is infinitely ruled by the curves of $\mathcal{C}$ contains more than π curves from C. Let $\mathcal{P}^*$ be the family of curves in dual 3-space that are dual to the (suitable subset of) points of $\mathbb{R}^3$, with respect to the curves of $\mathcal{C}$, and assume that no surface that is infinitely ruled by curves of $\mathcal{P}^*$ contains more than δ curves dual to the points of P, and that not all pairs of curves of $\mathcal{C}$ intersect. Then*

$$I(P, C) = O\left(m^{3/5}n^{3/5} + (m^{11/15}n^{2/5} + n^{8/9})\delta^{1/3} + m^{2/3}n^{1/3}\pi^{1/3} + m + n\right).$$

If $\pi = O(n^{1/2})$ and $\delta = O(n^{1/3})$ then $I(P, C) = O(m^{3/5}n^{3/5} + m + n)$.

[4] In general, there might exist a lower-dimensional set of points whose duals are two-dimensional.

5 Conclusion

The elegant bound $O(m^{3/5}n^{3/5}+m+n)$ on the number of incidences, derived in Theorems 10 and 14 (and in the favorable subcase of Theorem 17), improves upon the best bounds for a family of curves with standard two degrees of freedom. Comparing this bound with the more cumbersome-looking general bound in Theorem 17, indicates that a major step in extending the technique of this paper to other instances of the problem, is analyzing the structure, or establishing the nonexistence, of surfaces that are infinitely ruled by the given curves or by the dual curves. This seems to be a rich area of further research, which calls for sophisticated tools from algebraic geometry.

Specific subproblems that are still not resolved, in their full generality, are: (a) Understand and characterize the existence of dual curves. (b) As just mentioned, understand and characterize the existence of surfaces that are infinitely ruled by the family of curves, as well as of dual surfaces that are infinitely ruled by the family of dual curves. (c) Obtain improved bounds, if at all possible, for the number of incidences between points and curves that lie on such a surface, both in the primal and in the dual setups.

In particular, it would be interesting to investigate whether ideas similar to those used in distinguishing between rich and poor points, given in Section 3.3, can be developed to reduce the threshold on the number of primal or dual curves that lie on a surface that is infinitely ruled by such curves.

A natural open problem, which we have yet to make progress on, is to generalize the bounds and techniques from this paper to families of curves in three dimension with almost s degrees of freedom, for larger constants $s \geq 3$. For instance, the problem of bounding the number of incidences between (non-anchored) unit circles and points in three dimensions falls under this general setup for $s = 3$, since unit circles (in any dimension) have almost three degrees of freedom. A specific goal here is to improve the bound (1) of [10] for non-anchored unit circles.

A simple, albeit unsatisfactory, way of handling the case $s \geq 3$ is to use anchoring. For example, for the case of unit circles in $\mathbb{R}^3$, we fix a point p_0 of P, consider the subfamily $\mathcal{C}_{p_0}$ of the unit circles that are incident to p_0, apply the bound obtained in Theorem 10 to P and the set C_{p_0} of the circles of C that are incident to p_0, and then combine these bounds, over all $p_0 \in P$, to obtain the desired bound. We believe that this coarse (and weak) approach can be considerably improved by replacing it by a direct approach that treats all the circles of C together, and leave this as yet another interesting open problem for further research.

A final open question is whether the bound $O(m^{3/5}n^{3/5}+m+n)$ is tight, for any instance of the setup considered in this paper. We strongly suspect that the bound is not tight.

References

1 Pankaj K. Agarwal, Eran Nevo, János Pach, Rom Pinchasi, Micha Sharir, and Shakhar Smorodinsky. Lenses in arrangements of pseudo-circles and their applications. *J. ACM*, 51(2):139–186, 2004.

2 David A. Cox, John Little, and Donal O'Shea. *Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra*. Springer-Verlag, Berlin, Heidelberg, 2007.

3 Jordan S. Ellenberg, Jozsef Solymosi, and Joshua Zahl. New bounds on curve tangencies and orthogonalities. *Discrete Analysis*, 22:1–22, 2016.

4 Larry Guth. Polynomial partitioning for a set of varieties. *Mathematical Proceedings of the Cambridge Philosophical Society*, 159(3):459—469, 2015. doi:10.1017/S0305004115000468.

5 Larry Guth and Nets Hawk Katz. On the Erdős distinct distances problem in the plane. *Annals Math.*, 181:155—190, 2015.

6 Larry Guth and Joshua Zahl. Algebraic curves, rich points, and doubly-ruled surfaces. *American Journal of Mathematics*, 140:1187—1229, March 2015.

7 Niels Lubbes. Families of circles on surfaces. *arXiv preprint*, 2013. arXiv:1302.6710.

8 Adam Marcus and Gábor Tardos. Intersection reverse sequences and geometric applications. *Journal of Combinatorial Theory, Series A*, 113(4):675–691, 2006. doi:10.1016/j.jcta.2005.07.002.

9 János Pach and Micha Sharir. On the number of incidences between points and curves. *Combinatorics, Probability and Computing*, 7:121–127, 1998.

10 Micha Sharir, Adam Sheffer, and Joshua Zahl. Improved bounds for incidences between points and circles. *Combinatorics, Probability and Computing*, 24(3):490—520, 2015.

11 Micha Sharir and Noam Solomon. Incidences with curves and surfaces in three dimensions, with applications to distinct and repeated distances. In *Proceedings 28th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '17, pages 2456–2475, 2017.

12 Micha Sharir and Oleg Zlydenko. Incidences between points and curves with almost two degrees of freedom. *arXiv preprint*, 2020. arXiv:2003.02190.

13 E. Szemerédi and W. T. Trotter. Extremal problems in discrete geometry. *Combinatorica*, 3(3):381–392, September 1983.

Connectivity of Triangulation Flip Graphs in the Plane (Part II: Bistellar Flips)

Uli Wagner
IST Austria, Am Campus 1, 3400 Klosterneuburg, Austria
uli@ist.ac.at

Emo Welzl
Department of Computer Science, ETH Zürich, Switzerland
emo@inf.ethz.ch

Abstract

Given a finite point set P in general position in the plane, a *full triangulation* is a maximal straight-line embedded plane graph on P. A *partial triangulation* on P is a full triangulation of some subset P' of P containing all extreme points in P. A *bistellar flip* on a partial triangulation either flips an edge, removes a non-extreme point of degree 3, or adds a point in $P \setminus P'$ as vertex of degree 3. The *bistellar flip graph* has all partial triangulations as vertices, and a pair of partial triangulations is adjacent if they can be obtained from one another by a bistellar flip. The goal of this paper is to investigate the structure of this graph, with emphasis on its connectivity.

For sets P of n points in general position, we show that the bistellar flip graph is $(n-3)$-connected, thereby answering, for sets in general position, an open questions raised in a book (by De Loera, Rambau, and Santos) and a survey (by Lee and Santos) on triangulations. This matches the situation for the subfamily of regular triangulations (i.e., partial triangulations obtained by lifting the points and projecting the lower convex hull), where $(n-3)$-connectivity has been known since the late 1980s through the secondary polytope (Gelfand, Kapranov, Zelevinsky) and Balinski's Theorem.

Our methods also yield the following results (see the full version [13]): (i) The bistellar flip graph can be covered by graphs of polytopes of dimension $n-3$ (products of secondary polytopes). (ii) A partial triangulation is regular, if it has distance $n-3$ in the Hasse diagram of the partial order of partial subdivisions from the trivial subdivision. (iii) All partial triangulations are regular iff the trivial subdivision has height $n-3$ in the partial order of partial subdivisions. (iv) There are arbitrarily large sets P with non-regular partial triangulations, while every proper subset has only regular triangulations, i.e., there are no small certificates for the existence of non-regular partial triangulations (answering a question by F. Santos in the unexpected direction).

2012 ACM Subject Classification Theory of computation → Computational geometry

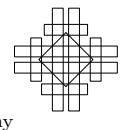
Keywords and phrases triangulation, flip graph, graph connectivity, associahedron, subdivision, convex decomposition, flippable edge, flip complex, regular triangulation, bistellar flip graph, secondary polytope, polyhedral subdivision

Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.67

Related Version A full version is available at [13], `https://arxiv.org/abs/2003.13557`.

Funding Research was supported by the Swiss National Science Foundation within the collaborative DACH project Arrangements and Drawings as SNSF Project 200021E-171681, and by IST Austria and Berlin Free University during a sabbatical stay of the second author.

Acknowledgements This research started at the 11$^{\text{th}}$ Gremo's Workshop on Open Problems (GWOP), Alp Sellamatt, Switzerland, June 24-28, 2013, motivated by a question posed by Filip Morić. We thank Michael Joswig, Jesús De Loera, and Francisco Santos for helpful discussions on the topics of this paper, and Daniel Bertschinger for carefully reading an earlier version and for many helpful comments.



1 Introduction

Throughout this paper we let P denote a finite planar point set in *general position* (no 3 points on a line) with $n \geq 3$ points. The set of *extreme points* of P (i.e., the vertices of the convex hull of P) is denoted by $\mathsf{xtr}P$, and $P^\circ := P \setminus \mathsf{xtr}P$ denotes the set of *inner* (i.e., non-extreme) *points* in P. We consistently use $h = h(P) := |\mathsf{xtr}P|$ and $n^\circ = n^\circ(P) := |P^\circ| = n - h$. We let $\mathsf{E}_{\mathsf{hull}} = \mathsf{E}_{\mathsf{hull}}(P) \subseteq \binom{P}{2}$ denote the h edges of the convex hull of P.

For graphs $G = (P', E)$, $P' \subseteq P$, $E \subseteq \binom{P'}{2}$, on P' we often identify edges $\{p, q\}$ with their corresponding straight line segments pq. We let $\mathsf{V}G := P'$ and $\mathsf{E}G := E$.

▶ **Definition 1** (plane)**.** A graph G on P is *plane* if no two straight line segments corresponding to edges in $\mathsf{E}G$ cross (i.e., they are disjoint except for possibly sharing an endpoint).

▶ **Definition 2** (full, partial triangulation)**.** A *full triangulation* of P is a maximal plane graph $T = (P, E)$. A *partial triangulation* of P is a full triangulation $T = (P', E)$ with $\mathsf{xtr}P \subseteq P' \subseteq P$ (hence $\mathsf{E}_{\mathsf{hull}} \subseteq \mathsf{E}T$). Points in $\mathsf{V}^\circ T := P^\circ \cap \mathsf{V}T$ are called *inner points*. Points in $P^\circ \setminus \mathsf{V}^\circ T$ are called *skipped* in T. Edges in $\mathsf{E}^\circ T := \mathsf{E}T \setminus \mathsf{E}_{\mathsf{hull}}$ are called *inner edges*. Edges in $\mathsf{E}_{\mathsf{hull}}$ are called *boundary edges*. $\mathcal{T}_{\mathsf{part}}(P)$ denotes the set of all partial triangulations of P.

▶ **Convention.** From now on, we will mostly use "triangulation" for "partial triangulation".

Figure 1 Edge flips and point flips (point removal, left to right; point insertion, right to left).

▶ **Definition 3** (bistellar flip)**.** Let T be a triangulation of P. An edge $e \in \mathsf{E}^\circ T$ is called *flippable in* T if removal of e in T creates a convex quadrilateral face Q, when $T[e]$ is the triangulation with the other diagonal $\overline{e}$ of Q added instead of e, i.e., $\mathsf{V}T[e] := \mathsf{V}T$ and $\mathsf{E}T[e] := \mathsf{E}T \setminus \{e\} \cup \{\overline{e}\}$; we call this an *edge flip*.

A point $p \in P^\circ$ is called *flippable in* T if $p \in P^\circ \setminus \mathsf{V}^\circ T$ or if $p \in \mathsf{V}^\circ T$, of degree 3 in T. (a) If $p \in P^\circ \setminus \mathsf{V}^\circ T$ then $T[p]$ is the triangulation with p added as a point of degree 3 (there is a unique way to do so); we call this a *point insertion flip*. (b) If $p \in \mathsf{V}^\circ T$ of degree 3 in T then $T[p]$ is obtained by removing p and its incident edges; we call this a *point removal flip*.

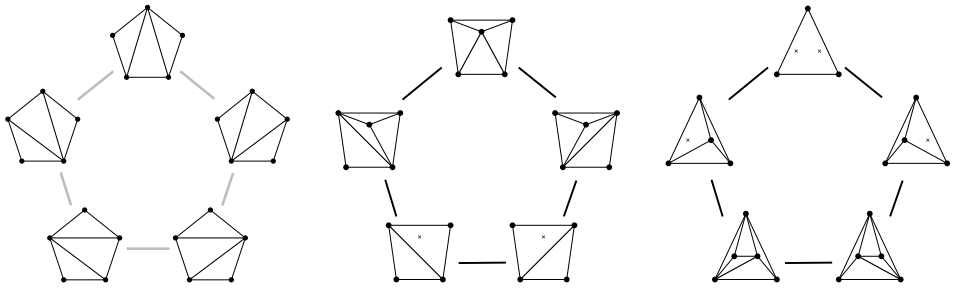


Figure 2 Bistellar flip graphs for 5 points. Small crosses indicate skipped points in P.

Whenever we write $T[x]$ for a triangulation T, then x is either a flippable point in P° or a flippable edge in $\mathsf{E}^\circ T$, and we write $T[x, y]$ short for $(T[x])[y]$, etc. The *bistellar flip graph* of P is the graph with vertex set $\mathcal{T}_{\mathsf{part}}(P)$ and edge set $\{\{T, T[x]\} \mid T \in \mathcal{T}_{\mathsf{part}}(P), x \text{ flippable in } T\}$.

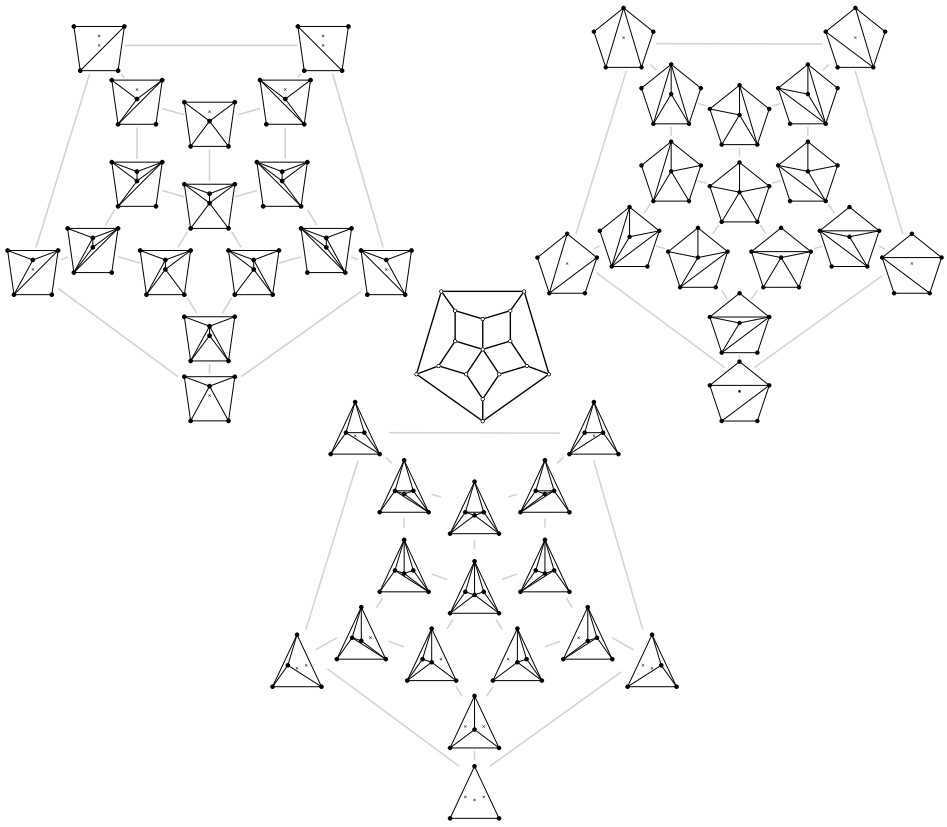


Figure 3 Sets of 6 points with isomorphic bistellar flip graphs of triangulations. (Points indicated by crosses are points in P skipped in the corresponding triangulation.)

The bistellar flip graph is connected (this follows easily from the connectedness of the edge flip graph of full triangulations, as established by Lawson in 1972 [8]). Here, we investigate how well connected the bistellar flip graph is. We refer to standard texts like [2, 6] for basics like the definition of *k-vertex connectivity* and *Menger's Theorem*. Our main result is:

▶ **Theorem 4.** *Let P be a set of $n \geq 3$ points in general position in the plane. Then the bistellar flip graph of P is $(n-3)$-vertex connected. (This is tight: Any triangulation of P that skips all inner points has degree $(n-3)$ in the bistellar flip graph.)*

This answers (for points in general position) a question by De Loera, Rambau, and Santos in 2010 [4, Exercise 3.23], and by Lee and Santos in 2017 [9, pg. 442]. A corresponding result, $\lceil \frac{n}{2} - 2 \rceil$-connectedness of the edge flip graph of full triangulations, is proved in [14].

A particular way of obtaining a triangulation of a point set P is to vertically lift the points to $\mathbb{R}^3$ such that no 4 points are coplanar, and then to project the lower convex hull of the lifted points back into the plane. Triangulations obtained in this way are called *regular triangulations* (e.g., [4]). It is well known that point sets may have non-regular triangulations.

Furthermore, we study the partially ordered set of *subdivisions* of P (see Def. 9 below, and, e.g., [4]), in which triangulations are the minimal elements. We introduce the notions of *slack* (Def. 10), *perfect coarsenings* (Def. 20), and *perfect coarseners* (Def. 21), and we prove the so-called *Coarsening Lemma 25*. We consider these our main contributions besides Thm. 4. As consequences, these yield several other results on the structure of the bistellar flip graph and regular triangulations (see abstract); in particular, they allow us to settle, in

an unexpected direction, another question by Santos [12] regarding the size of certificates for the existence of non-regular triangulations in the plane. Here, we focus on the proof of the connectivity, and we refer to the full version [13] for these additional results.

If P is in convex position, full, partial, and regular triangulations coincide. It is well-known that there is an $(n-3)$-dimensional convex polytope, the *associahedron*, whose vertices correspond to the triangulations of P and whose edges correspond to flips (Fig. 4, see [3] for a historical account). A classical theorem of Balinski [1], which asserts that the graph of any d-dimensional polytope is d-connected, immediately implies that the graph of the associahedron is $(n-3)$-connected. More generally, for arbitrary sets in the plane, it is known that there is an $(n-3)$-dimensional polytope, the *secondary polytope* defined by Gelfand et al. [7], whose vertices correspond to the regular triangulations of P and edges correspond to bistellar flips; again, Balinski's Theorem implies $(n-3)$-connectivity. Our result extends this to arbitrary triangulations of arbitrary sets in general position in the plane.

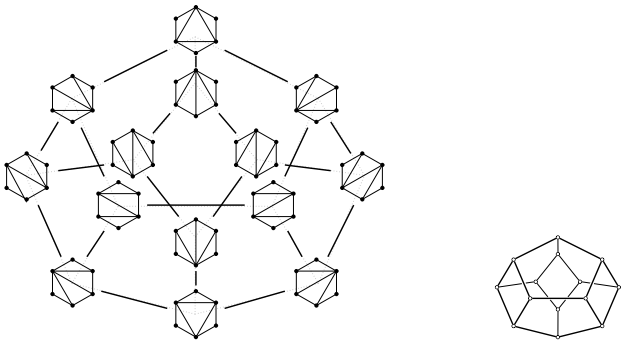


Figure 4 The flip graph of the convex hexagon, the graph of the order 5 associahedron.

Approach and Intuition

There is evidence that the bistellar flip graph of partial triangulations does not exhibit a polytopal structure as we see it with regular triangulations [4]. Still, the intuition behind our approach is to "pretend" that such a structure exists, at least locally for the small dimensional features. This will become clearer below, and is made more explicit in the full version [13] where it shown that the bistellar flip graph can be covered by polytopal structures.

2 Preliminaries, Terminology, and Notation

▶ **Definition 5** (legal graph; region)**.** For a graph $G = (P', E)$, $P' \subseteq P$, we let $\mathsf{V}^{\mathsf{by}}G$ be the points in P' which are isolated in G, called *bystanders* in G. G is called *legal* if it is plane, if $\mathsf{E}_{\mathsf{hull}}(P) \subseteq \mathsf{E}G$ (hence $\mathsf{xtr}P \subseteq P'$), and if the graph $(\mathsf{V}G \setminus \mathsf{V}^{\mathsf{by}}G, \mathsf{E}G)$ is 2-edge connected.

Let G be a legal graph. Similar to triangulations, we define $\mathsf{E}^\circ G := \mathsf{E}G \setminus \mathsf{E}_{\mathsf{hull}}$ and $\mathsf{V}^\circ G := \mathsf{V}G \cap P^\circ$. Moreover, we let $\mathsf{V}^{\mathsf{inv}}G := \mathsf{V}^\circ G \setminus \mathsf{V}^{\mathsf{by}}G$ (the *involved points*). Bounded faces of $(\mathsf{V}G \setminus \mathsf{V}^{\mathsf{by}}G, \mathsf{E}G)$ are called *regions* of G, i.e., these are bounded connected components in the complement of the straight line embedding of G, ignoring its bystanders.

Regions of legal graphs are bounded simply connected polygonal open sets, pairwise disjoint. We state the following well-known facts for ease of reference.

▶ **Lemma 6.** *For a full triangulation T of P, $|\mathsf{E}T| = |\mathsf{E}^\circ T| + h = 3n - 3 - h = 3n^\circ - 3 + 2h$ and the number of regions (which does not include the unbounded face) is $2n - 2 - h = 2n^\circ - 2 + h$.*

▶ **Definition 7** (locked). In a legal graph G on P, an edge $e \in \mathsf{E}G$ is *locked at endpoint* p if the angle obtained at p (between the edges adjacent to e at p) after removal of e exceeds π.

An edge in a triangulation is flippable iff it is locked by none of its endpoints. Edges locked at a common endpoint p have to be consecutive around p. There can be at most 3 edges locked at a given point p, and 3 edges can be locked at p only if p has degree 3.

Given a legal graph G, we consider *partial orientations* $\vec{G}$: These assign orientations to *some* (not all) of the edges in $\mathsf{E}G$, with no edge oriented in both directions, and with the boundary edges not oriented. We need the following [14, Lemma 5.1(i)]:

▶ **Lemma 8** (Unoriented Edges Lemma). *Let G be a legal graph with $\mathsf{V}^{\mathsf{by}}G = \emptyset$, $N := |\mathsf{V}G|$, and $D := 3N - 3 - h - |\mathsf{E}G|$, i.e., the number of edges missing in G towards a full triangulation of $\mathsf{V}G$. For $\vec{G}$ a partial orientation of G, let C_i be the number of inner points of $\vec{G}$ with indegree i and suppose $C_i = 0$ for $i \geq 4$. Then the number of unoriented inner edges is at least $N - 3 - C_3 - D$.*

To indicate, how this can be useful in our context, consider $G = T$, T a triangulation, i.e., $D = 0$. Orient every locked inner edge to the endpoint where it is locked. Then $C_i = 0$ for $i \geq 4$, C_3 is exactly the number of inner points of degree 3, and the inner unoriented edges are exactly the unlocked, i.e., flippable edges. It follows that there are C_3 point removal flips, at least $N - 3 - C_3$ edge flips, and obviously $n - N$ point insertion flips. Altogether, there are at least $n - 3$ flips.

3 Partial Subdivisions – Slack and Order

We now define *partial subdivisions*, which form a poset in which the partial triangulations of P are the minimal elements.

▶ **Definition 9** (full, partial subdivision). A *partial subdivision* S on P is a legal graph with all of its regions convex. For a region r of S, let $\mathsf{V}r := \overline{r} \cap \mathsf{V}S$ ($\overline{r}$ the closure of r). $S_{\mathsf{triv}} = S_{\mathsf{triv}}(P) := (P, \mathsf{E}_{\mathsf{hull}})$ is called the *trivial subdivision of* P. If $\mathsf{V}S = P$ and $\mathsf{V}^{\mathsf{by}}S = \emptyset$, then S is called a *full subdivision* on P.

▶ **Convention.** From now on, we will mostly use "subdivision" for "partial subdivision".

$\mathsf{V}S$ is essential in the definition of a subdivision, it is *not* simply the set of endpoints of edges in S, there are also bystanders; e.g., for T a triangulation of P, all graphs $(P', \mathsf{E}T)$, $\mathsf{V}T \subseteq P' \subseteq P$, are subdivisions of P, all different. $\mathsf{V}S$ partitions into boundary points, involved points, and bystanders, i.e., $\mathsf{V}S = \mathsf{xtr}P \dot\cup \mathsf{V}^{\mathsf{inv}}S \dot\cup \mathsf{V}^{\mathsf{by}}S$. Moreover there are the skipped points, $P \setminus \mathsf{V}S$.

A first important example of a subdivision is obtained from a triangulation T and an element x flippable in T, i.e., $\{T, T[x]\}$ is an edge of the bistellar flip graph:

$$T_{\pm x} := (\mathsf{V}T \cup \mathsf{V}T[x], \mathsf{E}T \cap \mathsf{E}T[x])$$

If $x = e$ is a flippable edge, then $T_{\pm e}$ has one convex quadrilateral region Q; all other regions are triangular. We obtain T and $T[e]$ from $T_{\pm e}$ by adding one or the other of the 2 diagonals of Q to $T_{\pm e}$. If $x = p$ is a flippable point, then $T_{\pm p}$ is almost a triangulation, all regions are triangular, except that $p \in \mathsf{V}T_{\pm p}$ is a bystander. We obtain T and $T[p]$ by either removing this point from $T_{\pm p}$ or by adding the 3 edges from p to the points of the triangular region in which p lies. The subdivision $T_{\pm x}$ is close to a triangulation and, in a sense, represents the flip between T and $T[x]$. To formalize and generalize this we introduce the following notion:

▶ **Definition 10** (slack)**.** Given a subdivision S of P, we call a region of S *active* if it is not triangular or if it contains at least one point in $\mathsf{V}S$ (necessarily a bystander) in its interior.

For a region r of S, we define its *slack* $\mathsf{sl}\, r := |\mathsf{V}r| - 3$. The *slack of* S, $\mathsf{sl}\, S$, is the sum of slacks of its regions.

▶ **Lemma 11.** *For a subdivision S with s bystanders we have*

$$\mathsf{sl}\, S = 3(|VS| - s) - 3 - h - |\mathsf{E}S| + s = 3|VS| - 3 - h - |\mathsf{E}S| - 2s\ .$$

Proof. The slack of a region r equals the number of edges it takes to triangulate r (ignoring bystanders) plus the number of bystanders. Thus, $\mathsf{sl}\, S$ is the number of edges it takes to triangulate $(\mathsf{V}S \setminus \mathsf{V}^{\mathsf{by}}S, \mathsf{E}S)$ plus $|\mathsf{V}^{\mathsf{by}}S|$. Now the claim follows from Lemma 6. ◀

▶ **Observation 12.** *Let S be a subdivision. (i) $\mathsf{sl}\, S = 0$ iff S is a triangulation iff S has no active region. (ii) $\mathsf{sl}\, S = 1$ iff S has exactly one active region of slack 1; this region is either a convex quadrilateral, or a triangular region with one bystander in its interior. (iii) $\mathsf{sl}\, S = 2$ iff S has either (a) exactly 2 active regions, both of slack 1, or (b) exactly one active region of slack 2, where this region is either a convex pentagon, or a convex quadrilateral with one bystander in its interior, or a triangular region with 2 bystanders in its interior (Fig. 2).*

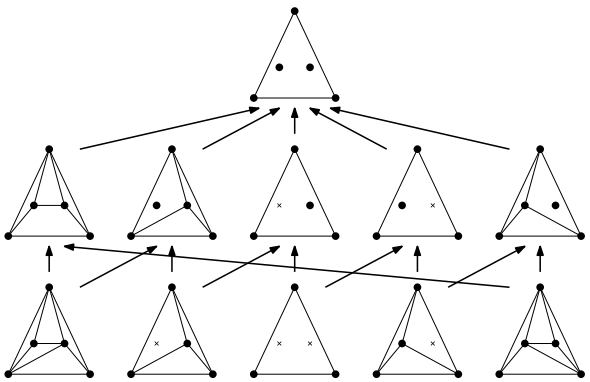


Figure 5 Hasse diagram of the partial order $\preceq$ for a set of 5 points.

▶ **Definition 13** (coarsening, refinement)**.** For subdivisions S_1 and S_2 of P, S_2 *coarsens* S_1, in symbols $S_2 \succeq S_1$, if $\mathsf{V}S_2 \supseteq \mathsf{V}S_1$, and $\mathsf{E}S_2 \subseteq \mathsf{E}S_1$. We also say that S_1 *refines* S_2, $(S_1 \preceq S_2)$.

The example in Fig. 5 hides some of the intricacies of the partial order $\preceq$; e.g., in general, it is not true that all paths from a triangulation to S_{triv} have the same length $n-3$. S_{triv} is the unique coarsest (maximal) element. The triangulations (i.e., subdivisions of slack 0) are the minimal elements.

▶ **Definition 14** (set of refining triangulations)**.** For a subdivision S of P we let $\mathcal{T}_{\mathsf{part}}\langle S\rangle := \{T \in \mathcal{T}_{\mathsf{part}}(P) \,|\, T \preceq S\}$.

Note that $\mathcal{T}_{\mathsf{part}}\langle S_{\mathsf{triv}}\rangle = \mathcal{T}_{\mathsf{part}}(P)$ and for x flippable in T, $\mathcal{T}_{\mathsf{part}}\langle T_{\pm x}\rangle = \{T, T[x]\}$.

▶ **Observation 15.** *(i) Any subdivision S of slack 1 of P equals $T_{\pm x}$ for some triangulation $T \preceq S$ and some x flippable in T. (ii) Let S be a subdivision of slack 2 of P. If there are exactly 2 active regions in S (of slack 1 each), then $\mathcal{T}_{\mathsf{part}}\langle S\rangle$ has cardinality 4, spanning a 4-cycle in the bistellar flip graph of P (Fig. 6). If there is exactly 1 active region in S (of slack 2), then $\mathcal{T}_{\mathsf{part}}\langle S\rangle$ has cardinality 5, spanning a 5-cycle (Fig. 2).*

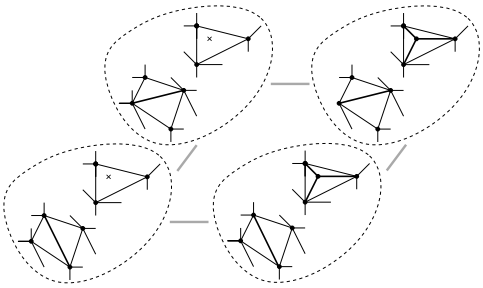


Figure 6 A subdivision S with 2 active regions of slack 1 each with $\mathcal{T}_{\mathsf{part}}\langle S\rangle$ spanning a 4-cycle.

▶ **Lemma 16.** *Any proper refinement of a subdivision of slack* 2 *has slack at most* 1.

Proof. Let $\mathsf{sl}\, S' = 2$ and let S be a proper refinement of S'. For a refinement we add m edges, thereby involving s' bystanders, and we remove s'' bystanders (some of these parameters may be 0, but not all, since the refinement is assumed to be proper). We have $\mathsf{sl}\, S = \mathsf{sl}\, S' - (m - 2s' + s'')$ (easy consequence of Lemma 11) and want to show $m - 2s' + s'' > 0$.

Since $\mathsf{sl}\, S' = 2$, S' has at most 2 bystanders and thus $s' \leq 2$. If $s' = 0$, then $m - 2s' + s'' > 0$ holds, since some of the 3 parameters have to be positive. If $s' = 1$, we observe that we need at least 3 edges to involve a bystander and $m - 2s' \geq 3 - 2 \cdot 1$. If $s' = 2$, we need at least 5 edges to involve 2 bystanders and $m - 2s' \geq 5 - 2 \cdot 2$. ◀

For $D \geq 3$, a proper refinement of a subdivision of slack D can have slack D or even higher (Fig. 7). The proof fails, since we can involve 3 bystanders with 6 edges.

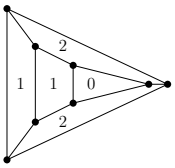


Figure 7 8 points, with a subdivision of slack 6, a refinement of S_{triv} of slack $8 - 3 = 5$.

Intuitively, as briefly alluded to at the end of Sec. 1, one can think of the subdivisions as the faces of a higher-dimensional geometric structure behind the bistellar flip graph, with the slack playing the role of dimension, somewhat analogous to the secondary polytope for regular triangulations. (For the edge flip graph of full triangulations, an analogous higher-dimensional *flip complex* is treated in [11, 10], and provides a similar geometric intuition for the arguments in [14].) The following lemma shows that – for slack at most 2 – we have the property corresponding to the fact that faces of dimension d are either equal, or intersect in a common face of smaller dimension (possibly empty).

▶ **Lemma 17.**

(i) *For subdivisions S_1 and S_2 of slack* 2, $\mathcal{T}_{\mathsf{part}}\langle S_1\rangle \cap \mathcal{T}_{\mathsf{part}}\langle S_2\rangle$ *is either* (a) *empty,* (b) *equals $\{T\}$ for some triangulation T,* (c) *equals $\{T, T[x]\}$ for some triangulation T and some flippable element x, or* (d) $S_1 = S_2$.

(ii) *Let x and y be two distinct flippable elements in triangulation T. If there is a subdivision S of slack* 2 *with $\{T[x], T, T[y]\} \subseteq \mathcal{T}_{\mathsf{part}}\langle S\rangle$, then this S is unique.*

Proof. If $\mathcal{T}_{\mathsf{part}}\langle S_1\rangle \cap \mathcal{T}_{\mathsf{part}}\langle S_2\rangle$ contains some triangulation, then we easily see that $S_1 \wedge S_2 := (\mathsf{V}S_1 \cap \mathsf{V}S_2, \mathsf{E}S_1 \cup \mathsf{E}S_2)$ is a subdivision, and $\mathcal{T}_{\mathsf{part}}\langle S_1 \wedge S_2\rangle = \mathcal{T}_{\mathsf{part}}\langle S_1\rangle \cap \mathcal{T}_{\mathsf{part}}\langle S_2\rangle$.

(i) If (a) does not apply, let $S := S_1 \wedge S_2$, a subdivision with $\mathcal{T}_{\mathsf{part}}\langle S\rangle = \mathcal{T}_{\mathsf{part}}\langle S_1\rangle \cap \mathcal{T}_{\mathsf{part}}\langle S_2\rangle$. If $\mathsf{sl}\, S = 0$ we have property (b), if $\mathsf{sl}\, S = 1$ we have property (c). In the remaining case $\mathsf{sl}\, S \geq 2$, S is a refinement of S_1 and of S_2. Lemma 16 tells us that S cannot be a proper refinement of S_1, hence $S = S_1$; similarly, $S = S_2$, hence $S_1 = S_2$.

(ii) Suppose S_1 and S_2 are subdivisions of slack 2 with $\{T[x], T, T[y]\} \subseteq \mathcal{T}_{\mathsf{part}}\langle S_1\rangle \cap \mathcal{T}_{\mathsf{part}}\langle S_2\rangle$. Since options (a-c) above cannot apply, we are left with $S_1 = S_2$. ◀

Two edges incident to a vertex of a polytope may span a 2-face, or not; same here:

▶ **Definition 18** (compatible elements)**.** Two distinct flippable elements $x, y \in \mathsf{V}^\circ T \cup \mathsf{E}^\circ T$ are called *compatible in* T, in symbols $x \diamond y$, if there is a subdivision $T_{\pm x,y} \succeq T$ of slack 2, s.t. $\{T[x], T, T[y]\} \subseteq \mathcal{T}_{\mathsf{part}}\langle T_{\pm x,y}\rangle$. (Note that $T_{\pm x,y}$ is unique, by Lemma 17(ii).) Otherwise, x and y are called *incompatible in* T, in symbols $x \not\diamond y$.

This needs some time to digest. In particular, if two flippable edges e and f share a common endpoint of degree 4, then they are compatible, see Fig. 8 (bottom left), quite contrary to the situation for full triangulations as treated in [14]. The configurations of 2 flippable but incompatible are shown in Fig. 8, rightmost examples: (a) Two flippable edges e and f whose removal creates a nonconvex pentagon and whose common endpoint q has degree at least 5. (b) A flippable edge e and a flippable point p of degree 3 whose removal creates a nonconvex quadrilateral region whose reflex point q has degree at least 5 in the triangulation.

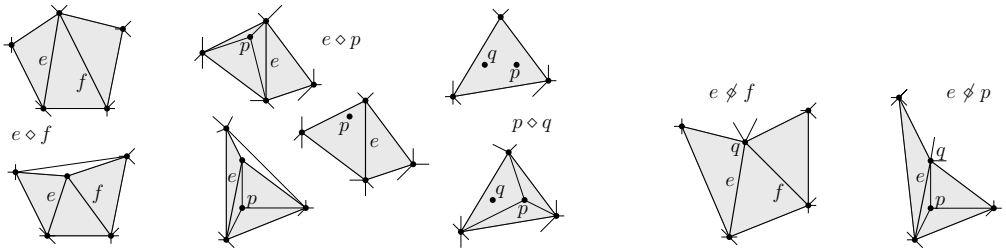


Figure 8 Compatible elements (with overlapping incident regions, all contained in a 5-cycle, see Fig. 2 and incompatible elements (two rightmost, where q is assumed to have degree at least 5). Shaded areas are unions of incident regions of flippable elements (not the active region in $T_{\pm x,y}$!).

What is essential for us is that whenever x and y are compatible in a triangulation T, then there is a cycle of length 4 or 5 containing $(T[x], T, T[y])$, and therefore, apart from the path $(T[x], T, T[y])$, there exists a T-avoiding $T[x]$-$T[y]$-path of length 2 or 3.

▶ **Observation 19.** *Let $T \in \mathcal{T}_{\mathsf{part}}(P)$. (i) A skipped point $p \in P^\circ \setminus \mathsf{V}^\circ T$ is compatible with every flippable element of T. (ii) Any two flippable points $p, q \in P^\circ$ are compatible.*

4 Coarsening Partial Subdivisions

As in [14] for full triangulations, the existence of many coarsenings is essential for our connectivity result. In order to motivate the definitions below, note that for full subdivisions (as employed in [14]), if $S_1 \preceq S_2$, then (S_1, S_2) is an edge in the Hasse-diagram of the partial order $\preceq$ iff $\mathsf{sl}\, S_2 = \mathsf{sl}\, S_1 + 1$. For partial subdivisions, this is not the case (Fig. 9).

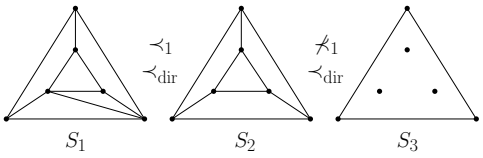


Figure 9 $\mathsf{sl}\, S_1 = 2$, $\mathsf{sl}\, S_2 = 3$, $\mathsf{sl}\, S_3 = 3$. Note that $S_2 \prec_{\text{dir}} S_3$ but $S_2 \not\prec_1 S_3$, and that $S_1 \preceq S_3$ with $\mathsf{sl}\, S_3 = \mathsf{sl}\, S_1 + 1$ but $S_1 \not\prec_1 S_3$.

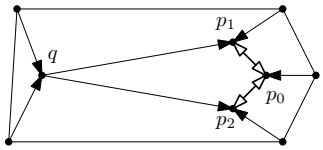


Figure 10 A subdivision, edges are oriented to endpoints where locked (not what we called a partial orientation, since some edges are doubly oriented). Removing the 3 edges incident to p_0 does not yield a subdivision, since a reflex angle occurs at p_1 and p_2. The edges incident to $\{p_0, p_1, p_2\}$ are not looked outside this set, but removing all incident edges creates a reflex angle at point q.

▶ **Definition 20** (direct, perfect coarsening)**.** Let S_1 and S_2 be subdivisions. (i) We call S_2 a *direct coarsening of* S_1 (and S_1 a *direct refinement of* S_2), in symbols $S_1 \prec_{\text{dir}} S_2$, if $S_1 \preceq S_2$ and any subdivision S with $S_1 \preceq S \preceq S_2$ satisfies $S \in \{S_1, S_2\}$ (equivalently, if (S_1, S_2) is an edge in the Hasse diagram of $\preceq$). (ii) We call S_2 a *perfect coarsening of* S_1 (S_1 a *perfect refinement of* S_2), in symbols $S_1 \prec_1 S_2$, if $S_1 \prec_{\text{dir}} S_2$ and $\mathsf{sl}\, S_2 = \mathsf{sl}\, S_1 + 1$. (iii) $\prec_1^*$ is the reflexive transitive closure of $\prec_1$.

The reflexive transitive closure of $\prec_{\text{dir}}$ is exactly $\preceq$, while $\prec_1^* \subseteq \preceq$ and, in general, the inclusion is proper.

To motivate the upcoming definitions, let us discuss a few possibilities of coarsenings, direct coarsenings and perfect coarsenings. There are the simple operations of removing an unlocked edge, and of adding a point $p \in P \setminus \mathsf{V}S$ as a bystander. For a triangulation, we can isolate a point of degree 3. How does this generalize to subdivisions? Removing the edges incident to a point of degree 3 does not work if some incident edge might be locked at its other endpoint (e.g., p_0 in Fig. 10). If, however, no edge incident to a given point p (of any degree) is locked at the respective other endpoint, then we can isolate this point for a coarsening S'. Unless p has degree 3, S' is not a direct coarsening of S, though. If p has degree at least 4, one of the incident edges, say e, is not locked at p, thus not locked at all, and therefore, $S \preceq S'' \preceq S'$ for $S'' := (\mathsf{V}S, \mathsf{E}S \setminus \{e\})$. Finally, suppose we want to isolate all points in a set U of points for obtaining a coarsening S'. For this to work, it is necessary that no edge e connecting U with the outside is locked at the endpoint of e not in U. However, this is not a sufficient condition, because several edges connecting U with a point not in U can collectively create a reflex vertex by their removal (e.g., $U = \{p_0, p_1, p_2\}$ in Fig. 10). Moreover, for $S \prec_{\text{dir}} S'$ to hold, U cannot be incident to unlocked edges, and no nonempty subset of U can be suitable for such an isolation operation.

▶ **Definition 21** (prime, perfect coarsener; increment)**.** Let S be a subdivision and let $U \subseteq \mathsf{V}S \cap P^\circ$. (i) U is called a *coarsener*, if (a) U is incident to at least one edge in S, and (b) removal of the set E_U of all edges incident to U in S yields a subdivision. (ii) If U is a coarsener, the *increment of* U, $\mathsf{inc}U$, is defined as $|E_U| - 2|U|$. (iii) U is called a *prime coarsener*, if (a) U is a coarsener, (b) U is a minimal coarsener, i.e., no proper subset of U is a coarsener, and (c) all edges incident to U are locked. (iv) U is called a *perfect coarsener*, if (a) U is a prime coarsener, and (b) $\mathsf{inc}U = 1$.

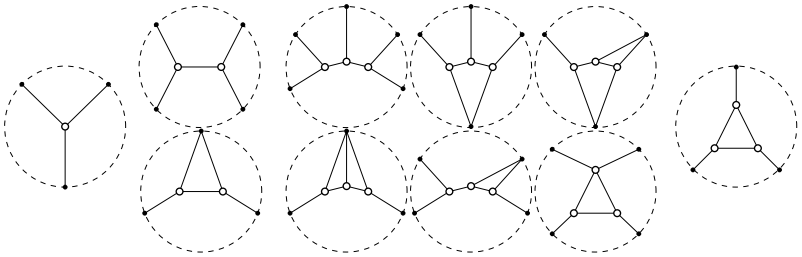


Figure 11 Prime coarseners, all perfect, except for the rightmost one (with $\mathsf{inc} = 0$).

The following observation, a simple consequence of Lemma 11, explains the term "increment".

▶ **Observation 22.** *Let S be a subdivision with coarsener U, and let S' be the subdivision obtained from S by removing all edges incident to U. Then $\mathsf{sl}S' = \mathsf{sl}S + \mathsf{inc}U$.*

▶ **Observation 23.**

(i) *Every subdivision S with $\mathsf{E}^\circ S \neq \emptyset$ has a coarsener (the set $\mathsf{V}^\circ S$).*

(ii) *If U_1 and U_2 are coarseners, then $U_1 \cap U_2$ is a coarsener, unless there is no edge of S incident to $U_1 \cap U_2$.*

(iii) *If U_1 and U_2 are prime coarseners, then $U_1 = U_2$ or $U_1 \cap U_2 = \emptyset$.*

(iv) *If U is a prime coarsener, then the subgraph of S induced by U is connected.*

The following observation lists all ways of obtaining direct and perfect coarsenings.

▶ **Observation 24.** *Let $S = (V, E)$ and S' be subdivisions.*

(i) *S' is a direct coarsening of S iff it is obtained from S by one of the following.*

Adding a single point. *For $p \in P \setminus V$, $S' = (V \cup \{p\}, E)$ (with $\mathsf{sl}S' = \mathsf{sl}S + 1$).*

Removing a single unlocked edge. *For $e \in E$, not locked by either of its two endpoints, $S' = (V, E \setminus \{e\})$ (with $\mathsf{sl}S' = \mathsf{sl}S + 1$).*

Isolating a prime coarsener. *For U a prime coarsener in S, S' is obtained from S by removal of the set, E_U, of all edges incident to points in U, i.e., $S' = (V, E \setminus E_U)$ (with $\mathsf{sl}S' = \mathsf{sl}S + \mathsf{inc}U$).*

(ii) *S' is a perfect coarsening of S iff it is obtained from S by adding a single point, removing a single unlocked edge, or by isolating a perfect coarsener.*

▶ **Lemma 25** (Coarsening Lemma for Partial Subdivisions)**.** *Every subdivision of slack D has at least $n - 3 - D$ perfect coarsenings (i.e., direct coarsenings of slack $D + 1$).*

Proof. We start with the case $D = 0$, i.e., we have a triangulation T and we want to show that there are at least $n - 3$ direct coarsenings of slack 1. Let $N := |\mathsf{V}T|$. We orient inner locked edges to their locking endpoints (recall that in a triangulation there is at most one such endpoint for each inner edge). Let C_i, $i \in \mathbb{N}_0$, be the number of points $p \in \mathsf{V}^\circ T$ with indegree i. The number of unoriented, thus unlocked edges is at least $N - 3 - C_3$ (Lemma 8).

There are $n - N$ subdivisions obtained from T by adding a single point, there are at least $N - 3 - C_3$ subdivisions obtained from T by removing a single unlocked edge, and there are C_3 direct coarsenings obtained from T by isolating an inner point of degree 3. Adding up these numbers gives at least $n - 3$ perfect coarsenings of T.

We let S be a subdivision of slack $D \geq 1$ assuming the assertion holds for slack less than D.

Case 1. *There is a bystander* $p_0 \in \mathsf{V}S \cap P^\circ$. Then $(\mathsf{V}S \setminus \{p_0\}, \mathsf{E}S)$ is a subdivision of slack $D-1$ of $P \setminus \{p_0\}$ with at least $(n-1)-3-(D-1) = n-3-D$ perfect coarsenings of slack D. For each such perfect coarsening S', the subdivision $(\mathsf{V}S' \cup \{p_0\}, \mathsf{E}S')$ is a direct coarsening of S of slack $D+1$, thus a perfect coarsening.

Case 2. *There is no bystander in* S. Again we employ a partial orientation of S. The choice of the orientation is somewhat more intricate and we will proceed in three phases (Fig. 12). We keep the invariant that the unoriented inner edges are exactly the unlocked inner edges.

In a *first phase*, we orient all locked inner edges to *all* of their locking endpoints, i.e., we temporarily allow edges to be directed to both ends (to be corrected in the third phase); edges directed to both endpoints are called *mutual edges*. We can give the following interpretation to an edge directed from p to q: If we decide to isolate p (i.e., remove all incident edges of p) for a coarsening of S, then q becomes a reflex point of some region and we have to isolate q as well (i.e., every coarsener containing p must contain q as well). In particular, if $\{p, q\}$ is a mutual edge, then either both or none of the points p and q will be isolated. In fact, if we consider the graph G on $\mathsf{V}^\circ S$ with all mutual edges in the current orientation, then in any coarsening of S either all points in a connected component of G are isolated, or none.

A connected component K of G is called a *candidate component*, (a) if all edges connecting K with points outside are directed towards K, (b) no point in K is incident to an unoriented edge, (c) all points in K have indegree 3, and (d) the mutual edges in K do not form any cycle (i.e., they have to form a spanning tree of K). It follows that if K has k points then the number of edges is $3k-(k-1) = 2k+1$. The term "candidate" refers to the fact that removing all edges incident to K *seems* like a direct coarsening step with incrementing the slack by 1 (Lemma 11); however, while individual edges connecting K to the rest of the graph are not locked at their endpoints outside K, some of these edges collectively may actually create a reflex vertex in this way (see K and q in Fig. 12, left). So K is only a candidate for a perfect coarsener.

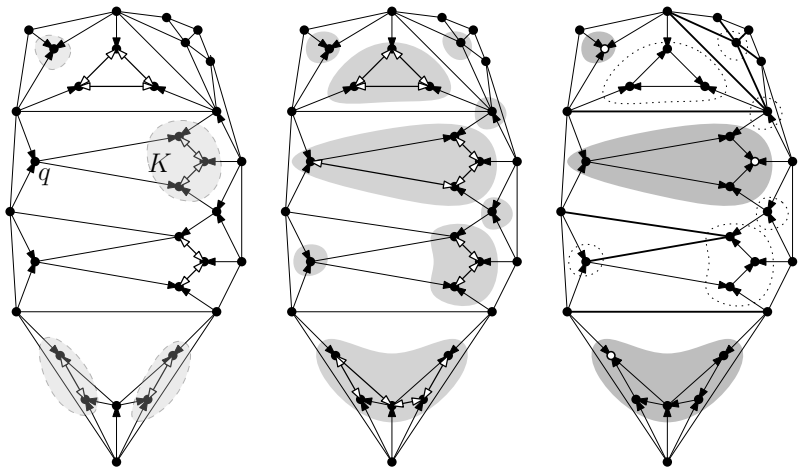


Figure 12 Left: orientation after phase 1, with candidate components shaded; middle: after phase 2, with the connected components of G^*; right: after phase 3, with unoriented edges bold (each of these can be removed for a coarsening of slack 1 larger), and with the candidate components with a leader shaded (perfect coarseners).

We start the *second phase* of orienting edges further. In the spirit of our remarks about candidate components of G, suppose q is an inner point outside a candidate K of G (thus all edges connecting q to K are directed from q to K), such that removing the edges connecting

q to K creates a reflex angle at q. Then we orient one (and only one) of the edges connecting q to K, say $\{p, q\}$, also to q (thereby making this edge mutual). We call all the edges connecting K to q, except for $\{p, q\}$, the *witnesses of the extra new orientation of* $\{p, q\}$ *from* p *to* q. We successively proceed orienting edges, with the graph G of mutual edges evolving in this way (and candidate components growing or disappearing).[1] The process will clearly stop at some point when the second phase is completed. We freeze G and denote it by G^*.

Before we start the third phase, let us make a few crucial observations:

(i) If p, q are inner points in the same connected component of G^*, then any coarsener contains both or none (i.e., if a connected component is a coarsener, then it is prime). This holds after phase 1, and whenever we expand a connected component, it is maintained.

(ii) During the second phase, an edge can be witness only once, and it is and will never be directed to the endpoint where it witnesses. Why? (a) Before it becomes a witness, it connects different connected components of G, after that it is and stays in a connected component of G. (b) Before it becomes a witness, it is not directed to the endpoint to which it witnesses an orientation, after that it is and stays in a connected component of G and can therefore not get an extra direction. (An unoriented edge can never get an orientation and it can never be a witness.)

(iii) If we remove, conceptually, for each incoming edge of a point q the witnesses (which direct away from q) for the orientation of this edge to q, then among remaining incident edges, all the incoming edges are locked at q (an incoming edge that was oriented already in the first phase to q has no witness). In particular, the indegree of q cannot exceed 3, and if q is incident to some not ingoing edge which is not a witness for any edge incoming at q, then the indegree of q is at most 2. (We might generate incoming edges to a point q that are not consecutive around q.)

(iv) If an unoriented edge e connects two points of the same connected component of G^*, then both endpoints have indegree at most 2 (recall that this edge e cannot be a witness at its endpoints). If an edge e is directed from a connected component K of G^* to a point outside K, then the tail of this edge e has indegree at most 2 (recall that e cannot be a witness at all, since its endpoints are in different connected components if G^*).

(v) A candidate component K of G^* is a perfect coarsener. It is a coarsener (otherwise, we would have expanded it further), it is a prime coarsener (see (i) above) and $\mathsf{inc}\, K = 1$ (we have argued before that a candidate component increases the slack by exactly 1).

The *third phase* will make sure that each mutual edge loses exactly one direction. Our goal is to have in every connected component K of G^* at most one point with indegree 3. To be more precise, only candidate components have exactly one point with indegree 3, others don't. Consider a connected component K.

(a) If the mutual edges form cycles in K, choose such a cycle c and keep for each edge on c one orientation so that we have a directed cycle, counterclockwise, say. All other mutual edges in K keep the direction in decreasing distance in G^* to c, ties broken arbitrarily. This completed, no point in K has indegree 3, since there is always a mutual edge incident that decreases the distance to c and the incoming direction of this edge will be removed.

[1] The reader will correctly observe that our approach is very conservative towards prime coarseners, but by what we observed and by what will follow, since we are interested only in perfect coarseners, we can afford to leave alone connected components other than the candidate components.

(b) If K has points of indegree at most 2, choose one such point p with indegree at most 2, orient all mutual edges in K in decreasing distance in G^* to p, ties broken arbitrarily. Again, this completed, no point in K will have indegree 3.

(c) If none of the above applies, the mutual edges of K form a spanning tree and all points in K have indegree 3. Moreover, all edges connecting K with points outside are directed towards K and no edge within K is unoriented (violation of these properties force a point of indegree at most 2). So this is a candidate component. We choose an arbitrary point p in K, call it the *leader of* K, and for all mutual edges keep the orientation of decreasing distance in G^* to p (ties cannot occur, mutual edges form a tree). Now the leader p is the only point of K with indegree 3, all other points in K have indegree exactly 2.

Phase 3 is completed. Let us denote the obtained partial orientation on S as $\vec{S}^*$. It has identified certain connected components of G^* which have a leader of indegree 3. In fact, every point of indegree 3 after phase 3 is part of a perfect coarsener (probably of size 1).

We can now describe a sufficient supply of perfect coarsenings of S. Let $N := |\mathsf{V}S|$ and let C_3 be the number of points of indegree 3 in $\vec{S}^*$. We know that there are at least $N - 3 - D - C_3$ unoriented inner edges (Lemma 8).

(I) There are $n - N$ perfect coarsenings obtained by adding a single point $p \in P \setminus \mathsf{V}S$.

(II) There are at least $N - 3 - D - C_3$ perfect coarsenings obtained by removing a single unoriented inner edge in $\vec{S}^*$.

(III) And there are C_3 perfect coarsenings obtained by isolating all points in a candidate component in G^* (with a leader of indegree 3).

In this way we have identified at least $n - 3 - D$ perfect coarsenings. ◀

▶ **Corollary 26.** *Let T be a triangulation.* (i) *T has at least $n-3$ flippable elements.* (ii) *For every x flippable in T there are at least $n-4$ elements compatible with x.*

Part (i) of the corollary was proved, without general position assumption, in [5, Thm. 2.1].

5 The Link of a Triangulation – Proof of $(n-3)$-Connectivity

To complete the proof of the connectivity bound for the bistellar flip graph, we need two further ingredients. The first is the following variant of Menger's Theorem [14, Lemma 3.1].

▶ **Lemma 27** (Local Menger). *Let $k \geq 2$ be an integer and let G be a* connected *simple undirected graph. Then G is k-vertex connected iff G has at least $k+1$ vertices and for any pair of vertices u and v* at distance 2 *there are k pairwise internally vertex disjoint u-v-paths.*

The second ingredient are *links* of triangulations, which are graphs that represent the compatibility relation among flippable elements (Def. 18). Recall that if x is a flippable element in a triangulation T then $T_{\pm x}$ denotes the subdivision with $\mathcal{T}_{\mathsf{part}}\langle T_{\pm x}\rangle = \{T, T[x]\}$, and if y is compatible with x, denoted $x \diamond y$, then $T_{\pm x,y}$ denotes the unique coarsening of slack 2 of T with $\{T[x], T, T[y]\} \subseteq \mathcal{T}_{\mathsf{part}}\langle T_{\pm x,y}\rangle$ (Def. 18).

▶ **Definition 28** (link). For $T \in \mathcal{T}_{\mathsf{part}}(P)$, the *link of* T, denoted $\mathsf{Lk}T$, is the edge-weighted graph with vertices $\mathsf{F}T := \{x \in \mathsf{V}^\circ T \cup \mathsf{E}^\circ T \mid x \text{ flippable in } T\}$ and edge set $\{\{x,y\} \in \binom{\mathsf{F}T}{2} \mid x \diamond y\}$. The *weight of* an edge $\{x,y\}$ is $|\mathcal{T}_{\mathsf{part}}\langle T_{\pm x,y}\rangle| - 2$ (which is 2 or 3).

We will see that it is enough to prove $(n-4)$-vertex connectivity of all links. Again, the intuition can be explained for polytopes: Recall that for a vertex v in a d-polytope $\mathcal{P}$, its vertex figure is the $(d-1)$-polytope $\mathcal{P}'$ obtained by intersecting $\mathcal{P}$ with a hyperplane that separates v from the remaining vertices of the polytope. Vertices of $\mathcal{P}'$ correspond to edges

of $\mathcal{P}$, edges in the graph of $\mathcal{P}'$ correspond to 2-faces of $\mathcal{P}$. There is a natural way of mapping paths in the graph of $\mathcal{P}'$ to paths in the graph of $\mathcal{P}$. This can be easily made an inductive proof of Balinski's Theorem, as mentioned in Sec. 1 (using the Local Menger Lemma 27). We follow exactly this line of thought in our setting, except that we will not need induction – the link is a dense graph which directly yields $(n-4)$-vertex connectivity.

Note that, indeed, the following lemma implies that the complement of the link is sparse, hence the link is dense.

▶ **Lemma 29.** *The complement of $\mathsf{Lk}T$ has no cycle of length 4, i.e., if (x_0, x_1, x_2, x_3) are flippable elements in T, then there exists $i \in \{0,1,2,3\}$ such that $x_i \diamond x_{i+1 \bmod 3}$.*

Proof. Recall that all $p \in P^\circ \setminus \mathsf{V}^\circ T$ are flippable and compatible with every flippable element (Obs. 19), hence let us assume $\{x_0, x_1, x_2, x_3\} \subseteq \mathsf{V}^\circ T \cup \mathsf{E}^\circ T$. Moreover, if $p, q \in \mathsf{V}^\circ T$ are two distinct points flippable in T, then $p \diamond q$. Hence, we assume that no two consecutive elements in the cyclic sequence (x_0, x_1, x_2, x_3) are points; w.l.o.g. let $x_0 = e$ and $x_2 = f$ be edges.

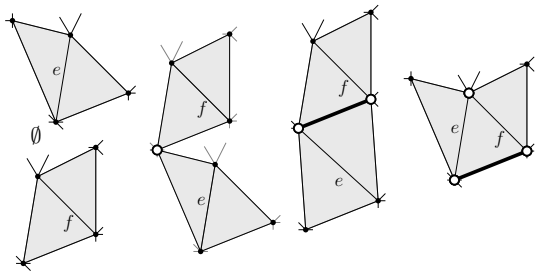


Figure 13 Intersections of boundaries of territories of two flippable edges.

For an inner edge e in a triangulation T, we define its *territory* $\mathsf{terr}\, e = \mathsf{terr}_T\, e$, as the interior of the closure of the union of the two regions in T incident to e. Obviously, e is flippable in T iff the quadrilateral $\mathsf{terr}_T\, e$ is convex. Note that for an element x to be incompatible with edge e, x must appear on the boundary of $\mathsf{terr}\, e$, and analogously elements incompatible with f must appear on the boundary of $\mathsf{terr}\, f$.

We show that there is at most one flippable element in the intersection of the boundaries of $\mathsf{terr}\, e$ and $\mathsf{terr}\, f$ (Fig. 13). This is obvious, if $\overline{\mathsf{terr}\, e} \cap \overline{\mathsf{terr}\, f}$ is empty or a single point (recall that $\overline{A}$ denotes the closure of $A \subseteq \mathbb{R}^2$). If this intersection is an edge and its two endpoints, we observe that among any edge and its two incident points, at most one element can be flippable (inner degree 3 points cannot be adjacent and cannot be incident to a flippable edge). This covers already all possibilties if $\mathsf{terr}\, e$ and $\mathsf{terr}\, f$ are disjoint (since they are convex). Finally, $\overline{\mathsf{terr}\, e} \cap \overline{\mathsf{terr}\, f}$ can be a triangle, in which case the common boundary consists of the common endpoint of e and f, clearly not flippable, and an edge with its two endpoints; again, only one of these three can be flippable. ◀

▶ **Lemma 30.** *Given a triangulation T with x and y flippable elements, $x \neq y$, every x-y-path of weight w in $\mathsf{Lk}T$ induces a T-avoiding $T[x]$-$T[y]$-path of length w in the bistellar flip graph. Interior vertex disjoint x-y-paths in the link induce interior vertex disjoint $T[x]$-$T[y]$-paths.*

Proof. Given an x-y-path in $\mathsf{Lk}T$, we replace every edge $\{z', z''\}$ on this path by $\mathsf{path}_T(z', z'') = (T[z'], \ldots, T[z''])$ (of length 2 or 3) which draws its (1 or 2) interior vertices from $\mathcal{T}_{\mathsf{part}}\langle T_{\pm z', z''}\rangle \setminus \{T[z'], T, T[z'']\}$ (Fig. 14); these vertices must have distance 2 from T in the flip graph, while $T[z']$ and $T[z'']$ have distance 1. In the resulting $T[x]$-$T[y]$-path,

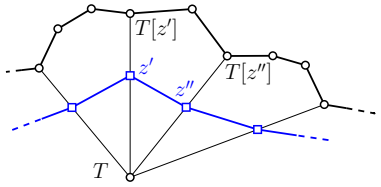


Figure 14 From a path in the link to a path in the bistellar flip graph.

all interior vertices adjacent to T (i.e., of the form $T[z]$) are distinct from interior vertices at other paths by assumption on the initial paths in the link. For vertices at distance 2, suppose $T_1 \in \mathcal{T}_{\mathsf{part}}\langle T_{\pm z_1', z_1''}\rangle$ coincides with $T_2 \in \mathcal{T}_{\mathsf{part}}\langle T_{\pm z_2', z_2''}\rangle$, both at distance 2 from T. Since $\mathsf{sl}\, T_{\pm z_1', z_1''} = \mathsf{sl}\, T_{\pm z_2', z_2''} = 2$, we have that $\mathcal{T}_{\mathsf{part}}\langle T_{\pm z_1', z_1''}\rangle \cap \mathcal{T}_{\mathsf{part}}\langle T_{\pm z_2', z_2''}\rangle$ either (a) equals $\{T\}$, (b) equals $\{T, T[z]\}$ for some z, or (c) $T_{\pm z_1', z_1''} = T_{\pm z_2', z_2''}$ (Lemma 17). In (a-b) $T_{\pm z_1', z_1''}$ and $T_{\pm z_2', z_2''}$ cannot possibly share a vertex at distance 2 from T. Thus (c) holds. $T_{\pm z_1', z_1''} = T_{\pm z_2', z_2''}$ implies $\{z_1', z_1''\} = \{z_2', z_2''\}$. ◀

▶ **Lemma 31.** *The link $\mathsf{Lk}T$ satisfies:* (i) *There are at least $n-3$ vertices.* (ii) *Every vertex has degree at least $n-4$.* (iii) *Every pair of non-adjacent vertices has at least $n-4$ connecting interior vertex disjoint paths (all of length at most 3).* (iv) *It is $(n-4)$-vertex connected.*

Proof. (i) x is a vertex in $\mathsf{Lk}T$ iff x is flippable in T iff $T_{\pm x}$ is a subdivision of slack 1, a perfect coarsening of T. Lemma 25 ensures the existence of at least $n-3$ such perfect coarsenings.

(ii) Let x be a vertex of $\mathsf{Lk}T$. $T_{\pm x}$, a subdivision of slack 1, has at least $n-4$ perfect coarsenings of slack 2 (Lemma 25). Each such coarsening equals $T_{\pm x, y}$ for some $y \diamond x$, i.e., y is a neighbor of x in $\mathsf{Lk}T$.

(iii) Let x and y be non-adjacent vertices of $\mathsf{Lk}T$, i.e., $x \not\diamond y$. Let $z_1, z_2, \ldots, z_\ell$ be all flippable elements in T compatible with both x and y. Each such element z_i gives rise to a path (x, z_i, y) of length 2 in the link. If $\ell \geq n-4$, we are done. Otherwise, there is an extra supply of elements $x_1, x_2, \ldots, x_{n-4-\ell}$ compatible with x but not with y, and elements $y_1, y_2, \ldots, y_{n-4-\ell}$ compatible with y but not with x. For all $i = 1, 2, \ldots, n-4-\ell$, $y_i \not\diamond x$, $x \not\diamond y$, and $y \not\diamond x_i$. By Lemma 29, $x_i \diamond y_i$, hence we have a path (x, x_i, y_i, y) of length 3 in the link. Obviously, these paths of length 3 are pairwise internally vertex disjoint, and also internally vertex disjoint from all x-y-paths of length 2 (interior vertices on length 2 paths are connected to x and y, interior vertices on the constructed length 3 paths are not).

(iv) We apply the Local Menger Lemma 27. Indeed, $\mathsf{Lk}T$ has at least $(n-4)+1 = n-3$ vertices (see (i)), and every pair of vertices at distance 2 has at least $n-4$ internally vertex disjoint paths (see (iii)). Hence, the link is $(n-4)$-vertex connected. ◀

$(n-3)$-Connectivity of the Bistellar Flip Graph – Proof of Thm. 4

Proof of Thm. 4. If $n \leq 4$, $(n-3)$-vertex connectivity can be easily checked according to the definition of k-vertex connectivity. For $n \geq 5$, we employ the Local Menger Lemma 27. Thus (apart from the presence of at least $n-2$ vertices), we have to show that for any triangulation T and flippable elements x and y, there are at least $n-3$ internally vertex disjoint $T[x]$-$T[y]$-paths in the bistellar flip graph. We know that in $\mathsf{Lk}T$ has at least $n-4$ internally vertex disjoint x-y-paths (Menger's Theorem, [2, 6]). Therefore, there are at least $n-4$ interior vertex disjoint $T[x]$-$T[y]$-paths disjoint from T (Lemma 30). Together with the path $(T[x], T, T[y])$, the claim is established. ◀

References

1 M. L. Balinski. On the graph structure of convex polyhedra in n-space. *Pacific J. Math.*, 11(2):431–434, 1961. URL: https://projecteuclid.org:443/euclid.pjm/1103037323.

2 Béla Bollobás. *Modern graph theory (Graduate texts in mathematics)*. New York: Springer, 1998.

3 Cesar Ceballos, Francisco Santos, and Günter M. Ziegler. Many non-equivalent realizations of the associahedron. *Combinatorica*, 35(5):513–551, October 2015. doi:10.1007/s00493-014-2959-9.

4 Jesús A De Loera, Jörg Rambau, and Francisco Santos. *Triangulations Structures for algorithms and applications*. Springer, 2010.

5 Jesús A. De Loera, Francisco Santos, and Jorge Urrutia. The number of geometric bistellar neighbors of a triangulation. *Discrete & Computational Geometry*, 21(1):131–142, 1999. doi:10.1007/PL00009405.

6 Reinhard Diestel. *Graph theory*. Springer, 1997.

7 Israel M. Gelfand, Mikhail M. Kapranov, and Andrei V. Zelevinsky. Newton polyhedra and principal A-determinants. *Soviet Math. Dokl.*, 40:278–281, 1990.

8 Charles L. Lawson. Transforming triangulations. *Discrete Math.*, 3(4):365–372, January 1972. doi:10.1016/0012-365X(72)90093-3.

9 Carl W. Lee and Francisco Santos. Subdivisions and triangulations of polytopes. In Csaba D. Toth, Jacob E. Goodman, and Joseph O'Rourke, editors, *Handbook of Discrete and Computational Geometry, 3rd Edition*, pages 415–477. Chapman and Hall/CRC, New York, USA, 2017.

10 Anna Lubiw, Zuzana Masárová, and Uli Wagner. A proof of the orbit conjecture for flipping edge-labelled triangulations. *Discrete & Computational Geometry*, 61(4):880–898, June 2019. doi:10.1007/s00454-018-0035-8.

11 David Orden and Francisco Santos. The polytope of non-crossing graphs on a planar point set. *Discrete & Computational Geometry*, 33(2):275–305, February 2005. doi:10.1007/s00454-004-1143-1.

12 Francisco Santos Leal, 2019. personal communication.

13 Uli Wagner and Emo Welzl. Connectivity of triangulation flip graphs in the plane. *CoRR*, abs/2003.13557, 2020. arXiv:2003.13557.

14 Uli Wagner and Emo Welzl. Connectivity of triangulation flip graphs in the plane (Part I: Edge flips). In *Proceedings of the 31st Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2823–2841. ACM-SIAM, 2020.

On the Planar Two-Center Problem and Circular Hulls

Haitao Wang
Department of Computer Science, Utah State University, Logan, UT 84322, USA
haitao.wang@usu.edu

Abstract

Given a set S of n points in the Euclidean plane, the two-center problem is to find two congruent disks of smallest radius whose union covers all points of S. Previously, Eppstein [SODA'97] gave a randomized algorithm of $O(n \log^2 n)$ expected time and Chan [CGTA'99] presented a deterministic algorithm of $O(n \log^2 n \log^2 \log n)$ time. In this paper, we propose an $O(n \log^2 n)$ time deterministic algorithm, which improves Chan's deterministic algorithm and matches the randomized bound of Eppstein. If S is in convex position, we solve the problem in $O(n \log n \log \log n)$ deterministic time. Our results rely on new techniques for dynamically maintaining circular hulls under point insertions and deletions, which are of independent interest.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms; Theory of computation → Computational geometry

Keywords and phrases two-center, disk coverage, circular hulls, dynamic data structures

Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.68

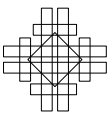
Related Version A full version of this paper is available at `https://arxiv.org/abs/2002.07945`.

1 Introduction

Given a set S of n points in the Euclidean plane, we consider the planar 2-center problem that is to find two congruent disks of smallest radius whose union covers all points of S.

The classical 1-center problem for a set of points is to find the smallest disk covering all points, and the problem can be solved in linear time in any fixed dimensional space [9, 13, 24]. As a natural generalization, the 2-center problem has attracted much attention. Hershberger and Suri [19] first solved the decision version of the problem in $O(n^2 \log n)$ time, which was later improved to $O(n^2)$ time [18]. Using this result and parametric search [23], Agarwal and Sharir [2] gave an $O(n^2 \log^3 n)$ time algorithm for the 2-center problem. Katz and Sharir [21] achieved the same running time by using expanders instead of parametric search. Eppstein [15] presented a randomized algorithm of $O(n^2 \log^2 n \log \log n)$ expected time. Later, Jaromczyk and Kowaluk [20] proposed an $O(n^2)$ time algorithm. A breakthrough was achieved by Sharir [26], who proposed the first subquadratic algorithm for the problem, and the running time is $O(n \log^9 n)$. Afterwards, following Sharir's algorithmic scheme, Eppstein [16] derived a randomized algorithm of $O(n \log^2 n)$ expected time, and then Chan [6] developed an $O(n \log^2 n \log^2 \log n)$ time deterministic algorithm and a randomized algorithm of $O(n \log^2 n)$ time with high probability. Recently, Tan and Jiang [27] proposed a simple algorithm of $O(n \log^2 n)$ time based on binary search, but unfortunately, the algorithm is not correct (see the full paper for details). The problem has an $\Omega(n \log n)$ time lower bound in the algebraic decision tree model [16], by a reduction from the max-gap problem.

In this paper, we present a new deterministic algorithm of $O(n \log^2 n)$ time, which improves the $O(n \log^2 n \log^2 \log n)$ time deterministic algorithm by Chan [6] and matches the randomized bound of $O(n \log^2 n)$ [6, 16]. This is the first progress on the problem since Chan's work [6] was published twenty years ago. Further, if S is in convex position (i.e., the points




of S are all on the convex hull of S), our technique solves the problem in $O(n \log n \log\log n)$ time. Previously, Kim and Shin [22] announced an $O(n \log^2 n)$ time algorithm for this convex position case, but Tan and Jiang [27] found errors in their time analysis.

Some variations of the 2-center problem have also been considered in the literature. Agarwal et al. [3] studied the discrete 2-center problem where the centers of the two disks must be in S, and they solved the problem in $O(n^{4/3} \log^5 n)$ time. Agarwal and Phillips [1] considered an outlier version of the problem where k points of S are allowed to be outside the two disks, and they presented a randomized algorithm of $O(nk^7 \log^3 n)$ expected time. In addition to the set S, the problem of Halperin et al. [17] also involves a set of pairwise disjoint simple polygons, and the centers of the two disks are required to lie outside all polygons. Both exact and approximation algorithms are given in [17]. Arkin et al. [4] studied a bichromatic 2-center problem for a set of n pairs of points in the plane, and the goal is to assign a red color to a point and a blue color to the other point for every pair, such that $\max\{r_1, r_2\}$ is minimized, where r_1 (resp., r_2) is the radius of the smallest disk covering all red (resp., blue) points. Arkin et al. [4] gave an $O(n^3 \log^2 n)$ time algorithm, which was recently improved to $O(n^2 \log^2 n)$ time by Wang and Xue [28].

1.1 Our techniques

Let D_1^* and D_2^* be two congruent disks in an optimal solution such that the distance of their centers is minimized. Let r^* be their radius and δ^* the distance of their centers. If $\delta^* \geq r^*$, we call it the *distant case*; otherwise, it is the *nearby case*.

Eppstein [16] already solved the distant case in $O(n \log^2 n)$ deterministic time. Solving the nearby case turns out to be the bottleneck in all previous three sub-quadratic time algorithms [6, 16, 26]. Specifically, Sharir [26] first solved it in $O(n \log^9 n)$ deterministic time. Eppstein [16] gave a randomized algorithm of $O(n \log n \log\log n)$ expected time. Chan [16] proposed a randomized algorithm of $O(n \log n)$ time with high probability and a deterministic algorithm of $O(n \log^2 n \log^2 \log n)$ time. Our contribution is an $O(n \log n \log\log n)$ time deterministic algorithm for the nearby case, which improves Chan's algorithm by a factor of $\log n \log\log n$. Combining with the $O(n \log^2 n)$ time deterministic algorithm of Eppstein [16] for the distant case, the 2-center problem can now be solved in $O(n \log^2 n)$ deterministic time. Interestingly, solving the distant case now becomes the bottleneck of the problem.

Our algorithm (for the nearby case) is based on Chan's framework [6]. Our improvement is twofold. First, Chan [6] derived an $O(n \log n)$ time algorithm for the *decision problem*, i.e., given r, decide whether $r^* \leq r$. We improve the algorithm to $O(n)$ time, after $O(n \log n)$ time preprocessing. Second, Chan [6] solved the optimization problem (i.e., the original 2-center problem) by parametric search. To this end, Chan developed a parallel algorithm for the decision problem and the algorithm runs in $O(\log n \log^2 \log n)$ parallel steps using $O(n \log n)$ processors. By applying Cole's parametric search [10] and using his $O(n \log n)$ time decision algorithm, Chan solved the optimization problem in $O(n \log^2 n \log^2 \log n)$ time. We first notice that simply replacing Chan's $O(n \log n)$ time decision algorithm with our new $O(n)$ time algorithm does not lead to any improvement. Indeed, in Chan's parallel algorithm, the number of processors times the number of parallel steps is $O(n \log^2 n \log^2 \log n)$. We further design another parallel algorithm for the decision problem, which runs in $O(\log n \log\log n)$ parallel steps using $O(n)$ processors. Consequently, by applying Cole's parametric search with our $O(n)$ time decision algorithm, we solve the optimization problem in $O(n \log n \log\log n)$ time. Note that although Cole's parametric search is used, our algorithm mainly involves independent binary searches and no sorting networks are needed.

In addition, we show that our algorithm can be easily applied to solving the convex position case in $O(n \log n \log\log n)$ time.

Circular hulls. To obtain our algorithm for the decision problem, we develop new techniques for *circular hulls* [19] (also known as α-*hulls* with $\alpha = 1$ [14]). A circular hull of radius r for a set Q of points is the common intersection of all disks of radius r containing Q (to see how circular hulls are related to the two-center problem, notice that there exists a disk of radius r covering all points of Q if and only if the circular hull of Q of radius r exists). Although circular hulls have been studied before, our result needs more efficient algorithms for certain operations. For example, two algorithms [14, 19] were known for constructing the circular hull for a set of n points; both algorithms run in $O(n \log n)$ time, even if the points are given sorted. We instead present a linear-time algorithm once the points are sorted. Also, Hershberger and Suri [19] gave a linear-time algorithm to find the common tangents of two circular hulls separated by a line, and we design a new algorithm of $O(\log n)$ time. We also need to maintain a dynamic circular hull for a set of points under point insertions and deletions. Hershberger and Suri [19] gave a semi-dynamic data structure that can support deletions in $O(\log n)$ amortized time each. In our problem, we need to handle both insertions and deletions but with the following special properties: the point in each insertion must be to the right of all points of Q and the point in each deletion must be the leftmost point of Q. Our data structure can handle each update in $O(1)$ amortized time (which leads to the linear time decision algorithm for the 2-center problem[1]). We believe that these results on circular hulls are interesting in their own right.

Outline. We introduce notation and review some previous work in Section 2. In Section 3, we present our decision algorithm, and the algorithm needs a data structure to maintain circular hulls dynamically, which is given in the full paper. Section 4 solves the optimization problem. The convex position case is discussed in Section 5.

2 Preliminaries

We begin with some notation. It suffices to solve the nearby case. Thus, we assume $\delta^* < r^*$ in the rest of the paper. In the nearby case, it is possible to find in $O(n)$ time a constant number of points such that at least one of them, denoted by o, is in $D_1^* \cap D_2^*$ [16]. We assume that o is the origin of the plane. We make a general position assumption: no two points of S are collinear with o and no two points of S have the same x-coordinate. This assumption does not affect the running time of the algorithm, but simplifies the presentation.

For any set P of points in the plane, let $\tau(P)$ denote the radius of the smallest enclosing disk of P. For a connected region B in the plane, let ∂B denote the boundary of B.

The boundaries of the two disks D_1^* and D_2^* have exactly two intersections, and let ρ_1 and ρ_2 be the two rays through these intersections emanating from o (e.g., see Fig. 1). As argued in [6], one of the two coordinate axes must separate ρ_1 and ρ_2 since the angle between the two rays lies in $[\pi/2, 3\pi/2]$, and without loss of generality, we assume it is the x-axis.

Let S^+ denote the subset of points of S above the x-axis, and $S^- = S \setminus S^+$. For notational simplicity, let $|S^+| = |S^-| = n$. Let $p_1, p_2, \ldots, p_n$ be the sorted list of S^+ counterclockwise around o, and $q_1, q_2, \ldots, q_n$ the sorted list of S^- also counterclockwise around o (e.g., see Fig. 2). For each $i = 0, 1, \ldots, n$ and $j = 0, 1, \ldots, n$, define $L_{ij} = \{p_{i+1} \ldots, p_n, q_1, \ldots, q_j\}$ and $R_{ij} = \{q_{j+1}, \ldots, q_n, p_1, \ldots, p_i\}$. Note that if $i = n$, then $L_{ij} = \{q_1, \ldots, q_j\}$, and if $j = n$, then $R_{ij} = \{p_1, \ldots, p_i\}$. In words, if we consider a ray emanating from o and between p_i and p_{i+1}, and another ray emanating from o and between q_j and q_{j+1}, then L_{ij} (resp., R_{ij}) consist of all points to the left (resp., right) of the two rays (e.g., see Fig. 2).

[1] As will be clear later, the points involved in our dynamic circular hull problem are actually sorted radially by a point; we can extend the result for the left-right sorted case to the radically sorted case.

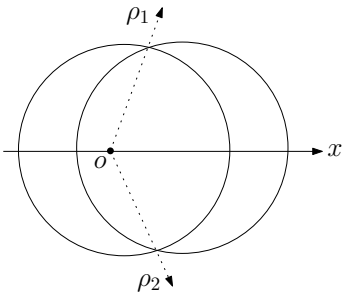


Figure 1 Illustrating the nearby case.

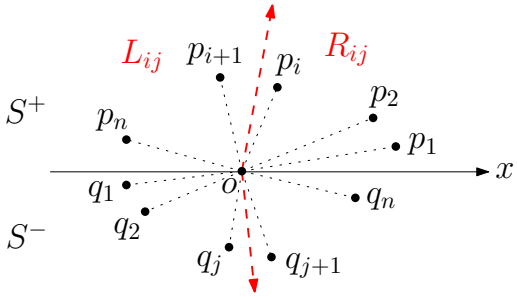


Figure 2 Illustrating the points of S^+ and S^-.

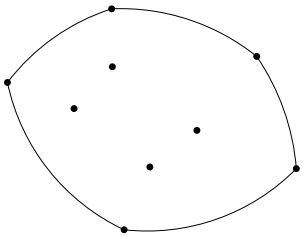


Figure 3 The circular hull of a set of points.

Note that the partition of S by the two rays $\rho_1 \cup \rho_2$ is $\{L_{ij}, R_{ij}\}$ for some i and j, and thus $r^* = \max\{\tau(L_{ij}), \tau(R_{ij})\}$. Define $A[i,j] = \tau(L_{ij})$ and $B[i,j] = \tau(R_{ij})$, for all $0 \leq i, j \leq n$. Then, $r^* = \min_{0 \leq i,j \leq n} \max\{A[i,j], B[i,j]\}$. If we consider A and B as $(n+1) \times (n+1)$ matrices, then each row of A (resp., B) is monotonically increasing (resp., decreasing) and each column of A (resp., B) is monotonically decreasing (resp., increasing). For each $i \in [0, n]$, define $r_i^* = \min_{0 \leq j \leq n} \max\{A[i,j], B[i,j]\}$. Thus, $r^* = \min_{0 \leq i \leq n} r_i^*$.

2.1 Circular hulls

For any point c in the plane and a value r, we use $D_r(c)$ to denote the disk centered at c with radius r. For a set Q of points in the plane, define $\mathcal{I}_r(Q) = \bigcap_{c \in Q} D_r(c)$, i.e., the common intersection of the disks $D_r(c)$ for all points $c \in Q$. Note that $\mathcal{I}_r(Q)$ is convex. A dual concept of $\mathcal{I}_r(Q)$ is the *circular hull* [19] (also known as α-*hull* with $\alpha = 1$ [14]; e.g., see Fig 3), denoted by $\alpha_r(Q)$, which is the common intersection of all disks of radius r containing Q. $\alpha_r(Q)$ is convex and unique. The vertices of $\alpha_r(Q)$ is a subset of Q and the edges are arcs of circles of radius r. $\mathcal{I}_r(Q)$ and $\alpha_r(Q)$ are dual to each other: Every arc of $\alpha_r(Q)$ is on the circle of radius r centered at a vertex of $\mathcal{I}_r(Q)$ and every arc of $\mathcal{I}_r(Q)$ is on the circle of radius r centered at a vertex of $\alpha_r(Q)$. Note that $\alpha_r(Q)$ exists if and only if $\mathcal{I}_r(Q) \neq \emptyset$, which is true if and only $\tau(Q) \leq r$. For brevity, we often drop the subscript r from $\mathcal{I}_r(Q)$ and $\alpha_r(Q)$ if it is clear from the context.

Circular hulls are critical to our algorithm. As discussed in [19], circular hulls have many properties similar to convex hulls. However, circular hulls also have special properties that convex hulls do not possess. For example, the circular hull for a set of points may not exist. Also, the leftmost point of a set Q of points must be a vertex of the convex hull of Q, but this may not be the case for the circular hull. Due to these special properties, extending algorithms on convex hulls to circular hulls sometimes is not trivial, as will be seen later. In the following, we introduce some concepts on circular hulls that will be needed later.

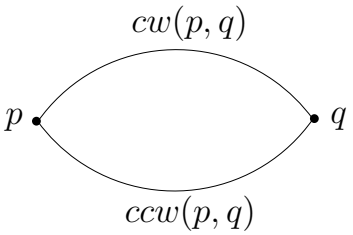


Figure 4 Illustrating two minor arcs of p and q.

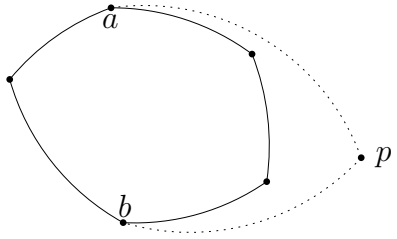


Figure 5 Illustrating the two tangents from p to $\alpha(Q)$: $cw(a, p)$ and $ccw(b, p)$.

We assume that $r = 1$ and thus a disk of radius r is a *unit disk* (whose boundary is a *unit circle*). We use $\alpha(Q)$ to refer to $\alpha_r(Q)$. We assume that $\alpha(Q)$ exists.

For any arc of a circle, the circle is called the *supporting circle* of the arc, and the disk enclosed in the circle is called the *supporting disk* of the arc. If a disk D contains all points of a set P, we say D *covers* P. We say that a set P of points in the plane is *unit disk coverable* if there is a unit disk containing all points of P, which is true if and only if $\alpha(P)$ exists.

Consider two points p and q that are unit disk coverable. There must be a unit circle with p and q on it, and we call the arc of the circle subtending an angle of at most 180° a *minor arc* [19]. Note that there are two minor arcs connecting p and q, we use $cw(p, q)$ to refer to the one clockwise around the center of the supporting circle of the arc from p to q, and use $ccw(p, q)$ to refer to the other one (e.g., see Fig. 4). Note that $cw(p, q) = ccw(q, p)$ and $ccw(p, q) = cw(q, p)$. For any minor arc w, we use $D(w)$ to denote the supporting disk of w, i.e., the disk whose boundary contains w. Note that all arcs of $\alpha(Q)$ are minor arcs. We make a general position assumption that no point of Q is on a minor arc of two other points of Q. The following observation has already been discovered previously [14, 19].

▶ **Observation 1** ([14, 19]).
1. *A point p of Q is a vertex of $\alpha(Q)$ iff there is a unit disk covering Q and with p on the boundary.*
2. *A minor arc connecting two points of Q is an arc of $\alpha(Q)$ iff its supporting disk covers Q.*
3. *$\alpha(Q)$ is the common intersection of the supporting disks of all arcs of $\alpha(Q)$.*
4. *A unit disk covers Q iff it contains $\alpha(Q)$.*
5. *For any subset Q' of Q, $\alpha(Q') \subseteq \alpha(Q)$.*

For any vertex v of $\alpha(Q)$, we refer to the clockwise neighboring vertex of v on $\alpha(Q)$ the *clockwise neighbor* of v, and the *counterclockwise* neighbor is defined analogously. We use $cw(v)$ and $ccw(v)$ to denote v's clockwise and counterclockwise neighbors, respectively.

Tangents. Consider a vertex v in the circular hull $\alpha(Q)$. Consider the arc $cw(ccw(v), v)$ of $\alpha(Q)$. Let D be the disk $D(cw(ccw(v), v))$. By Observation 1(2) and (4), D contains $\alpha(Q)$. Observe that if we rotate D around v clockwise until ∂D contains the arc $cw(v, cw(v))$, D always contains $\alpha(Q)$, and in fact, this continuum of disks D are the only unit disks that contain $\alpha(Q)$ and have v on the boundaries. For each of such disk D, we say that D (and any part of ∂D containing v) is *tangent* to $\alpha(Q)$ at v. We have the following observation.

▶ **Observation 2.** *A unit disk D that contains a vertex v of $\alpha(Q)$ on its boundary is tangent to $\alpha(Q)$ at v if and only if D contains both $cw(v)$ and $ccw(v)$.*

Let p be a point outside $\alpha(Q)$. If there is a vertex a on $\alpha(Q)$ such that $D(cw(a, p))$ is tangent to $\alpha(Q)$ at a, then the arc $cw(a, p)$ is an *upper tangent* from p to $\alpha(Q)$; e.g., see Fig 5. If there is a vertex b on $\alpha(Q)$ such that $D(ccw(b, p))$ is tangent to $\alpha(Q)$ at b, then

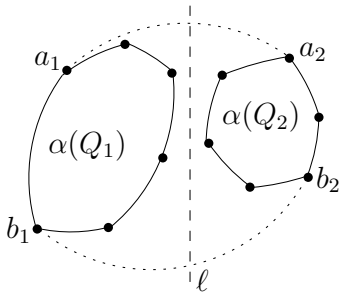


Figure 6 Illustrating the upper common tangent $cw(a_1, a_2)$ and the lower common tangent $ccw(b_1, b_2)$ of $\alpha(Q_1)$ and $\alpha(Q_2)$.

the arc $ccw(b, p)$ is a *lower tangent* from p to $\alpha(Q)$. By replacing the arcs of $\alpha(Q)$ clockwise from a to b with the two tangents from p, we obtain $\alpha(Q \cup \{p\})$. This also shows that p has tangents to $\alpha(Q)$ if and only if $Q \cup \{p\}$ is unit disk coverable and p is outside $\alpha(Q)$. Note that $a = b$ is possible, in which case $\alpha(Q \cup \{p\}) = \alpha(\{a, p\})$.

Common tangents of two circular hulls. Let Q_1 and Q_2 be two sets of points in the plane such that all points of Q_1 (resp., Q_2) are to the left (resp., right) of a vertical line ℓ. Let $Q = Q_1 \cup Q_2$. A unit disk D that is tangent to $\alpha(Q_1)$, say at a vertex a, and is also tangent to $\alpha(Q_2)$, say at a vertex b, is said to be *commonly tangent* to $\alpha(Q_1)$ and $\alpha(Q_2)$. The minor arc of D connecting a and b is called a *common tangent* of the two circular hulls. It is an *upper* (resp, *lower*) tangent if it is clockwise (resp., counterclockwise) from a to b along the minor arc (e.g., see Fig. 6). The common tangents of $\alpha(Q_1)$ and $\alpha(Q_2)$ may not exist. Indeed, if $\alpha(Q)$ does not exist, then the common tangents do not exist. Otherwise the common tangents do not exist either if all points of Q_2 are contained in $\alpha(Q_1)$, which happens only if Q_2 is covered by $D(w)$ for the rightmost arc w of $\alpha(Q_1)$ and we call it the Q_1-*dominating case*, or if all points of Q_1 are contained in $\alpha(Q_2)$, which happens only if Q_1 is covered by $D(w')$ for the leftmost arc w' of $\alpha(Q_2)$ and we call it the Q_2-*dominating case*. If none of the above cases happens, then there are exactly two common tangents between the two hulls. Each tangent intersects the vertical line ℓ, which separates Q_1 and Q_2, and the upper tangent intersects ℓ higher than the lower tangent does.

3 The decision problem

This section is concerned with the decision problem: Given a value r, decide whether $r^* \leq r$. Previously, Chan [6] solved the problem in $O(n \log n)$ time (Chan actually considered a slightly different problem: decide whether $r^* < r$, but the idea is similar). We present an $O(n)$ time algorithm, after $O(n \log n)$ time preprocessing to sort all points of S^+ and S^- to obtain the sorted lists $p_1, \ldots, p_n$ and $q_1, \ldots, q_n$.

Given r, we use the following algorithmic framework in Algorithm 1 from [6] (see Theorem 3.3), which can decide whether $r^* \leq r$, and if yes, report all indices i with $r_i^* \leq r$.

The algorithm is simple, but the technical crux is in how to decide whether $A[i, j+1] \leq r$ and whether $B[i, j] \leq r$. Chan [6] built a data structure in $O(n \log n)$ time so that each of these two steps can be done in $O(\log n)$ time, which leads to an overall $O(n \log n)$ time for his decision algorithm. Our innovation is a new data structure that can perform each of the two steps in $O(1)$ amortized time, resulting in an $O(n)$ time algorithm. Our idea is motivated by the following observation.

Algorithm 1 The decision algorithm of Chan [6].

```
1 j ← −1;
2 for i ← 0 to n do
3     while A[i, j + 1] ≤ r do j + + ;
4     if B[i, j] ≤ r then report i ;
5 end
```

▶ **Observation 3.** *All such elements $A[i, j + 1]$ that are checked in the algorithms (i.e., Line 3) are in a path of the matrix A from $A[0, 0]$ to an element in the bottom row and the path only goes rightwards or downwards. The same holds for the elements of B that are checked in the algorithms (i.e., Line 4).*

We call such a path in A as specified in the observation a *monotone path*, which has at most $2(n + 1)$ elements of A. We show that we can determine in $O(n)$ time whether $A[i, j] \leq r$ for all elements $A[i, j]$ in a monotone path of A. The same algorithm works for B as well.

Let π be a monotone path of A, starting from $A[0, 0]$. Consider any element $A[i, j]$ on π. Recall that $A[i, j] = \tau(L_{ij})$. The next value of π after $A[i, j]$ is either $A[i, j + 1]$ or $A[i + 1, j]$, i.e., either $\tau(L_{i,j+1})$ or $\tau(L_{i+1,j})$. Note that $L_{i,j+1}$ can be obtained from L_{ij} by inserting q_{j+1} and $L_{i+1,j}$ can be obtained from L_{ij} by deleting p_{i+1}. Because the points $p_1, p_2, \ldots, p_n, q_1, q_2 \ldots, q_n$ are ordered around o counterclockwise, our problem becomes the following. Maintain a sublist Q of the above sorted list of S, with $Q = S^+$ initially, to determine whether $\tau(Q) \leq r$ (or equivalently whether $\alpha_r(Q)$ exists) under deletions and insertions, such that a deletion operation deletes the first point of Q and an insertion operation inserts the point of S following the last point of Q. Further, deletions only happen to points of S^+ (i.e., once p_n is deleted from Q, no deletions will happen). We refer to the problem as the *dynamic circular hull problem*. In the full paper we solve the problem in $O(n)$ time, i.e., each update takes $O(1)$ amortized time. This leads to the following result.

▶ **Theorem 4.** *Assume that points of S are sorted cyclically around o. Given any r, whether $r^* \leq r$ can be decided in $O(n)$ time.*

▶ Remark. For the nearby case, Chan proposed (in Theorem 3.4 [16]) a randomized algorithm of $O(n \log n)$ time with high probability (i.e., $1 - 2^{-\Omega(n/\log^{12} n)}$) by using his $O(n \log n)$ time decision algorithm. Applying our decision algorithm and following Chan's algorithm (specifically, setting m to $\lfloor n/\log^7 n \rfloor$ instead of $\lfloor n/\log^6 n \rfloor$ in the algorithm of Theorem 3.4 in [16]), we can obtain the following result: After $O(n \log n)$ deterministic time preprocessing, we can compute r^* for the nearby case in $O(n)$ time with high probability (i.e., $1 - 2^{-\Omega(n/\log^{14} n)}$).

4 The optimization problem

With Theorem 4, we solve the optimization problem by parametric search [10, 23]. As Chan's algorithm [6], because our decision algorithm is inherently sequential, we need to design a parallel decision algorithm. Chan [6] gave a parallel decision algorithm that runs in $O(\log n \log^2 \log n)$ parallel steps using $O(n \log n)$ processors. Consequently, by using his $O(n \log n)$ time decision algorithm and applying Cole's parametric search [10], Chan [6] solved the optimization problem in $O(n \log^2 n \log^2 \log n)$ time. By following Chan's algorithmic scheme, we develop a new parallel decision algorithm that runs in $O(\log n \log \log n)$ parallel steps using $O(n)$ processors. Then, with the serial decision algorithm in Theorem 4 and applying Cole's parametric search [10] on our new parallel decision algorithm, we solve the optimization problem in $O(n \log n \log \log n)$ time.

Our algorithm relies on Lemma 5, whose proof is quite independent of the remainder of this section and is given in the full paper. Note that Hershberger and Suri [19] gave a linear-time algorithm to achieve the same result as Lemma 5, which suffices for their purpose.

▶ **Lemma 5.** *Given the circular hull (with respect to a radius r) of a set L of points and the circular hull of another set R of points such that the points of L and R are separated by a line, one can do the following in $O(\log(|L|+|R|))$ time (assuming that the vertices of each circular hull are stored in a data structure that supports binary search): determine whether the circular hull of $L \cup R$ (with respect to r) exists; if yes, either determine which dominating case happens (i.e., all points of a set are contained in the circular hull of the other set) or compute the two common tangents between the circular hulls of L and R.*

For any $0 \le i \le j \le n$, let $S^+[i,j] = \{p_i, p_{i+1}, \ldots, p_j\}$ and $S^-[i,j] = \{q_i, q_{i+1}, \ldots, q_j\}$. By using Lemma 5, we have the following lemma.

▶ **Lemma 6.** *We can preprocess S and compute an interval $(r_1, r_2]$ containing r^* in $O(n \log n)$ time so that given any $r \in (r_1, r_2)$ and any pair (i,j) with $1 \le i \le j \le n$, we can determine whether $\alpha_r(S^+[i,j])$ (resp., $\alpha_r(S^-[i,j])$) exists, and if yes, return the root of a balanced binary search tree representing the circular hull, in $O(\log k \log\log k)$ parallel steps using $O(\log k)$ processors, or in $O(\log^2 k)$ time using one processor, where $k = j - i + 1$.*

Proof. As in [6, 16], we use the following geometric transformation. For any point $p = (a,b)$, let $h(p)$ denote the halfspace $\{(x,y,z) : z \ge a^2 + b^2 - 2ax - aby\}$. Then, for any set P of points in the plane, $(\tau(P))^2$ is the minimum of $x^2 + y^2 + z$ over all points (x,y,z) in the polyhedron $\mathcal{H}(P) = \bigcap_{p \in P} h(p)$.

Preprocessing. We build a complete binary search tree T^+ on the set $S^+ = \{p_1, p_2, \ldots, p_n\}$ such that the leaves of T^+ from left to right storing the points of S^+ in their index order. Each internal node v of T^+ stores a hierarchical representation [11] of the polyhedron $\mathcal{H}(P)$, where P is the set of points stored in the leaves of the subtree rooted at v (P is called a *canonical subset*). Computing the polyhedrons of all internal nodes of T^+ can be done in $O(n \log n)$ time in a bottom-up manner using linear time polyhedra intersection algorithms [7, 8]. Similarly, we build a tree T^- on the set $S^- = \{q_1, q_2, \ldots, q_n\}$.

Consider a vertex $v = (x,y,z)$ of $\mathcal{H}(P)$ for a canonical subset P of T^+. Define $r(v) = \sqrt{x^2 + y^2 + z}$. Let C be the set of the values $r(v)$ of all vertices v of $\mathcal{H}(P)$ for all canonical subsets P of T^+. Note that $|C| = O(n \log n)$. We find the smallest value $r(v) \in C$ such that $r^* \le r(v)$, and let r_2 denote such $r(v)$. The value r_2 can be found in $O(n \log n)$ using our linear time decision algorithm and doing binary search on C using the linear time selection algorithm [5]. Next, we find the largest value in C that is smaller than r_2, and let r_1 denote that value. By definition, $r^* \in (r_1, r_2]$ and (r_1, r_2) does not contain any element of C.

Consider a canonical subset P of T^+ and any $r \in (r_1, r_2)$. We construct $\mathcal{I}_r(P)$ for each canonical subset P of T^+ by intersecting the facets of $\mathcal{H}(P)$ with the paraboloid $W(r) = \{(x,y,z) : x^2 + y^2 + z = r^2\}$ and projecting them vertically to the xy-plane. By the definitions of r_1 and r_2, the paraboloid $W(r)$ intersects the same set of edges of $\mathcal{H}(P)$ for all $r \in (r_1, r_2)$; this implies that $\mathcal{I}_r(P)$ is combinatorially the same for all $r \in (r_1, r_2)$. Hence, we can consider $\alpha_r(P)$, which is the dual of $\mathcal{I}_r(P)$, as a parameterized circular hull of P. We store the (parameterized) vertices of $\alpha_r(P)$ in a balanced binary search tree. Since $\mathcal{H}(P)$ is convex, we can obtain $\mathcal{I}_r(P)$ and thus the balanced binary search tree for $\alpha_r(P)$ in $O(|P|)$ time; we associate the tree at the node of T^+ for P. Because the total size of $\mathcal{H}(P)$ for all canonical subsets P in T^+ is $O(n \log n)$, we can obtain the balanced binary search trees for $\alpha_r(P)$ of all canonical subsets P in T^+ in $O(n \log n)$ time.

We do the same for T^- as above, which will obtain two values r_1' and r_2' correspondingly as above r_1 and r_2. We update $r_1 = \max\{r_1, r_1'\}$ and $r_2 = \min\{r_2, r_2'\}$; so $r^* \in (r_1, r_2]$ still holds. This finishes our processing on S, which takes $O(n \log n)$ time and is independent of r.

Queries. Given any $r \in (r_1, r_2)$ and any pair (i, j) with $i < j$, we determine whether $\alpha_r(S^+[i, j])$ exists, and if yes, return the root of a balanced binary search tree representing it, as follows (the case for $S^-[i, j]$ is similar). Let $k = j - i + 1$ and let $P = S^+[i, j]$.

By the standard method, we first find $O(\log k)$ canonical subsets of T^+ whose union is exactly $S^+[i, j]$. Our following computation procedure can be described as a complete binary tree T where the leaves corresponding to the above $O(\log k)$ canonical subsets. So T has $O(\log k)$ leaves, and its height is $O(\log \log k)$. For each leave of T, its circular hull is already available due to the preprocessing. For each internal node v that is the parent of two leaves, we compute the circular hull of the union of the two subsets P_1 and P_2 of the two leaves. As the points of S^+ are ordered radially by o, the two subsets are separated by a line through o. Hence, we can find the common tangents (if exist) using Lemma 5 in $O(\log k)$ time because the size of each subset is no more than k. Recall that the circular hull of each canonical subset is represented by a balanced binary search tree. After having the common tangents, we split and merge the two balanced binary search trees to obtain a balanced binary search tree for $\alpha_r(P_1 \cup P_2)$. In addition, we keep unaltered the two original trees for $\alpha_r(P_1)$ and $\alpha_r(P_2)$ respectively, and this can be done by using persistent data structures (e.g., using the copy-path technique [12, 25]) in $O(\log k)$ time. In this way, the original trees for $\alpha_r(P_1)$ and $\alpha_r(P_2)$ can be used in parallel for other computations. If the algorithm detects that $\alpha_r(P_1 \cup P_2)$ does not exist, then we halt the algorithm and report that $\alpha_r(S^+[i, j])$ does not exist. Also, if the algorithm finds that a dominating case happens, e.g., the P_1-dominating case, then $\alpha_r(P_1 \cup P_2) = \alpha_r(P_1)$ and thus we simply return the root of the tree for $\alpha_r(P_1)$.

We do this for all internal nodes in the second level of T (i.e., the level above the leaves) in parallel by assigning a processor for each node. As T has $O(\log k)$ leaves, we can compute the circular hulls for the second level in $O(\log k)$ parallel steps using $O(\log k)$ processors. Then, we proceed on the third level similarly. At the root of T, we will have the root of a balanced binary search tree for $\alpha_r(P)$. Using $O(\log k)$ processors, this takes $O(\log k \log \log k)$ parallel steps because each level needs $O(\log k)$ parallel steps and the height of T is $O(\log \log k)$.

Alternatively, if we only use one processor, then since T has $O(\log k)$ nodes and we spend $O(\log k)$ time on each node, the total time is $O(\log^2 k)$. ◀

Armed with Lemma 6, to determine whether $r^* \le r$, we use the algorithm framework in Theorem 4.2 of Chan [6], but we provide a more efficient implementation, as follows.

Recall the definitions of the matrices A and B in Section 2, and in particular, each row of A (resp., B) is monotonically increasing while each column of A (resp., B) is monotonically decreasing. For convenience, let $A[i, -1] = 0$ and $A[i, n+1] = B[i, -1] = \infty$ for all $0 \le i \le n$. Let $m = \lfloor n/\log^6 n \rfloor$. Let $j_t = t \cdot \lfloor n/m \rfloor$ for $t = 1, 2, \ldots, m-1$. Set $j_0 = -1$ and $j_m = n$. For each $t \in [0, m]$, find the largest $i_t \in [0, n]$ with $A[i_t, j_t] \ge B[i_t, j_t]$ (set $i_t = -1$ if no such index exists; note that $i_0 = -1$). Observe that $i_0 \le i_1 \le \cdots \le i_m$. Each i_t can be found in $O(\log^7 n)$ time by binary search using Lemma 7. Hence, computing all i_t's takes $O(n \log n)$ time. This is part of our preprocessing, independent of r.

▶ **Lemma 7** ([6, 16]). *After $O(n \log n)$ time preprocessing, $A[i, j]$ and $B[i, j]$ can be computed in $O(\log^6 n)$ time for any given pair (i, j).*

Given $r > 0$, our goal is to decide whether $r^* \leq r$. Let $(r_1, r_2]$ be the interval obtained by Lemma 6. Since $r^* \in (r_1, r_2]$, if $r \leq r_1$, then $r^* > r$; if $r \geq r_2$, then $r^* \leq r$. It remains to resolve the case $r \in (r_1, r_2)$, as follows. In this case the result of Lemma 6 applies.

We will decide whether $r_i^* \leq r$ for all $i = 0, 1, \ldots, n$ (recall that $r^* \leq r$ iff some $r_i^* \leq r$). Let $t \in [0, m-1]$ such that $i_t < i \leq i_{t+1}$. If $A[i, j_t] > r$, then return $r_i^* > r$. Otherwise, find (by binary search) the largest $j \in [j_t, j_{t+1}]$ with $A[i, j] \leq r$, and return $r_i^* \leq r$ if and only if $B[i, j] \leq r$. See the pseudocode below. See Theorem 4.2 of [6] for the algorithm correctness.

Algorithm 2 The decision algorithm of Theorem 4.2 by Chan [6].

1 Let $t \in [0, m-1]$ such that $i_t < i \leq i_{t+1}$;
2 **if** $A[i, j_t] > r$ **then** return $r_i^* > r$;
3 find the largest $j \in [j_t, j_{t+1}]$ with $A[i, j] \leq r$;
4 return $r_i^* \leq r$ iff $B[i, j] \leq r$;

Chan [6] implemented the algorithm in $O(\log n \log^2 \log n)$ parallel steps using $O(n \log n)$ processors. With Lemma 6, we provide a faster implementation of $O(\log n \log \log n)$ parallel steps using $O(n)$ processors. Line 1 can be done in $O(n)$ time as part of the preprocessing, independent of r. We first discuss how to implement Line 3 for all indices i, and we will show later that Lines 2 and 4 can be implemented in a similar (and faster) way.

For each $t = 0, 1, \ldots, m-1$, if $i_{t+1} - i_t \leq \log^6 n$, then we form a group of at most $\log^6 n$ indices: $i_t + 1, i_t + 2, \ldots, i_{t+1}$. Otherwise, starting from $i_t + 1$, we form a group for every consecutive $\log^6 n$ indices until i_{t+1}, so every group has exactly $\log^6 n$ indices except that the last group may have less than $\log^6 n$ indices. In this way, we have at most $2m$ groups, each of which consists of at most $\log^6 n$ consecutive indices in $(i_t, i_{t+1}]$ for some $t \in [0, m-1]$.

Consider a group $G = \{a, a+1, \ldots, a+b\}$ of indices in $(i_t, i_{t+1}]$. Note that $b < \log^6 n$. For each $i \in [a, a+b]$ such that $A[i, j_t] \leq r$, we need to perform binary search on $[j_t, j_{t+1}]$ to find the largest index j with $A[i, j] \leq r$. To this end, we do the following. We compute the two circular hulls $\alpha(S^+[a+b, n])$ and $\alpha(S^-[1, j_t])$, in $O(\log n \log \log n)$ parallel steps using $O(\log n)$ processors by Lemma 6. Note that by "compute the two circular hulls", we mean that the two circular hulls are computed implicitly in the sense that each of them is represented by a balanced binary search tree and we have the access of its root. If $\alpha(S^+[a+b, n])$ (resp., $\alpha(S^-[1, j_t])$) does not exist, we set it to *null*. We do this for all $2m$ groups in parallel, which takes $O(\log n \log \log n)$ parallel steps using $O(m \log n) \in O(n)$ processors.

Consider the group G defined above again. For each $i \in [a, a+b]$, we need to do binary search on $[j_t, j_{t+1}]$ for $O(\log(j_{t+1} - j_t)) = O(\log \log n)$ iterations. In each iteration, the goal is to determine whether $A[i, j] \leq r$ for an index $j \in [j_t, j_{t+1}]$. To this end, it suffices to determine whether $\alpha(U_{ij})$ exists. Notice that $U_{ij} = S^+[i+1, a+b-1] \cup S^+[a+b, n] \cup S^-[1, j_t] \cup S^-[j_t+1, j]$. $\alpha(S^+[a+b, n])$ and $\alpha(S^-[1, j_t])$ are already computed above. If one of them does not exist, then $\alpha(U_{ij})$ does not exist and thus $A[i, j] > r$. Otherwise, we compute the circular hull $\alpha(S^+[i+1, a+b-1])$, which can be done in $O(\log^2 \log n)$ time using one processor by Lemma 6 because $a + b - 1 - i \leq b - 1 \leq \log^6 n$. We also compute $\alpha(S^-[j_t + 1, j])$ in $O(\log^2 \log n)$ time using one processor. Then, we compute the common tangents of $\alpha(S^+[i+1, a+b-1])$ and $\alpha(S^+[a+b, n])$ by Lemma 5 (note that $S^+[i+1, a+b-1]$ and $S^+[a+b, n]$ are separated by a line through o), in $O(\log n)$ time using one processor. Then, we merge the two hulls with the two common tangents to obtain a balanced binary search tree for $\alpha(S^+[i+1, n])$. Because we want to keep the tree for $\alpha(S^+[a+b, n])$ unaltered so that it can participate in other computations in parallel, we use a persistent tree to represent it. Similarly, we obtain a tree for $\alpha(S^-[1, j])$, in $O(\log n)$ time using one processor. If one of $\alpha(S^+[i+1, n])$ and $\alpha(S^-[1, j])$

does not exist, then we return $A[i,j] > r$. Note that $S^+[a+b,n]$ and $S^-[1,j]$ are separated by ℓ and $U_{ij} = S^+[a+b,n] \cup S^-[1,j]$. By Lemma 5, we can determine whether $\alpha(U_{ij})$ exists in $O(\log n)$ time using one processor and consequently determine whether $A[i,j] \leq r$. Hence, the above algorithm determines whether $A[i,j] \leq r$ in $O(\log n)$ time using one processor.

If we do the above for all i's in parallel, then we can determine whether $A[i,j] \leq r$ in $O(\log n)$ time using $n+1$ processors, for each iteration of the binary search. As there are $O(\log\log n)$ iterations, the binary search procedure (i.e., Line 3) for all $i = 0, 1, \ldots, n$ runs in $O(\log n \log\log n)$ parallel steps using $n+1$ processors.

For implementing Line 2, we can use the same approach as above by grouping the indices i into $2m$ groups. The difference is that now each i has a specific index j, i.e., $j = j_t$, for deciding whether $A[i,j] \leq r$, and thus we do not have to do binary search. Hence, using $n+1$ processors, we can implement Line 2 for all $i = 0, 1, \ldots, n$ in $O(\log n)$ parallel steps. We can do the same for Line 4. As a summary, we have the following theorem.

▶ **Theorem 8.** *After $O(n \log n)$ time preprocessing on S, given any r, we can decide whether $r^* \leq r$ in $O(\log n \log\log n)$ parallel steps using $O(n)$ processors.*

With the serial decision algorithm in Theorem 4 and applying Cole's parametric search [10] on the parallel decision algorithm in Theorem 8, the following result follows.

▶ **Theorem 9.** *The value r^* can be computed in $O(n \log n \log\log n)$ time.*

Proof. Suppose there is a serial decision algorithm of time T_S and another parallel decision algorithm that runs in T_p parallel steps using P processors. Then, Megiddo's parametric search [23] can compute r^* in $O(PT_p + T_sT_p \log P)$ time by simulating the parallel decision algorithm on r^* and using the serial decision algorithm to resolve comparisons with r^*. If the parallel decision algorithm has a "bounded fan-in or bounded fan-out" property, then Cole's technique [10] can reduce the time complexity to $O(PT_p + T_s(T_p + \log P))$. Like Chan's algorithm [6], our algorithm has this property because it mainly consists of $O(\log\log n)$ rounds of independent binary search (i.e., the algorithm of Lemma 5). In our case, $T_s = O(n)$, $T_p = O(\log n \log\log n)$, and $P = O(n)$. Thus, applying Cole's technique, r^* can be computed in $O(n \log n \log\log n)$ time. ◀

▶ **Corollary 10.** *The planar two-center problem can be solved in $O(n \log^2 n)$ time.*

Proof. This follows by combining Theorem 9, which is for the nearby case, with the $O(n \log^2 n)$ time algorithm for the distant case [16]. ◀

5 The convex position case

In this section, we consider the case where S is in convex position (i.e., every point of S is a vertex of the convex hull of S). We show that our above $O(n \log n \log\log n)$ time algorithm can be applied to solving this case in the same time asymptotically.

We first compute the convex hull $CH(S)$ of S and order all vertices clockwise as $p_1, p_2, \ldots, p_n$. A key observation [22] is that there is an optimal solution with two congruent disks D_1^* and D_2^* of radius r^* such that D_1^* covers the points of S in a chain of $\partial CH(S)$ and D_2^* covers the rest of the points. In other words, the cyclic list of $p_1, p_2, \ldots, p_n$ can be cut into two lists such that one list is covered by D_1^* and the other list is covered by D_2^*.

Let o be any point in the interior of $CH(S)$. By the above observation, there exists a pair of rays ρ_1 and ρ_2 emanating from o such that D_1^* covers all points of S on one side of the two rays and D_2^* covers the points of S in the other side. To apply our previous algorithm, we need to find a line ℓ that separates the two rays. For this, we propose the following approach.

For any $i, j \in [1, n]$, let $S_{cw}[i, j]$ denote the subset of vertices on $CH(S)$ clockwise from p_i to p_j, and $S_{cw}[i, j] = \{p_i\}$ if $i = j$. Due to the above observation, $r^* = \min_{i,j\in[1,n]} \max\{\tau(S_{cw}[i, j]), \tau(S_{cw}[j + 1, i - 1])\}$, with indices modulo n. For each $i \in [1, n]$, define $r(i) = \min_{h\in[i,i+n-1]} \max\{\tau(S_{cw}[i, h]), \tau(S_{cw}[h + 1, i - 1])\}$. Notice that as h increases in $[1, n-1]$, $\tau(S_{cw}[1, h])$ is monotonically increasing while $\tau(S_{cw}[h+1, n])$ is monotonically decreasing. Define k to be the largest index in $[1, n-1]$ such that $\tau(S_{cw}[1, k]) \leq \tau(S_{cw}[k+1, n])$.

▶ **Lemma 11.** *r^* is equal to the minimum of the following values: $r(1)$, $r(k+1)$, $r(k+2)$, and $\max\{\tau(S_{cw}[i, j]), \tau(S_{cw}[j + 1, i - 1])$ for all indices i and j with $i \in [1, k]$ and $j \in [k + 2, n]$.*

Proof. Observe that $r^* = \min_{i,j\in[1,n]} \max\{\tau(S_{cw}[i, j]), \tau(S_{cw}[j+1, i-1])\} = \min_{1\leq h\leq n} r(h)$. Hence, r^* is no larger than any of the values specified in the lemma statement.

Let i and j be two indices such that $r^* = \max\{\tau(S_{cw}[i, j]), \tau(S_{cw}[j + 1, i - 1])\}$ with $1 \leq i \leq j \leq n$. We claim that $r^* = r(i)$. Indeed, since $r^* = \min_{1\leq h\leq n} r(h)$, we have $r^* \leq r(i)$. On the other hand, as $r(i) \leq \max\{\tau(S_{cw}[i, j]), \tau(S_{cw}[j + 1, i - 1])\} = r^*$, we obtain $r(i) = r^*$. By a similar argument, $r^* = r(j + 1)$ also holds.

Without loss of generality, we assume that $r^* = \tau(S_{cw}[i, j]) \geq \tau(S_{cw}[j + 1, i - 1])$.

If $i \in [1, k]$ and $j \in [k+2, n]$, then the lemma follows. Otherwise, one of the following four cases must hold: $i = k + 1$, $j = k + 1$, $[i, j] \subseteq [1, k]$, and $[i, j] \subseteq [k + 2, n]$. If $i = k + 1$, then $r^* = r(k + 1)$. If $j = k + 1$, then $r^* = r(k + 2)$. Below we show that $r^* = r(1)$ if $[i, j] \subseteq [1, k]$ and we also show that the case $[i, j] \subseteq [k + 2, n]$ cannot happen, which will prove the lemma.

If $[i, j] \subseteq [1, k]$, then $\tau(S_{cw}[j+1, i-1]) \geq \tau(S_{cw}[k+1, n])$, for $S_{cw}[k+1, n] \subseteq S_{cw}[j+1, i-1]$. By the definition of k, we have $\tau(S_{cw}[k + 1, n]) \geq \tau(S_{cw}[1, k])$. Because $[i, j] \subseteq [1, k]$, $\tau(S_{cw}[1, k]) \geq \tau(S_{cw}[i, j])$. Combining the above three inequalities leads to the following: $\tau(S_{cw}[j+1, i-1]) \geq \tau(S_{cw}[k+1, n]) \geq \tau(S_{cw}[1, k]) \geq \tau(S_{cw}[i, j])$. Because $r^* = \tau(S_{cw}[i, j]) \geq \tau(S_{cw}[j + 1, i - 1])$, we obtain $r^* = \tau(S_{cw}[j + 1, i - 1]) = \tau(S_{cw}[k + 1, n]) = \tau(S_{cw}[1, k]) = \tau(S_{cw}[i, j])$. Notice that $r(1) \leq \max\{\tau(S_{cw}[1, k]), \tau(S_{cw}[k+1, n])\}$. Thus, we derive $r(1) \leq r^*$. Since $r^* \leq r(1)$, we finally have $r^* = r(1)$.

If $[i, j] \subseteq [k+2, n]$, then $\tau(S_{cw}[j+1, i-1]) \geq \tau(S_{cw}[1, k+1])$. By the definition of k, we have $\tau(S_{cw}[1, k+1]) > \tau(S_{cw}[k+2, n])$. Also, since $[i, j] \subseteq [k+2, n]$, $\tau(S_{cw}[k+2, n]) \geq \tau(S_{cw}[i, j])$ holds. Therefore, we obtain $\tau(S_{cw}[j + 1, i - 1]) \geq \tau(S_{cw}[1, k + 1]) > \tau(S_{cw}[k + 2, n]) \geq \tau(S_{cw}[i, j])$, which incurs contradiction since $r^* = \tau(S_{cw}[i, j]) \geq \tau(S_{cw}[j + 1, i - 1])$. Thus, the case $[i, j] \subseteq [k + 2, n]$ cannot happen. ◀

Based on the above lemma, our algorithm works as follows.

We first compute $r(1)$ and the index k. This can be easily done in $O(n \log n)$ time. Indeed, as h increases in $[1, n - 1]$, $\tau(S_{cw}[1, h])$ is monotonically increasing while $\tau(S_{cw}[h + 1, n])$ is monotonically decreasing. Thus, r_1^* and k can be found by binary search on $[1, n - 1]$. As both $\tau(S_{cw}[1, h])$ and $\tau(S_{cw}[h+1, n])$ can be computed in $O(n)$ time, the binary search takes $O(n \log n)$ time. Similarly, we can compute $r(k + 1)$ and $r(k + 2)$ in $O(n \log n)$ time.

If $r^* \notin \{r(1), r(k + 1), r(k + 2)\}$, then $r^* = \max\{\tau(S_{cw}[i, j]), \tau(S_{cw}[j + 1, i - 1])$ for two indices i and j with $i \in [1, k]$ and $j \in [k + 2, n]$. We can compute it as follows. Let ℓ be a line through v_{k+1} and intersecting the interior of $\overline{p_n p_1}$. Let o be any point on ℓ in the interior of $CH(S)$. Lemma 11 implies ℓ and o satisfy the property discussed above, i.e., ℓ separates the two rays ρ_1 and ρ_2. Consequently, we can apply our algorithm for Theorem 9 to compute r^* in $O(n \log n \log \log n)$ time.

▶ **Theorem 12.** *The planar two-center problem for a set of n points in convex position can be solved in $O(n \log n \log \log n)$ time.*

▶ Remark. The randomized result remarked after Theorem 4 also applies here: r^* can be computed in $O(n)$ time with high probability after $O(n \log n)$ deterministic time preprocessing.

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 Symposium on Algorithms (ESA)*, pages 64–75, 2008.
2. P.K. Agarwal and M. Sharir. Planar geometric location problems. *Algorithmica*, 11:185–195, 1994.
3. P.K. Agarwal, M. Sharir, and E. Welzl. The discrete 2-center problem. *Discrete and Computational Geometry*, 20:287–305, 1998.
4. E.M. Arkin, J.M. Díaz-Báñez, F. Hurtado, P. Kumar, J.S.B. Mitchell, B. Palop, P. Pérez-Lantero, M. Saumell, and R.I. Silveira. Bichromatic 2-center of pairs of points. *Computational Geometry: Theory and Applications*, 48:94–107, 2015.
5. M. Blum, R.W. Floyd, V. Pratt, R.L. Rivest, and R.E. Tarjan. Time bounds for selection. *Journal of Computer and System Sciences*, 7:448–461, 1973.
6. T.M. Chan. More planar two-center algorithms. *Computational Geometry: Theory and Applications*, 13:189–198, 1999.
7. T.M. Chan. A simpler linear-time algorithm for intersecting two convex polyhedra in three dimensions. *Discrete and Computational Geometry*, 56:860–865, 2016.
8. B. Chazelle. An optimal algorithm for intersecting three-dimensional convex polyhedra. *SIAM Journal on Computing*, 21(4):671–696, 1992.
9. B. Chazelle and J. Matoušek. On linear-time deterministic algorithms for optimization problems in fixed dimension. *Journal of Algorithms*, 21:579–597, 1996.
10. R. Cole. Slowing down sorting networks to obtain faster sorting algorithms. *Journal of the ACM*, 34(1):200–208, 1987.
11. D.P. Dobkin and D.G. Kirkpatrick. Determining the separation of preprocessed polyhedra – A unified approach. In *Proceedings of the 17th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 400–413, 1990.
12. J. Driscoll, N. Sarnak, D. Sleator, and R.E. Tarjan. Making data structures persistent. *Journal of Computer and System Sciences*, 38(1):86–124, 1989.
13. M.E. Dyer. On a multidimensional search technique and its application to the Euclidean one centre problem. *SIAM Journal on Computing*, 15(3):725–738, 1986.
14. H. Edelsbrunner, D. Kirkpatrick, and R. Seidel. On the shape of a set of points in the plane. *IEEE Transactions on Information Theory*, 29:551–559, 1983.
15. D. Eppstein. Dynamic three-dimensional linear programming. *ORSA Journal on Computing*, 4:360–368, 1992.
16. D. Eppstein. Faster construction of planar two-centers. In *Proc. of the 8th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 131–138, 1997.
17. D. Halperin, M. Sharir, and K. Goldberg. The 2-center problem with obstacles. *Journal of Algorithms*, 42:109–134, 2002.
18. J. Hershberger. A faster algorithm for the two-center decision problem. *Information Processing Letters*, 1:23–29, 1993.
19. J. Hershberger and S. Suri. Finding tailored partitions. *Journal of Algorithms*, 3:431–463, 1991.
20. J. Jaromczyk and M. Kowaluk. An efficient algorithm for the Euclidean two-center problem. In *Proceedings of the 10th Annual Symposium on Computational Geometry (SoCG)*, pages 303–311, 1994.
21. M. Katz and M. Sharir. An expander-based approach to geometric optimization. *SIAM Journal on Computing*, 26(5):1384–1408, 1997.
22. S.K. Kim and C.-S. Shin. Efficient algorithms for two-center problems for a convex polygon. In *Proceedings of the 6th International Computing and Combinatorics Conference (COCOON)*, pages 299–309, 2000.
23. N. Megiddo. Applying parallel computation algorithms in the design of serial algorithms. *Journal of the ACM*, 30(4):852–865, 1983.

24 N. Megiddo. Linear-time algorithms for linear programming in R^3 and related problems. *SIAM Journal on Computing*, 12(4):759–776, 1983.

25 N. Sarnak and R.E. Tarjan. Planar point location using persistent search trees. *Communications of the ACM*, 29:669–679, 1986.

26 M. Sharir. A near-linear algorithm for the planar 2-center problem. *Discrete and Computational Geometry*, 18:125–134, 1997.

27 X. Tan and B. Jiang. Simple $O(n \log^2 n)$ algorithms for the planar 2-center problem. In *Proceedings of the 23rd International Computing and Combinatorics Conference (COCOON)*, pages 481–491, 2017.

28 H. Wang and J. Xue. Improved algorithms for the bichromatic two-center problem for pairs of points. In *Proceedings of the 16th Algorithms and Data Structures Symposium (WADS)*, pages 578–591, 2019.

Algorithms for Subpath Convex Hull Queries and Ray-Shooting Among Segments

Haitao Wang
Department of Computer Science, Utah State University, Logan, UT 84322, USA
haitao.wang@usu.edu

Abstract

In this paper, we first consider the subpath convex hull query problem: Given a simple path π of n vertices, preprocess it so that the convex hull of any query subpath of π can be quickly obtained. Previously, Guibas, Hershberger, and Snoeyink [SODA 90'] proposed a data structure of $O(n)$ space and $O(\log n \log\log n)$ query time; reducing the query time to $O(\log n)$ increases the space to $O(n \log\log n)$. We present an improved result that uses $O(n)$ space while achieving $O(\log n)$ query time. Like the previous work, our query algorithm returns a compact interval tree representing the convex hull so that standard binary-search-based queries on the hull can be performed in $O(\log n)$ time each. Our new result leads to improvements for several other problems.

In particular, with the help of the above result, we present new algorithms for the ray-shooting problem among segments. Given a set of n (possibly intersecting) line segments in the plane, preprocess it so that the first segment hit by a query ray can be quickly found. We give a data structure of $O(n \log n)$ space that can answer each query in $(\sqrt{n} \log n)$ time. If the segments are nonintersecting or if the segments are lines, then the space can be reduced to $O(n)$. All these are classical problems that have been studied extensively. Previously data structures of $\widetilde{O}(\sqrt{n})$ query time[1] were known in early 1990s; nearly no progress has been made for over two decades. For all problems, our results provide improvements by reducing the space of the data structures by at least a logarithmic factor while the preprocessing and query times are the same as before or even better.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms; Theory of computation → Computational geometry

Keywords and phrases subpath hull queries, convex hulls, compact interval trees, ray-shooting, data structures

Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.69

Related Version A full version of this paper is available at `https://arxiv.org/abs/2002.10672`.

1 Introduction

We first consider the subpath convex hull query problem. Let π be a simple path of n vertices in the plane. A *subpath hull query* specifies two vertices of π and asks for the convex hull of the subpath between the two vertices. The goal is to preprocess π so that the subpath hull queries can be answered quickly. Ideally, the query should return a representation of the convex hull so that standard queries on the hull can be performed in logarithmic time.

The problem has been studied by Guibas, Hershberger, and Snoeyink [18], who proposed a method of using compact interval trees. After $O(n \log n)$ time preprocessing, Guibas et al. [18] built a data structure of $O(n)$ space that can answer each query in $O(\log n \log\log n)$ time. Their query algorithm returns a compact interval tree that represents the convex hull so that all binary-search-based queries on the hull can be performed in $O(\log n)$ time each. The queries on the hull include (but are not limited to) the following: find the most extreme vertex of the convex hull along a query direction; find the intersection between a query

[1] The notation $\widetilde{O}$ suppresses a polylogarithmic factor.

line and the convex hull; find the common tangents from a query point to the convex hull; determine whether a query point is inside the convex hull, etc. Guibas et al. [18] reduced the subpath hull query time to $O(\log n)$ but the space becomes $O(n \log\log n)$. A trade-off was also made with $O(\log n \log^* n)$ query time and $O(n \log^* n)$ space [18].

As compact interval trees are quite amenable, the results of Guibas et al. [18] have found many applications, e.g., [4, 9–13, 25]. Clearly, there is still some room for improvement on the results of Guibas et al. [18]; the ultimate goal might be an $O(n)$ space data structure with $O(\log n)$ query time. We achieve this goal. The preprocessing time of our data structure is $O(n)$, after the vertices of π are sorted by x-coordinate. Like the results of Guibas et al. [18], our query algorithm also returns a compact interval tree that can support logarithmic time queries for all binary-search-based queries on the convex hull of the query subpath; the edges of the convex hull can be retrieved in time linear in the number of vertices of the convex hull. Note that like those in [18] our results are for the random access machine (RAM) model.

With our new result, previous applications that use the results of Guibas et al. [18] can now be improved accordingly. These include the problem of enclosing polygons by two minimum area rectangles [4, 5], computing a guarding set for simple polygons in wireless location [12], computing optimal time-convex hulls [13], L_1 top-k weighted sum aggregate nearest and farthest neighbor searching [25], etc. For all these problems, we reduce the space of their algorithms by a $\log\log n$ factor while the time complexities are the same as before or even better. See the full paper for the details of our improvements.

Wagener [24] proposed a parallel algorithm for computing a data structure, called *bridge tree*, for representing the convex hull of a simple path π. If using one processor, for any query subpath of π, Wagener [24] showed that the bridge tree can be used to answer decomposable queries on the convex hull of the query subpath in logarithmic time each. Wagener [24] claimed that some non-decomposable queries can also be handled; however no details were provided. In contrast, our approach returns a compact interval tree that is more amenable (indeed, the bridge trees [24] were mainly designed for parallel processing) and can support both decomposable and non-decomposable queries. In addition, if one wants to output the convex hull of the query subpath, our approach can do so in time linear in the number of the vertices of the convex hull while the method of Wagener [24] needs $O(n)$ time.

1.1 Ray-shooting

With the help of our above result and other new techniques, we present improved results for several classical ray-shooting problems. Previously, data structures of $\widetilde{O}(\sqrt{n})$ query time and near-linear space were known in early 1990s; nearly no progress has been made for over two decades. Our results reduce the space by at least a logarithmic factor while achieving the same or better preprocessing and query times. In the following, we use $O(T(n), S(n), Q(n))$ to represent the complexity of a data structure, where $T(n)$ is the preprocessing time, $S(n)$ is the space, and $Q(n)$ is the query time. We will confine the discussion of the previous work to data structures of linear or near-linear space. Refer to Table 1 for a summary. Throughout the paper, we use δ to refer to an arbitrarily small positive constant.

Ray-shooting among lines. Given a set of n lines in the plane, the problem is to build a data structure so that the first line hit by a query ray can be quickly found.

Bar-Yehuda and Fogel [3] gave a data structure of complexity $O(n^{1.5}, n\log^2 n, \sqrt{n}\log n)$. Cheng and Janardan [11] gave a data structure of complexity $O(n^{1.5}\log^2 n, n\log n, \sqrt{n}\log n)$. Agarwal and Sharir [2] developed a data structure of complexity $O(n\log n, n\log n, n^{1/2+\delta})$.

Table 1 Summary of the results. The big-O notation is omitted. δ can be any small positive constant. The results marked with * hold with high probability (except that the result of Chan [6] is expected).

	Preprocessing time	Space	Query time	Source
Ray-shooting among lines	$n^{1.5}$	$n\log^2 n$	$\sqrt{n}\log n$	BF [3]
	$n^{1.5}\log^2 n$	$n\log n$	$\sqrt{n}\log n$	CJ [11]
	$n\log n$	$n\log n$	$n^{0.5+\delta}$	AS [2]
	$n^{1.5}$	n	$\sqrt{n}\log n$	this paper
	$n\log n$	n	$\sqrt{n}\log n$ *	this paper
Intersection detection	$n^{1.5}\log^2 n$	$n\log n$	$\sqrt{n}\log n$	CJ [11]
	$n^{1.5}$	n	$\sqrt{n}\log n$	this paper
	$n\log n$	n	$\sqrt{n}\log n$ *	this paper
Ray-shooting among intersecting segments	$n\alpha(n)\log^3 n$	$n\log^2 n$	$n^{0.695}\log n$	OSS [23]
	$n\alpha(n)\log^3 n$	$n\alpha(n)$	$n^{2/3+\delta}$	GOS [19]
	$n^{1.5}\log^{4.33} n$	$n\alpha(n)\log^4 n$	$\sqrt{n\alpha(n)}\log^2 n$	A [1]
	$(n\alpha(n))^{1.5}$	$n\alpha(n)\log^2 n$	$\sqrt{n\alpha(n)}\log n$	BF [3]
	$n^{1.5}\log^2 n$	$n\log^2 n$	$\sqrt{n}\log n$	CJ [11]
	$n\log^2 n$	$n\log^2 n$	$n^{0.5+\delta}$	AS [2]
	$n\log^3 n$	$n\log^2 n$	$\sqrt{n}\log^2 n$ *	C [6]
	$n^{1.5}$	$n\log n$	$\sqrt{n}\log n$	this paper
	$n\log^2 n$	$n\log n$	$\sqrt{n}\log n$ *	this paper
Ray-shooting among nonintersecting segments	$n\log n$	n	$n^{0.695}\log n$	OSS [23]
	$n^{1.5}\log^{4.33} n$	$n\alpha(n)\log^3 n$	$\sqrt{n}\log^2 n$	A [1]
	$n^{1.5}$	$n\log n$	$\sqrt{n}\log n$	BF [3]
	$n^{1.5}$	n	$\sqrt{n}\log n$	this paper
	$n\log n$	n	$\sqrt{n}\log n$ *	this paper

By using our subpath hull query data structure and a result from Chazelle and Guibas [7], we present a new data structure of complexity $O(n^{1.5}, n, \sqrt{n}\log n)$.

In addition, we also consider a more general *first-k-hits* query, i.e., given a query ray and an integer k, report the first k lines hit by the ray. This problem was studied by Bar-Yehuda and Fogel [3], who gave a data structure of complexity $O(n^{1.5}, n\log^2 n, \sqrt{n}\log n + k\log^2 n)$. Our new result is a data structure of complexity $O(n^{1.5}, n, \sqrt{n}\log n + k\log n)$.

Intersection detection. Given a set of n line segments in the plane, the problem is to build a data structure to determine whether a query line intersects at least one segment. Cheng and Janardan [11] gave a data structure of complexity $O(n^{1.5}\log^2 n, n\log n, \sqrt{n}\log n)$. By adapting the interval partition trees of Overmars et al. [23] to the partition trees of Matoušek [20, 21], we obtain a data structure of complexity $O(n^{1.5}, n, \sqrt{n}\log n)$.

Ray-shooting among segments. Given a set of n (possibly intersecting) line segments in the plane, we want to build a data structure to find the first segment hit by a query ray.

The result of Overmars et al. [23] has complexity $O(n\alpha(n)\log^3 n, n\log^2 n, n^{0.695}\log n)$, where $\alpha(n)$ is the inverse Ackermann's function. Guibas et al. [19] presented a data structure of complexity $O(n\alpha(n)\log^3 n, n\alpha(n), n^{2/3+\delta})$. Agarwal [1] gave a data structure of complexity $O(n^{1.5}\log^{4.33} n, n\alpha(n)\log^4 n, \sqrt{n\alpha(n)}\log^2 n)$. Bar-Yehuda and Fogel [3] gave a data structure

of complexity $O((n\alpha(n))^{1.5}, n\alpha(n)\log^2 n, \sqrt{n\alpha(n)}\log n)$. Cheng and Janardan [11] developed a data structure of complexity $O(n^{1.5}\log^2 n, n\log^2 n, \sqrt{n}\log n)$. Agarwal and Sharir [2] proposed a data structure of complexity $O(n\log^2 n, n\log^2 n, n^{0.5+\delta})$. Chan's randomized result [6] has complexity $O(n\log^3 n, n\log^2 n, \sqrt{n}\log^2 n)$, where the query time is expected.

Cheng and Janardan's algorithm [11] relies on their results for the ray-shooting problem among lines and the intersection detection problem. Following their algorithmic scheme and using our above new results for these two problems, we obtain a data structure for the ray-shooting problem among segments with complexity $O(n^{1.5}, n\log n, \sqrt{n}\log n)$. This is the first data structure of $\widetilde{O}(\sqrt{n})$ query time that uses only $O(n\log n)$ space.

If the segments are nonintersecting, Overmars et al. [23] gave a data structure of complexity $O(n\log n, n, n^{0.695}\log n)$. Agarwal [1] presented a data structure of complexity $O(n^{1.5}\log^{4.33} n, n\alpha(n)\log^3 n, \sqrt{n}\log^2 n)$. Bar-Yehuda and Fogel [3] proposed a data structure of complexity $O(n^{1.5}, n\log n, \sqrt{n}\log n)$. Our result has complexity $O(n^{1.5}, n, \sqrt{n}\log n)$.

Randomized results. Using Chan's randomized techniques [6], the preprocessing time of all our above results can be reduced to $O(n\log n)$ (except $O(n\log^2 n)$ time for the ray-shooting problem among intersecting segments), while the same query time complexities hold with high probability (i.e., probability at least $1 - 1/n^c$ for any large constant c).

Outline. Section 2 reviews some previous work of the subpath hull queries; Section 3 presents our new data structure for the problem. Section 4 solves the ray-shooting problem.

2 Preliminaries

Let $p_1, \ldots, p_n$ be the vertices of a simple path π ordered along π. For any two indices i and j with $1 \le i \le j \le n$, we use $\pi(i, j)$ to refer to the subpath of π from p_i to p_j. Given a pair (i, j) of indices with $1 \le i \le j \le n$, the *subpath hull query* asks for the convex hull of $\pi(i, j)$.

The convex hull of a simple path can be found in linear time, e.g., [17, 22]. Note that the convex hull of a simple path is the same as the convex hull of its vertices. For this reason, in our discussion a subpath π' of π actually refers to its vertex set. For each subpath π' of π, we use $|\pi'|$ to denote the number of vertices of π'.

For any set P of points in the plane, let $H(P)$ denote the convex hull of P. Denote by $H_U(P)$ and $H_L(P)$ the upper and lower hulls, respectively.

Interval trees. Let S be a set of n points in the plane. The *interval tree* $T(S)$ is a complete binary tree whose leaves from left to right correspond to the points of S sorted from left to right. Each internal node corresponds to the interval between the rightmost leaf in its left subtree and the leftmost leaf in its right subtree. We say that a segment joining two points of S *spans* an internal node v if the projection of the interval of v on the x-axis is contained in the projection of the segment on the x-axis.

We store each edge e of the upper hull $H_U(S)$ at the highest node of $T(S)$ that e spans (e.g., see Fig. 1). By also storing the edges of the lower hull $H_L(S)$ in $T(S)$ in the same way, we can answer all standard binary-search-based queries on the convex hull $H(S)$ in $O(\log n)$ time, by following a path from the root of $T(S)$ to a leaf (see Lemma 4.1 of [18] for details).

Compact interval trees. As the size of $T(S)$ is $\Theta(n)$ while $|H(S)|$ may be much smaller than n, where $|H(S)|$ is the number of edges of $H(S)$, using $T(S)$ to store $H(S)$ may not be space-efficient. Guibas et al. [18] proposed to use a *compact interval tree* $T_U(S)$ of $O(|H_U(S)|)$

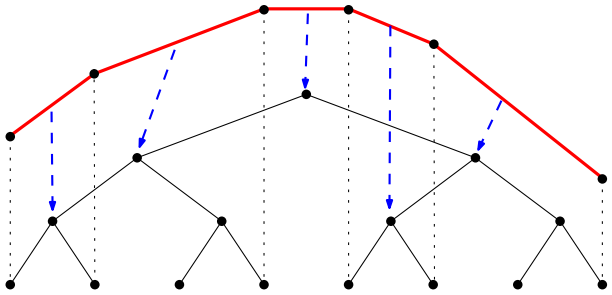


Figure 1 Illustrating an interval tree that stores upper hull edges.

size to store $H_U(S)$, as follows. In $T(S)$, a node v is *empty* if it does not store an edge of $H_U(S)$; otherwise it is *full*. It was shown in [18] that if two nodes of $T(S)$ are full, then their lowest common ancestor is also full. We remove empty nodes from $T(S)$ by relinking the tree to make each full node the child of its nearest full ancestor. Let $T_U(S)$ be the new tree and we still use $T(S)$ to refer to the original interval tree without storing any hull edges. Each node of $T_U(S)$ stores exactly one edge of $H_U(S)$, and thus $T_U(S)$ has $|H_U(S)|$ nodes. After $O(n)$ time preprocessing on $T(S)$, $T_U(S)$ can be computed from $H_U(S)$ in $O(|H_U(S)|)$ time (see Lemma 4.4 in [18]). Similarly, we use a compact interval tree $T_L(S)$ of $|H_L(S)|$ nodes to store $H_L(S)$. Then, using the three trees $T_U(S)$, $T_L(S)$, and $T(S)$, all standard binary-search-based queries on $H(S)$ can be answered in $O(\log n)$ time. The main idea is that the algorithm walks down through the compact interval trees while keeping track of the corresponding position in $T(S)$ (see Lemma 4.3 [18] for details). We call $T(S)$ a *reference tree*. In addition, using $T_U(S)$ and $T_L(S)$, $H(S)$ can be output in $O(|H(S)|)$ time.

As discussed above, to represent $H(S)$, we need two compact interval trees, one for $H_U(S)$ and the other for $H_L(S)$. To make our discussion more concise, we will simply say "the compact interval tree" for S and use $T^+(S)$ to refer to it, which actually includes two trees.

Compact interval trees for π. Consider two *consecutive* subpaths π_1 and π_2 of π. Suppose their compact interval trees $T^+(\pi_1)$ and $T^+(\pi_2)$ and the interval tree $T(\pi)$ of π are available. We know that $H(\pi_1)$ and $H(\pi_2)$ have at most two common tangents [7]. Using the path-copying method of persistent data structures [14], Guibas et al. [18] obtained the following.

▶ **Lemma 1** (Guibas et al. [18]). *Without altering $T^+(\pi_1)$ and $T^+(\pi_2)$, the compact interval tree $T^+(\pi_1 \cup \pi_2)$ can be produced in $O(\log n)$ time and $O(\log n)$ additional space.*

▶ **Lemma 2** (Guibas et al. [18]). *Given the interval tree $T(\pi)$, with $O(n)$ time preprocessing, we can compute $T^+(\pi')$ for any subpath π' of π in $O(|\pi'|)$ time.*

3 Subpath convex hull queries

We present our new data structure for subpath hull queries. We first sort all vertices of π by x-coordinate. The rest of the preprocessing of our data structure takes $O(n)$ time in total.

3.1 A decomposition tree

After having the interval tree $T(\pi)$, we construct a *decomposition tree* $\Psi(\pi)$, which is a segment tree on the vertices of π following their order along π. Specifically, $\Psi(\pi)$ is a complete binary tree with n leaves corresponding to the vertices of π in order along π. Each internal

node v of $\Psi(\pi)$ corresponds to the subpath $\pi(a_v, b_v)$, where a_v (resp., b_v) is defined to be the index of the vertex of π corresponding to the leftmost (resp., rightmost) leaf of the subtree of $\Psi(\pi)$ rooted at v; we call $\pi(a_v, b_v)$ a *canonical subpath* of π and use $\pi(v)$ to denote it.

Next, we remove some nodes in the lower part of $\Psi(\pi)$, as follows. For each node v whose canonical path has at most $\log^2 n$ vertices and whose parent canonical subpath has more than $\log^2 n$ vertices, we remove both the left and the right subtrees of v from $\Psi(\pi)$ but explicitly store $\pi(v)$ at v, after which v becomes a leaf of the new tree. From now on we use $\Psi(\pi)$ to refer to the new tree. It is not difficult to see that $\Psi(\pi)$ now has $O(n/\log^2 n)$ nodes.

We then compute compact interval trees $T^+(\pi(v))$ for all nodes v of $\Psi(\pi)$ in a bottom-up manner. Specifically, if v is a leaf, then $\pi(v)$ has at most $\log^2 n$ vertices, and we compute $T^+(\pi(v))$ from scratch, which takes $O(\log^2 n)$ time by Lemma 2. If v is not a leaf, then $T^+(\pi(v))$ can be obtained by merging the two compact interval trees of its children, which takes $O(\log n)$ time by Lemma 1. In this way, computing compact interval trees for all nodes of $\Psi(\pi)$ takes $O(n)$ time in total, for $\Psi(\pi)$ has $O(n/\log^2 n)$ nodes.

3.2 A preliminary query algorithm

Consider a subpath hull query (i, j). We first present an $O(\log^2 n)$ time query algorithm using $\Psi(\pi)$ and then reduce the time to $O(\log n)$. Depending on whether the two vertices p_i and p_j are in the same canonical subpath of a leaf of $\Psi(\pi)$, there are two cases.

Case 1. If yes, let v be the leaf. Then, $\pi(i, j)$ is a subpath of $\pi(v)$ and thus has at most $\log^2 n$ vertices. We compute $T^+(\pi(i, j))$ from scratch in $O(\log^2 n)$ time by Lemma 2.

Case 2. Otherwise, let v be the leaf of $\Psi(\pi)$ whose canonical subpath contains p_i and u the leaf whose canonical subpath contains p_j. Let w be the lowest common ancestor of u and v. As in [18], we partition $\pi(i, j)$ into two subpaths $\pi(i, k)$ and $\pi(k + 1, j)$, where $k = b_{w'}$ with w' being the left child of w (recall the definition of $b_{w'}$ given before). We will compute the compact interval trees for the two subpaths separately, and then merge them to obtain $T^+(\pi(i, j))$ in additional $O(\log n)$ time by Lemma 1. We only discuss how to compute $T^+(\pi(i, k))$, for the other tree can be computed likewise.

We partition $\pi(i, k)$ into two subpaths $\pi(i, b_v)$ and $\pi(b_v + 1, k)$. We will compute the trees for them separately and then merge the two trees to obtain $T^+(\pi(i, k))$.

For computing $T^+(\pi(i, b_v))$, as $\pi(i, b_v)$ is a subpath of $\pi(v)$, it has at most $\log^2 n$ vertices. Hence, we can compute $T^+(\pi(i, b_v))$ from scratch in $O(\log^2 n)$ time.

For $T^+(\pi(b_v + 1, k))$, observe that $\pi(b_v + 1, k)$ is the concatenation of canonical subpaths of $O(\log n)$ nodes of $\Psi(\pi)$; precisely, these nodes are the right children of their parents that are in the path of $\Psi(\pi)$ from v's parent to w' and these nodes themselves are not on the path. Since the compact interval trees of these nodes are already computed in the preprocessing, we can produce $T^+(\pi(b_v + 1, k))$ in $O(\log^2 n)$ time by merging these trees.

3.3 Reducing the query time to $O(\log n)$

We now reduce the query time to $O(\log n)$, with additional preprocessing (but still $O(n)$).

To reduce the time for Case 1, we perform the following preprocessing. For each leaf v of $\Psi(\pi)$, we preprocess the path $\pi(v)$ in the same way as above for preprocessing π. This means that we construct an interval tree $T(\pi(v))$ as well as a decomposition tree $\Psi(\pi(v))$ for the subpath $\pi(v)$. To answer a query for Case 1, we instead use $\Psi(\pi(v))$ (and use $T(\pi(v))$ as the reference tree). The query time becomes $O(\log^2 \log n)$ as $|\pi(v)| \leq \log^2 n$. Note that to construct $T(\pi(v))$ and $\Psi(\pi(v))$ in $O(|\pi(v)|)$ time, we need to sort all vertices of $\pi(v)$ by

x-coordinate in $O(|\pi(v)|)$ time. Recall that we already have a sorted list of all vertices of π, from which we can obtain sorted lists for $\pi(v)$ for all leaves v of $\Psi(\pi)$ in $O(n)$ time altogether. Hence, the preprocessing for $\pi(v)$ for all leaves v of $\Psi(\pi)$ takes $O(n)$ time.

We proceed to Case 2. To reduce the query time to $O(\log n)$, we will discuss how to perform additional preprocessing so that $T^+(\pi(i,k))$ can be computed in $O(\log n)$ time. Computing $T^+(\pi(k+1,j))$ can be done in $O(\log n)$ time similarly. Finally we can merge the two trees to obtain $T^+(\pi(i,j))$ in additional $O(\log n)$ time by Lemma 1.

To compute $T^+(\pi(i,k))$ in $O(\log n)$ time, according to our algorithm it suffices to compute both $T^+(i,b_v)$ and $T^+(b_v+1,k)$ in $O(\log n)$ time. We discuss $T^+(i,b_v)$ first.

Dealing with $T^+(\pi(i,b_v))$. To compute $T^+(i,b_v)$ in $O(\log n)$ time, we preform the following additional preprocessing. For each leaf v of $\Psi(\pi)$, recall that $|\pi(v)| \le \log^2 n$; we partition $\pi(v)$ into $t_v \le \log n$ subpaths each of which contains at most $\log n$ vertices. We use $\pi_v(1), \pi_v(2), \ldots, \pi_v(t_v)$ to refer to these subpaths in order along $\pi(v)$. For each subpath $\pi_v(i)$, we compute $T^+(\pi_v(i))$ from scratch in $O(\log n)$ time. The total time for computing all such trees is $O(\log^2 n)$. Next, we compute compact interval trees for t_v *prefix subpaths* of $\pi(v)$. Specifically, for each $t \in [1,t_v]$, we compute $T^+(\pi_v[1,t])$, where $\pi_v[1,t]$ is the concatenation of the paths $\pi_v(1), \pi_v(2), \ldots, \pi_v(t)$. This can be done in $O(\log^2 n)$ time by computing $T^+(\pi_v[1,t])$ incrementally for $t = 1,2,\ldots,t_v$ using the merge algorithm of Lemma 1. Indeed, initially $T^+(\pi_v[1,t]) = T^+(\pi_v(1))$, which is already available. Then, for each $2 \le t \le t_v$, $T^+(\pi_v[1,t])$ can be produced by merging $T^+(\pi_v[1,t-1])$ and $T^+(\pi_v(t))$ in $O(\log n)$ time. Similarly, we compute compact interval trees for t_v *suffix subpaths* of $\pi(v)$: $T^+(\pi_v[t,t_v])$ for all $t = 1,2,\ldots,t_v$, where $\pi_v[t,t_v]$ is the concatenation of the paths $\pi_v(t), \pi_v(t+1), \ldots, \pi_v(t_v)$. This can be done in $O(\log^2 n)$ time by a similar algorithm as above. Thus, the preprocessing on v takes $O(\log^2 n)$ time; the preprocessing on all leaves of $\Psi(\pi)$ takes $O(n)$ time in total.

We can now compute $T^+(i,b_v)$ in $O(\log n)$ time as follows. Recall that $\pi(i,b_v)$ is a subpath of $\pi(v)$ and b_v is the last vertex of $\pi(v)$. We first determine the subpath $\pi_v(t)$ that contains i. Let g be the last vertex of $\pi_v(t)$. We partition $\pi(i,b_v)$ into two subpaths $\pi(i,g)$ and $\pi(g+1,b_v)$, and we will compute their compact interval trees separately and then merge them to obtain $T^+(\pi(i,b_v))$. For $\pi(i,g)$, as $\pi(i,g)$ is a subpath of $\pi_v(t)$ and $|\pi_v(t)| \le \log n$, we can compute $T^+(\pi(i,g))$ from scratch in $O(\log n)$ time. For $\pi(g+1,b_v)$, observe that $\pi(g+1,b_v)$ is exactly the suffix supath $\pi_v[t+1,t_v]$, whose compact interval tree has already been computed in the preprocessing. Hence, $T^+(i,b_v)$ can be produced in $O(\log n)$ time.

Dealing with $T^+(\pi(b_v+1,k))$. To compute $T^+(b_v+1,k)$ in $O(\log n)$ time, we perform the following preprocessing, which was also used by Guibas et al. [18]. Recall that $\pi(b_v+1,k)$ is the concatenation of the canonical paths of $O(\log n)$ nodes that are right children of the nodes on the path in $\Psi(\pi)$ from v's parent to the left child of w (and these nodes themselves are not on the path). Hence, this sequence of nodes can be uniquely determined by the leaf-ancestor pair (v,w); we use $\pi_{v,w}$ to denote the above concatenated subpath of π.

Correspondingly, in the preprocessing, for each leaf v we do the following. For each ancestor w of v, we compute the compact interval tree for the subpath $\pi_{v,w}$. As v has $O(\log n)$ ancestors, computing the trees for all ancestors takes $O(\log^2 n)$ time using the merge algorithm of Lemma 1. Hence, the total preprocessing time on v is $O(\log^2 n)$, and thus the total preprocessing time on all leaves of $\Psi(\pi)$ is $O(n)$, for $\Psi(\pi)$ has $O(n/\log^2 n)$ leaves. Due to the above preprocessing, $T^+(b_v+1,k)$ is available during queries.

Wrapping up. In summary, the query time is $O(\log n)$. Comparing with the method of Guibas et al. [18], our innovation is threefold. First, we process subpaths individually to handle queries of Case 1. Second, we precompute compact interval trees for convex hulls of the prefix and suffix subpaths of $\pi(v)$ for each leaf v of $\Psi(\pi)$. Third, we use a smaller decomposition tree $\Psi(\pi)$ of only $O(n/\log^2 n)$ nodes. Theorem 3 summarizes our result.

▶ **Theorem 3.** *After all vertices of π are sorted by x-coordinate, a data structure of $O(n)$ space can be built in $O(n)$ time so that each subpath hull query can be answered in $O(\log n)$ time. The query algorithm produces a compact interval tree representing the convex hull of the query subpath, which can support all binary-search-based operations on the convex hull in $O(\log n)$ time each. These operations include (but are not limited to) the following (let π' denote the query subpath and let $H(\pi')$ be its convex hull):*

1. *Given a point, decide whether the point is in $H(\pi')$.*
2. *Given a point outside $H(\pi')$, find the two tangents from the point to $H(\pi')$.*
3. *Given a direction, find the most extreme point of π' along the direction.*
4. *Given a line, find its intersection with $H(\pi')$.*
5. *Given a convex polygon (represented in a data structure supporting binary search), decide whether it intersects $H(\pi')$, and if not, find their common tangents (both outer and inner).*

In addition, $H(\pi')$ can be output in time linear in the number of vertices of $H(\pi')$.

4 Ray-shooting

The ray-shooting problem among lines is discussed in Section 4.1. Section 4.2 is concerned with the intersection detection problem and the ray-shooting problem among segments.

4.1 Ray-shooting among lines

Given a set of n lines in the plane, we wish to build a data structure so that the first line hit by a query ray can be found efficiently. The problem is usually tackled in the dual plane, e.g., [11]. Let P be the set of dual points of the lines. In the dual plane, the problem is equivalent to the following: Given a query line l_q, a pivot point $q \in l_q$, and a rotation direction (clockwise or counterclockwise), find the first point of P hit by rotating l_q around q.

A spanning path $\pi(P)$ of P is a polygonal path connecting all points of P such that P is the vertex set of the path. Hence, $\pi(P)$ corresponds to a permutation of P. For any line l in the plane, let $\sigma(l)$ denote the number of edges of $\pi(P)$ crossed by l. The *stabbing number* of $\pi(P)$ is the largest $\sigma(l)$ of all lines l in the plane. It is known that a spanning path of P with stabbing number $O(\sqrt{n})$ always exists [8], which can be computed in $O(n^{1+\delta})$ time using Matoušek's partition tree [21] (e.g., by a method in [8]). Let $\pi'(P)$ denote such a path. Note that $\pi'(P)$ may have self-intersections. Using $\pi'(P)$, Edelsbrunner et al. [15] gave an algorithm that can produce another spanning path $\pi(P)$ of P such that the stabbing number of $\pi(P)$ is also $O(\sqrt{n})$ and $\pi(P)$ has no self-intersections (i.e., $\pi(P)$ is a simple path); the runtime of the algorithm is $O(n^{1.5})$. Below we will use $\pi(P)$ to solve our problem.

▶ **Lemma 4** (Chazelle and Guibas [7])**.** *We can build a data structure of $O(n)$ size in $O(n \log n)$ time for any simple path of n vertices, so that given any query line l_q, if l_q intersects the path in k edges, then these edges can be found in $O(k \log \frac{n}{k})$ time.*

We first build the data structure in Lemma 4 for $\pi(P)$. Then, we construct the subpath hull query data structure of Theorem 3 for $\pi(P)$. This finishes our preprocessing.

Given a query line l_q, along with the pivot q and the rotation direction, we first use Lemma 4 to find the edges of $\pi(P)$ intersecting l_q. As the stabbing number of $\pi(P)$ is $O(\sqrt{n})$, this steps finds $O(\sqrt{n})$ edges intersecting l_q in $O(\sqrt{n}\log n)$ time. Then, using these edges we can partition $\pi(P)$ into $O(\sqrt{n})$ subpaths each of which does not intersect l_q. For each subpath, we use our subpath hull query data structure to compute its convex hull in $O(\log n)$ time. Next, we compute the tangents from the pivot q to each of these $O(\sqrt{n})$ convex hulls, in $O(\log n)$ time each by Theorem 3. Using these $O(\sqrt{n})$ tangents, based on the rotation direction of l_q, we can determine the first point of P hit by l_q in additional $O(\sqrt{n})$ time. Hence, the total time of the query algorithm is $O(\sqrt{n}\log n)$.

▶ **Theorem 5.** *There exists a data structure of complexity $O(n^{1.5}, n, \sqrt{n}\log n)$ for the ray-shooting problem among lines. The preprocessing time can be reduced to $O(n\log n)$ time by a randomized algorithm while the query time is bounded by $O(\sqrt{n}\log n)$ with high probability.*

Proof. The deterministic result has been discussed above. For the randomized result, Chan [6] gave an $O(n\log n)$ time randomized algorithm to compute a spanning path $\pi''(P)$ for P such that $\pi''(P)$ is a simple path and the stabbing number of $\pi''(P)$ is at most $O(\sqrt{n})$ with high probability. After having $\pi''(P)$, we build the data structure for Lemma 4 and the subpath hull query data structure. Hence, the preprocessing takes $O(n\log n)$ time and $O(n)$ space, and the query time is bounded by $O(\sqrt{n}\log n)$ with high probability. ◀

We can extend the algorithm to obtain the result for the first-k-hit queries. The details are omitted but can be found in the full paper.

4.2 Intersection detection and ray-shooting among segments

Given a set S of n segments in the plane, an intersection detection query asks whether a query line intersects at least one segment of S. One motivation to study the problem is that it is a subproblem in our algorithm for the ray-shooting problem among segments.

To find a data structure to store the segments of S, we adapt the techniques of Overmars et al. [23] to the partition trees of Matoušek [20, 21] (to obtain the deterministic result) as well as that of Chan [6] (to obtain the randomized result). To store segments, Overmars et al. [23] used a so-called *interval partition tree*, whose underling structure is a conjugation tree of Edelsbrunner and Welzl [16]. The idea is quite natural due to the nice properties of conjugation trees: Each parent region is partitioned into exactly two disjoint children regions by a line. The drawback of conjugation trees is the slow $\widetilde{O}(n^{0.695})$ query time. When adapting the techniques to more query-efficient partition trees such as those in [6, 20, 21], two issues arise. First, each parent region may have more than two children. Second, children regions may overlap. Chan's partition tree [6] does not have the second issue while both issues appear in Matoušek's partition trees [20, 21]. As a matter of fact, the second issue incurs a much bigger challenge. In the following, we first present our randomized result by using Chan's partition tree [6], which is relatively easy, and then discuss the more complicated deterministic result using Matoušek's partition trees [20, 21].

We begin with the following lemma, which solves a special case of the problem. The lemma will be needed in both our randomized and deterministic results.

▶ **Lemma 6.** *Suppose all segments of S intersect a given line segment.*

1. *We can build a data structure of $O(n)$ space in $O(n\log n)$ time so that whether a query line intersects any segment of S can be determined in $O(\log n)$ time.*
2. *If the segments of S are nonintersecting, we can build a data structure of $O(n)$ space in $O(n\log n)$ time so that the first segment hit by a query ray can be found in $O(\log n)$ time.*

4.2.1 The randomized result

We briefly review Chan's partition tree [6] (for simplicity we only discuss it in 2D, which suffices for our problem). Chan's tree for a set P of n points, denoted by T, is a hierarchical structure by recursively subdividing the plane into triangles. Each node v of T corresponds to a triangle, denoted by $\triangle(v)$. If v is the root, then $\triangle(v)$ is the entire plane. If v is not a leaf, then v has $O(1)$ children whose triangles form a disjoint partition of $\triangle(v)$. Define $P(v) = P \cap \triangle(v)$. The set $P(v)$ is not explicitly stored at v unless v is a leaf, in which case $|P(v)| = O(1)$. The height of T is $O(\log n)$. Let $\kappa(T)$ denote the maximum number of triangles of T that are crossed by any line in the plane. Chan [6] gave an $O(n \log n)$ time randomized algorithm to compute T such that $\kappa(T)$ is at most $O(\sqrt{n})$ with high probability.

Let P be the set of the endpoints of all segments of S (so $|P| = 2n$). We first build the tree T as above. We then store the segments of S in T, as follows. For each segment s, we do the following. Starting from the root of T, for each node v, we assume that s is contained in $\triangle(v)$, which is true when v is the root. If v is a leaf, then we store s at v; let $S(v)$ denote all segments stored at v. If v is not a leaf, then we check whether s is in $\triangle(u)$ for a child u of v. If yes, we proceed on u. Otherwise, for each child u, for each edge e of $\triangle(u)$, if s intersects e, then we store s at the edge e (in this case we do not proceed to the children of u); denote by $S(e)$ the set of edges stored at e. This finishes the algorithm for storing s. As each node of T has $O(1)$ children, s is stored $O(1)$ times and the algorithm runs in $O(\log n)$ time. In this way, it takes $O(n \log n)$ time to store all segments of S, and the total sum of $|S(e)|$ and $|S(v)|$ for all triangle edges e and all leaves v is $O(n)$. In addition, $|S(v)| = O(1)$ for any leaf v, since $|P(v)| = O(1)$ and both endpoints of each segment $s \in S(v)$ are in $P(v)$.

Next, for each triangle edge e, since all edges of $S(e)$ intersect e, we preprocess $S(e)$ using Lemma 6(1). Doing this for all triangle edges e takes $O(n \log n)$ time and $O(n)$ space.

Consider a query line l. Our goal is to decide whether l intersects a segment of S. Starting from the root, we determine the set of nodes v whose triangles $\triangle(v)$ are crossed by l. For each such node v, if v is a leaf, then we check whether s intersects l for each segment $s \in S(v)$; otherwise, for each edge e of $\triangle(v)$, we use the query algorithm of Lemma 6(1) to determine whether l intersects any segment of $S(e)$. As the number of nodes v whose triangles $\triangle(v)$ crossed by l is at most $\kappa(T)$ and $S(v) = O(1)$ for each leaf v, the total time of the query algorithm is $O(\kappa(T) \cdot \log n)$. The algorithm correctness is discussed in the proof of Theorem 7.

▶ **Theorem 7.** *There exists a data structure of complexity $O(n \log n, n, \sqrt{n} \log n)$ for the intersection detection problem, where the query time holds with high probability.*

Proof. We have discussed the preprocessing time and space. Since the query time is $O(\kappa(T) \cdot \log n)$ and $\kappa(T)$ is at most $O(\sqrt{n})$ with high probability, the query time is bounded by $O(\sqrt{n} \log n)$ with high probability. For the correctness of the query algorithm, suppose l intersects a segment s, say, at a point p. If s is stored at $S(v)$ for a leaf v, then l must cross $\triangle(v)$ and thus our algorithm will detect the intersection. Otherwise, s must be stored in $S(e)$ for an edge e of a triangle $\triangle(u)$ that contains p. Since $p \in l$, l must cross $\triangle(u)$. According to our query algorithm, the query algorithm of Lemma 6(1) will be invoked on $S(e)$, and thus the algorithm will report the existence of intersection. ◀

If the segments of S are nonintersecting, by replacing Lemma 6(1) with Lemma 6(2) in both the above preprocessing and query algorithms, we can obtain the following result.

▶ **Theorem 8.** *There exists a data structure of complexity $O(n \log n, n, \sqrt{n} \log n)$ for the ray-shooting among nonintersecting segments, where the query time holds with high probability.*

To solve the ray-shooting problem among (possibly intersecting) segments, as discussed in Section 1.1, using our results in Theorems 5 and 7 and following the algorithmic scheme of Cheng and Janardan [11], we can obtain Theorem 9 (see the full paper for details).

▶ **Theorem 9.** *There exists a data structure of complexity* $O(n\log^2 n, n\log n, \sqrt{n}\log n)$ *for the ray-shooting among intersecting segments, where the query time holds with high probability.*

4.2.2 The deterministic result

To obtain the deterministic result, we resort to Matoušek's partition trees [20, 21].

An overview. To solve the simplex range searching problem (e.g., the counting problem), Matoušek built a partition tree in [20] with complexity $O(n\log n, n, \sqrt{n}(\log n)^{O(1)})$; subsequently, he presented a more query-efficient result in [21] with complexity $O(n^{1+\delta}, n, \sqrt{n})$. Ideally, we want to use his second approach. In order to achieve the $O(n^{1+\delta})$ preprocessing time, Matoušek used multilevel data structures (called partial simplex decomposition scheme in [21]). In our problem, however, the multilevel data structures do not work any more because they do not provide a "nice" way to store the segments of S. Without using multilevel data structures, the preprocessing time would be too high (indeed Matoušek [21] gave a *basic algorithm* without using multilevel data structures but he only showed that its runtime is polynomial). By a careful implementation, we can bound the preprocessing time by $O(n^2)$. To improve it, we resort to the *simplicial partition* in [20]. Roughly speaking, let P be the set of endpoints of the segments of S; we partition P into $r = \Theta(\sqrt{n})$ subsets of size $\sqrt{n}$ each, using r triangles such that any line in the plane only crosses $O(\sqrt{r})$ triangles. Then, for each subset, we apply the algorithm of [21]. This guarantees the $O(n^{1.5})$ upper bound on the preprocessing time for all subsets. To compute the simplicial partition, Matoušek [20] first provided a *basic algorithm* of polynomial time and then used other techniques to reduce the time to $O(n\log n)$. For our purpose, these techniques are not suitable (for a similar reason to multilevel data structures). Hence, we can only use the basic algorithm, whose time complexity is only shown to be polynomial in [20]. Further, we cannot directly use the algorithm because the produced triangles may overlap (the algorithm in [21] has the same issue). Nevertheless, we manage to modify the algorithm and bound its time complexity by $O(n^{1.5})$. Also, even with the above modification that avoids certain triangle overlap, using the approach in [21] directly still cannot lead to an $O(\sqrt{n}\log n)$ time query algorithm. Instead we have to further modify the algorithm (e.g., choose a different weight function).

In the following, we first describe our algorithm for computing the simplicial partition.

4.2.3 Computing a simplicial partition

Recall that P is the set of the endpoints of S and $|S| = n$. To simplify the notation, we let $|P| = n$ in the following (and thus $|S| = n/2$).

A *simplicial partion* of size m for P is a collection $\Pi = \{(P_1, \triangle_1), \ldots, (P_m, \triangle_m)\}$ with the following properties: (1) The subsets P_i's form a disjoint partition of P; (2) each $\triangle_i$ is an open triangle containing P_i; (3) $\max_{1\le i\le m}|P_i| \le 2\cdot\min_{1\le i\le m}|P_i|$; (4) the triangles may overlap and a triangle $\triangle_i$ may contain points in $P\setminus P_i$. We define the *crossing number* of Π as the largest number of triangles that are intersected by any line in the plane.

▶ **Lemma 10** ([20]). *For any integer* z *with* $2 \le z < |P|$*, there exists a simplicial partition* Π *of size* $\Theta(r)$ *for* P*, whose subsets* P_i*'s satisfy* $z \le |P_i| < 2z$*, and whose crossing number is* $O(\sqrt{r})$*, where* $r = |P|/z$*.*

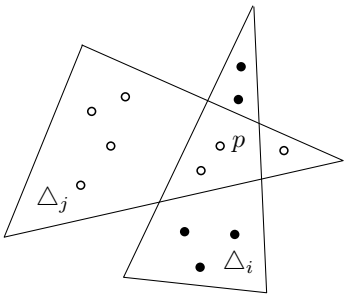


Figure 2 Illustrating the weakly-overlapped property: P_j consists of all circle points and P_i consists of all disk points. A point $p \in P_j$ is also contained in $\triangle_i$, but all points of P_i are outside $\triangle_j$.

To compute such a simplicial partition as in Lemma 10, Matoušek [20] first presented a basic algorithm whose runtime is polynomial and then improved the time to $O(n \log n)$ by other techniques. As discussed before, the techniques are not suitable for our purpose and we can only use the basic algorithm. In addition, the above property (4) prevents us from using the partition directly. Instead we use an *enhanced simplicial partition* with the following modified/changed properties. In property (2), each $\triangle_i$ is either a triangle or a convex quadrilateral; we now call $\triangle_i$ a *cell*. In property (4), the cells may still overlap, and a cell $\triangle_i$ may still contain points in $P \setminus P_i$; however, if $\triangle_i$ contains a point $p \in P_j$ with $j \neq i$, then all points of P_i are outside $\triangle_j$ (e.g., see Fig. 2). This modified property (4), which we call *the weakly-overlapped property*, is the key to guarantee the success of our approach. We use convex quadrilaterals instead of only triangles to make sure the weakly-overlapped property can be achieved. The crossing number of the enhanced partition is defined as the largest number of cells that are intersected by any line in the plane. By modifying Matoušek's basic algorithm [20], we can compute in $O(n^{1.5})$ time an enhanced simplicial partition $\Pi = \{(P_1, \triangle_1), \ldots, (P_m, \triangle_m)\}$ with $m = \Theta(r)$, which satisfies the property of Lemma 10 with $z = \sqrt{n}$ (and thus $r = \sqrt{n}$); in particular, the crossing number of Π is $O(\sqrt{r})$. The algorithm is omitted but can be found in the full paper.

Storing the segments in Π. For each segment s of S, if both endpoints of s are in the same subset P_i of Π, then s is in the cell $\triangle_i$ and we store s in $\triangle_i$; let S_i denote the set of segments stored in $\triangle_i$. Otherwise, let P_i and P_j be the two subsets that contain the endpoints of s, respectively. The weakly-overlapped property of Π leads to the following observation.

▶ **Observation 11.** *The segment s intersects the boundary of at least one cell of $\triangle_i$ and $\triangle_j$.*

By Observation 11, we find a cell $\triangle$ of $\triangle_i$ and $\triangle_j$ whose boundary intersects s. Let e be an edge of $\triangle$ that intersects s. We store s at e; let $S(e)$ denote the set of segments of S that are stored at e. In this way, each segment of S is stored exactly once. Next, for each cell $\triangle \in \Pi$ and for each edge e of $\triangle$, we preprocess $S(e)$ using Lemma 6. With Π, the above preprocessing on S takes $O(n \log n)$ time and $O(n)$ space.

We further have the following Lemma 12, whose proof can be found in the full paper.

▶ **Lemma 12.**

1. *For each subset P_i of Π, with $O(|P_i|^2)$ time and $O(|P_i|)$ space preprocessing, we can decide whether a query line intersects any segment of S_i in $O(\sqrt{|P_i|} \log |P_i|)$ time.*
2. *If segments of S_i are nonintersecting, with $O(|P_i|^2)$ time and $O(|P_i|)$ space preprocessing, we can find the first segment of S_i hit by a query ray in $O(\sqrt{|P_i|} \log |P_i|)$ time.*

We preprocess each P_i using Lemma 12. As Π has $O(\sqrt{n})$ subsets P_i and the size of each P_i is $O(\sqrt{n})$, the total preprocessing time is $O(n^{1.5})$ and the total space is $O(n)$.

Answering queries. Consider a query line ℓ. First, for each cell $\triangle_i$ of Π, for each edge e of $\triangle_i$, we determine whether ℓ intersects a segment of $S(e)$, in $O(\log n)$ time by Lemma 6(1). As Π has $\Theta(\sqrt{n})$ cells and each cell has at most four edges, the total time of this step is $O(\sqrt{n}\log n)$. Second, by checking every cell of Π, we find those cells that are crossed by ℓ. For each such cell $\triangle_i$, by Lemma 12(1), we determine whether ℓ intersects any segment of S_i in $O(n^{1/4}\log n)$ time, for $|P_i| = \Theta(\sqrt{n})$. As ℓ can cross at most $O(n^{1/4})$ cells of Π, this step takes $O(\sqrt{n}\log n)$ time. Hence, the query time is $O(\sqrt{n}\log n)$.

If the segments of S_i are nonintersecting, the ray-shooting query algorithm is similar. We can thus obtain our results for the segment intersection and the ray-shooting problems.

References

1. P.K. Agarwal. Ray shooting and other applications of spanning trees with low stabbing number. *SIAM Journal on Computing*, 21:540–570, 1992.
2. P.K. Agarwal and M. Sharir. Applications of a new space-partitioning technique. *Discrete and Computational Geometry*, 9(1):11–38, 1993.
3. R. Bar-Yehuda and S. Fogel. Variations on ray shootings. *Algorithmica*, 11:133–145, 1994.
4. B. Becker, P.G. Franciosa, S. Gschwind, S. Leonardi, T. Ohler, and P. Widmayer. Enclosing a set of objects by two minimum area rectangles. *Journal of Algorithms*, 21:520–541, 1996.
5. B. Becker, P.G. Franciosa, S. Gschwind, T. Ohler, T. Ohler, G. Thiemt, and P. Widmayer. An optimal algorithm for approximating a set of rectangles by two minimum area rectangles. In *Workshop on Computational Geometry*, pages 13–25, 1991.
6. T.M. Chan. Optimal partition trees. *Discrete and Computational Geometry*, 47:661–690, 2012.
7. B. Chazelle and L. Guibas. Fractional cascading: II. Applications. *Algorithmica*, 1(1):163–191, 1986.
8. B. Chazelle and E. Welzl. Quasi-optimal range searching in spaces of finite VC-dimension. *Discrete and Computational Geometry*, 4(5):467–489, 1989.
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. S.W. Cheng and R. Janardan. Algorithms for ray-shooting and intersection searching. *Journal of Algorithms*, 13:670–692, 1992.
12. T. Christ, M. Hoffmann, Y. Okamoto, and T. Uno. Improved bounds for wireless localization. *Algorithmica*, 57:499–516, 2010.
13. B.-S. Dai, M.-J. Kao, and D.T. Lee. Optimal time-convex hull under the L_p metrics. In *Proceedings of the 13rd Algorithms and Data Structures Symposium (WADS)*, pages 268–279, 2013.
14. J. Driscoll, N. Sarnak, D. Sleator, and R.E. Tarjan. Making data structures persistent. *Journal of Computer and System Sciences*, 38(1):86–124, 1989.
15. H. Edelsbrunner, L. Guibas, J. Hershberger, R. Seidel, M. Sharir, J. Snoeyink, and E. Welzl. Implicitly representing arrangements of lines or segments. *Discrete and Computational Geometry*, 4:433–466, 1989.
16. H. Edelsbrunner and E. Welzl. Halfplanar range search in linear space and $O(n^{0.695})$ query time. *Information Processing Letters*, 23:289–293, 1986.
17. R.L. Graham and F.F. Yao. Finding the convex hull of a simple polygon. *Journal of Algorithms*, 4:324–331, 1983.

18 L. Guibas, J. Hershberger, and J. Snoeyink. Compact interval trees: A data structure for convex hulls. *International Journal of Computational Geometry and Applications*, 1(1):1–22, 1991. First appeared in SODA 1990.

19 L. Guibas, M. Overmars, and M. Sharir. Intersecting line segments, ray shooting, and other applications of geometric partitioning techniques. In *Proceedings of the 1st Scandinavian Workshop on Algorithm Theory (SWAT)*, pages 64–73, 1988.

20 J. Matoušek. Efficient partition trees. *Discrete and Computational Geometry*, 8(3):315–334, 1992.

21 J. Matoušek. Range searching with efficient hierarchical cuttings. *Discrete and Computational Geometry*, 10(1):157–182, 1993.

22 A. Melkman. On-line construction of the convex hull of a simple polygon. *Information Processing Letters*, 25:11–12, 1987.

23 M.H. Overmars, H. Schipper, and M. Sharir. Storing line segments in partition trees. *BIT Numerical Mathematics*, 30:385–403, 1990.

24 H. Wagener. Optimal parallel hull construction for simple polygons in $O(\log \log n)$ time. In *Proceedings of the 33rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 593–599, 1992.

25 H. Wang and W. Zhang. On top-k weighted sum aggregate nearest and farthest neighbors in the L_1 plane. *International Journal of Computational Geometry and Applications*, 29:189–218, 2019.

GPU-Accelerated Computation of Vietoris-Rips Persistence Barcodes

Simon Zhang
Department of Computer Science and Engineering, The Ohio State University, Columbus, OH, USA
zhang.680@osu.edu

Mengbai Xiao
Department of Computer Science and Engineering, The Ohio State University, Columbus, OH, USA
xiao.736@osu.edu

Hao Wang
Department of Computer Science and Engineering, The Ohio State University, Columbus, OH, USA
wang.2721@osu.edu

Abstract

The computation of Vietoris-Rips persistence barcodes is both execution-intensive and memory-intensive. In this paper, we study the computational structure of Vietoris-Rips persistence barcodes, and identify several unique mathematical properties and algorithmic opportunities with connections to the GPU. Mathematically and empirically, we look into the properties of apparent pairs, which are independently identifiable persistence pairs comprising up to 99% of persistence pairs. We give theoretical upper and lower bounds of the apparent pair rate and model the average case. We also design massively parallel algorithms to take advantage of the very large number of simplices that can be processed independently of each other. Having identified these opportunities, we develop a GPU-accelerated software for computing Vietoris-Rips persistence barcodes, called Ripser++. The software achieves up to 30x speedup over the total execution time of the original Ripser and also reduces CPU-memory usage by up to 2.0x. We believe our GPU-acceleration based efforts open a new chapter for the advancement of topological data analysis in the post-Moore's Law era.

2012 ACM Subject Classification Theory of computation → Massively parallel algorithms; Software and its engineering → Massively parallel systems; Theory of computation → Randomness, geometry and discrete structures

Keywords and phrases Parallel Algorithms, Topological Data Analysis, Vietoris-Rips, Persistent Homology, Apparent Pairs, High Performance Computing, GPU, Random Graphs

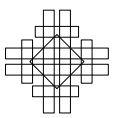
Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.70

Related Version A full version of the paper is available at `https://arxiv.org/abs/2003.07989`.

Supplementary Material Open Source Software: `https://www.github.com/simonzhang00/ripser-plusplus`

Funding This work has been partially supported by the National Science Foundation under grants CCF-1513944, CCF-1629403, CCF-1718450, and an IBM Fellowship.

Acknowledgements We would like to thank Ulrich Bauer for technical discussions on Ripser and Greg Henselman for discussions on Eirene. We also thank Greg Henselman, Matthew Kahle, and Cheng Xin on discussions about probability and apparent pairs. We acknowledge Birkan Gokbag for his help in developing Python bindings for Ripser++. We appreciate the constructive comments and suggestions of the anonymous reviewers. Finally, we are grateful for the insights and expert judgement in many discussions with Tamal Dey.



1 Introduction

Topological data analysis (TDA) [12] is an emerging field in the era of big data, which has a strong mathematical foundation. As a subfield of TDA, persistent homology seeks to find topological or qualitative features of data (usually represented by a finite metric space). It has many applications, such as in neural networks [25], sensor networks [15], bioinformatics [14], deep learning [28], manifold learning [36], and neuroscience [32]. One of the most popular and useful topological signatures persistent homology can compute are Vietoris-Rips barcodes. There are two challenges to Vietoris-Rips barcode computation. The first one is its highly computing- and memory-intensive nature in part due to the exponentially growing number of simplices it must process. The second one is its irregular computation patterns with high dependencies such as its matrix reduction step [47]. Therefore, sequential computation is still the norm in computing persistent homology. There are several CPU-based software packages in sequential mode for computing persistent homology [8, 9, 27, 33, 5]. Ripser [5, 46] is a representative and computationally efficient software specifically designed to compute Vietoris-Rips barcodes, achieving state of the art performance [6, 37] by using effective and mathematically based algorithmic optimizations.

The usage of hardware accelerators like GPU is inevitable for computation in many areas. To continue advancing the computational geometry field, we must include hardware-aware algorithmic efforts. The ending of Moore's law [45] and the termination of Dennard scaling [19] technically limits the performance improvement of general-purpose CPUs [23]. The computing ecosystem is rapidly evolving from conventional CPU computing to a new disruptive accelerated computing environment where hardware accelerators such as GPUs play the main roles of computation for performance improvement.

Our goal in this work is to develop GPU-accelerated computation for Vietoris-Rips barcodes, not to only significantly improve the performance, but also to lead a new direction in computing for topological data analysis. We have looked into the two major computational components of Vietoris-Rips barcodes, namely filtration construction with clearing and matrix reduction, and identified hidden parallelisms and data locality. Having laid mathematical foundations, we develop parallel algorithms for each component.

Our contributions explained in this paper are as follows:

1. We introduce and prove the Apparent Pairs Lemma for Vietoris-Rips barcode computation. It has a natural algorithmic connection to the GPU. We furthermore prove theoretical bounds on the number of so-called "apparent pairs".

2. We design and implement hardware-aware massively parallel algorithms that accelerate the two major computation components of Vietoris-Rips barcodes as well as a data structure for persistence pairs for matrix reduction.

3. We perform extensive experiments justifying our algorithms' computational effectiveness as well as dissecting the nature of Vietoris-Rips barcode computation.

4. We achieve up to 30x speedup over the original Ripser software and, surprisingly, up to 2.0x CPU memory efficiency and requires, at best, 60% of the CPU memory used by Ripser on the GPU device memory.

5. Ripser++ is an open source software in the public domain to serve the TDA community and relevant application areas.

2 Preliminaries

2.1 Persistent Homology

Computing Vietoris Rips barcodes involves the measurement of "birth" and "death" [4, 8, 21] of topological features as we grow combinatorial objects on top of the data with respect to some real-valued time parameter. We call the pairs of birth and death times with respect to the combinatorial objects "persistence barcodes." Persistence barcodes give a topological signature of the original data (finite metric space) and have many further applications with statistical meaning in TDA [1, 11, 24, 39].

2.2 Vietoris-Rips Filtrations

When computing persistent homology, data is usually represented by a finite metric space X, a finite set of points with real-valued distances determined by an underlying metric d between each pair of points. X is defined by its distance matrix D, which is defined as $D[i,j]= d(\text{point } i, \text{point } j)$ with $D[i,i] = 0$.

Define an (abstract) simplicial complex K as a collection of simplices closed under the subset relation, where a simplex s is defined as a subset of X. We call a "filtration" as a totally ordered sequence of growing simplicial complexes. A particularly popular and useful [3] filtration is a Vietoris-Rips filtration. See Figure 1 for an illustration. Let

$$Rips_t(X) = \{\emptyset \neq s \subset X \mid diam(s) \leq t\}, \tag{1}$$

where $t \in \mathbb{R}$ and $diam(s)$ is the maximum distance between pairs of points in s as determined by D. The Vietoris-Rips filtration is defined as the sequence: $(Rips_t(X))_t$, indexed by growing $t \in \mathbb{R}$ where $Rips_t(X)$ strictly increases in cardinality for growing t.

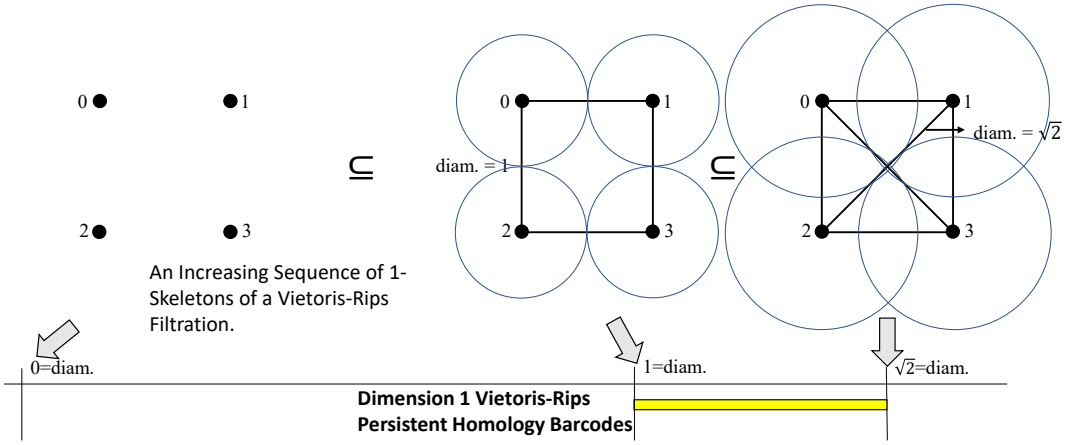


Figure 1 A filtration on an example finite metric space of four points of a square in the plane. The 1-skeleton at each diameter value where "birth" or "death" occurs is shown. The 1 dimensional Vietoris-Rips barcode is below it: a 1-cycle is "born" at diameter 1 and "dies" at diameter $\sqrt{2}$.

2.2.1 The Simplex-wise Refinement of the Vietoris-Rips Filtration

For computation (see Section 2.4) of Vietoris-Rips persistence barcodes, it is necessary to construct a simplex-wise refinement S of a given filtration F. F is equivalent to a partial order on the simplices of K, where K is the largest simplicial complex of F. To construct S,

we assign a total order on the simplices $\{s_i\}_{i=1..|K|}$ of K, extending the partial order induced by F so that the increasing sequence of subcomplexes $S = (\bigcup_{i \leq j} \{s_i\})_{j=1..|K|}$ ordered by inclusion grows subcomplexes by one simplex at a time. There are many ways to order a simplex-wise refinement S of F [32]; in the case of Ripser and Ripser++, we use the following simplex-wise filtration ordering criterion on simplices:

1. by increasing diameter: denoted by $diam(s)$,
2. by increasing dimension: denoted by $dim(s)$, and
3. by decreasing combinatorial index: denoted by $cidx(s)$ (equivalently, by decreasing lexicographical order on the decreasing sequence of vertex indices) [41, 31, 38].

Every simplex in the simplex-wise refinement will correspond to a "column" in a uniquely associated (co-)boundary matrix for persistence computation. Thus we will use the terms "column" and "simplex" interchangeably to explain our algorithms.

Define *persistence pairs* as a pair of "birth" and "death" simplices from K [22].

2.3 The Combinatorial Number System

We use the combinatorial number system to encode simplices. The combinatorial number system is simply a bijection between ordered fixed-length $\mathbb{N}$-tuples and $\mathbb{N}$. It provides a minimal representation of simplices and an easy extraction of simplex facets (see Algorithm 3), cofacets, and vertices. When not mentioned, we assume that all simplices are encoded by their combinatorial index. The bijection is stated as follows:

$$\mathbb{N}^{d+1} \ni (v_d ... v_0) \iff \binom{v_d}{d+1} + ... + \binom{v_0}{1} \in \mathbb{N}, v_d > ... > v_0 \geq 0. \tag{2}$$

For a proof of this bijection see [41, 31, 38].

2.4 Computation

The general computation of persistent barcodes involves two inter-relatable stages. One stage is to construct a simplex-wise refinement [9] of the given filtration. The other stage is to "reduce" the corresponding boundary matrix by a "standard algorithm" [21]. In Algorithm 1, let $low_R(j)$ be the maximum nonzero row of column j, -1 if column j is zero for a given matrix R. For fully reduced R, $(low_R(j), j)$ over all j are in bijection with *persistence pairs*.

Algorithm 1 Standard Persistent Homology Computation.

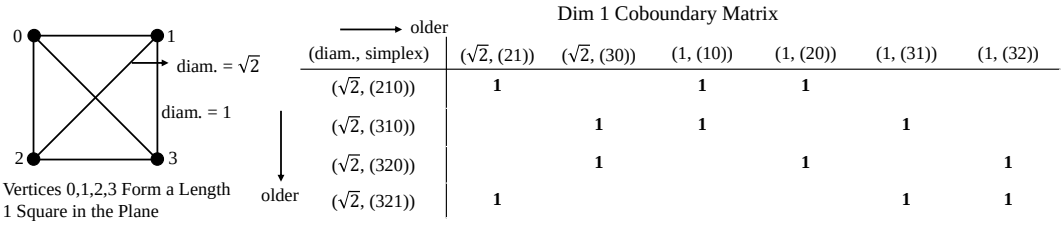
Require: filtered simplicial complex $\boldsymbol{K}$
Ensure: $\boldsymbol{P}$ persistence barcodes

```
F ← F_K                                   ▷ let F be the filtration of K
S ←simplex-wise-refinement(F)             ▷ F = S ∘ r where r is injective
R ← ∂(S)
for every column j in R do                ▷ begin the standard matrix reduction algorithm
    while ∃ k < j s.t. low_R(j)=low_R(k) do
        column j ← column k + column j
    if low_R(j) ≠ −1 then
        P ← P ∪ r^{-1}([low(j), j))       ▷ we call the pair (low(j), j) a pivot in the matrix R.
```

The construction stage can be optimized [6, 29, 33, 44, 48]. Furthermore, all persistent homology software are based on the standard algorithm [2, 6, 8, 9, 26, 33, 35, 47].

2.4.1 The Coboundary Matrix

We compute cohomology [17, 16, 20] in Ripser++, like in Ripser, for performance reasons specific to Rips filtrations mentioned in [6]. Thus we introduce the coboundary matrix of a simplex-wise filtration. This is defined as the matrix of coboundaries (each column is made up of the cofacets of the corresponding simplex) where the columns/simplices are ordered in reverse to the order given in Section 2.2.1 (see [16]). If certain columns can be zeroed/cleared [13] in the coboundary matrix, we will still denote the cleared matrix as a coboundary matrix since the matrix reduction does nothing on zero columns (see Algorithm 1).



(diam., simplex)	(√2, (21))	(√2, (30))	(1, (10))	(1, (20))	(1, (31))	(1, (32))
(√2, (210))	**1**		**1**	**1**		
(√2, (310))		**1**	**1**		**1**	
(√2, (320))		**1**		**1**		**1**
(√2, (321))	**1**				**1**	**1**

Figure 2 The full 1-skeleton for the point cloud of Figure 1. Its 1-dimensional coboundary matrix is shown on the right. Let $(e, (a_d...a_0))$ be a d-dimensional simplex with vertices $a_d...a_0$ and diameter $e \in \mathbb{R}^+$. For example, simplex (1,(10)) has vertices 1 and 0 with diameter 1. The order of the columns/simplices is the reverse of the simplex-wise refinement of the Vietoris-Rips filtration.

2.5 Computation in Ripser

The sequential computation in Ripser follows the two stages given in Algorithm 1, however with four key optimizations [6]. We use and build on top of all of these four optimizations.

1. The clearing lemma [7, 13, 47],
2. Computing cohomology [16, 20], with a low complexity 0-dim. persistence algorithm,
3. Implicit matrix reduction [6], and
4. The emergent pairs lemma [6].

3 Mathematical and Algorithmic Foundations in GPU Acceleration

3.1 Overview of GPU-Accelerated Computation

Figure 3(a) shows a high-level structure of Ripser, which processes simplices dimension by dimension. In each dimension starting at dimension 1, the filtration is constructed and the clearing lemma is applied followed by a sort operation. The simplices to reduce are further processed in the matrix reduction stage, where the cofacets of each simplex are enumerated to form coboundaries and the column addition is applied iteratively.

Running Ripser intensively on many datasets, we have observed its inefficiency on CPU. There are two major performance issues. First, in each dimension, the matrix reduction of Ripser uses an enumeration-and-column-addition style to process each simplex. Although the computation is highly dependent among columns, a large percentage of columns (see Table 1 in Section 5) do NOT need the column addition. Only the cofacet enumeration and a possible persistence pair insertion (into the hashmap of Ripser) are needed on these columns. In Ripser, a subset of these columns are identified by the "emergent pair" lemma [6] as columns containing "shortcut pairs". Ripser follows the serial framework of Figure 3(a) to process these columns one by one, where rich parallelisms are hidden. Second, in the filtration

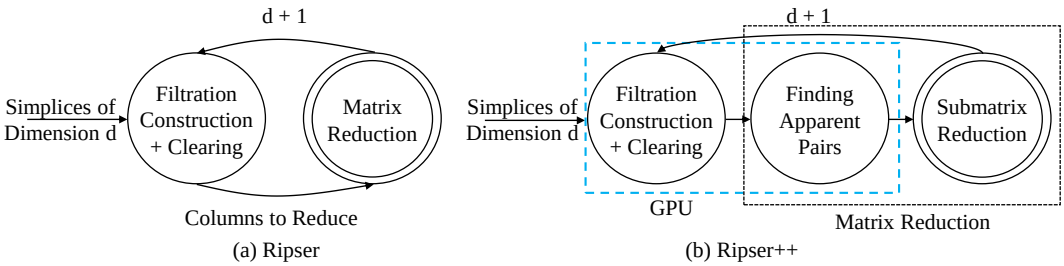


Figure 3 A High-level computation framework comparison of Ripser and Ripser++ starting at dimension $d \geq 1$. Ripser follows the two stage standard persistence computation of sequential Algorithm 1 with optimizations. In contrast, Ripser++ finds the hidden parallelism inside Vietoris-Rips barcode computation, extracts "Finding Apparent Pairs" out from Matrix Reduction, and parallelizes "Filtration Construction with Clearing" on GPU. These two steps are designed and implemented with new parallel algorithms on GPU, as shown in (b) with the dashed rectangle.

construction with clearing stage, applying the clearing lemma and predefined threshold is independent among simplices. Furthermore, on GPU the performance of sorting for filtration construction with clearing can be further improved due to the massive parallelism and the high memory bandwidth of GPU [40, 42].

We aim to turn these hidden parallelisms and data localities into reality for accelerated computation by GPU for high performance. Utilizing SIMT (single instruction, multiple threads) parallelism and achieving coalesced device memory accesses are our major intentions because they are unique advantages of GPU architecture. Our efforts are based on mathematical foundation, algorithms development, and effective implementations interacting with GPU hardware. Figure 3(b) gives the high-level structure of Ripser++, showing the components of Vietoris-Rips barcode computation offloaded to GPU. We will elaborate on the computation mathematically and algorithmically in this section.

3.2 Matrix Reduction

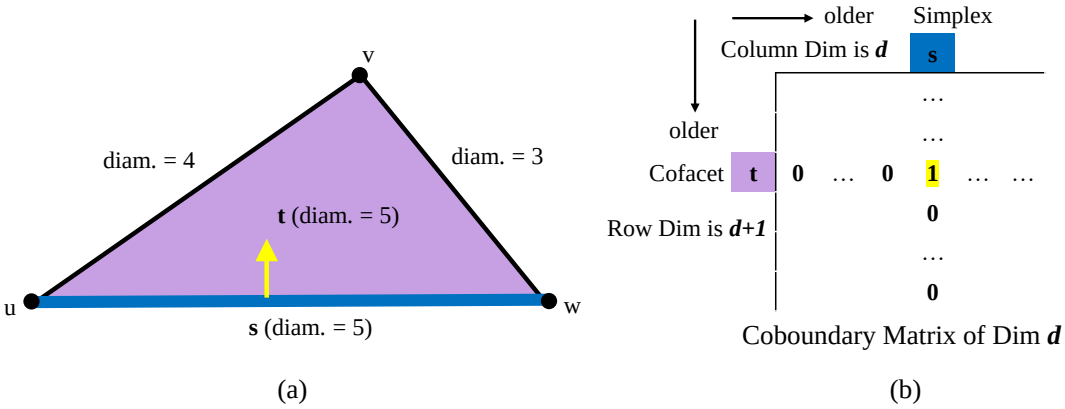
Matrix reduction is a fundamental component of computing Rips barcodes. Its computation can be highly skewed [47], involving very few columns for column additions. We prove and present the Apparent Pairs Lemma and a GPU algorithm to find apparent pairs in an implicitly represented coboundary matrix. We then design and implement a 2-layer data structure that optimizes the performance of the hashmap storing persistence pairs for subsequent matrix reduction on the non-apparent columns, which we term "submatrix reduction".

3.2.1 The Apparent Pairs Lemma

▶ **Definition 1.** *A pair of simplices (s, t) is an apparent pair iff:*

1. *s is the youngest facet of t and*
2. *t is the oldest cofacet of s.*

We will use the simplex-wise order of Section 2.2.1 for Definition 1. In a (co-)boundary matrix, a nonzero entry having all zeros to its left and below is equivalent to an apparent pair. We call a column containing such a nonzero entry as an apparent column. An example of an apparent pair geometrically and computationally is shown in Figure 4. Furthermore, apparent pairs have zero persistence in Rips filtrations by property 1 of Definition 1.



(a) (b)

Figure 4 (a) A dimension 1 0-persistence apparent pair (s, t) on a single 2-dimensional simplex. s is an edge of diameter 5 and t is a cofacet of s with diameter 5. The light arrow denotes the pairing between s and t. (b) In the dimension d coboundary matrix, (s, t) is an apparent pair iff entry (t, s) has all zeros to its left and below. See Figure 2 for an example coboundary matrix.

In the explicit matrix reduction, where every column is stored in memory, it is easy to determine apparent pairs by checking the positions of s and t in the (co-)boundary matrix. However, in the implicit matrix reduction used in Ripser and Ripser++, we need to enumerate cofacets t from s and facets s from t at runtime. We first notice a property of the facets of a cofacet t of simplex s where $diam(s) = diam(t)$.

▶ **Proposition 2.** *Let t be the cofacet of simplex s with $diam(s) = diam(t)$. s' is a strictly younger facet of t than s iff*

1. *$diam(s') = diam(s) = diam(t)$ and*
2. *$cidx(s') < cidx(s)$. (s' is strictly lexicographically smaller than s)*

Proof. ($\Longrightarrow$) s' as a facet of t implies that $diam(s') \leq diam(t) = diam(s)$. If s' is strictly younger than s, then $diam(s') \geq diam(s)$. Thus 1. $diam(s') = diam(s) = diam(t)$. Furthermore, if s' is strictly younger than s and $diam(s') = diam(s)$, then the only way for s' to be younger than s is if 2. $cidx(s') < cidx(s)$.

($\Longleftarrow$) If $diam(s') = diam(s) = diam(t)$ and $cidx(s') < cidx(s)$ then certainly s' is a strictly younger facet of t than s is as a facet of t. ◀

We propose the following lemma to find apparent pairs:

▶ **Lemma 3** (The Apparent Pairs Lemma). *Given simplex s and its cofacet t,*

1. *t is the lexicographically greatest cofacet of s with $diam(s) = diam(t)$ and*
2. *no facet s' of t is strictly lexicographically smaller than s with $diam(s') = diam(s)$,*

iff (s, t) is an apparent pair.

Proof. ($\Longrightarrow$) Since $diam(t) \geq diam(s)$ for all cofacets t, Condition 1 is equivalent to having chosen the cofacet t of s of minimal diameter at the largest combinatorial index, by the filtration ordering we have defined in Section 2.2.1; this implies t is the oldest cofacet of s.

Assuming condition 1, by the negation of the iff in Proposition 2, there are no simplices s' with $diam(s') = diam(s) = diam(t)$ and $cidx(s') < cidx(s)$ iff s is the youngest facet of t.

($\Longleftarrow$) If $diam(t) > diam(s)$ then there exists a younger s' with same cofacet t and thus s is not the youngest facet of t. Thus (s, t) being an apparent pair implies Condition 1. Furthermore, (s, t) being apparent with Condition 1 implies Condition 2 by Proposition 2.

Thus (Conditions 1 and 2) is equivalent to Definition 1. ◀

▶ **Corollary 4.** *The Apparent Pairs Lemma can be applied for massively parallel operations on every column s of the coboundary matrix.*

Proof. Notice we may generate the cofacets of simplex s and facets of cofacet t of s independently with other simplices $s' \neq s$. ◀

▶ Remark 5. The effectiveness of the Apparent Pairs Lemma hinges on an important empirical fact and common dataset property: namely that there are a lot of apparent pairs [47, 6]. By Table 1 in Section 5, in many datasets up to 99% of persistence pairs are apparent pairs. Further theoretical results are in Section 3.2.3 and more results are illustrated by Figure 10.

3.2.2 Finding Apparent Pairs in Parallel on GPU

Based on Lemma 3, finding apparent pairs from a cleared coboundary matrix without explicit coboundaries becomes feasible. There is no dependency for identifying an apparent pair as Corollary 4 states, giving us a unique opportunity to develop an efficient GPU algorithm by exploiting the massive parallelism.

Algorithm 2 Finding Apparent Pairs on GPU.

Require: $\mathbf{C}$: the simplices to reduce; $vertices(\cdot)$: the vertices of a simplex; $diam(\cdot)$: the diameter of a simplex; $cidx(\cdot)$: the combinatorial index of a simplex; $dist(\cdot)$: the distance between two vertices; $enumerate\text{-}facets(\cdot)$: enumerates facets of a simplex. ▷ global to all threads
tid: the thread id. ▷ local to each thread
Ensure: $\mathbf{A}$: the apparent pair set from the coboundary matrix of dimension dim.

```
s ← C[tid]                          ▷ each thread fetches a distinct simplex from the set of simplices
V ← vertices(s)                     ▷ this only depends on the combinatorial index of s
for each cofacet t of s in lexicographically decreasing order do
    for v' in V do                  ▷ t and s differ by one vertex v
        diam(t) ← max(dist(v', v), diam(s))   ▷ calculate the diameter of t
    if diam(t) = diam(s) then       ▷ t is the oldest cofacet of s
        S ← ∅
        enumerate-facets(t, S)      ▷ S are facets of t in lexicographical increasing order
        for s' in S do
            if diam(s') = diam(s) then
                if cidx(s') = cidx(s) then      ▷ s is the youngest facet of t
                    A ← A ∪ {(s, t)}
                return              ▷ exit if (s, t) is apparent or if s' is strictly younger than s
```

Algorithm 2 shows how a GPU kernel finds all apparent pairs in a massively parallel manner. A GPU thread fetches a distinct simplex from an ordered array of simplices in GPU device memory, and checks if this simplex and one of its cofacets can form an apparent pair. Lastly, it inserts into a data structure containing all apparent pairs in the GPU device memory. The complexity of one GPU thread is $\mathrm{O}(log(n) \cdot (d{+}1){+}(n\text{-}d\text{-}1) \cdot (d{+}1))$, in which n is the number of points and d is the dimension of the simplex s. The first term represents a binary search for $d{+}1$ simplex vertices from a combinatorial index, and the second term says the algorithm checks at most $d{+}1$ facets of all $n\text{-}d\text{-}1$ cofacets of the simplex s.

Enumerating cofacets/facets in a lexicographically decreasing/increasing order is substantial to our algorithm. Algorithm 3 shows how to enumerate facets of a simplex. A facet of a simplex is enumerated by removing one of its vertices. Due to properties of the combinatorial number system, if the removed vertex index follows a decreasing order, the combinatorial indices of the generated facets will lexicographically increase.

Algorithm 3 Enumerating Facets of a Simplex.

Require: $\boldsymbol{X} = \{0..n-1\}$: n points of a finite metric space; s: a simplex with vertices in $\boldsymbol{X}$; $vertices(\cdot)$: the vertices of a simplex; $cidx(\cdot)$: the combinatorial index of a simplex; $last(\cdot)$: the last simplex of a sequence.
Ensure: $\boldsymbol{S}$: the facets of s in lexicographically increasing order.

```
1: procedure ENUMERATE-FACETS(s, S)
2:     V ← vertices(s)
3:     prev ← ∅; k ← |V|
4:     for v ∈ V ⊂ X in decreasing order do
5:         if prev ≠ ∅ then
6:             cidx(s') ← cidx(last(S)) − C(v,k) + C([prev],k)   ▷ [x] is the only element of singleton x
7:         else
8:             cidx(s') ← cidx(last(S)) − C(v,k)
           append(S, s')                                          ▷ append s' to the end of S
9:         prev ← {v}; k ← k − 1
```

3.2.3 Theoretical Bounds on the Number of Apparent Pairs

Besides the existence of a large number of apparent pairs empirically (see Section 5), we show theoretically that there are tight upper and lower bounds to the number of apparent pairs. The proof of Theorem 6 is in this paper's full version.

▶ **Theorem 6** (Bounds on the Number of Apparent Pairs). *The ratio of the number of d-dimensional apparent pairs to the number of d-dimensional simplices for a full Rips-filtration on a n point $(d+1)$-skeleton where all d-dimensional simplices s' containing maximum vertex $n-1$ have $diam(s') \leq diam(s)$ for all d-dimensional simplices s not containing vertex $n-1$:*

- *theoretical upper bound: $(n-d-1)/n$; (tight for all $n \geq d+1$ and $d \geq 1$).*
- *theoretical lower bound: $1/(d+2)$; (tight for $d \geq 1$).*

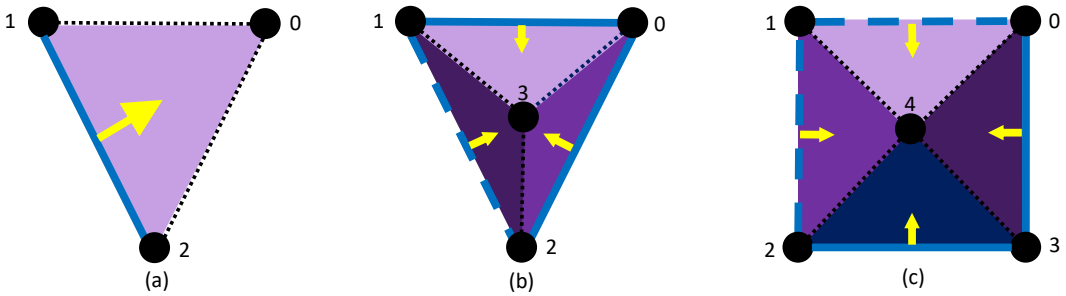


Figure 5 Geometric interpretation of the theoretical upper bound in Theorem 6. Edge distances are not to scale. (a),(b),(c) (constructed in this order) show the apparent pairs for $d = 1$ on the planar cone graph centered around the newest apex point: $n-1$ for $n = 3, 4, 5$ points. The yellow arrows denote the apparent pairs: blue edges paired with purple or navy triangles. The dashed (not dotted) blue edges denote apparent edges from the previous $n-1$ point subcomplex.

4 GPU and System Kernel Development for Ripser++

4.1 Core System Optimizations

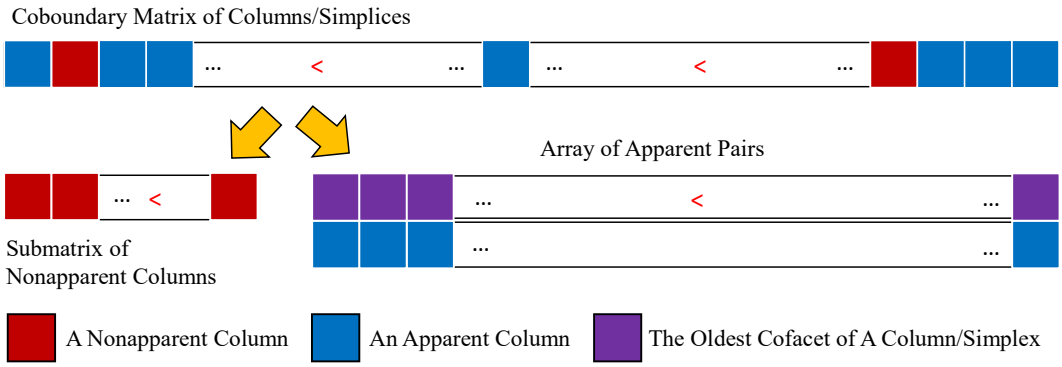


Figure 6 After finding apparent pairs, we partition the coboundary matrix columns into apparent and nonapparent columns. The apparent columns are sorted by the coboundary matrix row (the oldest cofacet of an apparent column) and stored in an array of pairs; while the nonapparent columns are collected and sorted by coboundary matrix order in another array for submatrix reduction.

The expected performance gain of finding apparent pairs on GPU comes from not only the parallel computation on thousands of cores but also the concurrent memory accesses at a high bandwidth, where the apparent pairs can be efficiently aggregated. In a sequential context, an apparent pair (a row index and a column index) of the coboundary matrix may be kept in a hashmap as a key-value pair with the complexity of $O(1)$. However building a hashmap is not as fast as constructing a sorted continuous array [30] in parallel. So in our implementation, the apparent pairs are represented by a key-value pair (t, s) where t is the oldest cofacet of simplex s and stored in an aligned continuous array of pairs. This slightly lowers the read performance because we need a binary search to locate a desired apparent pair. But this is cost-effective since the number of insertions of apparent pairs are actually three orders of magnitude higher than that of reads (See Table 3 in Section 5) after finding apparent pairs. Figure 6 presents how we collect apparent pairs on GPU, where each thread works on a column of coboundary matrix and writes to the output array in parallel.

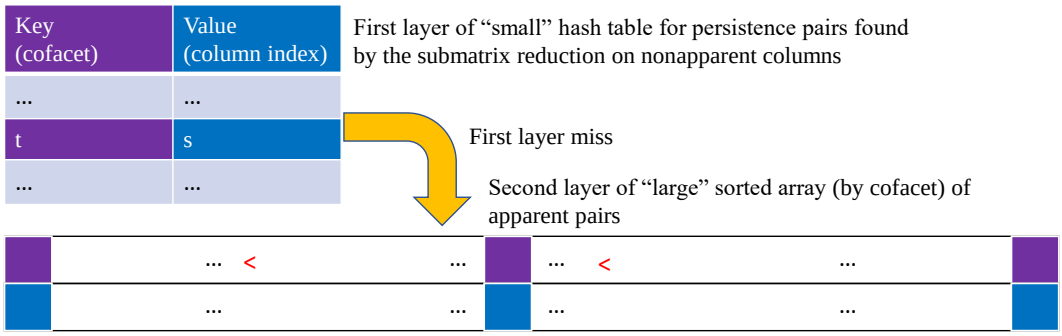


Figure 7 Two-layer data structure for persistence pairs. Apparent pair insertion to the second layer of the data structure is illustrated in Figure 6, followed by persistence pair insertion to a small hashmap during the submatrix reduction on CPU. A key-value read during submatrix reduction involves atmost two steps: first, check the hashmap; second, if the key is not found in the hashmap, use a binary search over the sorted array to locate the key-value pair (see the arrow in the figure).

We add a hashmap as one more layer to store persistence pairs discovered during the submatrix reduction. Figure 7 explains such a design in details.

4.2 Filtration Construction with Clearing

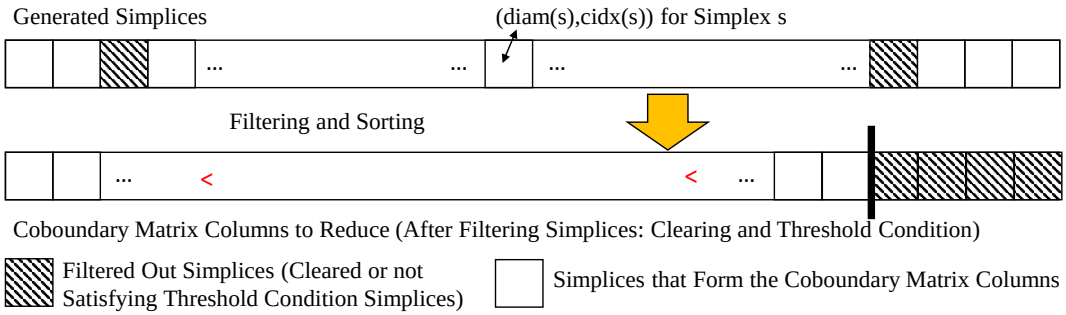


Figure 8 The Filtration Construction with Clearing Algorithm for Full Rips Filtrations.

Before entering the matrix reduction phase, the input simplex-wise filtration must be constructed and simplified to form coboundary matrix columns. We call this Filtration Construction with Clearing. This requires two steps: filtering and sorting. Both of which can be done in parallel. Filtering removes simplices that we don't need to reduce as they are equivalent to zeroed columns. As presented in Algorithm 4, the simplices having higher diameters than the *threshold* and paired simplices (the clearing lemma [13]) are filtered out.

Algorithm 4 Filtering the Columns on GPU.

Require: $\boldsymbol{P}$: the persistence pairs in the form (cofacet,simplex) discovered in the previous dimension; *threshold*: the max diameter allowed for a simplex; $diam(\cdot)$: the diameter of a simplex; $cidx(\cdot)$: the combinatorial index of a simplex. ▷ global to all threads
tid: the thread id. ▷ local to each thread

Ensure: $\boldsymbol{C}$: an array of simplices, in which an element is represented as a diameter paired with a combinatorial index; $flagarray$: an array of flags marking which columns are kept (filtered in).

1: **procedure** FILTER-COLUMNS-KERNEL($\boldsymbol{C}$, $\boldsymbol{P}$, *threshold*, $flagarray$)
2: $cidx(s) \leftarrow tid$
3: **if** $\nexists t$ s.t. $(t, s) \in \boldsymbol{P}$ AND $diam(s) \leq threshold$ **then**
4: $diam(\boldsymbol{C}[tid]) \leftarrow diam(s)$; $cidx(\boldsymbol{C}[tid]) \leftarrow cidx(s)$; $flagarray[tid] \leftarrow 1$;
5: **else**
6: $diam(\boldsymbol{C}[tid]) \leftarrow -\infty$; $cidx(\boldsymbol{C}[tid]) \leftarrow +\infty$; $flagarray[tid] \leftarrow 0$;

Sorting in the reverse of the order given in Section 2.2.1 is then conducted over the remaining simplices. This is the order for the columns of a coboundary matrix. The resulting sequence of simplices is then the columns to reduce for the following matrix reduction phase. Algorithm 5 presents how we construct the full Rips filtration with clearing. Our GPU-based algorithms leverage the massive parallelism of GPU threads and high bandwidth data processing in GPU device memory.

Algorithm 5 Use GPU for Full Rips Filtration Construction with Clearing.

Require: $\boldsymbol{P}, threshold, flagarray$: same as in Algorithm 4; n: the number of points; d: the current dimension for simplices to construct. len: the number of simplices selected.
Ensure: $\boldsymbol{C}$ same as in Algorithm 4.
1: $\boldsymbol{C} \leftarrow \emptyset$
2: $flagarray \leftarrow \{0, ..., 0\}$
3: $filter\text{-}columns\text{-}kernel(\boldsymbol{C}, \boldsymbol{P}, threshold, flagarray)$ ▷ $\binom{n}{d+1}$ threads launched
4: $len \leftarrow GPU\text{-}reduction(flagarray)$
5: $GPU\text{-}sort(\boldsymbol{C})$ ▷ sort entries of $\boldsymbol{C}$ in coboundary filtration order: decreasing diameters, increasing combinatorial indices; restrict $\boldsymbol{C}$ to indices 0 to $len - 1$ afterwards.

5 Experiments

All experiments are performed on a powerful computing server. It consists of an NVIDIA Tesla V100 GPU that has 5120 FP32 cores and 2560 FP64 cores for single- and double-precision floating-point computation. The GPU device memory is 32 GB High Bandwidth Memory 2 (HBM2) that can provide up to 900 GB/s memory access bandwidth. The node also has two 14 core Intel XEON E5-2680 v4 CPUs (28 cores in total) running at 2.4 GHz with a total of 100 GB of DRAM. The datasets are taken from the original Ripser repository on Github [5] and the repository of benchmark datasets from [37].

5.1 The Empirical Relationship amongst Apparent Pairs, Emergent Pairs, and Shortcut Pairs

There exists three kinds of persistence pairs of the Vietoris-Rips filtration, in fact for any filtration with a simplex-wise refinement. Using the terminology of [6], these are apparent (Definition 1) [18, 27, 6, 34], shortcut [6], and emergent pairs [6, 47]. By definition, they are known to form a tower of sets ordered by inclusion (expressed by Equation (3)). We will show a further empirical relationship amongst these pairs involving their cardinalities.

$$\overbrace{\underbrace{\text{apparent pairs}}_{\text{large cardinality}} \subset \text{shortcut pairs} \subset \text{emergent pairs} \subset \text{persistence pairs}}^{\text{the difference in cardinalities is ``small''}} \tag{3}$$

The cardinality difference amongst all of the sets of pairs is very small compared to the number of pairs, assuming Ripser's framework of computing cohomology and using the simplex-wise filtration ordering in Section 2.2.1. Thus there are a very large number of apparent pairs to be found.

Table 1 shows the percentage of apparent pairs up to dimension d is extremely high, around 99%. Since the number of columns of a cleared coboundary matrix equals to the number of persistence pairs, the number of nonapparent columns for submatrix reduction is a tiny fraction of the original number of columns in Ripser's matrix reduction phase.

5.2 Execution Time and Memory Usage

We perform extensive experiments that demonstrate the execution time and memory usage of Ripser++. We further look into the performance of both the apparent pairs search algorithm and the management of persistence pairs in the two layer data structure after finding apparent pairs. Variables n and d for each dataset are the same for all experiments.

Table 1 Empirical Results on Apparent, Shortcut, Emergent Pairs.

Datasets	n	d	apparent pairs	shortcut pairs	emergent pairs	all pairs	percentage of apparent pairs
celegans	297	3	317,664,839	317,723,916	317,723,974	317,735,650	99.9777139%
dragon1000	1000	2	166,132,946	166,160,587	166,160,665	166,167,000	99.9795062%
HIV	1088	2	214,000,996	214,030,431	214,040,521	214,060,736	99.9720920%
o3 (sparse: $t = 1.4$)	4096	3	43,480,968	43,940,030	43,940,686	44,081,360	98.6379912%
sphere_3_192	192	3	54,779,316	54,871,199	54,871,214	54,888,625	99.8008531%
Vicsek300_of_300	300	3	330,724,672	330,818,491	330,818,507	330,835,726	99.9664323%

Table 2 Total Execution Time and CPU/GPU Memory Usage.

Datasets	n	d	R.++ time	R. time	R.++ GPU mem.	R.++ CPU mem.	R. CPU mem.	Speedup
celegans	297	3	7.30 s	228.56 s	16.84 GB	10.53 GB	23.84 GB	31.33x
dragon1000	1000	2	5.79 s	48.98 s	8.81 GB	3.75 GB	5.79 GB	8.46x
HIV	1088	2	7.11 s	147.18 s	11.36 GB	6.68 GB	14.59 GB	20.69x
o3 (sparse: $t = 1.4$)	4096	3	11.62 s	64.18 s	18.76 GB	2.77 GB	3.86 GB	5.52x
sphere_3_192	192	3	2.43 s	36.96 s	2.92 GB	2.03 GB	4.32 GB	15.21x
Vicsek300_of_300	300	3	9.98 s	248.72 s	17.53 GB	11.46 GB	27.78 GB	24.92x

Table 2 shows the comparisons of execution time and memory usage for computation up to dimension d between Ripser++ and Ripser with six datasets, where R. stands for Ripser and R.++ stands for Ripser++. Memory usage on CPU and total execution time were measured with the `/usr/time -v` command on Linux. GPU memory usage was counted by the total displacement of free memory over program execution.

Table 2 shows Ripser++ can achieve 5.52x - 31.33x speedups of total execution time over Ripser in the evaluated datasets. The performance improvement mainly comes from massive parallel operations of finding apparent pairs on GPU, and from the fast filtration construction with clearing by GPU using filtering and sorting. We also notice that the speedups of execution time varies in different datasets. That is because the percentages of execution time in the submatrix reduction are different among datasets.

It is well known that the memory usage of full Vietoris-Rips filtration grows exponentially in the number of simplices with respect to the dimension of persistence computation. For example, 2000 points at dimension 4 computation may require $\binom{2000}{4+1} \times 8$ bytes = 2 million GB memory. Algorithmically, we avoid allocating memory in the cofacet dimension and keep the memory requirement of Ripser++ asymptotically same as Ripser. Table 2 also shows the memory usage of Ripser++ on CPU and GPU. Ripser++ can actually lower the memory usage on CPU. This is mostly because Ripser++ offloads the process of finding apparent pairs to GPU and the following matrix reduction only works on much fewer columns than that of Ripser (as the submatrix reduction). Table 2 also shows that the GPU device memory usage is usually lower than the total memory usage of Ripser. However, in the sparse computation case (dataset *o3*) the algorithm must change; Ripser++ thus allocates memory depending on the hardware instead of the input sizes.

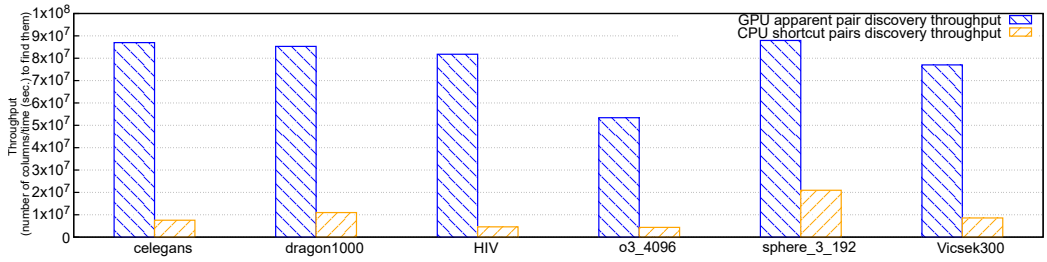


Figure 9 A comparison of column discovery throughput of apparent pair discovery with Ripser++ vs. Ripser's shortcut pair discovery. The corresponding time is greatly reduced due to Algorithm 2.

Table 3 Hashmap Access Throughput, Counts, and Times Comparisons.

Datasets	R.++ write throuput (pairs/s)	R. write throughput (pairs/s)	Num. of R.++ reads to data struct.	Num. of R. reads to hashmap	R.++ read time (s)	R. read time (s)
celegans	7.21×10^8	6.98×10^7	3.22×10^4	5.81×10^8	0.00100	11.43
dragon1000	7.62×10^8	6.29×10^7	1.19×10^5	1.12×10^8	0.00460	1.28
HIV	7.06×10^8	8.85×10^7	1.57×10^5	3.10×10^8	0.00130	5.52
o3 (sparse: $t = 1.4$)	4.78×10^8	6.88×10^7	1.65×10^6	8.85×10^7	0.01500	0.56
sphere_3_192	7.32×10^8	9.41×10^7	2.71×10^5	9.37×10^7	0.00068	0.30
Vicsek300_of_300	6.80×10^8	8.82×10^7	2.12×10^5	5.67×10^8	0.00053	10.81

5.3 Throughput of Apparent Pairs Discovery with Ripser++ vs. Throughput of Shortcut Pairs Discovery in Ripser

Discovering shortcut pairs in Ripser and discovering apparent pairs in Ripser++ account for a significant part of the computation. Let the throughput be calculated as the number of a specific type of pair divided by the time to find and store them. We can find in Figure 9 that for all datasets, our GPU-based solution outperforms the CPU-based algorithm used in Ripser by 4.2x-12.3x. Since the two types of pairs' counts are almost the same (see Table 1), such throughput improvement can lead to a significant saving in computation time.

5.4 Two-layer Data Structure for Memory Access Optimizations

Table 3 first presents the write throughput of persistence pairs in pairs/s. In Ripser, we use the measured time of writing pairs to the hashmap to divide the total persistence pair number; while in Ripser++, the time includes writing to the two-layer data structure and sorting the array on GPU. The results show that Ripser++ consistently has one order of magnitude higher write throughput than that of Ripser.

Table 3 also gives the number of reads as well as the time consumed in the read operations (in seconds). The number of reads in Ripser means the number of reads to its hashmap, while Ripser++ counts the number of reads to the data structure. The reported results confirm that Ripser++ can reduce at least two orders of magnitude memory reads over Ripser. A similar performance improvement can also be observed in the measured read time.

5.5 The Apparent Fraction Depending on the Number of Points

Figure 10 shows 3 curves for the apparent fraction depending on the number of points.

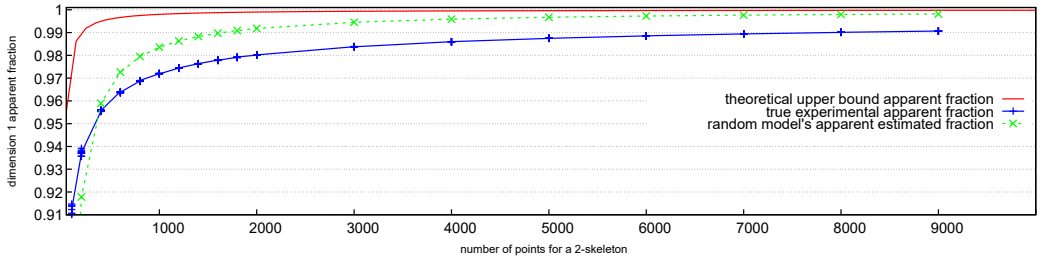


Figure 10 Three different curves of the apparent fraction: $(\frac{num.\ apparent\ pairs}{num.\ d\text{-}simplices})$ for $d = 1$ as a function of the number of points. The theoretical upper bounding curve for the case of all equivalent edge diameters is shown as well as the true experimental curve. The dotted curve is the piecewise linear interpolated curve of a uniform random mathematical model that matches the shape of the empirical and theoretical curve. (See the paper's full version for more explanations.)

6 Conclusion

Ripser++ can achieve significant speedup (up to 20x-30x) on representative datasets in our work and thus opens up unprecedented opportunities in many application areas. For example, fast streaming applications [43] or point clouds from neuroscience [10] that spent minutes can now be computed in seconds, significatly advancing the domain fields.

We identify specific properties of Vietoris-Rips filtrations such as the simplicity of diameter computations by individual threads on GPU for Ripser++. Related discussions, both theoretical and empirical, suggest that our approach be applicable to other filtration types such as cubical [8], flag [32], and alpha shapes [44]. We strongly believe that our acceleration methods are widely applicable beyond computing Rips persistence barcodes.

We have described the mathematical, algorithmic, and experimental-based foundations of Ripser++. We hope our efforts open a new chapter for the advancement of TDA.

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. *The Journal of Machine Learning Research*, 18(1):218–252, 2017.

2 Henry Adams and Andrew Tausz. Javaplex tutorial. *Google Scholar*, 2011.

3 Mehmet E Aktas, Esra Akbas, and Ahmed El Fatmaoui. Persistence homology of networks: methods and applications. *Applied Network Science*, 4(1):61, 2019.

4 Sergey Barannikov. The framed morse complex and its invariants, 1994.

5 Ulrich Bauer. Ripser: efficient computation of vietoris–rips persistence barcodes, 2018. URL: `https://github.com/Ripser/ripser`.

6 Ulrich Bauer. Ripser: efficient computation of vietoris-rips persistence barcodes. *arXiv preprint*, 2019. `arXiv:1908.02518`.

7 Ulrich Bauer, Michael Kerber, and Jan Reininghaus. Clear and compress: Computing persistent homology in chunks. In *Topological methods in data analysis and visualization III*, pages 103–117. Springer, 2014.

8 Ulrich Bauer, Michael Kerber, and Jan Reininghaus. Distributed computation of persistent homology. In *2014 proceedings of the sixteenth workshop on algorithm engineering and experiments (ALENEX)*, pages 31–38. SIAM, 2014.

9 Ulrich Bauer, Michael Kerber, Jan Reininghaus, and Hubert Wagner. Phat–persistent homology algorithms toolbox. *Journal of symbolic computation*, 78:76–90, 2017.

10 Paul Bendich, James S Marron, Ezra Miller, Alex Pieloch, and Sean Skwerer. Persistent homology analysis of brain artery trees. *The annals of applied statistics*, 10(1):198, 2016.

11 Peter Bubenik. Statistical topological data analysis using persistence landscapes. *The Journal of Machine Learning Research*, 16(1):77–102, 2015.

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

13 Chao Chen and Michael Kerber. Persistent homology computation with a twist. In *Proceedings 27th European Workshop on Computational Geometry*, volume 11, 2011.

14 Yuri Dabaghian, Facundo Mémoli, Loren Frank, and Gunnar Carlsson. A topological paradigm for hippocampal spatial map formation using persistent homology. *PLoS computational biology*, 8(8):e1002581, 2012.

15 Vin De Silva and Robert Ghrist. Coverage in sensor networks via persistent homology. *Algebraic & Geometric Topology*, 7(1):339–358, 2007.

16 Vin De Silva, Dmitriy Morozov, and Mikael Vejdemo-Johansson. Dualities in persistent (co) homology. *Inverse Problems*, 27(12):124003, 2011.

17 Vin De Silva, Dmitriy Morozov, and Mikael Vejdemo-Johansson. Persistent cohomology and circular coordinates. *Discrete & Computational Geometry*, 45(4):737–759, 2011.

18 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, 2014.

19 Robert H Dennard, Fritz H Gaensslen, V Leo Rideout, Ernest Bassous, and Andre R LeBlanc. Design of ion-implanted mosfet's with very small physical dimensions. *IEEE Journal of Solid-State Circuits*, 9(5):256–268, 1974.

20 Tamal K Dey, Fengtao Fan, and Yusu Wang. Computing topological persistence for simplicial maps. In *Proceedings of the thirtieth annual symposium on Computational geometry*, page 345. ACM, 2014.

21 Herbert Edelsbrunner and John Harer. *Computational topology: an introduction*. American Mathematical Soc., 2010.

22 Herbert Edelsbrunner, David Letscher, and Afra Zomorodian. Topological persistence and simplification. In *Proceedings 41st annual symposium on foundations of computer science*, pages 454–463. IEEE, 2000.

23 Hadi Esmaeilzadeh, Emily Blem, Renee St Amant, Karthikeyan Sankaralingam, and Doug Burger. Dark silicon and the end of multicore scaling. *IEEE Micro*, 32(3):122–134, 2012.

24 Brittany Terese Fasy, Fabrizio Lecci, Alessandro Rinaldo, Larry Wasserman, Sivaraman Balakrishnan, Aarti Singh, et al. Confidence sets for persistence diagrams. *The Annals of Statistics*, 42(6):2301–2339, 2014.

25 William H Guss and Ruslan Salakhutdinov. On characterizing the capacity of neural networks using algebraic topology. *arXiv preprint*, 2018. `arXiv:1802.04443`.

26 G Henselman. Eirene: a platform for computational homological algebra, 2016.

27 Gregory Henselman and Robert Ghrist. Matroid filtrations and computational persistent homology. *arXiv preprint*, 2016. `arXiv:1606.00199`.

28 Christoph Hofer, Roland Kwitt, Marc Niethammer, and Andreas Uhl. Deep learning with topological signatures. In *Advances in Neural Information Processing Systems*, pages 1634–1644, 2017.

29 Alan Hylton, Janche Sang, Greg Henselman-Petrusek, and Robert Short. Performance enhancement of a computational persistent homology package. In *2017 IEEE 36th International Performance Computing and Communications Conference (IPCCC)*, pages 1–8. IEEE, 2017.

30 Changkyu Kim, Tim Kaldewey, Victor W. Lee, Eric Sedlar, Anthony D. Nguyen, Nadathur Satish, Jatin Chhugani, Andrea Di Blas, and Pradeep Dubey. Sort vs. hash revisited: Fast join implementation on modern multi-core cpus. *Proc. VLDB Endow.*, 2(2):1378–1389, August 2009. doi:10.14778/1687553.1687564.

31 Donald Ervin Knuth. *The art of computer programming*, volume 3. Pearson Education, 1997.

32 Daniel Luetgehetmann, Dejan Govc, Jason Smith, and Ran Levi. Computing persistent homology of directed flag complexes. *arXiv preprint*, 2019. arXiv:1906.10458.

33 Clément Maria, Jean-Daniel Boissonnat, Marc Glisse, and Mariette Yvinec. The gudhi library: Simplicial complexes and persistent homology. In *International Congress on Mathematical Software*, pages 167–174. Springer, 2014.

34 Rodrigo Mendoza-Smith and Jared Tanner. Parallel multi-scale reduction of persistent homology filtrations. *arXiv preprint*, 2017. arXiv:1708.04710.

35 Dmitriy Morozov. Dionysus software, 2017. URL: http://www.mrzv.org/software/dionysus/.

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

37 Nina Otter, Mason A Porter, Ulrike Tillmann, Peter Grindrod, and Heather A Harrington. A roadmap for the computation of persistent homology. *EPJ Data Science*, 6(1):17, 2017.

38 Ernesto Pascal. Sopra una formula numerica, 1887.

39 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.

40 Nadathur Satish, Mark Harris, and Michael Garland. Designing efficient sorting algorithms for manycore gpus. In *Proceedings of the 2009 IEEE International Symposium on Parallel&Distributed Processing*, IPDPS '09, pages 1–10, Washington, DC, USA, 2009. IEEE Computer Society. doi:10.1109/IPDPS.2009.5161005.

41 Abu Bakar Siddique, Saadia Farid, and Muhammad Tahir. Proof of bijection for combinatorial number system. *arXiv preprint*, 2016. arXiv:1601.05794.

42 Erik Sintorn and Ulf Assarsson. Fast parallel gpu-sorting using a hybrid algorithm. *J. Parallel Distrib. Comput.*, 68(10):1381–1388, October 2008. doi:10.1016/j.jpdc.2008.05.012.

43 Meirman Syzdykbayev and Hassan A Karimi. Persistent homology for detection of objects from mobile lidar point cloud data in autonomous vehicles. In *Science and Information Conference*, pages 458–472. Springer, 2019.

44 The GUDHI Project. *GUDHI User and Reference Manual*. GUDHI Editorial Board, 2015. URL: http://gudhi.gforge.inria.fr/doc/latest/.

45 Thomas N Theis and H-S Philip Wong. The end of moore's law: A new beginning for information technology. *Computing in Science & Engineering*, 19(2):41, 2017.

46 Christopher Tralie, Nathaniel Saul, and Rann Bar-On. Ripser. py: A lean persistent homology library for python. *J. Open Source Software*, 3(29):925, 2018.

47 Simon Zhang, Mengbai Xiao, Chengxin Guo, Liang Geng, Hao Wang, and Xiaodong Zhang. Hypha: a framework based on separation of parallelisms to accelerate persistent homology matrix reduction. In *Proceedings of the ACM International Conference on Supercomputing*, pages 69–81. ACM, 2019.

48 Afra Zomorodian. Fast construction of the vietoris-rips complex. *Computers & Graphics*, 34(3):263–271, 2010.

The Spiroplot App

Casper van Dommelen
Utrecht University, The Netherlands

Marc van Kreveld
Utrecht University, The Netherlands
m.j.vankreveld@uu.nl

Jérôme Urhausen
Utrecht University, The Netherlands
j.e.urhausen@uu.nl

Abstract

We introduce an app for generating spiroplots, based on a new discrete-time, linear, dynamic system that repeatedly rotates a pair of points, and plots points where they land. The app supports easy definition of the initial situation and has various visualization settings. It can be accessed at `https://spiroplot.sites.uu.nl`.

2012 ACM Subject Classification Applied computing → Media arts; Computing methodologies → Animation

Keywords and phrases generative art, dynamic system, pattern generation tool

Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.71

Category Media Exposition

Supplementary Material `https://spiroplot.sites.uu.nl`

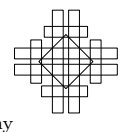
Funding Supported by the Netherlands Organisation for Scientific Research (NWO) under project no. 612.001.651.

1 Spiroplots

Spiroplots are a new type of discrete-time, linear, dynamic system that produces various nice geometric patterns. They are introduced in [2], where a comparison is made with other systems that generate patterns like L-systems, chaotic systems, and fractals [1]. A spiroplot is defined by a finite set V of points in the plane with their initial positions, a finite sequence R of triplets consisting of two points from V and a rotation angle, and a rotation count k. Each triplet in R – called *r-triplet* – defines a repositioning of the two specified points in V by rotating them around their middle over the specified rotation angle (in degrees in this paper). Whenever points of V land somewhere, they plot a small dot. Each point plots in a different color. We refer to both the system and to the resulting pattern as a spiroplot.

The sequence of k rotations is done by following the sequence in R and repeating it from the start, until k rotations in total have been done, and $2k$ dots are plotted. Even simple spiroplots give nice patterns. For example, a spiroplot with just three points $V = \{v_1, v_2, v_3\}$, and just two r-triplets $R = \langle (v_1, v_2, 90), (v_2, v_3, 90) \rangle$ will show a pattern with eight ellipses after a few thousand rotations (Figure 1, left), regardless of the initial coordinates.

The name spiroplot is derived from the spirograph, a drawing tool for kids from the 1960s. A more complete description and various properties of spiroplots are given in [2], including preservation of the center of mass and the existence of cyclic and non-cyclic spiroplots. The latter ones plot infinitely many points when $k \to \infty$.



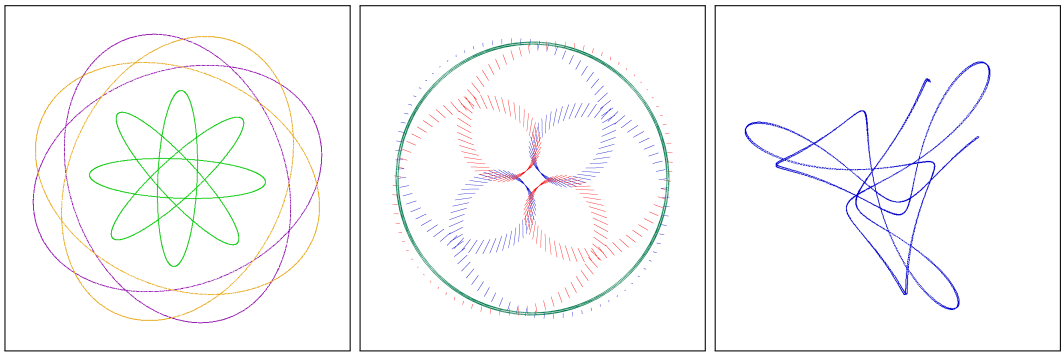


Figure 1 Three spiroplots. Left, with $R = \langle (v_1, v_2, 90), (v_2, v_3, 90) \rangle$, after 5000 rotations. Middle, with $R = \langle (v_1, v_2, 4), (v_2, v_3, 4) \rangle$, after 50,000 rotations. Right, with $R = \langle (v_1, v_2, 1), (v_2, v_3, 1), (v_3, v_4, -1), (v_4, v_5, 1) \rangle$, after 8000 rotations; only v_3 is shown.

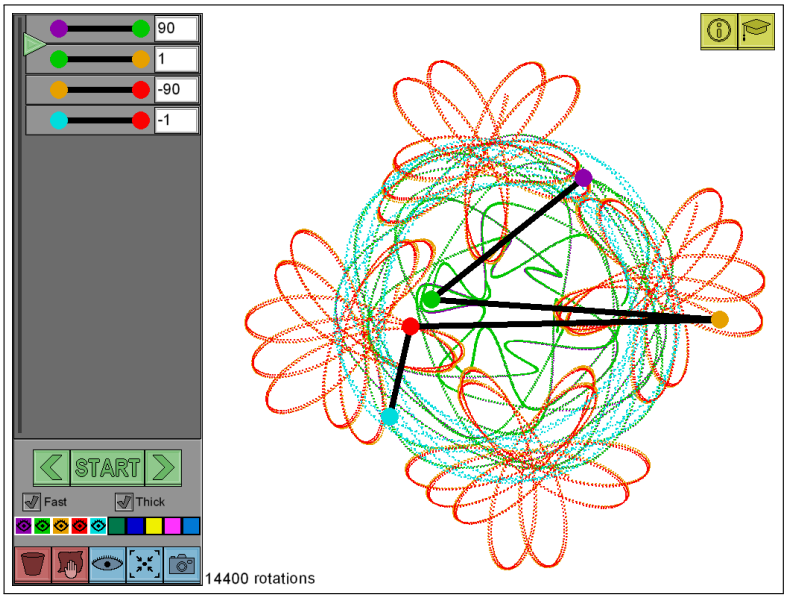


Figure 2 User interface. On the canvas, pairs of rotating points are shown by an edge.

2 A web app for spiroplots

This abstract presents a web app that generates spiroplots with a simple user interface.

`https://spiroplot.sites.uu.nl`

We believe it can serve to make young people enthusiastic about mathematical patterns, dynamic systems, and procedural generation of structures.

The app has a canvas and a control panel, see Figure 2. It allows the user to place points using a mouse left-click, and generate pairs by dragging (mouse-down) from one point to the other point. The default rotation angle is 90 degrees, but other angles can be specified in a text box for that pair. We may repeat a pair of points in the r-triplet sequence. With a right-click on a point, it can be moved, removed, or assigned a different color.

The left panel shows the sequence of r-triplets, each by the colors of the points and its own rotation angle. A green indicator shows what the next rotation will be.

To the bottom in the left panel, options to perform one or many rotations are given, along with visualization options, wiping, resetting, and saving. The center-and-scale option places and scales the point set to ensure that the spiroplot stays inside the canvas and uses most of it. The Fast option performs 100 rotations before the next canvas redraw, doing close to 7,000 rotations per second (web app, standard laptop, wall clock time). On the desktop app version, it is a few times faster. Simple patterns often start to emerge within seconds; complex patterns may show a pattern only after minutes. Sometimes we do not discern any pattern. The Thick option draws 2×2 pixels when plotting a point. Every color can be toggled on or off in the row of colored eyes, and so can the graph of the spiroplot (bottom eye). Colors are stacked, so there is always a top color. The stacking order can be changed by clicking the colored eyes, which brings the last toggled color to the top. The app has a tutorial built in.

3 Various settings

When trying out different settings in the app, we can make various observations.

Spiroplots with $V = \{v_1, v_2, v_3\}$, $R = \langle (v_1, v_2, 90), (v_2, v_3, 90) \rangle$ always look like ellipses after sufficiently many rotations, regardless of the initial coordinates of v_1, v_2 and v_3. The points plotted for v_2 lie on four ellipses and the points plotted for v_1 or v_3 lie on two ellipses each. If one of the angles is -90 degrees, there are only four ellipses in total, two for v_2 and one for each of v_1 and v_3. When the two angles are 30 or 45 degrees, we see more ellipses.

Spiroplots like $V = \{v_1, v_2, v_3\}$, $R = \langle (v_1, v_2, 90), (v_2, v_3, 90), (v_1, v_3, 90) \rangle$ also show ellipses only, and all points plot on the same eight ellipses.

When spiroplots use small angles only, like 1 degree (or even smaller), the plotted points trace a curve with small steps, essentially drawing it (see Figure 1, right). The types of curves can appear rather complex when there are five or more points in the system.

Certain spiroplots repeat their current points after a few rotations. For example, every spiroplot with $V = \{v_1, v_2, v_3, v_4\}$, $R = \langle (v_1, v_2, 90), (v_2, v_3, 90), (v_3, v_4, 90) \rangle$ will have the same positions for $v_1, \ldots, v_4$ after 36 rotations, regardless of the initial coordinates. Hence, no interesting pattern appears. The three rotations can be represented together by an 8×8 matrix operating on the eight coordinates of the four points. Raising this matrix to the power 12 yields the identity matrix.

The order of r-triplets is very important. A spiroplot with $R = \langle (v_1, v_2, 90), (v_2, v_3, 90), (v_3, v_4, 90), (v_1, v_4, 90) \rangle$ shows complex patterns while $R = \langle (v_1, v_2, 90), (v_3, v_4, 90), (v_2, v_3, 90), (v_1, v_4, 90) \rangle$ yields a cyclic spiroplot.

Many spiroplots with more complex specifications will not show a pattern in the app. This may be due to the limited resolution that is available. Sometimes one point (color) shows a pattern but another point does not.

For simpler situations, patterns show up after a few thousand rotations. For complex spiroplots, it may take hundreds of thousands rotations. Figure 3 shows how more iterations yields a fuller pattern. Figure 4 shows two more examples of spiroplots.

4 Discussion

There are many interesting open problems related to spiroplots.

1. By observation we noticed that spiroplots with three points only show ellipses, and each ellipse has the same center. Can we prove this, or find a counterexample?
2. Do all points of a spiroplot lie on a finite set of algebraic curves? Or are there positive-area regions of the plane that are covered arbitrarily densely by plotted points, if $k \to \infty$?

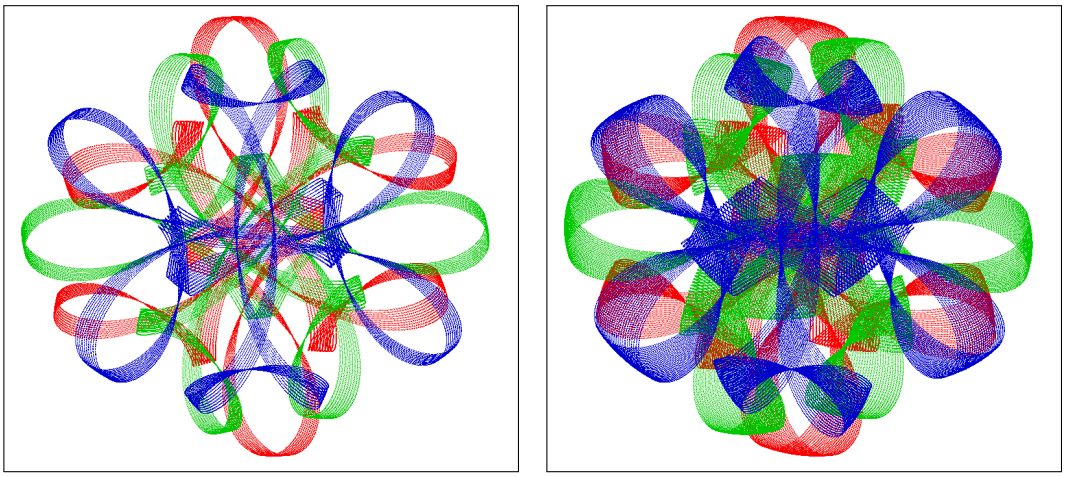


Figure 3 Spiroplots that are the same except for the number of rotations. Left, after 201,000 rotations. Right, after 500,800 rotations. The rotation sequence is $\langle(v_1, v_2, 120), (v_3, v_4, -120), (v_5, v_6, 120), (v_2, v_3, 0.4), (v_4, v_5, 0.4), (v_6, v_1, -0.4)\rangle$. Only three of the six colors are shown: v_1 is red, v_2 is green, and v_5 is blue.

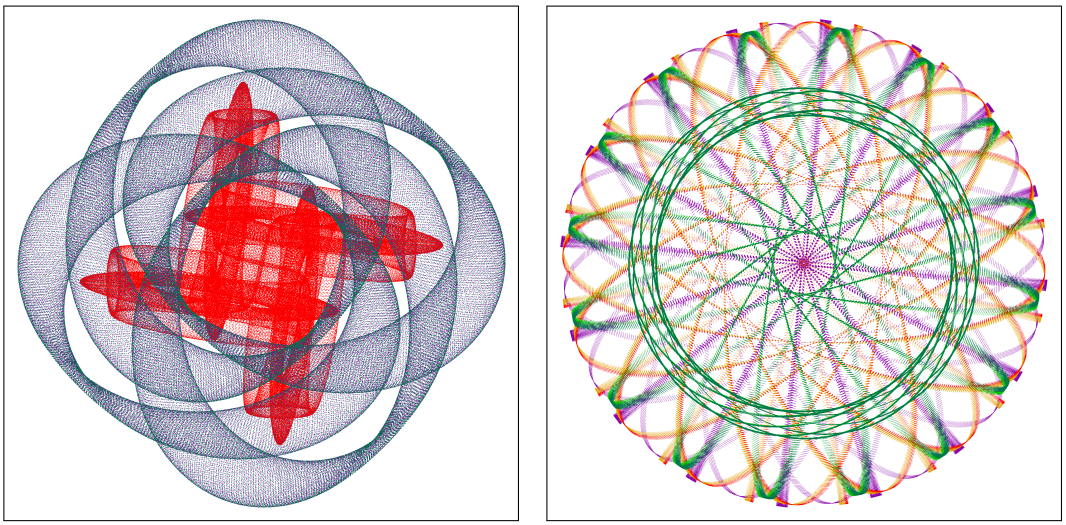


Figure 4 Two spiroplots created with the program. Left, $R = \langle(v_1, v_2, 90), (v_3, v_4, 90), (v_5, v_6, 90), (v_2, v_3, 90), (v_4, v_5, 90)\rangle$. Right, $R = \langle(v_1, v_2, 0.2), (v_1, v_2, 0.2), (v_1, v_2, -0.6), (v_1, v_2, -0.2), (v_1, v_2, 90.4), (v_2, v_3, 0.2), (v_2, v_3, 0.2), (v_2, v_3, -0.6), (v_2, v_3, -0.2), (v_2, v_3, 90.4), (v_1, v_3, 45)\rangle$ (some colors were changed during the run).

3. Can spiroplots be interpreted as projections of higher-dimensional curves?
4. Can we characterize all cyclic spiroplots?
5. How should we define 3D spiroplots? Rotations of two points about their center are no longer unique; we need a rotation axis.

A more practical question is the one of control: How can we provide intuitive control on the spiroplot to be produced? If we generated a spiroplot and want it slightly different, it is unclear how to change the parameters to realize this.

References

1 Heinz-Otto Peitgen, Hartmut Jürgens, and Dietmar Saupe. *Chaos and Fractals: new frontiers of science*. Springer Science & Business Media, 2006.

2 Casper van Dommelen, Marc van Kreveld, and Jérôme Urhausen. Spiroplots: a new discrete-time dynamical system to generate curve patterns, 2020. Submitted.

Coordinated Particle Relocation with Global Signals and Local Friction

Victor M. Baez
Department of Electrical and Computer Engineering, University of Houston, TX, USA
vjmontan@uh.edu

Aaron T. Becker
Department of Electrical and Computer Engineering, University of Houston, TX, USA
atbecker@uh.edu

Sándor P. Fekete
Department of Computer Science, TU Braunschweig, Germany
s.fekete@tu-bs.de

Arne Schmidt
Department of Computer Science, TU Braunschweig, Germany
arne.schmidt@tu-bs.de

Abstract

In this video, we present theoretical and practical methods for achieving arbitrary reconfiguration of a set of objects, based on the use of external forces, such as a magnetic field or gravity: Upon actuation, each object is pushed in the same direction. This concept can be used for a wide range of applications in which particles do not have their own energy supply or in which they are subject to the same global control commands.

A crucial challenge for achieving any desired target configuration is breaking global symmetry in a controlled fashion. Previous work (some of which was presented during SoCG 2015) made use of specifically placed barriers; however, introducing precisely located obstacles into the workspace is impractical for many scenarios. In this paper, we present a different, less intrusive method: making use of the interplay between static friction with a boundary and the external force to achieve arbitrary reconfiguration. Our key contributions are *theoretical* characterizations of the critical coefficient of friction that is sufficient for rearranging two particles in triangles, convex polygons, and regular polygons; a method for reconfiguring multiple particles in rectangular workspaces, and deriving *practical* algorithms for these rearrangements. Hardware experiments show the efficacy of these procedures, demonstrating the usefulness of this novel approach.

2012 ACM Subject Classification Theory of computation → Computational geometry; Computer systems organization → Embedded and cyber-physical systems

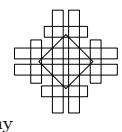
Keywords and phrases Global control, reconfiguration, geometric algorithms, friction

Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.72

Category Media Exposition

1 Introduction

Reconfiguring a large set of objects in a prespecified manner is a fundamental task for a large spectrum of applications, including swarm robotics, smart materials and advanced manufacturing. In many of these scenarios, the involved items are not equipped with individual motors or energy supplies, so actuation must be performed from the outside. Moreover, reaching into the workspace to manipulate individual particles of an arrangement is often impractical or even impossible; instead, global external forces (such as gravity or a magnetic force) may have to be employed, targeting each object in the same, uniform manner. These limitations of individual navigation apply even in scenarios of swarm robotics, e.g.



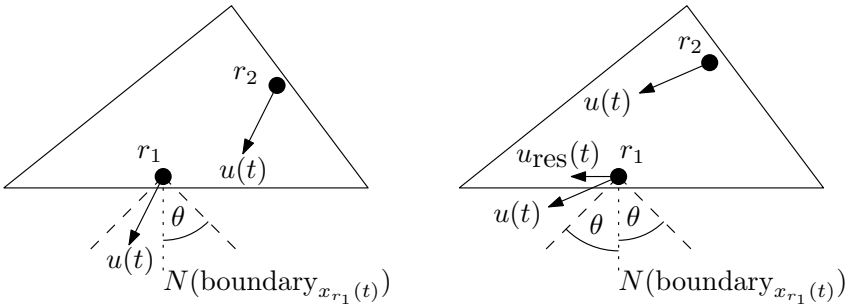



Figure 1 Left: An input force command $u(t)$ within the cone $\pm\theta$ about the normal to the boundary results in no motion of r_1. Right: An input force command $u(t)$ outside the cone results in a motion of both particles. Observe that r_1 slides along the boundary with a resulting force $u_{\text{res}}(t)$.

for the well-known kilobots [12] that can be directed by switching on a light beacon, which works just like activating an external force. This concept of global control has also been studied for using biological cells as reactive robots controlled by magnetic fields, see Arbuckle and Requicha [3] and Kim et al. [9]. Global control also has applications in assembling nano- and micro-structures. Related work shows how to assemble shapes by adding one particle at a time [8, 4], or combining multiple pairs of subassemblies in parallel in one time step [14].

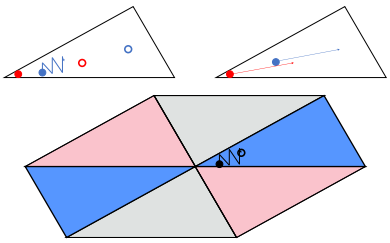
Considering this approach of navigation by a global external force gives rise to a number of problems, including navigation of one particle from a start to a goal position [10], particle computation [6, 7], or emptying a polygon [2]. Zhang et al. [15, 16] show how to rearrange a rectangle of agents in a workspace that is only constant times larger than the number of agents. Akella et al. [1] consider the problem of reconfiguring an object on a conveyor belt with a simple robot, and Lynch et al. [11] use a mobile robot with a flat pusher plate as the gripper to manipulate objects. A crucial issue for all these tasks is how to combine the use of a uniform force (which is the same for all involved items) with the individual requirements of object relocation (which may be distinct for different particles): How can we achieve an *arbitrary* arrangement of particles if all of them are subjected to the same external force? Previous work (such as [7]) has shown how arbitrary reconfiguration of an ensemble is possible with the help of specifically placed barriers; this has also been the subject of a previous multimedia contribution to SoCG [5]. However, introducing precisely located obstacles into the workspace is impractical for many scenarios.

In this contribution, we present a different, less intrusive method: making use of the interplay between static friction with a boundary of the workspace and the external force to achieve any desired configuration. For more details, see our journal paper [13].

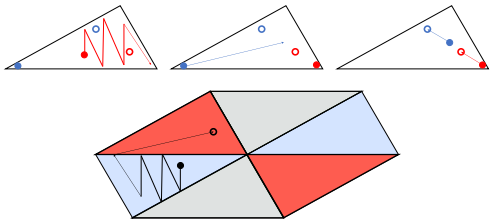
2 Using friction for reconfiguration

The *coefficient of friction* μ is the ratio between the strength of an orthogonal force against a surface and the resistance parallel to the surface. Geometrically, this corresponds to a (static) *angle of friction* θ: which is the critical angle when sliding commences, satisfying $\mu := \tan\theta$. See Fig. 1 for an illustration.

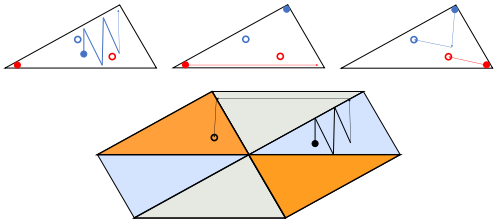
Just like in the context of sorting algorithms in computer science or discrete mathematics, a critical component for achieving arbitrary reconfiguration of larger ensembles is the ability to rearrange two specific particles. The idea is to completely cover the Δ configuration, which is the set of all differences between all pairs of possible particle locations. To this end, we employ a number of different strategies (shown in Fig. 2 and visualized in the video). As shown in Fig. 3, these can be combined to yield an overall lower bound for θ, as follows.



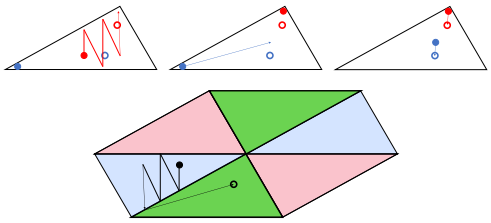
(a) Blue strategy.



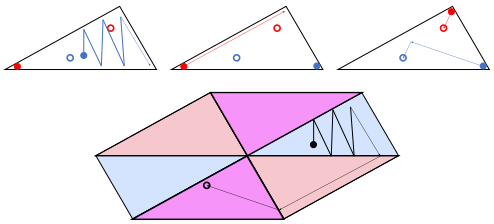
(b) Red strategy.



(c) Orange strategy.



(d) Green strategy.



(e) Violet strategy.

Figure 2 Illustration of the five strategies for dealing with different portions of Δ space. In each case, shown are two particles in actual space (top), and in Δ space (bottom).

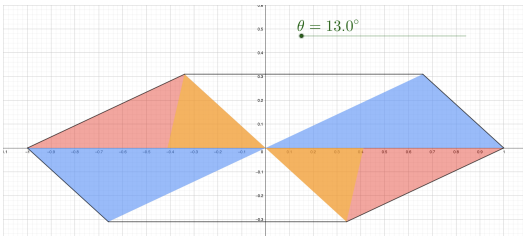
▶ **Theorem 1.** *Let T be a triangle with angles $\alpha \leq \beta \leq \gamma$. If $\theta > \frac{\pi}{2} - \beta$, then we can guarantee any reconfiguration of two particles, i.e., Δ_T is completely covered by our strategies.*

This can be generalized to other environments, as follows.

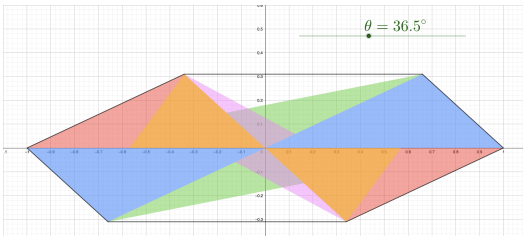
▶ **Theorem 2.** *Let P be a convex polygon with vertices $C_0, \ldots, C_{n-1}$ and angles $\gamma_0, \ldots, \gamma_{n-1}$. If $\theta > \max_{0 \leq i < n} \left(\min_{j \in P_i} \left(\frac{\gamma_i}{2}, \max\left(\frac{\gamma_j}{2}, \eta^+_{i,j} - \frac{\pi}{2}, \eta^-_{i,j} - \frac{\pi}{2} \right) \right) \right)$, where $\eta^+_{i,j} := \sum_{C_k \in P^+_{i+1,j-1}} \delta_k$ and $\eta^-_{i,j} := \sum_{C_k \in P^-_{i-1,j+1}} \delta_k$, then every configuration of two particles can be reached.*

▶ **Theorem 3.** *If P is a regular polygon with n vertices and if $\mu > \cot(\pi/n)$, then every reconfiguration is possible.*

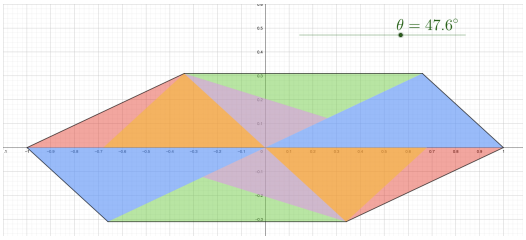
These results for static friction can be extended to rearranging multiple particles; this is visualized in the video, and demonstrated for a real-world application.



(a) Small angle of friction.



(b) Medium angle of friction.



(c) Large angle of friction.

Figure 3 Combining the different strategies for covering Δ space. **(a)** For small θ, a portion remains uncovered, while the rest is covered by the blue, orange and red strategies. **(b)** For growing θ, green and violet strategies cover an increasing portion of the remaining sections. **(c)** For large enough θ, the whole Δ space is covered.

▶ **Theorem 4.** *Consider the class $\mathcal{C}$ of configurations of three particles in a square, where one of the particles lies within the bounding rectangle of the other two particles. If $\theta > \frac{\pi}{4}$, then we can reconfigure any configuration to any configuration of $\mathcal{C}$.*

Using induction, we can achieve arbitrary reconfiguration of a set of collinear particles, as demonstrated in the video for an example with six particles.

▶ **Theorem 5.** *For $\theta > \frac{\pi}{4}$, we can sort any permuted set of collinear particles.*

3 The video

The video starts with an introduction of controlling a swarm of particles or robots by a uniform global force, and the problem of controlled reconfiguration. After a brief review of previous work (which employed obstacles), we introduce the approach of using local differences in boundary friction for breaking symmetry between different particles. This is followed by a description of involved parameters, the concept of Δ space and our five different, "colored" strategies for using static friction to achieve arbitrary reconfiguration of two particles. This ultimately leads to Theorem 1 and can be extended to Theorems 2 and 3; it also can be extended to multiple particles, which yields Theorems 4 and 5. We conclude with practical demonstrations with real particles that are rearranged by a robot controller, and a pair of particles of size 1 mm that are relocated in the stomach of a cow.

References

1 Srinivas Akella, Wesley H Huang, Kevin M Lynch, and Matthew T Mason. Parts feeding on a conveyor with a one joint robot. *Algorithmica*, 26(3-4):313–344, 2000.

2 Greg Aloupis, Jean Cardinal, Sébastien Collette, Ferran Hurtado, Stefan Langerman, and Joseph O'Rourke. Draining a polygon – or – rolling a ball out of a polygon. *Computational Geometry*, 47(2):316–328, 2014.

3 DJ Arbuckle and Aristides AG Requicha. Self-assembly and self-repair of arbitrary shapes by a swarm of reactive robots: algorithms and simulations. *Autonomous Robots*, 28(2):197–211, 2010.

4 Jose Balanza-Martinez, Austin Luchsinger, David Caballero, Rene Reyes, Angel A Cantu, Robert Schweller, Luis Angel Garcia, and Tim Wylie. Full tilt: Universal constructors for general shapes with uniform external forces. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2689–2708, 2019.

5 A. Becker, Erik D. Demaine, Sándor P. Fekete, S. H. Mohtasham Shad, and R. Morris-Wright. Tilt: The video. designing worlds to control robot swarms with only global signals. In *Symposium on Computational Geometry (SoCG)*, pages 16–18, 2015.

6 Aaron Becker, Erik D Demaine, Sándor P Fekete, and James McLurkin. Particle computation: Designing worlds to control robot swarms with only global signals. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 6751–6756, 2014.

7 Aaron T. Becker, Erik D. Demaine, Sándor P. Fekete, Jarrett Lonsford, and Rose Morris-Wright. Particle computation: complexity, algorithms, and logic. *Natural Computing*, 18(1):181–201, 2019.

8 Aaron T Becker, Sándor P Fekete, Phillip Keldenich, Dominik Krupke, Christian Rieck, Christian Scheffer, and Arne Schmidt. Tilt assembly: algorithms for micro-factories that build objects with uniform external forces. *Algorithmica*, pages 1–23, 2017.

9 Paul Seung Soo Kim, Aaron T. Becker, Yan Ou, Anak Agung Julius, and Min Jun Kim. Imparting magnetic dipole heterogeneity to internalized iron oxide nanoparticles for microorganism swarm control. *Journal of Nanoparticle Research*, 17(3):1–15, 2015.

10 Jeremy S. Lewis and Jason M. O'Kane. Planning for provably reliable navigation using an unreliable, nearly sensorless robot. *The International Journal of Robotics Research*, 32(11):1342–1357, 2013.

11 Kevin M Lynch and Matthew T Mason. Stable pushing: Mechanics, controllability, and planning. *The International Journal of Robotics Research*, 15(6):533–556, 1996.

12 Michael Rubenstein, Christian Ahler, Nick Hoff, Adrian Cabrera, and Radhika Nagpal. Kilobot: A low cost robot with scalable operations designed for collective behaviors. *Robotics and Autonomous Systems*, 62(7):966–975, 2014.

13 Arne Schmidt, Victor M. Baez, Aaron T. Becker, and Sándor P. Fekete. Coordinated particle relocation using finite static friction with boundary walls. *Robotics and Automation Letters*, 2:985–992, 2020.

14 Arne Schmidt, Sheryl Manzoor, Li Huang, Aaron T Becker, and Sándor P Fekete. Efficient parallel self-assembly under uniform control inputs. *IEEE Robotics and Automation Letters*, 3(4):3521–3528, 2018.

15 Y. Zhang, X. Chen, H. Qi, and D. Balkcom. Rearranging agents in a small space using global controls. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3576–3582, 2017. doi:10.1109/IROS.2017.8206202.

16 Yinan Zhang, Emily Whiting, and Devin Balkcom. Assembling and disassembling planar structures with divisible and atomic components. *Transactions on Automation Science and Engineering*, 15(3):945–954, 2018.

Space Ants: Constructing and Reconfiguring Large-Scale Structures with Finite Automata

Amira Abdel-Rahman
Center for Bits and Atoms, MIT, Cambridge, MA, USA
amira.abdel-rahman@cba.mit.edu

Aaron T. Becker
Department of Electrical and Computer Engineering, University of Houston, TX, USA
atbecker@uh.edu

Daniel E. Biediger
Department of Electrical and Computer Engineering, University of Houston, TX, USA
dbiediger@gmail.com

Kenneth C. Cheung
Coded Structures Lab, NASA Ames Research Center, Moffett Field, CA, USA
kenny@nasa.gov

Sándor P. Fekete
Department of Computer Science, TU Braunschweig, Germany
s.fekete@tu-bs.de

Neil A. Gershenfeld
Center for Bits and Atoms, MIT, Cambridge, MA, USA
neil.gershenfeld@cba.mit.edu

Sabrina Hugo
Department of Computer Science, TU Braunschweig, Germany
s.hugo@tu-bs.de

Benjamin Jenett
Center for Bits and Atoms, MIT, Cambridge, MA, USA
bej@mit.edu

Phillip Keldenich
Department of Computer Science, TU Braunschweig, Germany
p.keldenich@tu-bs.de

Eike Niehs
Department of Computer Science, TU Braunschweig, Germany
e.niehs@tu-bs.de

Christian Rieck
Department of Computer Science, TU Braunschweig, Germany
c.rieck@tu-bs.de

Arne Schmidt
Department of Computer Science, TU Braunschweig, Germany
arne.schmidt@tu-bs.de

Christian Scheffer
Department of Computer Science, TU Braunschweig, Germany
c.scheffer@tu-bs.de

Michael Yannuzzi
Department of Electrical and Computer Engineering, University of Houston, TX, USA
mcyannuz@central.uh.edu

Abstract

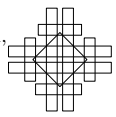
In this video, we consider recognition and reconfiguration of lattice-based cellular structures by very simple robots with only basic functionality. The underlying motivation is the construction and modification of space facilities of enormous dimensions, where the combination of new materials with extremely simple robots promises structures of previously unthinkable size and flexibility. We present algorithmic methods that are able to detect and reconfigure arbitrary polyominoes, based on finite-state robots, while also preserving connectivity of a structure during reconfiguration. Specific results include methods for determining a bounding box, scaling a given arrangement, and adapting more general algorithms for transforming polyominoes.

2012 ACM Subject Classification Theory of computation → Computational geometry; Theory of computation → Formal languages and automata theory

Keywords and phrases Finite automata, reconfiguration, construction, scaling

Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.73

Category Media Exposition



36th International Symposium on Computational Geometry (SoCG 2020).
Editors: Sergio Cabello and Danny Z. Chen; Article No. 73; pp. 73:1–73:6

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

1 Introduction

Building and modifying large-scale structures is an important and natural objective in a vast array of applications. In many cases, the use of autonomous robots promises significant advantages, but also a number of additional difficulties. This is particularly true in space, where the difficulties of expensive supply chains, scarcity of building materials, dramatic costs and consequences of even small errors, and the limitations of outside intervention in case of malfunctions pose a vast array of extreme challenges.

In recent years, a number of significant advances have been made to facilitate overall breakthroughs. One important step has been the development of ultra-light and scalable composite lattice materials [29] that allow the construction of modular, reconfigurable, lattice-based structures [35]; see Figure 1. A second step has been the design of simple autonomous robots [32, 34] that are able to move on the resulting lattice structures and move their cell components, allowing the reconfiguration of the overall edifice; see Figure 2.

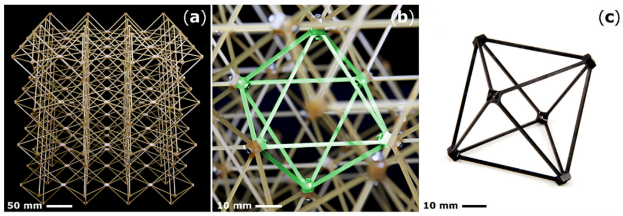


Figure 1 (a) An assembled cuboctahedral lattice specimen, made from octahedral unit cells (highlighted), termed *voxels*. (c) A single injection molded voxel. (See [29].)

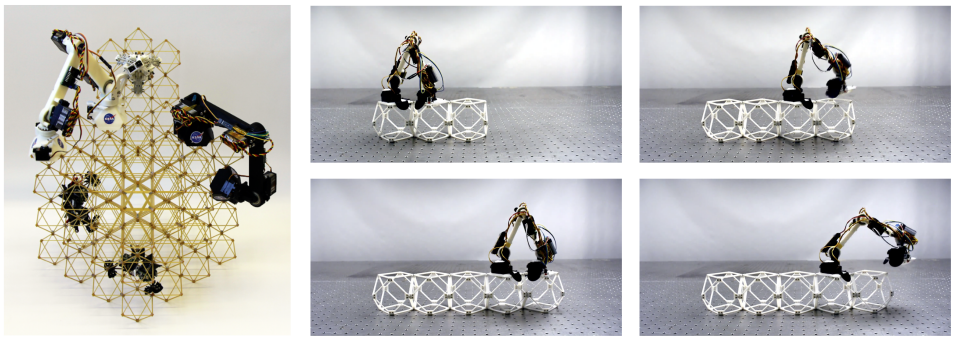


Figure 2 (Left) Modular reconfigurable 3D lattice structure and mobile robots; note how robots are similar in size to lattice cells, and the parallel use of multiple robots. (See [7].) (Right) A sequence of images from the video: a BILL-E robot moving on an expanding row of voxels. (See [31].)

We address the next step in this hierarchy: Can we enable extremely simple robots to perform a more complex spectrum of construction tasks for cellular structures in space, such as patrolling and marking the perimeter, scaling up a given seed construction, and a number of other design operations? As we demonstrate, finite automata can achieve these tasks.

2 Related Work

The structures considered in this work are based on *ultra-light material*, as described by Cheung and Gershenfeld [6] and Gregg et al. [29]. Modular two-dimensional elements mechanically link in 3D to form reversibly assembled composite lattices. This process is

not limited by scale, and it enables disassembly and reconfiguration. As shown by Cramer et al. [8] and Jenett et al. [33], large but light-weight structures can be built from these components. Jenett et al. have developed autonomous robots that move on the surface [32, 31] or within the cellular structure [34]. With the help of these robots, individual cells can be attached to an existing assembly, or moved to a different location [31]. An approach for global optimization of a corresponding motion plan has been described by Costa et al. [7], while the design of hierarchical structures was addressed by Jenett et al. [36].

Assembly by simple robots has also been considered at the micro scale, where global control is used for supplying the necessary force for moving agents, e.g., see Becker et al. [2] for the corresponding problem of motion planning, Schmidt et al. [39] for using this model for assembling structures, and Balanza-Martinez et al. [1] for theoretical characterizations. On the algorithmic side, work dealing with *robots or agents on graphs* includes Blum and Kozen [4], who showed that two finite automata can jointly search any unknown maze. Other work has focused on exploring general graphs (e.g.,[38, 23, 20]), as a distributed or collaborative problem using multiple agents (e.g. [3, 21, 9, 5]) or with space limitations (e.g. [22, 23, 17, 24, 25]).

From an algorithmic view, we are interested in *different models representing programmable matter* and further recent results. Inspired by the single-celled amoeba, Derakhshandeh et al. introduced the Amoebot model [11] and later a generalized variant, the general Amoebot model [15]; see [13, 10, 16, 14, 12] for various results in this model. Other models with active particles were introduced in [40] as the Nubot model and in [30] with modular robots. In [26], Gmyr et al. introduced a model with two types of particles: active robots acting like a deterministic finite automaton and passive tile particles. Furthermore, they presented algorithms for shape formation [28] and shape recognition [27] using robots on tiles.

3 Results for Finite Automata

We consider a set of N two-dimensional orthogonal *tiles* that form a *polyomino* P of total width w and height h. We use *robots* as active particles, which work like *finite deterministic automata* that can move between adjacent grid positions, where they can place or remove a tile. We assume that different robots cannot occupy the same position at the same time, and communication between robots is limited to adjacent positions. A basic step for recognizing and possibly reconfiguring P is based on constructing its bounding box $bb(P)$, which is the boundary of the smallest axis-aligned rectangle enclosing but not touching P; this implies that there is a gap of one tile between the two, so we use a robot to keep the two parts connected.

The first result demonstrated in the video deals with constructing the bounding box, and thus recognizing the extent of a shape. See [19, 18, 37] for technical details.

▶ **Theorem 1.** *Given a polyomino P of width w and height h, we can build a bounding box surrounding P with the boundary and P always being connected, with two finite-state robots in $O(\max(w,h) \cdot (wh + k \cdot |\partial P|))$ steps, where k is the number of convex corners in P.*

The second result demonstrated in the video achieves *scaling* of a given shape.

▶ **Theorem 2.** *After building $bb(P)$, scaling a polyomino P of width w and height h by a constant scaling factor c without loss of connectivity can be done with one finite-state robot in $O(wh \cdot (c^2 + cw + ch))$ steps.*

Further reconfiguration results mentioned in the video are as follows.

▶ **Theorem 3.** *Copying a polyomino P columnwise can be done within $\mathcal{O}(wh^2)$ steps using $\mathcal{O}(N)$ of auxiliary particles and $\mathcal{O}(wh)$ additional space in $O(h)$ extra rows and columns.*

▶ **Theorem 4.** *Reflecting a polyomino P horizontally can be done in $\mathcal{O}(w^2h)$ steps, using $\mathcal{O}(w)$ of additional space and $\mathcal{O}(w)$ auxiliary particles.*

▶ **Theorem 5.** *There is a strategy to rotate a polyomino P by $\pm\frac{\pi}{2}$ within $\mathcal{O}((w+h)wh)$ steps, using $\mathcal{O}(w+h+|w-h|h)$ of additional space in $\mathcal{O}(|w-h|+1)$ extra rows and columns and $\mathcal{O}(w+h)$ auxiliary particles.*

Finally, the video demonstrates how we can carry out any geometric transformation by finite-state robots, if and only if there is a corresponding Turing machine for transforming the corresponding one-dimensional string $S(P_1)$ (arising from a row-wise scan of P_1) into $S(P_2)$.

▶ **Theorem 6.** *Let P_1 and P_2 be two polyominos with $|P_1| = |P_2| = N$. There is a strategy transforming P_1 into P_2 if there is a Turing machine transforming the corresponding one-dimensional string $S(P_1)$ into $S(P_2)$. The finite-state robot needs $\mathcal{O}(\partial P_1 + \partial P_2 + S_{TM})$ auxiliary particles, $\mathcal{O}(N^4 + T_{TM})$ steps, and $\Theta(N^2 + S_{TM})$ of additional space, where T_{TM} and S_{TM} are the number of steps and additional space needed by the Turing machine.*

4 The Video

The video starts with a discussion of the problems faced when building large-scale structures in space, and an introduction of digital, ultra light-weight materials and simple robots currently developed at MIT and NASA. This is followed by a description of finite automata corresponding to finite-state robots. As a first algorithmic demonstration, the connected construction of the bounding box of a given polyomino shape is shown, followed by producing a scaled copy of a shape. Then we show how general constructions can be built based on methods of Turing machines. The video concludes with a 3D simulation.

References

1 Jose Balanza-Martinez, Austin Luchsinger, David Caballero, Rene Reyes, Angel A Cantu, Robert Schweller, Luis Angel Garcia, and Tim Wylie. Full tilt: universal constructors for general shapes with uniform external forces. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2689–2708, 2019.

2 Aaron T Becker, Sándor P Fekete, Phillip Keldenich, Dominik Krupke, Christian Rieck, Christian Scheffer, and Arne Schmidt. Tilt assembly: Algorithms for micro-factories that build objects with uniform external forces. *Algorithmica*, 82:165–187, 2020.

3 M. A. Bender and D. K. Slonim. The power of team exploration: two robots can learn unlabeled directed graphs. In *Symposium on Foundations of Computer Science (FOCS*, pages 75–85, 1994. doi:10.1109/SFCS.1994.365703.

4 M. Blum and D. Kozen. On the power of the compass (or, why mazes are easier to search than graphs). In *Symposium on Foundations of Computer Science (FOCS)*, pages 132–142, 1978. doi:10.1109/SFCS.1978.30.

5 P. Brass, F. Cabrera-Mora, A. Gasparri, and J. Xiao. Multirobot tree and graph exploration. *IEEE Transactions on Robotics*, 27(4):707–717, 2011. doi:10.1109/TRO.2011.2121170.

6 Kenneth C Cheung and Neil Gershenfeld. Reversibly assembled cellular composite materials. *Science*, 341(6151):1219–1221, 2013.

7 Allan Costa, Amira Abdel-Rahman, Benjamin Jenett, Neil Gershenfeld, Irina Kostitsyna, and Kenneth Cheung. Algorithmic approaches to reconfigurable assembly systems. In *IEEE Aerospace Conference*, pages 1–8, 2019.

8 Nicholas B Cramer, Daniel W Cellucci, Olivia B Formoso, Christine E Gregg, Benjamin E Jenett, Joseph H Kim, Martynas Lendraitis, Sean S Swei, Greenfield T Trinh, Khanh V Trinh, and Kenneth C. Cheung. Elastic shape morphing of ultralight structures by programmable assembly. *Smart Materials and Structures*, 28(5):055006, 2019.

9 Shantanu Das, Paola Flocchini, Shay Kutten, Amiya Nayak, and Nicola Santoro. Map construction of unknown graphs by multiple agents. *Theoretical Computer Science*, 385(1):34–48, 2007. doi:10.1016/j.tcs.2007.05.011.

10 Joshua J. Daymude, Robert Gmyr, Andréa W. Richa, Christian Scheideler, and Thim Strothmann. Improved leader election for self-organizing programmable matter. In *Algorithms for Sensor Systems (ALGOSENSORS)*, pages 127–140, 2017. doi:10.1007/978-3-319-72751-6_10.

11 Zahra Derakhshandeh, Shlomi Dolev, Robert Gmyr, Andréa W. Richa, Christian Scheideler, and Thim Strothmann. Brief announcement: Amoebot – a new model for programmable matter. In *ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 220–222, 2014. doi:10.1145/2612669.2612712.

12 Zahra Derakhshandeh, Robert Gmyr, Alexandra Porter, Andréa W. Richa, Christian Scheideler, and Thim Strothmann. On the runtime of universal coating for programmable matter. In Yannick Rondelez and Damien Woods, editors, *DNA Computing and Molecular Programming*, pages 148–164, 2016. doi:10.1007/978-3-319-43994-5_10.

13 Zahra Derakhshandeh, Robert Gmyr, Andréa W. Richa, Christian Scheideler, and Thim Strothmann. An algorithmic framework for shape formation problems in self-organizing particle systems. In *International Conference on Nanoscale Computing and Communication (NANOCOM)*, pages 21:1–21:2, 2015. doi:10.1145/2800795.2800829.

14 Zahra Derakhshandeh, Robert Gmyr, Andréa W. Richa, Christian Scheideler, and Thim Strothmann. Universal coating for programmable matter. *CoRR*, abs/1601.01008, 2016. arXiv:1601.01008.

15 Zahra Derakhshandeh, Robert Gmyr, Thim Strothmann, Rida Bazzi, Andréa W. Richa, and Christian Scheideler. Leader election and shape formation with self-organizing programmable matter. In Andrew Phillips and Peng Yin, editors, *DNA Computing and Molecular Programming*, pages 117–132, 2015. doi:10.1007/978-3-319-21999-8_8.

16 Giuseppe Antonio Di Luna, Paola Flocchini, Nicola Santoro, Giovanni Viglietta, and Yukiko Yamauchi. Shape formation by programmable particles. *CoRR*, abs/1705.03538, 2017. arXiv:1705.03538.

17 Krzysztof Diks, Pierre Fraigniaud, Evangelos Kranakis, and Andrzej Pelc. Tree exploration with little memory. *Journal of Algorithms*, 51(1):38–63, 2004. doi:10.1016/j.jalgor.2003.10.002.

18 Sándor P. Fekete, Robert Gmyr, Sabrina Hugo, Phillip Keldenich, Christian Scheffer, and Arne Schmidt. CADbots: Algorithmic aspects of manipulating programmable matter with finite automata. *CoRR*, abs/1810.06360, 2018. To appear in: Proceedings of Workshop of Algorithmic Foundations of Robotics (WAFR 2018). arXiv:1810.06360.

19 Sándor P. Fekete, Eike Niehs, Christian Scheffer, and Arne Schmidt. Connected assembly and reconfiguration by finite automata. *CoRR*, abs/1909.03880, 2019. arXiv:1909.03880.

20 Rudolf Fleischer and Gerhard Trippen. Exploring an unknown graph efficiently. In *European Symposium on Algorithms (ESA)*, pages 11–22, 2005.

21 Pierre Fraigniaud, Leszek Gasieniec, Dariusz R. Kowalski, and Andrzej Pelc. Collective tree exploration. *Networks*, 48(3):166–177, 2006. doi:10.1002/net.20127.

22 Pierre Fraigniaud and David Ilcinkas. Digraphs exploration with little memory. In *Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 246–257, 2004.

23 Pierre Fraigniaud, David Ilcinkas, Guy Peer, Andrzej Pelc, and David Peleg. Graph Exploration by a Finite Automaton. *Theoretical Computer Science*, 345(2-3):331–344, 2005. doi:10.1016/j.tcs.2005.07.014.

24 Leszek Gasieniec, Andrzej Pelc, Tomasz Radzik, and Xiaohui Zhang. Tree exploration with logarithmic memory. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 585–594, 2007. URL: http://dl.acm.org/citation.cfm?id=1283383.1283446.

25 Leszek Gasieniec and Tomasz Radzik. Memory efficient anonymous graph exploration. In *Graph-Theoretic Concepts in Computer Science (WG)*, pages 14–29, 2008.

26 R. Gmyr, I. Kostitsyna, F. Kuhn, C. Scheideler, and T. Strothmann. Forming tile shapes with a single robot. In *European Workshop on Computational Geometry (EuroCG 2017)*, pages 9–12, 2017. URL: https://research.tue.nl/en/publications/forming-tile-shapes-with-a-single-robot.

27 Robert Gmyr, Kristian Hinnenthal, Irina Kostitsyna, Fabian Kuhn, Dorian Rudolph, and Christian Scheideler. Shape Recognition by a Finite Automaton Robot. In *Mathematical Foundations of Computer Science (MFCS)*, pages 52:1–52:15, 2018. doi:10.4230/LIPIcs.MFCS.2018.52.

28 Robert Gmyr, Kristian Hinnenthal, Irina Kostitsyna, Fabian Kuhn, Dorian Rudolph, Christian Scheideler, and Thim Strothmann. Forming tile shapes with simple robots. In *DNA Computing and Molecular Programming*, pages 122–138, 2018. doi:10.1007/978-3-030-00030-1_8.

29 Christine E Gregg, Joseph H Kim, and Kenneth C Cheung. Ultra-light and scalable composite lattice materials. *Advanced Engineering Materials*, 20(9):1800213, 2018.

30 Ferran Hurtado, Enrique Molina, Suneeta Ramaswami, and Vera Sacristán. Distributed reconfiguration of 2d lattice-based modular robotic systems. *Autonomous Robots*, 38(4):383–413, 2015. doi:10.1007/s10514-015-9421-8.

31 B. Jenett, A. Abdel-Rahman, K. Cheung, and N. Gershenfeld. Material–robot system for assembly of discrete cellular structures. *IEEE Robotics and Automation Letters*, 4:4019–4026, 2019.

32 Ben Jenett and Kenneth Cheung. BILL-E: Robotic platform for locomotion and manipulation of lightweight space structures. In *AIAA/AHS Adaptive Structures Conference*, pages 1876–ff., 2017.

33 Benjamin Jenett, Sam Calisch, Daniel Cellucci, Nick Cramer, Neil Gershenfeld, Sean Swei, and Kenneth C Cheung. Digital morphing wing: active wing shaping concept using composite lattice-based cellular structures. *Soft Robotics*, 4(1):33–48, 2017.

34 Benjamin Jenett and Daniel Cellucci. A mobile robot for locomotion through a 3D periodic lattice environment. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5474–5479, 2017.

35 Benjamin Jenett, Daniel Cellucci, Christine Gregg, and Kenneth Cheung. Meso-scale digital materials: modular, reconfigurable, lattice-based structures. In *International Manufacturing Science and Engineering Conference*. American Society of Mechanical Engineers Digital Collection, 2016.

36 Benjamin Jenett, Christine Gregg, Daniel Cellucci, and Kenneth Cheung. Design of multi-functional hierarchical space structures. In *IEEE Aerospace Conference*, pages 1–10, 2017.

37 Eike Niehs, Arne Schmidt, Christian Scheffer, Daniel E. Biediger, Michael Yanuzzi, Benjamin Jenett, Amira Abdel-Rahman, Kenneth C. Cheung, Aaron T. Becker, and Sándor P. Fekete. Recognition and reconfiguration of lattice-based cellular structures by simple robots. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2020. To appear.

38 Petrişor Panaite and Andrzej Pelc. Exploring unknown undirected graphs. *Journal of Algorithms*, 33(2):281–295, 1999. doi:10.1006/jagm.1999.1043.

39 Arne Schmidt, Sheryl Manzoor, Li Huang, Aaron T Becker, and Sándor P Fekete. Efficient parallel self-assembly under uniform control inputs. *IEEE Robotics and Automation Letters*, 3(4):3521–3528, 2018.

40 Damien Woods, Ho-Lin Chen, Scott Goodfriend, Nadine Dabby, Erik Winfree, and Peng Yin. Active self-assembly of algorithmic shapes and patterns in polylogarithmic time. In *Innovations in Theoretical Computer Science (ITCS)*, pages 353–354, 2013. doi:10.1145/2422436.2422476.

How to Make a CG Video

Aaron T. Becker
Department of Electrical and Computer Engineering, University of Houston, TX, USA
atbecker@uh.edu

Sándor P. Fekete
Department of Computer Science, TU Braunschweig, Germany
s.fekete@tu-bs.de

Abstract

In this video we describe why producing a Computational Geometry video is a good idea, what it takes to make one, and how to actually do it. This includes a guide for the overall process, a number of examples, and a variety of tips and tricks.

2012 ACM Subject Classification Theory of computation → Computational geometry; Applied computing → Education

Keywords and phrases Videos, animation, education, SoCG Multimedia

Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.74

Category Media Exposition

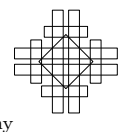
1 Introduction

Developing and communicating abstract concepts is one of the fundamental capabilities of humans that distinguish them from other beings [20]. For tens of thousands of years, this communication was based on direct human interaction; one of the crucial breakthroughs of humankind was the development of methods that allow conveying information by other means, in particular, by written language [21]. This has allowed spreading and preserving information and ideas, independent of physical presence.

In this evolution, mathematics has been at the forefront of exploring and expanding the realm of abstract concepts; moreover, the development of mathematics has greatly benefitted from a rigorous separation of human emotions and experience from logic and truth. As a consequence, many mathematical ideas are difficult to convey and comprehend; however, in the words of Terry Tao [22]: "There's more to mathematics than rigour and proofs... The point of rigour is not to destroy all intuition; instead, it should be used to destroy bad intuition while clarifying and elevating good intuition."

What are the best ways to convey mathematical ideas and intuition? While human brains have dealt with visual information since the beginning of humankind, processing letters and mathematical notation has been a very recent development. Moreover, the evolution of visual information processing by humans has largely been fueled by the need to recognize the position, motion and interaction of objects in space, so the affinity of human brains to geometry is deeply rooted [2]. Therefore, it is quite natural to make use of geometry to convey mathematical ideas – in particular, in the field of geometry itself, as demonstrated by the visual power of Byrne's classic illustrated version of Euclid's elements [10].

Over the years, many other scientific areas have contributed to advances in visualization [18, 19], at the same time demonstrating the power of turning complex information into convincing geometric representations. This development has been fueled both by the need for processing and representing large amounts of data, as well as the availability of more and more powerful computational devices, algorithms and interfaces.




2 Videos in Computational Geometry

All this makes it natural to use dynamic visual presentation of concepts and ideas in Computational Geometry. Now in its 29th year, the multimedia track of SoCG has demonstrated how videos not only allow making use of dynamically changing objects, but can also make use of the fundamental concept of storytelling, already mentioned above [21].

There are a number of other good reasons for capturing and visualizing scientific ideas in the form of videos. They allow presenting results and ideas in a *dynamic* fashion, instead of statically as on paper; because they allow demonstrating how things really work, they are more *convincing*. With concise and well-planned presentation, they are also *compact*, allowing them to present ideas and results in just a few minutes.

Particularly important is the *portability* of digital content, which can travel large distances without requiring physical presence. While this video was designed and submitted before the onset of recent world events, the new reality of social distancing and remote teaching has made it irrefutably clear that the need to convey complex ideas without direct human proximity will greatly benefit from a large spectrum of methods for digital representation.

Other benefits of a video are a lasting, *memorable* impression, and the *reproducibility* of running it. Finally, the process of carefully planning and producing a video is also *inspiring*, helping to make regular talks better. This is particularly true for long-distance presentations, which benefit from mastering all aspects of the involved technologies, a clear, crisp delivery, and the ability to quickly turn a talk into a video production when a conference moves from physical to virtual presence. A good example was the 2020 *European Workshop on Computational Geometry*, for which 77 talks were turned into videos within just three days [11].

The process described in the following is based on the experience of producing a number of SoCG videos [15, 16, 9, 4, 5, 6, 7, 14, 13, 1, 3]), as well as serving on several multimedia committees. There is a wide range of other tools that can be used, and the need for remote presentations will lead to further progress; however, the underlying process of planning the story, planning visual elements, sound, process and mastering the tools is pretty much independent of those specifics.

3 The *How to* video

The video starts with a number of introductory scenes filmed in front of a *lightboard* [8], an illuminated piece of glass that can be used similar to a whiteboard, with the speaker facing the camera; see Figure 1. This produces mirror-inverted writing, which is flipped in video editing. After an initial motivation for producing videos, the main components are introduced: results, story, visual elements, sound, process, tools, and abstract.

The following part discusses putting together the story, based on combining key ideas with visual elements and developing an overall plot. A top-down approach for structuring a screenplay is recommended, which is then refined with a storyboard for planning the visual elements (Figure 2 (Left)). These are discussed in the context of presenting mathematical results (such as the ones from paper [12] and video [13]), in particular, turning ideas into dynamic visualizations, and presenting mathematical theorems as *images* of mathematical theorems. This is followed by a demonstration of how effective use of sound can turn even a simple sequence of images into a story, and by a brief glimpse at a completed screenplay.

The next step focuses on producing animation based on standard presentation software, such as Keynote or PowerPoint, which are then exported into a video format or converted by a screen capture; the particular demonstration visualizes the dimensions of large-scale structures in space (as shown in Figure 2 (Right)), which is part of the SoCG video [1].

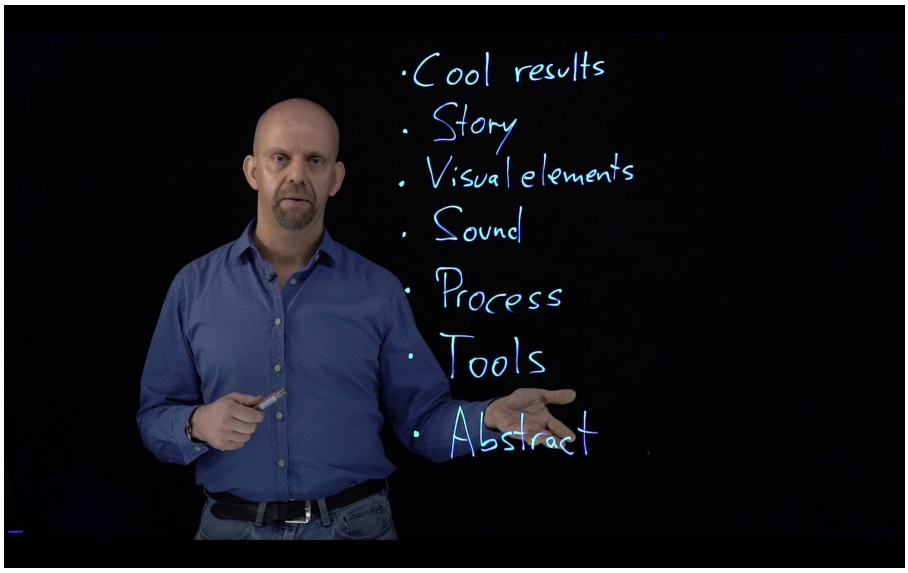


Figure 1 Lightboard presentation: components for making a video.

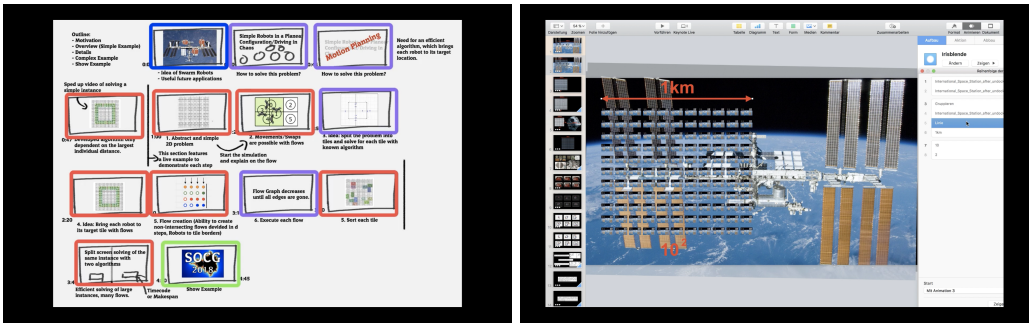


Figure 2 (Left) A storyboard for the video [6]. (Right) Generating animations for the video [1].

This is followed by recommendations for producing the voiceover. A particularly helpful tip is to simply record the sound by itself in one continuous session, without the need for completely avoiding coughing or misspeaking, or synchronicity to animations: If an error occurs, there is no need to start over; it suffices to simply repeat the line until it is correct. Any issues can be worked out later in video editing.

The remainder deals with tools and processes by showing a variety of capabilities of a good video program, demonstrated in Camtasia as a concrete example. The main focus is on editing sound and video to create a well-synchronized movie, described for major parts of this "How to" video itself, as shown in Figure 3. The first step consists in cutting video and sound into relatively small individual pieces, which can then easily be arranged appropriately. In most cases, it works best to let the sound be the guide for the timing of the video, as it is easier to speed up or slow down the latter; this is demonstrated with the opening of this very video, for which the animation steps are synchronized to the beats of the music. Another tip is the combination of different video elements (such as slides, real-world video, and labels) into the same scene; a demonstration uses a snippet of a mathematical talk [17] with a dance video featuring one of the authors (Figure 4). Finally, everything is exported into one integral video.

Figure 3 Four levels of editing for this video, shown in Camtasia. (Level 0, white rectangle) The title sequence for this video. (Level 1, inner black frame) Editing the title sequence. (Level 2, medium black frame) Showing how to synchronize sound and video in the title sequence. (Level 3, outer black frame) Screenshot of editing Level 2 for this video.

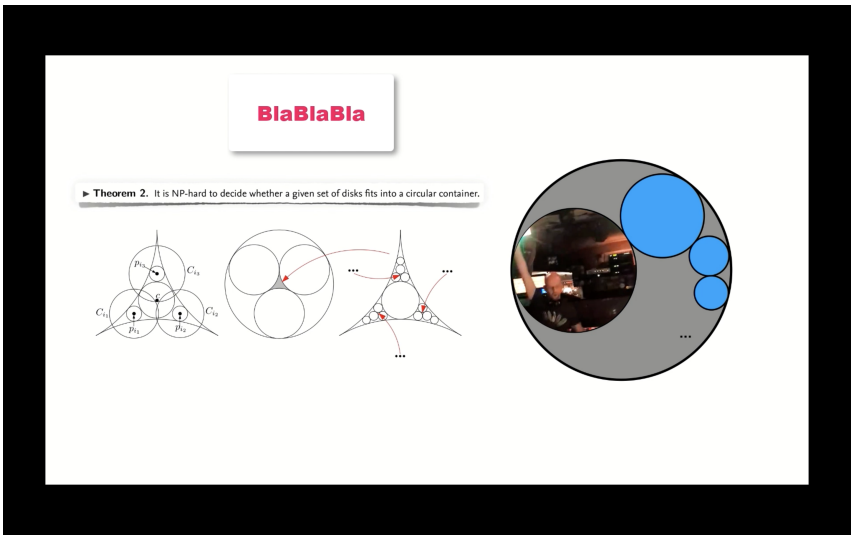


Figure 4 Combining multiple elements in the same scene with the help of a video editor, such as a slide, a label (“BlaBlaBla”), and some real-world video.

References

1. Amira Abdel-Rahman, Aaaron T. Becker, Daniel E. Biedinger, Kenneth C. Cheung, Sándor P. Fekete, Sabrina Hugo, Benjamin Jenett, Phillip Keldenich, Eike Niehs, Christian Rieck, Arne Schmidt, Christian Scheffer, and Michael Yannuzzi. Space ants: Constructing and reconfiguring large-scale structures with finite automata. In *36th International Symposium on Computational Geometry (SoCG)*, volume 164 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 73:1–73:6, 2020. These proceedings. doi:10.4230/LIPIcs.SoCG.2020.73.
2. William L Abler. The human mind: origin in geometry. *Science progress*, 93(4):403–427, 2010.
3. Victor Baez, Aaron T. Becker, Sándor P. Fekete, and Arne Schmidt. Coordinated particle relocation with global signals and local friction. In *36th International Symposium on Computational Geometry (SoCG)*, volume 164 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 72:1–72:5, 2020. These proceedings. doi:10.4230/LIPIcs.SoCG.2020.72.
4. A. T. Becker, Erik D. Demaine, Sándor P. Fekete, S. H. Mohtasham Shad, and R. Morris-Wright. Tilt: The video. designing worlds to control robot swarms with only global signals. In *31st International Symposium on Computational Geometry (SoCG)*, pages 16–18, 2015. Video available at http://www.computational-geometry.org/SoCG-videos/socg15video/.
5. Aaron T. Becker, Mustapha Debboun, Sándor P. Fekete, Dominik Krupke, and An Nguyen. Zapping Zika with a Mosquito-Managing Drone: Computing Optimal Flight Patterns with Minimum Turn Cost. In *33rd International Symposium on Computational Geometry (SoCG)*, volume 77 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 62:1–62:5, 2017. Video available at http://www.computational-geometry.org/SoCG-videos/socg17video/. doi:10.4230/LIPIcs.SoCG.2017.62.
6. Aaron T. Becker, Sándor P. Fekete, Phillip Keldenich, Matthias Konitzny, Lillian Lin, and Christian Scheffer. Coordinated Motion Planning: The Video (Multimedia Exposition). In *34th International Symposium on Computational Geometry (SoCG)*, volume 99 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 74:1–74:6, 2018. Video available at http://www.computational-geometry.org/SoCG-videos/socg18video/. doi:10.4230/LIPIcs.SoCG.2018.74.
7. Aaron T. Becker, Sándor P. Fekete, Phillip Keldenich, Sebastian Morr, and Christian Scheffer. Packing Geometric Objects with Optimal Worst-Case Density (Multimedia Exposition). In *35th International Symposium on Computational Geometry (SoCG)*, volume 129 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 63:1–63:6, 2019. Video available at http://www.computational-geometry.org/SoCG-videos/socg19video/. doi:10.4230/LIPIcs.SoCG.2019.63.
8. J Alex Birdwell and Michael Peshkin. Capturing technical lectures on lightboard. In *ASEE Annual Conference & Exposition*, volume 26, pages 12851:1–12851:9, 2015.
9. Dorit Borrmann, Pedro de Rezende, Clécio de Souza, Sándor P. Fekete, Alexander Kröller, Andreas Nüchter, and Christiane Schmidt. Point guards and point clouds: Solving general art gallery problems. In *29th Annual ACM Symposium on Computational Geometry (SoCG)*, pages 347–348, 2013. Video available at http://www.computational-geometry.org/SoCG-videos/socg13video/.
10. Oliver Byrne. *The first six books of the Elements of Euclid: in which coloured diagrams and symbols are used instead of letters for the greater ease of learners*. William Pickering, 1847.
11. Stephen Chaplick, Philipp Kindermann, and Alexander Wolff, editors. *36th European Workshop on Computational Geometry (EuroCG)*, 2020. URL: http://www1.pub.informatik.uni-wuerzburg.de/eurocg2020/.
12. Sándor P. Fekete, Utkarsh Gupta, Phillip Keldenich, Christian Scheffer, and Sahil Shah. Worst-Case Optimal Covering of Rectangles by Disks. In *Proceedings 36th International Symposium on Computational Geometry (SoCG)*, volume 164 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 42:1–42:23, 2020. These proceedings. doi:10.4230/LIPIcs.SoCG.2020.42.

13 Sándor P. Fekete, Phillip Keldenich, and Christian Scheffer. Covering rectangles by disks: The video. In *Proceedings of the 36th International Symposium on Computational Geometry (SoCG)*, volume 164 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 75:1–75:4, 2020. These proceedings. doi:10.4230/LIPIcs.SoCG.2020.75.

14 Sándor P. Fekete, Rolf Klein, and A. Nüchter. Searching with an autonomous robot. In *20th Annual ACM Symposium on Computational Geometry (SoCG)*, pages 449–450, 2004. Video available at http://www.computational-geometry.org/SoCG-videos/socg04video/. doi:10.1145/997817.997885.

15 Sándor P. Fekete and Alexander Kröller. Geometry-based reasoning for a large sensor network. In *22th Annual ACM Symposium on Computational Geometry (SoCG)*, pages 475–476, 2006. Video available at http://www.computational-geometry.org/SoCG-videos/socg06video/. doi:10.1145/1137856.1137926.

16 Sándor P. Fekete, Alexander Kröller, L.S. Kyou, and J. McKurkin Christiane Schmidt. Triangulating unknown environments using robot swarms. In *29th Annual ACM Symposium on Computational Geometry (SoCG)*, pages 345–346, 2013. Video available at http://www.computational-geometry.org/SoCG-videos/socg13video/.

17 Sándor P. Fekete, Sven von Höveling, and Christian Scheffer. Online circle packing. In *International Algorithms and Data Structures Symposium (WADS)*, pages 366–369, 2019.

18 Michael Friendly. Milestones in the history of data visualization: A case study in statistical historiography. In *Classification—the Ubiquitous Challenge*, pages 34–52. Springer, 2005.

19 Charles D Hansen and Chris R Johnson. *Visualization handbook*. Elsevier, 2011.

20 Yuval Noah Harari. The tree of knowledge. In *Sapiens: A brief history of humankind*, chapter 2, pages 22–44. Random House, 2014.

21 Yuval Noah Harari. The storytellers. In *Homo Deus: A brief history of tomorrow*, chapter 4, pages 181–207. Random House, 2016.

22 Terence Tao. There's more to mathematics than rigour and proofs. *What's New blog*, 2009. URL: http://terrytao.wordpress.com/career-advice/theres-more-to-mathematics-than-rigour-and-proofs.

Covering Rectangles by Disks: The Video

Sándor P. Fekete
Department of Computer Science, TU Braunschweig, Germany
s.fekete@tu-bs.de

Phillip Keldenich
Department of Computer Science, TU Braunschweig, Germany
p.keldenich@tu-bs.de

Christian Scheffer
Department of Computer Science, TU Braunschweig, Germany
scheffer@ibr.cs.tu-bs.de

Abstract

In this video, we motivate and visualize a fundamental result for covering a rectangle by a set of non-uniform circles: For any $\lambda \geq 1$, the critical covering area $A^*(\lambda)$ is the minimum value for which any set of disks with total area at least $A^*(\lambda)$ can cover a rectangle of dimensions $\lambda \times 1$. We show that there is a threshold value $\lambda_2 = \sqrt{\sqrt{7}/2 - 1/4} \approx 1.035797\ldots$, such that for $\lambda < \lambda_2$ the critical covering area $A^*(\lambda)$ is $A^*(\lambda) = 3\pi\left(\frac{\lambda^2}{16} + \frac{5}{32} + \frac{9}{256\lambda^2}\right)$, and for $\lambda \geq \lambda_2$, the critical area is $A^*(\lambda) = \pi(\lambda^2 + 2)/4$; these values are tight. For the special case $\lambda = 1$, i.e., for covering a unit square, the critical covering area is $\frac{195\pi}{256} \approx 2.39301\ldots$. We describe the structure of the proof, and show animations of some of the main components.

2012 ACM Subject Classification Theory of computation → Packing and covering problems; Theory of computation → Computational geometry

Keywords and phrases Disk covering, critical density, covering coefficient, tight worst-case bound, interval arithmetic, approximation

Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.75

Category Media Exposition

Related Version This contribution visualizes the main result of paper [1], `https://doi.org/10.4230/LIPIcs.SoCG.2020.42`, which is part of SoCG 2020.

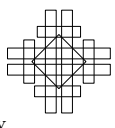
Supplementary Material `https://github.com/phillip-keldenich/circlecover`

Acknowledgements We thank Sebastian Morr, Utkarsh Gupta and Sahil Shah for joint related work.

1 Introduction

Given a collection of (not necessarily equal) disks, is it possible to arrange them so that they completely cover a given region, such as a square or a rectangle? Problems of this type have a variety of applications, but are notoriously difficult; see our related conference paper [1] for a more detailed overview.

In this contribution, we illustrate a fundamental result: If the total area of the disks is sufficiently large, they can always cover the region. More precisely, for any given λ, we identify the minimum value $A^*(\lambda)$ for which any collection of disks with total area at least $A^*(\lambda)$ can cover a rectangle of dimensions $\lambda \times 1$. We call $A^*(\lambda)$ the *critical covering area* for $\lambda \times 1$ rectangles and give a complete and tight characterization, along with a visual illustration of the involved proof techniques.



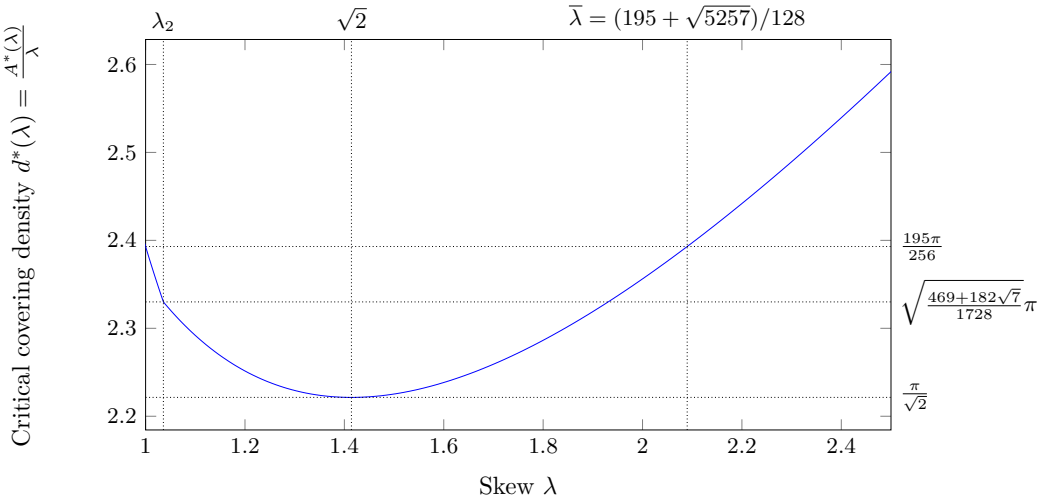


Figure 1 The critical covering density $d^*(\lambda)$ depending on λ and its values at the threshold value λ_2, the global minimum at $\sqrt{2}$ and the skew $\overline{\lambda}$ at which the density becomes as bad as for the square.

▶ **Theorem 1.** *Let $\lambda \geq 1$ and let $\mathcal{R}$ be a rectangle of dimensions $\lambda \times 1$. Let*

$$\lambda_2 = \sqrt{\frac{\sqrt{7}}{2} - \frac{1}{4}} \approx 1.035797\ldots, \text{ and } A^*(\lambda) = \begin{cases} 3\pi\left(\frac{\lambda^2}{16} + \frac{5}{32} + \frac{9}{256\lambda^2}\right), & \text{if } \lambda < \lambda_2, \\ \pi\frac{\lambda^2+2}{4}, & \text{otherwise.} \end{cases}$$

(1) *For any $a < A^*(\lambda)$, there is a set D^- of disks with $A(D^-) = a$ that cannot cover $\mathcal{R}$.*
(2) *Let $D = \{r_1, \ldots, r_n\} \subset \mathbb{R}$, $r_1 \geq r_2 \geq \ldots \geq r_n > 0$ be any collection of disks identified by their radii. If $A(D) \geq A^*(\lambda)$, then D can cover $\mathcal{R}$.*

See Figure 1 for a graph showing the (normalized) critical covering area, called critical covering density $d^*(\lambda) = {}^{A^*(\lambda)}/_{\lambda}$, and Figure 2 for examples of worst-case configurations. The point $\lambda = \lambda_2$ is the unique real number greater than 1 for which the two bounds $3\pi\left(\frac{\lambda^2}{16} + \frac{5}{32} + \frac{9}{256\lambda^2}\right)$ and $\pi\frac{\lambda^2+2}{4}$ coincide; see Figure 1. At this so-called *threshold value*, the worst case changes from three identical disks to two disks, which are the circumcircle $r_1^2 = \frac{\lambda^2+1}{4}$ and a disk $r_2^2 = \frac{1}{4}$; see Figure 2. The intuition behind the behavior of $d^*(\lambda)$ is as follows. The three-disk worst case is bad due to the fact that one of the three disks has to cover an entire edge of the rectangle. The efficiency of this placement improves when λ increases, because the size of the largest disk increases as well, while the length of the shorter edge remains constant. For the two-disk worst case, increasing λ initially improves the density, because the constant area contributed by the second disk becomes less significant. After this initial improvement, the quadratic growth of the largest disk compared to the linear growth of the rectangle dominates, leading to an overall linear increase in density.

2 High-level description

As shown in the video and illustrated in Figure 3, the proof consists of several components. In addition, there are a number of lemmas, which we describe first for easier reference.

2.1 Mathematical components

First is a lemma that describes the worst cases and shows tightness of our result.

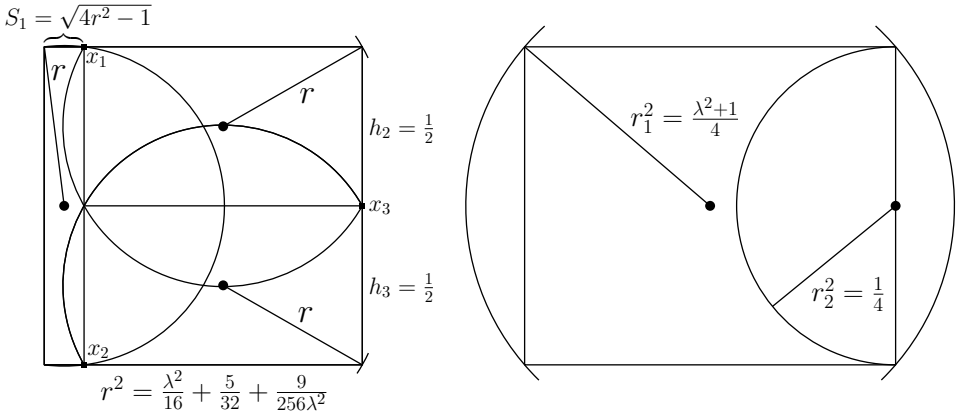


Figure 2 Worst-case configurations for small $\lambda \leq \lambda_2$ (left) and for large skew $\lambda \geq \lambda_2$ (right).

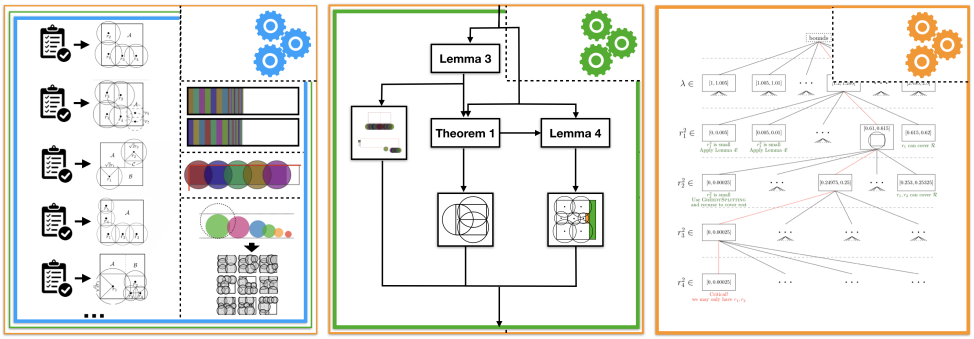


Figure 3 The different proof components. (Left) Individual covering routines. (Center) Recursive logic of the overall algorithmic approach. (Right) Case analysis for the computer-assisted proof.

▶ **Lemma 2.** *Let $\lambda \geq 1$ and let $\mathcal{R}$ be a rectangle of dimensions $\lambda \times 1$. (1) Two disks of radius $r_1 = \sqrt{\frac{\lambda^2+1}{4}}$ and $r_2 = \frac{1}{2}$ suffice to cover $\mathcal{R}$. (2) For any $\varepsilon > 0$, two disks of radius $r_1 - \varepsilon$ and r_2 do not suffice to cover $\mathcal{R}$. (3) Three identical disks of radius $r = \sqrt{\frac{\lambda^2}{16} + \frac{5}{32} + \frac{9}{256\lambda^2}}$ suffice to cover $\mathcal{R}$. (4) For $\lambda \leq \lambda_2$ and any $\varepsilon > 0$, three identical disks of radius $r_- := r - \varepsilon$ do not suffice to cover $\mathcal{R}$.*

For large λ, the *critical covering coefficient* $E^*(\lambda) := \frac{A^*(\lambda)}{\lambda\pi}$ of Theorem 1 becomes worse, as large disks cannot be used to cover the rectangle efficiently. If the *weight*, i.e., squared radius, of each disk is bounded by some $\sigma \geq r_1^2$, we provide the following lemma achieving a better covering coefficient $E(\sigma)$ for large λ.

▶ **Lemma 3.** *Let $\hat{\sigma} := \frac{195\sqrt{5257}}{16384} \approx 0.8629$. Let $\sigma \geq \hat{\sigma}$ and $E(\sigma) := \frac{1}{2}\sqrt{\sqrt{\sigma^2+1}+1}$. Let $\lambda \geq 1$ and $D = \{r_1, \ldots, r_n\}$ be any collection of disks with $\sigma \geq r_1^2 \geq \ldots \geq r_n^2$ and $W(D) := \sum_{i=1}^{n} r_i^2 \geq E(\sigma)\lambda$. Then D can cover a rectangle $\mathcal{R}$ of dimensions $\lambda \times 1$.*

The final component is the following Lemma 4, which also gives a better covering coefficient if the size of the largest disk is bounded. The bound required for Lemma 4 is smaller than for Lemma 3, yielding a better covering coefficient in return.

▶ **Lemma 4.** *Let $\lambda \geq 1$ and let $\mathcal{R}$ be a rectangle of dimensions $\lambda \times 1$. Let $D = \{r_1, \ldots, r_n\}$, $0.375 \geq r_1 \geq \ldots \geq r_n > 0$ be a collection of disks. If $W(D) \geq 0.61\lambda$, or equivalently $A(D) \geq 0.61\pi\lambda \approx 1.9164\lambda$, then D suffices to cover $\mathcal{R}$.*

2.2 Proof overview

The proofs of Theorem 1 and Lemmas 3 and 4 work by induction on the number of disks. For proving Lemma 3 for n disks, we use Theorem 1 for n disks. For proving Theorem 1 for n disks, we use Lemma 4 for n disks; Lemma 3 is only used for fewer than n disks. For proving Lemma 4 for n disks, we only use Theorem 1 and Lemma 3 for fewer than n disks. Therefore, there are no cyclic dependencies in our argument; however, we have to perform the induction for Theorem 1 and Lemmas 3 and 4 simultaneously.

The proofs of our result are constructive; they are based on an efficient recursive algorithm that uses a set of simple *routines*. These routines were derived by hand, in many cases based on problematic instances that were identified by the automatic prover and could not be handled by the routines that were already present. We go through the list of routines in some fixed order. For each routine, we check a sufficient criterion for the routine to work. We call these criteria *success criteria*. They only depend on the total available weight and a constant number of largest disks. If we cannot guarantee that a routine works by its success criterion, we simply disregard the routine; this means that our algorithm does not have to backtrack. We prove that, regardless of the distribution of the disks' weight, at least one success criterion is met, implying that we can always apply at least one routine. The number of routines and thus success criteria is large; this is where the need for automatic assistance comes from.

Typical routines are recursive; they consist of splitting the collection of disks into smaller parts, splitting the rectangle accordingly, and recursing, or recursing after fixing the position of a constant number of large disks. As a success criterion for recursion, we check whether Theorem 1 or Lemma 3 or 4 can be applied.

2.3 Interval arithmetic

We use interval arithmetic to prove that there always is a routine that works. In interval arithmetic, operations like addition, multiplication or taking a square root are performed on intervals $[a, b] \subset \mathbb{R}$ instead of numbers. After proving our result manually for large λ, this allows us to check a finite, discrete set of cases, instead of the continuum of all possible radii and λ. See our main paper [1] for details.

3 The video

The video starts with a motivation of the basic problem of covering a rectangle by disks, followed by a description of the main result. After an overview of the main three aspects of the proof (individual covering routines, recursive logic, case analysis), these are explained and illustrated in detail.

References

1 Sándor P. Fekete, Utkarsh Gupta, Phillip Keldenich, Christian Scheffer, and Sahil Shah. Worst-Case Optimal Covering of Rectangles by Disks. In *Proceedings 36th International Symposium on Computational Geometry (SoCG 2020)*, pages 42:1–42:19, 2020. doi:10.4230/LIPIcs.SoCG.2019.35.

Step-By-Step Straight Skeletons

Günther Eder
Universität Salzburg, FB Computerwissenschaften, Austria
geder@cs.sbg.ac.at

Martin Held
Universität Salzburg, FB Computerwissenschaften, Austria
held@cs.sbg.ac.at

Peter Palfrader
Universität Salzburg, FB Computerwissenschaften, Austria
palfrader@cs.sbg.ac.at

Abstract

We present two software packages for computing straight skeletons: Monos, our implementation of an algorithm by Biedl et al. (2015), computes the straight skeleton of a monotone input polygon, and Surfer2 implements a generalization of an algorithm by Aichholzer and Aurenhammer (1998) to handle multiplicatively-weighted planar straight-line graphs as input.

The graphical user interfaces that ship with our codes support step-by-step computations, where each event can be investigated and studied by the user. This makes them a canonical candidate for educational purposes and detailed event analyses. Both codes are freely available on GitHub.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases weighted straight skeleton, implementation, visualization, graphical user interface, education

Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.76

Category Media Exposition

Funding Work supported by Austrian Science Fund (FWF): Grants ORD 53-VO and P31013-N31.

1 Introduction

The straight skeleton of a polygon P is a skeletal structure similar to the medial axis. It consists of straight-line segments only and was introduced to computational geometry by Aichholzer et al. [2]. Consider a shrinking process of P, called the wavefront propagation, where all edges of P move inwards in a parallel manner at unit speed. The shrinking polygon, called the wavefront, will change as edges collapse to zero length and are removed (*edge event*), and vertices move into non-incident wavefront edges, thereby splitting the wavefront into more components (*split event*). The propagation ends when all wavefront components have collapsed. The straight skeleton is then defined as the set of line segments which cover the traces of vertices of the wavefront polygons during this process; cf. Figure 1.

Straight skeletons can be generalized by using planar straight-line graphs (PSLGs) as input instead of just polygons [1], or by weighting the input edges either multiplicatively or additively which causes their wavefront edges to move either faster [1, 5] or to start moving at different times [6].

Several algorithms exist to compute straight skeletons. Aichholzer and Aurenhammer [1] describe an algorithm with a worst-case complexity of $\mathcal{O}(n^3 \log n)$ for an n-vertex PSLG which, however, seems to run in near-linear time in practice [4, 7]. Currently, the algorithm with the best known worst-case complexity for general input is due to Eppstein and Erickson [5] and

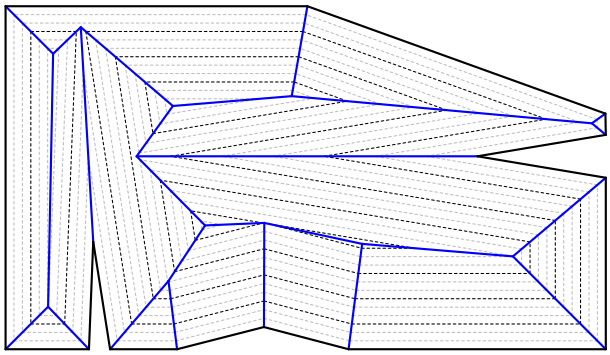


Figure 1 The straight skeleton (blue) of a polygon P (black) is given by the traces of the vertices of shrinking wavefront polygons (dashed) during the wavefront-propagation process.

runs in $\mathcal{O}(n^{17/11+\varepsilon})$ time and space for any positive ε. We are not aware of any implementation of their algorithm, though. Biedl et al. [3] present an $\mathcal{O}(n \log n)$ time algorithm to construct the straight skeleton of an n-vertex monotone polygon.

2 Contribution

We implemented the algorithms by Aichholzer and Aurenhammer [1] as well as Biedl et al. [3] and will report at SoCG 2020 on the engineering challenges of casting these algorithms into actual software [4]. Our code is freely available on GitHub: Surfer2 computes the weighted straight skeleton of planar straight-line graphs, and Monos is a special-purpose code to construct straight skeletons of monotone polygons[1].

In the course of implementing the algorithms, we also created graphical user interfaces (GUIs). These GUIs enable the user to "see" the algorithm in action. The interfaces are invaluable during development and debugging, but they also help to gain a deeper understanding of the various events handled during the wavefront propagation and, in the case of Monos, the merge step of the algorithm. As such they represent useful aids in teaching computational geometry classes. The concepts which they help to understand are not strictly limited to straight skeletons, but also include general event-based algorithms, kinetic data structures, and sweep-plane processes.

3 MonosGUI: Straight skeleton of a monotone polygon

Biedl et al. [3] describe an $\mathcal{O}(n \log n)$ time algorithm to compute the straight skeleton of a simple n-vertex monotone polygon $\mathcal{P}$. Their algorithm consists of two steps: (i) The polygon $\mathcal{P}$ is split into an upper and lower monotone chain, and the straight skeleton of each chain is computed individually. (ii) The final straight skeleton $\mathcal{S}(\mathcal{P})$ is obtained by merging these two straight skeletons.

We employ a classical wavefront propagation to obtain the straight skeleton of a monotone chain in $\mathcal{O}(n \log n)$ time [3]. In the second step, the skeletons of the upper and lower chains are merged to form $\mathcal{S}(\mathcal{P})$. This merge is based on a left-to-right traversal of the two chains and their respective straight-skeleton faces. Starting at the leftmost vertex of both chains, the angular bisector between the incident edges is constructed. It intersects arcs of both the

[1] Source available at `https://github.com/cgalab/surfer2` and `https://github.com/cgalab/monos`.

top and bottom straight skeleton. We stop at the first intersection reached and modify the respective straight skeleton locally by creating a node. Then the next bisector is constructed between the two edges that induce the faces of the upper and lower skeleton we are currently in. This process is repeated until the rightmost vertex of $\mathcal{P}$ is reached, thus obtaining the final skeleton $\mathcal{S}(\mathcal{P})$.

MonosGUI, the graphical user interface that comes with Monos, allows to step through both the skeleton computation of the chains and the merge process; cf. Figure 2. Pressing n, MonosGUI steps to the next event. We can skip the chain computation with c, then MonosGUI stops before the merge process starts. Simply pressing ⏎ computes the given input and displays its straight skeleton.

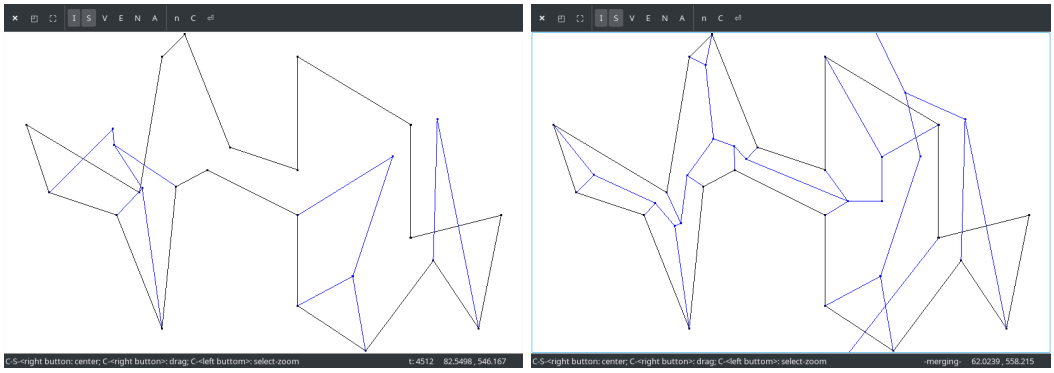


Figure 2 MonosGUI: A step-by-step construction of the straight skeleton of a single chain (left); The construction of the merge chain between the finished upper and lower chain's skeleton (right).

4 SurfGUI: Straight Skeleton of a PSLG

The algorithm by Aichholzer and Aurenhammer [1] constructs the straight skeleton by simulating the wavefront propagation. As the wavefront sweeps the plane, a kinetic triangulation of that part of the plane which has not yet been swept is maintained.

This triangulation is a kinetic data structure where every change in the structure of the wavefront, i.e., every edge event and every split event, is witnessed by a triangle collapse. However, not all triangle collapses witness a corresponding event of the wavefront. Some triangle collapses correspond to internal events only, where the triangulation changes as a wavefront vertex moves over a triangulation diagonal (*flip event*).

Our implementation of this algorithm, Surfer2, comes with a graphical user interface, SurfGUI, which shows the wavefront propagation step by step. By default, SurfGUI shows the input planar straight-line graph, the current kinetic triangulation and all straight skeleton arcs traced out so far. The next triangle to witness an event is highlighted.

The user can interactively control time, moving forwards or backwards. When moving forwards in time, event handling can optionally be skipped, which will result in the display of an invalid configuration after the missed event (time will reset to the first missed event once event-handling is re-enabled). Moving back will not undo events already processed. Shortcuts are provided to jump forward to (but not yet process) the next event or to jump backwards to the event just handled. See Figure 3 for some screenshots.

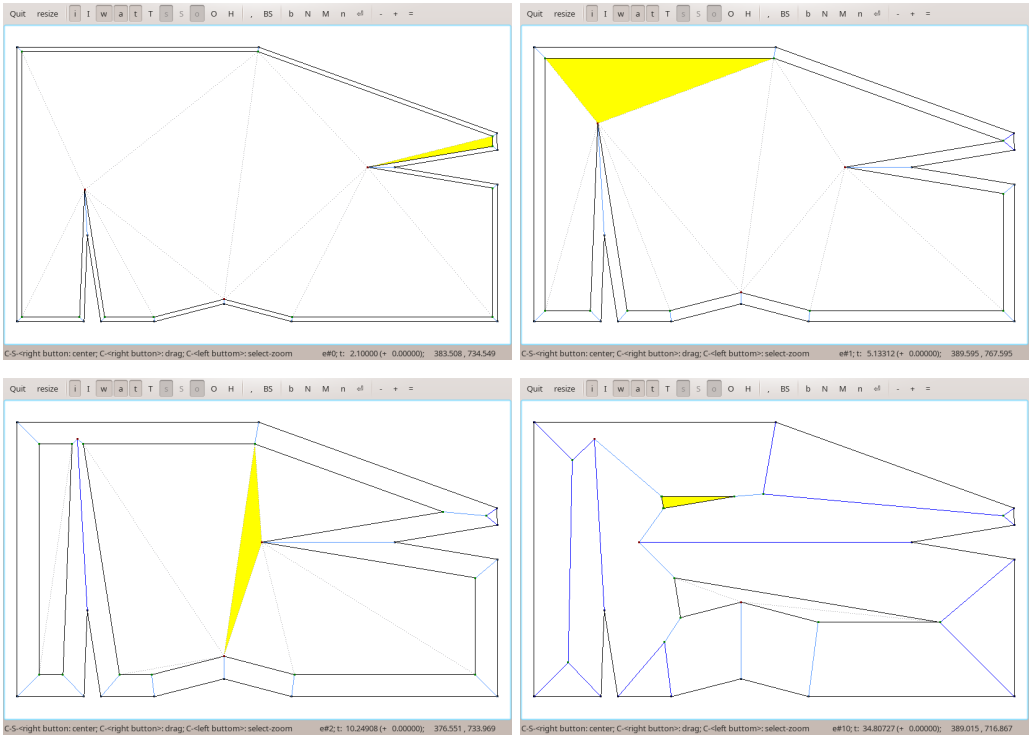


Figure 3 SurfGUI processing the input from Figure 1. The triangle witnessing the next event is always highlighted (in yellow). In reading order, the highlighted triangle witnesses an edge event, a split event, a flip event, and the collapse of an entire wavefront component.

References

1 Oswin Aichholzer and Franz Aurenhammer. Straight Skeletons for General Polygonal Figures in the Plane. In *Voronoi's Impact on Modern Sciences II*, volume 21, pages 7–21. Institute of Mathematics of the National Academy of Sciences of Ukraine, 1998. doi:10.1007/3-540-61332-3_144.

2 Oswin Aichholzer, Franz Aurenhammer, David Alberts, and Bernd Gärtner. A Novel Type of Skeleton for Polygons. *Journal of Universal Computer Science*, 1(12):752–761, 1995. doi:10.1007/978-3-642-80350-5_65.

3 Therese Biedl, Martin Held, Stefan Huber, Dominik Kaaser, and Peter Palfrader. A Simple Algorithm for Computing Positively Weighted Straight Skeletons of Monotone Polygons. *Information Processing Letters*, 115(2):243–247, 2015. doi:10.1016/j.ipl.2014.09.021.

4 Günther Eder, Martin Held, and Peter Palfrader. On Implementing Straight Skeletons: Challenges and Experiences. In *36th International Symposium on Computational Geometry, SoCG 2020*, volume 164 of *LIPIcs*, pages 38:1–38:16, Zürich, Switzerland, 2020.

5 David Eppstein and Jeff Erickson. Raising Roofs, Crashing Cycles, and Playing Pool: Applications of a Data Structure for Finding Pairwise Interactions. *Discrete & Computational Geometry*, 22(4):569–592, 1999. doi:10.1145/276884.276891.

6 Martin Held and Peter Palfrader. Straight Skeletons with Additive and Multiplicative Weights and Their Application to the Algorithmic Generation of Roofs and Terrains. *Computer-Aided Design*, 92(1):33–41, 2017. doi:10.1016/j.cad.2017.07.003.

7 Peter Palfrader, Martin Held, and Stefan Huber. On Computing Straight Skeletons by Means of Kinetic Triangulations. In *Proceedings of the 20th Annual European Symposium on Algorithms (ESA)*, pages 766–777, 2012. doi:10.1007/978-3-642-33090-2_66.

Computing Animations of Linkages with Rotational Symmetry

Sean Dewar
Johann Radon Institute for Computational and Applied Mathematics (RICAM), Austrian Academy of Sciences, Linz, Austria
sean.dewar@ricam.oeaw.ac.at

Georg Grasegger
Johann Radon Institute for Computational and Applied Mathematics (RICAM), Austrian Academy of Sciences, Linz, Austria
georg.grasegger@ricam.oeaw.ac.at

Jan Legerský
Johannes Kepler University Linz, Research Institute for Symbolic Computation (RISC), Linz, Austria
Department of Applied Mathematics, Faculty of Information Technology, Czech Technical University in Prague, Prague, Czech Republic
jan.legersky@risc.jku.at

Abstract

We present a piece of software for computing animations of linkages with rotational symmetry in the plane. We construct these linkages from an algorithm that utilises a special type of edge colouring to embed graphs with rotational symmetry.

2012 ACM Subject Classification Mathematics of computing → Graph algorithms

Keywords and phrases Flexibility, Linkages, Symmetry

Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.77

Category Media Exposition

Related Version A proof of the theorem can be found in [1], `https://arxiv.org/abs/2003.09328`.

Supplementary Material An implementation can be found in [3, 4], `https://doi.org/10.5281/zenodo.3719345`.

Funding *Sean Dewar*: Supported by the Austrian Science Fund (FWF): P31888.
Georg Grasegger: Supported by the Austrian Science Fund (FWF): P31888 and W1214-N15.
Jan Legerský: Supported by the Austrian Science Fund (FWF): P31061 and by the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 675789.

1 Introduction

A *framework* is a pair (G, p) where G is a (finite simple) graph and $p : V(G) \to \mathbb{R}^2$ – the *placement* of G – is a map where $p(u) \neq p(v)$ if uv is an edge. A framework is a *linkage* if there exists a continuous motion of its placed vertices that preserves the distances between each pair of vertices that share an edge, and the motion is not a rigid body motion of the framework; if such a motion does not exist then the framework is *rigid*. It was shown in [6] that a framework (G, p) with a *generic* placement of vertices (i.e. the set of coordinates of p is an algebraically independent set over the rational numbers) is rigid if and only if G contains a Laman graph as a spanning subgraph. This does not inform us whether we can construct a linkage from a graph; for example, any generic placement of the complete bipartite graph $K_{4,4}$ is rigid, however we can construct linkages from $K_{4,4}$ [2].

To determine whether a graph can be the graph of a linkage we introduce a special class of edge colourings. A red-blue colouring of the edges of a graph is a *NAC-colouring* if each colour is used at least once and each cycle is either monochromatic or contains at least two red and two blue edges (NAC comes from *No Almost Cycle*, which are cycles in which all edges but one have the same color). It was proven by [5, Theorem 3.1] that a connected graph with at least one edge is the graph of a linkage in the plane if and only if it has a NAC-colouring. The proof is done in two very distinct parts; the first part proves via valuation theory that any linkage induces a NAC-colouring, while the second gives an algorithmic method to construct a linkage from any given NAC-colouring. While the construction given for each NAC-colouring does give a linkage, the linkage will often not share any of the symmetries of the graph it was formed from (see for example Figure 1). A natural question now arises; can we adapt the result so as to preserve any chosen symmetry of our graph?

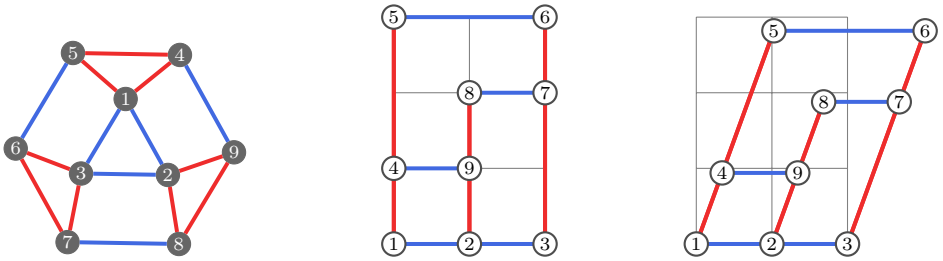


Figure 1 A graph (left) with a linkage constructed from the ilustrated NAC-coloring (middle). The motion of the linkage (right) is parametrised by the angle of the currently vertical lines to the fixed bottom horizontal line.

2 Linkages with rotational symmetry

We define a graph to have *n-fold rotational symmetry* if the group $\mathcal{C}_n := \langle \omega : \omega^n = 1 \rangle$ acts freely on the graph, i.e., each vertex has an orbit of exactly n elements. We define a placement p of G to be *n-fold rotational symmetric* if the placement of a rotated vertex is the rotation of the placement of the vertex, i. e. $p(\omega^k v) = \tau(\omega^k)p(v)$ for each vertex v and rotation $\omega^k \in \mathcal{C}_n$, where $\tau(\omega^k)$ is the rotation matrix for angle $2\pi k/n$.

If this holds then we define (G, p) to be a *n-fold rotational symmetric framework* with *n-fold rotational symmetric placement* p. We further define (G, p) to be a *n-fold rotational symmetric linkage* if there is a continuous edge-length preserving motion of (G, p) that maintains the rotational symmetry but is not a rotation of the framework. Remembering this, we can define the correct type of NAC-colouring to take into account the graph's rotational symmetry.

▶ **Definition 1.** *Let G be an n-fold rotational symmetric graph with NAC-colouring δ. We define δ to be an* n-fold rotational symmetric NAC-colouring *if the colouring respects the symmetry of the graph, and no two distinct blue, resp. red,* partially invariant *connected components are connected by an edge; a set of vertices U is partially invariant if there exists $\gamma \in \mathcal{C}_n \setminus \{1\}$ such that $\gamma U = U$.*

The ideas of n-fold rotational symmetric linkages and n-fold rotational symmetric NAC-colourings tie together nicely similarly to how linkages and NAC-colourings do.

▶ **Theorem 2.** *An n-fold rotational symmetric connected graph with at least one edge is the graph of an n-fold rotational symmetric linkage in the plane if and only if it has an n-fold rotational symmetric NAC-colouring δ.*

The full proof of Theorem 2 comes in two parts and can be found in [1]. We detail below the construction part that builds an n-fold rotational symmetric linkage from an n-fold rotational symmetric graph G with n-fold rotational symmetric NAC-colouring δ:

(1) We first need to label our red and blue connected components in a way that respects the symmetry; we will not, however, need to bother doing this for the partially invariant components as we will see in Step 3. We label the not partially invariant red components as $R_1^0, \dots, R_1^{n-1}, \dots, R_m^0, \dots, R_m^{n-1}$, where $R_j^i = \omega^i R_j^0$ for $0 \leq i < n$ and $1 \leq j \leq m$; similarly, we label the not partially invariant blue components as $B_1^0, \dots, B_1^{n-1}, \dots, B_k^0, \dots, B_k^{n-1}$.

(2) Next, we need to choose base vectors for each of the red and blue components that will determine the shape of the framework. The choice is actually (almost) arbitrary, which fortunately will allow us to pick "nice" vectors. Let $a_1, \dots, a_m$ and $b_1, \dots, b_k$ be our choice of points in the plane with the assumption that $a_j \neq \tau(\omega^i)a_{j'}$ and $b_j \neq \tau(\omega^i)b_{j'}$ for $j \neq j'$ and $1 \leq i < n$. This assumption is necessary to avoid overlapping vertices.

(3) Using our choices of a_i's and b_j's from Step 2, we now create a "coordinate system" in which vertices are placed depending on the red and blue component they belong to. To do this, we define the functions $\overline{a}, \overline{b} \colon V(G) \to \mathbb{R}^2$ by

$$\overline{a}(v) = \begin{cases} \tau(\omega^i)a_j & \text{if } v \in R_j^i \\ (0,0) & \text{otherwise,} \end{cases} \quad \text{and} \quad \overline{b}(v) = \begin{cases} \tau(\omega^i)b_j & \text{if } v \in B_j^i \\ (0,0) & \text{otherwise.} \end{cases}$$

We note that a vertex is mapped to the origin by $\overline{a}$ (respectively, $\overline{b}$) if and only if it lies in a red (respectively blue) partially invariant component.

(4) Finally, by using our "coordinate system" determined by $\overline{a}, \overline{b}$ we define for each $t \in [0, 2\pi]$ an n-fold rotational symmetric placement p_t of G, where for each $v \in V(G)$ we have

$$p_t(v) := \begin{pmatrix} \cos t & -\sin t \\ \sin t & \cos t \end{pmatrix} \overline{a}(v) + \overline{b}(v)\,.$$

This yields indeed an n-fold rotational symmetric linkage with the corresponding motion given by $t \mapsto (G, p_t)$. Further, we can change the linkage by choosing different a_i's and b_j's. This often allows us to construct "better" linkages by choosing new base vectors; whether "better" entails maintaining some other unspecified symmetry (for example, reflectional), or just a linkage that is more aesthetically pleasing.

▶ **Example 3.** By using our construction we can obtain the n-fold rotational symmetric linkages given in Figures 2 and 3.

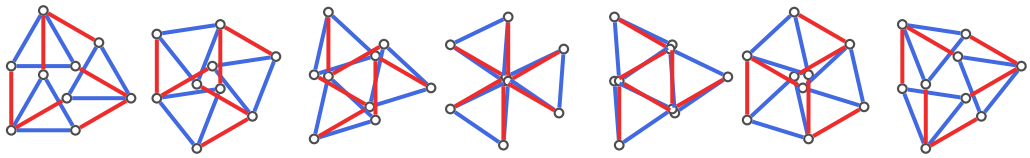


Figure 2 A 3-fold rotational symmetric linkage constructed from a given 3-fold rotational symmetric NAC-colouring.

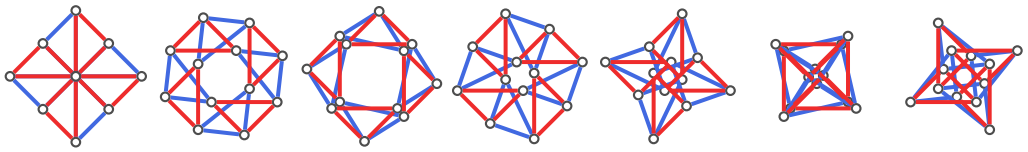


Figure 3 A 4-fold rotational symmetric linkage constructed from a given 4-fold rotational symmetric NAC-colouring.

3 Software for Animations

Animations can be created by an implementation of the above described algorithm using canonical choices for the base vectors. An updated version [4] of the package [3] can be used to study $\mathcal{C}_n$-symmetric frameworks. We encourage the reader to experiment[1] with the choice of a_i's and b_j's from Theorem 2. The implementation can also be used to find the symmetric NAC-colouring. However, in both graphs of Figure 4 we chose a simple NAC-colouring where two triangle subgraphs intersecting in a single vertex are coloured differently. Basic animations can be created with the software packages. The provided animation was constructed using graphical post-processing for the coordinate output.

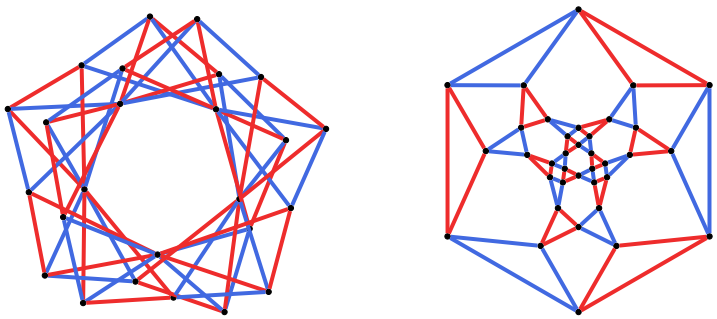


Figure 4 Two graphs with their symmetric NAC-colourings used for the animations. To illustrate the difficulty of finding the symmetric motion, we only show some arbitrary graph layouts. In fact, the animations will never reach either of these layouts.

References

1 S. Dewar, G. Grasegger, and J. Legerský. Flexible placements of graphs with rotational symmetry. Technical report, arXiv, 2020. `arXiv:2003.09328`.

2 A. C. Dixon. On certain deformable frameworks. *Messenger*, 29(2):1–21, 1899.

3 G. Grasegger and J. Legerský. FlexRiLoG – SageMath package for Flexible and Rigid Labelings of Graphs. Zenodo, March 2020. `doi:10.5281/zenodo.3719345`.

4 G. Grasegger and J. Legerský. FlexRiLoG – SageMath package for Flexible and Rigid Labelings of Graphs, repository, 2020. URL: `https://github.com/Legersky/flexrilog/`.

5 G. Grasegger, J. Legerský, and J. Schicho. Graphs with Flexible Labelings. *Discrete & Computational Geometry*, 62(2):461–480, 2019. `doi:10.1007/s00454-018-0026-9`.

6 H. Pollaczek-Geiringer. Über die Gliederung ebener Fachwerke. *Zeitschrift für Angewandte Mathematik und Mechanik (ZAMM)*, 7:58–72, 1927. `doi:10.1002/zamm.19270070107`.

[1] See file `examples/Rotationally_symmetric_frameworks_SoCGmedia.ipynb` of [4] or try it online: `https://jan.legersky.cz/SoCGmedia2020` redirecting to a Binder version of the notebook.

Hiding Sliding Cubes: Why Reconfiguring Modular Robots Is Not Easy

Tillmann Miltzow
Utrecht University, The Netherlands
t.miltzow@uu.nl

Irene Parada
TU Eindhoven, The Netherlands
i.m.de.parada.munoz@tue.nl

Willem Sonke
TU Eindhoven, The Netherlands
w.m.sonke@tue.nl

Bettina Speckmann
TU Eindhoven, The Netherlands
b.speckmann@tue.nl

Jules Wulms
TU Eindhoven, The Netherlands
j.j.h.m.wulms@tue.nl

Abstract

Face-connected configurations of cubes are a common model for modular robots in three dimensions. In this abstract and the accompanying video we study reconfigurations of such modular robots using so-called sliding moves. Using sliding moves, it is always possible to reconfigure one face-connected configuration of n cubes into any other, while keeping the robot connected at all stages of the reconfiguration. For certain configurations $\Omega(n^2)$ sliding moves are necessary. In contrast, the best current upper bound is $O(n^3)$. It has been conjectured that there is always a cube on the outside of any face-connected configuration of cubes which can be moved without breaking connectivity. The existence of such a cube would immediately imply a straight-forward $O(n^2)$ reconfiguration algorithm. However, we present a configuration of cubes such that no cube on the outside can move without breaking connectivity. In other words, we show that this particular avenue towards an $O(n^2)$ reconfiguration algorithm for face-connected cubes is blocked.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Sliding cubes, Reconfiguration, Modular robots

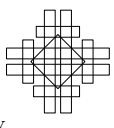
Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.78

Category Media Exposition

Supplementary Material The code used, along with a list of coordinates of the cubes in the construction, can be found at `https://github.com/tue-aga/cubes-checker`.

Funding This work was initiated at the 5th Workshop on Applied Geometric Algorithms (AGA 2020).
Tillmann Miltzow: Supported by the Netherlands Organisation for Scientific Research (NWO) under project no. 016.Veni.192.250.
Bettina Speckmann: Partially supported by the Netherlands Organisation for Scientific Research (NWO) under project no. 639.023.208.
Jules Wulms: Supported by the Netherlands eScience Center (NLeSC) under project no. 027.015.G02.

Acknowledgements We thank Tim Ophelders for his help with the computational verification of the properties of our configuration.



1 Introduction

Modular robots consist of several identical modules that can assemble themselves into various configurations. Often such modules live on a lattice and can move only relative to each other. One of the main algorithmic challenges in this context is to find efficient ways to reconfigure one configuration into any other. This topic has attracted significant attention in the computational geometry community [1, 2, 3, 4, 5, 7]; yet many fundamental questions remain open.

One frequently studied paradigm of modular robots is the so-called *sliding cube model*. Here, a *configuration* C is a face-connected set of cubes on the cubic grid. The model allows two types of moves: *slides* and *convex transitions* (see Figure 1). A cube c is *movable*, that is, c is allowed to perform a move, if and only if removing c from the configuration C leaves $C \setminus c$ face-connected.

Figure 1 The two types of moves in the sliding cube model.

To reconfigure a vertical pillar of n cubes into a horizontal line, $\Omega(n^2)$ moves are necessary, since that is the sum of the n (grid) distances to any horizontal plane through one of the cubes. Furthermore, Abel and Kominers [1] showed that $O(n^3)$ slides and convex transitions suffice to transform any configuration into a strip. Since both moves are reversible, this implies that any two configurations of n cubes can be reconfigured into each other with $O(n^3)$ moves. It has been conjectured that there is a simple way to close this gap, using so-called outer cubes. Specifically, we say that the *outside* of a configuration C is the unbounded maximal set of face-connected grid cells not in C. A cube c in a configuration C is an *outer cube* of C if it is face-connected to the outside. Furthermore, a *void* is a maximal face-connected set of grid cells neither in C, nor on the outside. If any configuration of cubes contains at least two movable outer cubes, then it is straight-forward to construct a horizontal line of cubes using $O(n^2)$ moves. In particular, we can iteratively construct the horizontal line to the right of a rightmost cube in the original configuration. If we are not done, we can find one movable outer cube c not yet part of the horizontal line. Then c can slide to the right of the horizontal line in $O(n)$ moves (we can actually find the shortest route by DFS). If the configuration does not contain voids, then there are indeed always two movable outer cubes. This fact about outer cubes has been used as the basis of reconfiguration algorithms [6, 8]. However, in this paper and in the associated video we show that there are configurations of cubes where no outer cube is movable.

2 The configuration

Consider the face-adjacency graph of the cubes. This graph has a spanning tree where all leaves correspond to cubes on the boundary of the configuration (either face-adjacent to the outside or to a void) [7]. Any tree with at least two vertices has at least two leaves. Hence, every configuration with more than one cube contains at least two movable cubes. It follows that the face-adjacency graph of configurations with the fewest movable cubes is a path. We

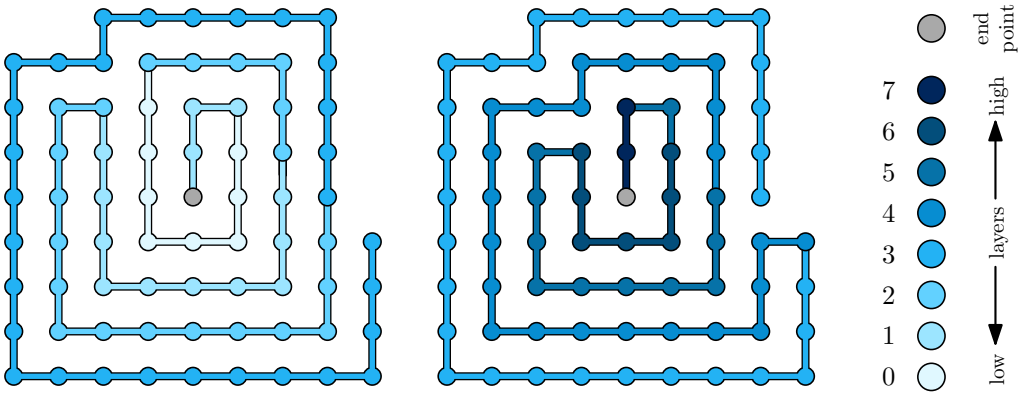


Figure 2 Adjacency graph of a basket, in top-down view: bottom *(left)* and lid *(right)*. The colors correspond to layers. Some positions contain two or three cubes stacked on top of each other. Note that layer 3 is drawn in both halves.

are hence basing our construction on such configurations. Intuitively, we are building two rope baskets out of the path of cubes. In each basket we are creating a void, in which we are hiding one endpoint of the path. The other endpoint becomes the top of the lid of the basket. Finally, we connect the two lids via a path.

We build each basket layer by layer (see Figure 2 (left) and Figure 3 (left)). This is a delicate construction, as we have to enclose a void while not creating cycles in the face-adjacency graph. Starting at an endpoint e of the path, on Layer 2, we first loop around e on Layers 1 and 0 in such a way that we trap e locally on the inside. The remainder of the construction closes this void, so that e is actually hidden inside it. To do so we build a lid for the basket, as shown in Figure 2 (right) and Figure 3 (right), ending with an endpoint e' on the top of the lid. All that remains now is to connect the top cubes of each basket via a path of cubes that does not introduce cycles (see Figure 4).

We computationally verified the correctness of our basket construction. The code used, along with a list of coordinates of the cubes in the construction, can be found at `https://github.com/tue-aga/cubes-checker`.

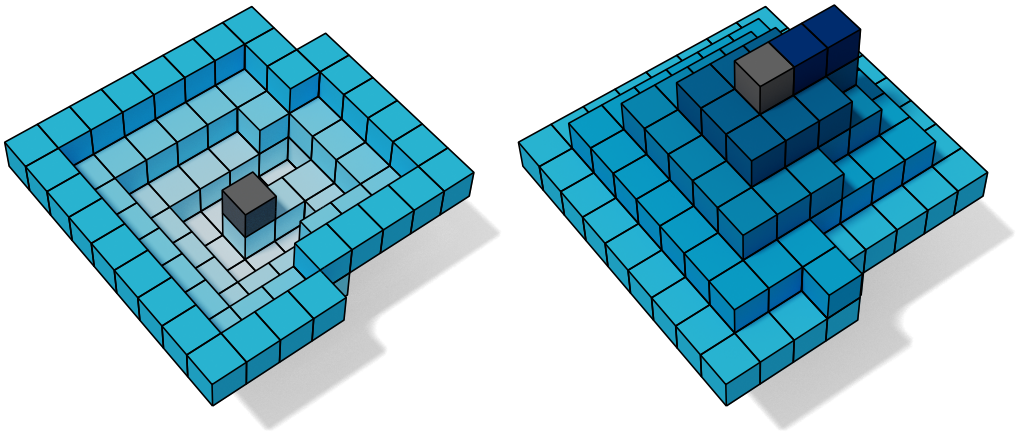


Figure 3 3D rendering of a basket: bottom only *(left)*, and entire basket *(right)*.

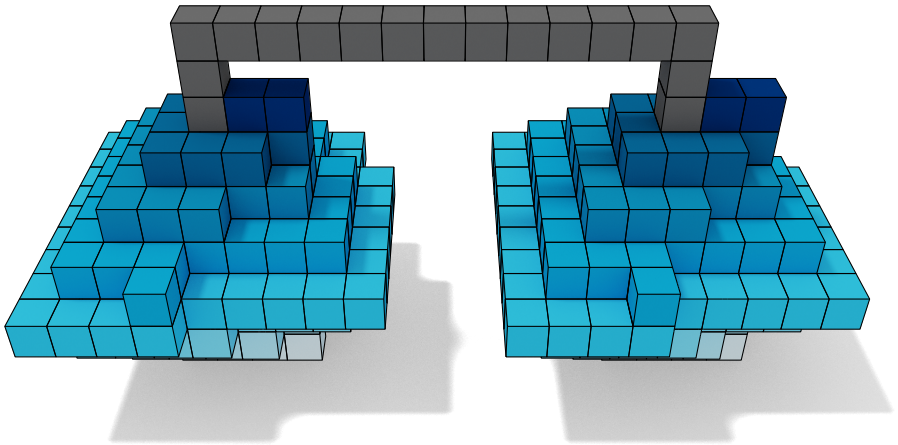


Figure 4 3D rendering of the complete construction.

3 The video

Our construction is inherently three dimensional and two-dimensional illustrations such as Figures 2 and 3 can convey only a schematic view of one of the baskets. To make our result more accessible and to stimulate interest in this fascinating class of modular robots, we hence produced a video using 3D animation in Blender. Our video first introduces modular robots and the sliding cube model, and then proceeds to build our configuration step-by-step.

4 Future work

The sliding cubes model extends to dimensions higher than three. Hence, concerning our specific construction, one can wonder if a similar example can be constructed in four or higher dimensions. Furthermore, we need two voids to trap both endpoints of our path. Is there an example that uses only one void? An obvious open question on the algorithmic side is the existence of an algorithm that reconfigures two configurations of n cubes in $O(n^2)$ time. Furthermore, the question arises if there are input sensitive algorithms to reconfigure cubes. That is, given two configurations C_1 and C_2, can we find the minimum number k of moves to reconfigure C_1 into C_2?

References

1 Zachary Abel and Scott Duke Kominers. Universal reconfiguration of (hyper-)cubic robots. *ArXiv e-Prints*, 2011. arXiv:0802.3414v3.

2 Hugo A. Akitaya, Esther M. Arkin, Mirela Damian, Erik D. Demaine, Vida Dujmovic, Robin Flatland, Matias Korman, Belen Palop, Irene Parada, André van Renssen, and Vera Sacristán. Universal reconfiguration of facet-connected modular robots by pivots: The $O(1)$ Musketeers. In *27th Annual European Symposium on Algorithms (ESA 2019)*, volume 144 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 3:1–3:14, 2019. doi:10.4230/LIPIcs.ESA.2019.3.

3 Greg Aloupis, Sébastien Collette, Mirela Damian, Erik D. Demaine, Robin Flatland, Stefan Langerman, Joseph O'Rourke, Val Pinciu, Suneeta Ramaswami, Vera Sacristán, and Stefanie Wuhrer. Efficient constant-velocity reconfiguration of crystalline robots. *Robotica*, 29(1):59–71, 2011. doi:10.1017/S026357471000072X.

4 Greg Aloupis, Sébastien Collette, Mirela Damian, Erik D. Demaine, Robin Flatland, Stefan Langerman, Joseph O'Rourke, Suneeta Ramaswami, Vera Sacristán, and Stefanie Wuhrer. Linear reconfiguration of cube-style modular robots. *Computational Geometry*, 42(6):652–663, 2009. doi:10.1016/j.comgeo.2008.11.003.

5 Adrian Dumitrescu and János Pach. Pushing squares around. *Graphs and Combinatorics*, 22:37–50, 2006. doi:10.1007/s00373-005-0640-1.

6 Robert Fitch, Zack Butler, and Daniela Rus. Reconfiguration planning for heterogeneous self-reconfiguring robots. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and System (IROS 2003)*, volume 3, pages 2460–2467, 2003. doi:10.1109/IROS.2003.1249239.

7 Ferran Hurtado, Enrique Molina, Suneeta Ramaswami, and Vera Sacristán. Distributed reconfiguration of 2D lattice-based modular robotic systems. *Autonomous Robots*, 38:383–413, 2015. doi:10.1007/s10514-015-9421-8.

8 Daniela Rus and Marsette Vona. Crystalline robots: self-reconfiguration with compressible unit modules. *Autonomous Robots*, 10:107–124, 2001. doi:10.1023/A:1026504804984.

Dots & Polygons

Kevin Buchin
Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands
k.a.buchin@tue.nl

Mart Hagedoorn
Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands
m.h.hagedoorn@student.tue.nl

Irina Kostitsyna
Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands
i.kostitsyna@tue.nl

Max van Mulken
Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands
m.j.m.v.mulken@student.tue.nl

Jolan Rensen
Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands
j.j.r.rensen@student.tue.nl

Leo van Schooten
Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands
l.g.t.v.schooten@student.tue.nl

Abstract

We present a new game, *Dots & Polygons*, played on a planar point set. We prove that its NP-hard and discuss strategies for the case when the point set is in convex position.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Dots & Boxes, NP-hard, game, cycle packing

Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.79

Category Media Exposition

Related Version A full version of this paper is available at `https://arxiv.org/abs/2004.01235`.

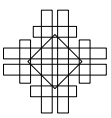
Supplementary Material
game: `https://www.win.tue.nl/~kbuchin/proj/ruler/dotsandpolygons/`,
source code: `https://github.com/kbuchin/ruler/tree/dots-and-polygons`

Acknowledgements We would like to thank David Eppstein for discussing [9] with us.

1 Introduction

Dots & Boxes [4] is a popular game, in which two players take turns in connecting nodes lying on the integer lattice, scoring when they surround unit squares. We introduce a more geometric variant of this game: *Dots & Polygons*.

The game is played on a planar point set P of size n. Two players, $\mathcal{B}$ (blue) and $\mathcal{R}$ (red), take turns, connecting two points $p, q \in P$ by a straight-line edge in a turn. The edge may not intersect other points or edges, and may not lie in a previously scored area. When a player closes a polygon, this player scores its area and makes another move. At the end, the player with the larger total area wins. We distinguish two variants. In *Dots & Polygons & Holes*, when a player closes a cycle, the player scores the enclosed area (excluding possibly



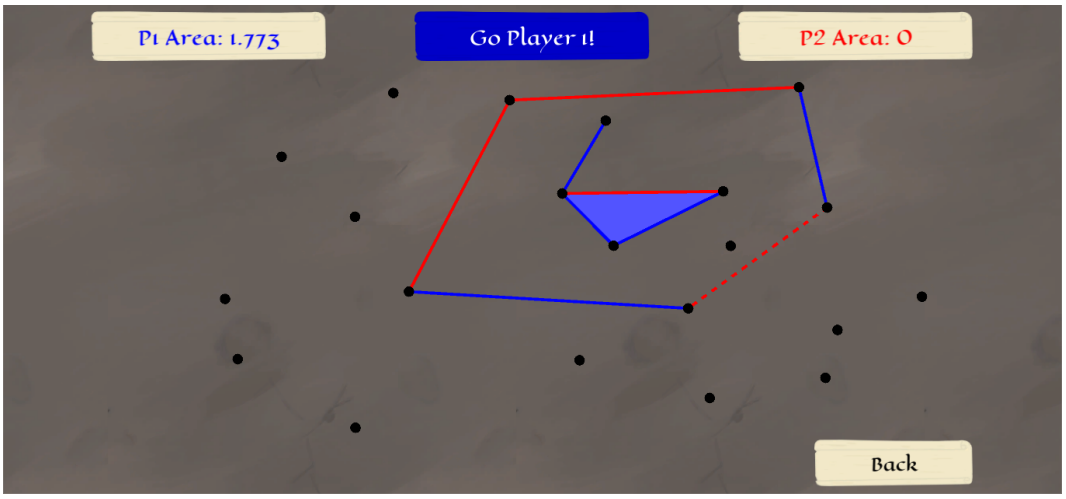


Figure 1 A screenshot of the *Dots & Polygons* game. In the *Dots & Polygons & Holes* version, if $\mathcal{R}$ draws the dotted edge, $\mathcal{R}$ will score the interior minus the blue triangle. In *Dots & Simple Polygons*, $\mathcal{R}$ will not score in this way.

previously enclosed parts). In *Dots & Simple Polygons*, a player only scores, when they close a simple polygon with no points inside. Figure 1 illustrates the difference between the variants.

A similar game is *Monochromatic Complete Triangulation Game* [1], but in that game only triangles are scored, and the score is the number of triangles. There is another variant of *Dots & Boxes* also called Dots & Polygons [12] that is played on the integer lattice.

Dots & Polygons is implemented on top of the *Ruler of the Plane* framework [2]. Both variants of the game can be played online (see supplementary materials). The ruler of the plane framework can be used to demonstrate different interesting geometric concepts and their applications. For example, to show a dynamic representation of the trapezoidal decomposition the user can press the `T` key while a game is active. The framework is extended with an implementation of a Doubly-Connected Edge List (DCEL) [3], a trapezoidal decomposition [3] and the Graham Scan algorithm [11].

Contributions. In Section 2 we show that *Dots & Simple Polygons* is NP-hard. We do so by a reduction from vertex-disjoint cycle packing in cubic planar graphs, including a self-contained reduction from planar 3-Satisfiability to this cycle-packing problem, and from the cycle-packing problem to *Dots & Boxes*. In Section 3 we discuss a greedy strategy for the case that P is in convex position.

2 Hardness

We show that *Dots & Simple Polygons* is NP-hard by a reduction from the maximum cycle packing problem in planar cubic graphs. The reduction is similar to the proof of NP-hardness of *Dots & Boxes*. The book *Winning Ways for your Mathematical Plays* [5] mentions that a generalization of *Dots & Boxes* can be shown to be NP-hard by a reduction from the maximum vertex-disjoint cycle packing (VCP) problem. The VCP problem can be viewed as a generalization of the triangle packing problem [6], which is known to be NP-hard [10].

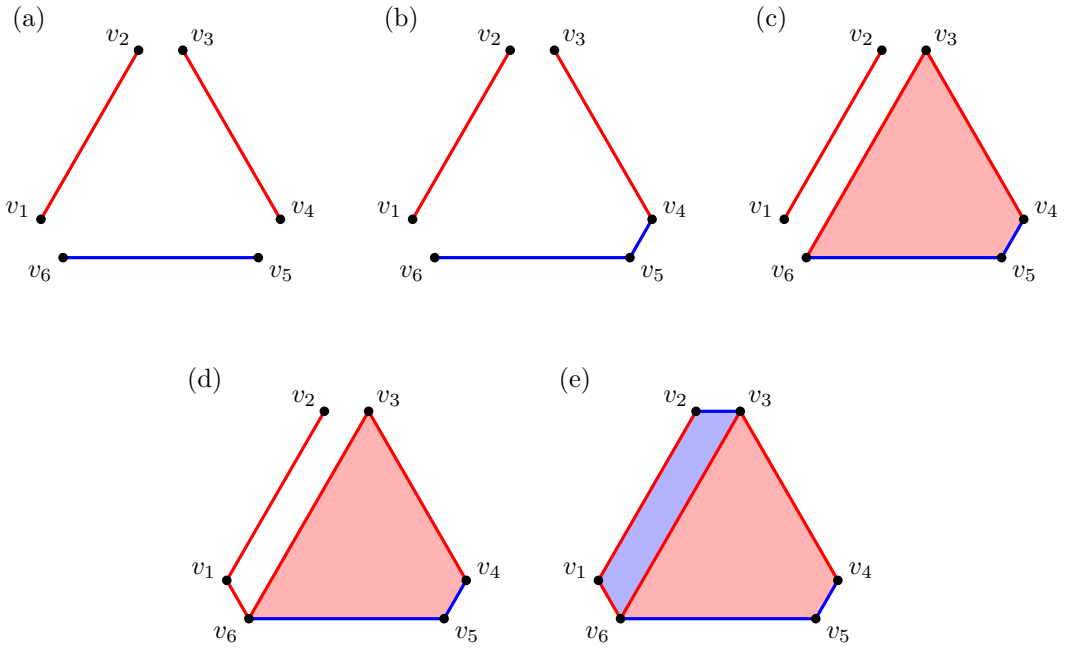


Figure 2 Player $\mathcal{R}$ could have won this game, but after reaching the state in (a) loses as shown in (b–e).

Eppstein notes that the NP-hardness, mentioned in [5], should apply to the classic *Dots & Boxes* by a reduction from the VCP problem in planar cubic graphs [9]. However, he does not cite a source of the hardness proof for this VCP variant. Furthermore, triangle packing is polynomial-time solvable in planar graphs with maximum degree three [8], and thus can no longer be used to justify the hardness of the VCP in planar cubic graphs. Thus, for the sake of completeness, we also show Theorem 1 and Theorem 2, which are used to prove Theorem 3. The full proofs for these theorems are given in [7].

▶ **Theorem 1.** *Maximal vertex-disjoint cycle packing in planar cubic graphs is NP-complete.*

▶ **Theorem 2.** *Given a state of* Dots & Boxes*, it is NP-hard to decide whether $\mathcal{B}$ can win.*

▶ **Theorem 3.** *Given a state of* Dots & Simple Polygons*, it is NP-hard to decide whether $\mathcal{B}$ can win.*

3 Strategy

What follows is a discussion of greedy strategies for *Dots & Polygons* played on a set of points P in convex position. Trivially, when the points are places in convex position, there exists no distinction between *Dots & Polygons & Holes* and *Dots & Simple Polygons*. In the related *Monochromatic Complete Triangulation Game* a greedy strategy is optimal for such points [1].

We first observe that in this setting the number of turns is exactly $n = |P|$: Consider connected components of the edges drawn by the players. If a player connects two points in the same component, this closes a polygon, and therefore the turn continues. If, however, the two points are in different components, the turn ends and the number of connected components decreases. Thus, the number of turns equals to the number of initial components.

Consider a game state in which the current player cannot close a polygon. Let E be the set of all edges that can still be drawn. Define the weight $w(e)$ for $e \in E$ to be the area the opponent can claim on their next turn if the current player draws e. For example an edge e between two isolated points has weight $w(e) = 0$. A *simple greedy strategy* is the following: if there is an edge that can close some area, immediately draw that edge. Otherwise, draw the edge $e_{min} = \min_{\forall e \in E} w(e)$. This strategy is not optimal, as shown in Figure 2.

The edges drawn partition the remaining area into *subproblems*. For an edge $e \in E$, $w(e)$ can only change if an edge in the same subproblem is drawn. Let $E' \subset E$ be the set of edges within a subproblem. We call a subproblem *easy*, if only two of the edges $e, e' \in E'$ lie on the convex hull of P. In such a subproblem, all edges have the same weight, namely the area of the subproblem. We call a game state in which all subproblems are easy, an *easy endgame*.

In the following we assume that points are placed in such a way that a draw is not possible. Consider the player that will go last (i.e., $\mathcal{B}$ for odd n, $\mathcal{R}$ for even n). If this player plays the *simple greedy strategy* in such a way that they reach an easy endgame, then this player wins. The reason is that from that point onward, anytime the opponent scores an area, this player will score an area that is at least as large in their next turn. For $n \leq 5$, an easy endgame is always reached. For $n = 6$ and $n = 7$, the player that will go last can enforce an easy endgame by playing a diagonal in their first move, preventing the situation of Figure 2. In this way, $\mathcal{B}$ can always win for $n = 3, 5, 7$, and $\mathcal{R}$ for $n = 4, 6$. We leave the problem for $n > 7$ open.

References

1 Oswin Aichholzer, David Bremner, Erik D. Demaine, Ferran Hurtado, Evangelos Kranakis, Hannes Krasser, Suneeta Ramaswami, Saurabh Sethia, and Jorge Urrutia. Games on triangulations. *Theoretical Computer Science*, 343(1–2):52–54, 2005.

2 Sander Beekhuis, Kevin Buchin, Thom Castermans, Thom Hurks, and Willem Sonke. Ruler of the plane – games of geometry (multimedia contribution). In *33rd International Symposium on Computational Geometry (SoCG)*, volume 77 of *LIPIcs*, pages 63:1–63:5. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017.

3 Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark Overmars. *Computational geometry: algorithms and applications*. Springer, 2008.

4 Elwyn R. Berlekamp. *The Dots and Boxes Game: Sophisticated Child's Play*. AK Peters/CRC Press, 2000.

5 Elwyn R. Berlekamp, John H. Conway, and Richard K. Guy. Chapter 16: Dots-and-boxes. In *Winning Ways for your Mathematical Plays*, volume 3, pages 541–584. A K Peters/CRC Press, 2nd edition, 2003.

6 Hans L. Bodlaender. On disjoint cycles. In *Graph-Theoretic Concepts in Computer Science*, pages 230–238, Berlin, Heidelberg, 1992. Springer Berlin Heidelberg.

7 Kevin Buchin, Mart Hagedoorn, Irina Kostitsyna, Max van Mulken, Jolan Rensen, and Leo van Schooten. Dots & polygons, 2020. `arXiv:2004.01235`.

8 Alberto Caprara and Romeo Rizzi. Packing triangles in bounded degree graphs. *Information Processing Letters*, 84(4):175–180, 2002.

9 David Eppstein. Computational complexity of games and puzzles. Last accessed on 14/02/2020. URL: `https://www.ics.uci.edu/~eppstein/cgt/hard.html`.

10 Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., USA, 1979.

11 Ronald L. Graham. An efficient algorithm for determining the convex hull of a finite planar set. *Information Processing Letters*, 1:132–133, 1972.

12 Sian Zelbo. Dots and polygons game. Last accessed on 14/02/2020. URL: `http://www.1001mathproblems.com/2015/03/for-printable-game-boards-click-here.html`.

Designing Art Galleries

Toon van Benthem
Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands
a.t.v.benthem@student.tue.nl

Kevin Buchin
Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands
k.a.buchin@tue.nl

Irina Kostitsyna
Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands
i.kostitsyna@tue.nl

Stijn Slot
Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands
s.j.slot@student.tue.nl

Abstract

We present a method for generating interesting levels based on several NP-hardness reductions for a puzzle game based on the Art Gallery problem.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Art Gallery problem, NP-hard, puzzle, level generation

Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.80

Category Media Exposition

Supplementary Material game: `https://www.win.tue.nl/~kbuchin/proj/ruler/art/`, source code: `https://github.com/kbuchin/ruler/`

1 Introduction

The *Art Gallery problem* is a classic visibility problem in computational geometry: given a polygon (i.e., an art gallery), find the smallest number of points (guards) inside the polygon such that the guards together see the interior of the polygon. The Art Gallery problem was shown to be NP-hard by Lee and Lin [3] by a reduction from 3SAT.

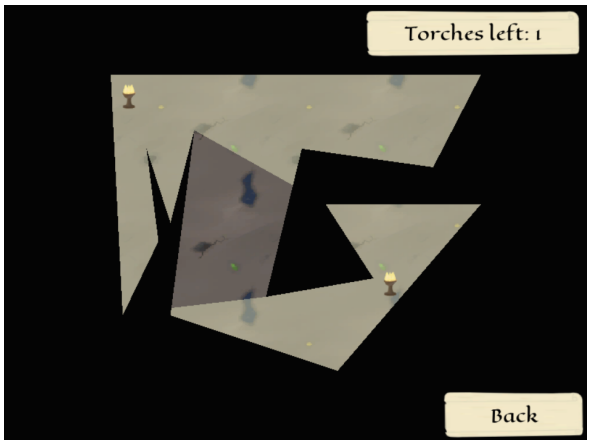


Figure 1 Screenshot from the Art Gallery Game: Two torches/guards have been placed so far, which is not sufficient to see the whole polygon.

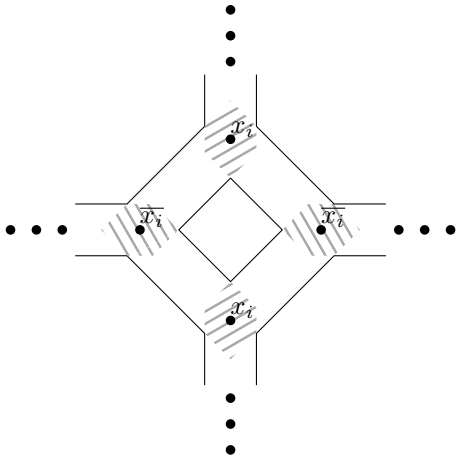


Figure 2 Variable gadget for 3SAT.

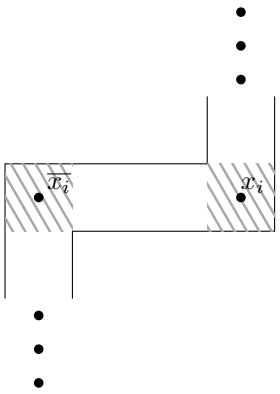


Figure 3 Wire gadget for 3SAT.

In a collection of games based on computational geometry [2], we recently published an Art Gallery puzzle[1], in which the player has to solve the Art Gallery problem, see Figure 1. The goal of this work is to develop methods for generating interesting levels for the Art Gallery puzzle based on NP-hardness reductions. We design levels, i.e., art galleries, based on three hardness reductions, from planar 3SAT, from planar vertex cover, and from geometric hitting set. We present the details of level generation using NP-hardness reductions in Section 2. We discuss how to measure the quality of the levels in Section 3. To evaluate the generated levels we have performed a user study, the results of which we discuss in Section 4.

2 Reductions

Next we present three NP-hardness reductions for the Art Gallery problem. The constructions described can be used to generate new levels, starting from an instance of the respective original problem, for the Art Gallery game.

Planar 3SAT. The first method is based on the reduction from the planar 3SAT [4]. Given an instance of the planar 3SAT, we construct an instance of the Art Gallery problem for polygons with holes. The 3SAT formula is satisfiable if and only if the resulting polygon can be guarded by K or fewer guards, for some K. For each variable and clause of the 3SAT instance we construct a *variable-* and *clause-gadget* respectively, and connect them with *wire gadgets*. To connect the same variable to multiple clauses we use *split gadgets*.

Variable gadget (Fig. 2) is a cycle with an even number of corners. The essential property of the gadget is that it (ignoring outgoing corridors) can be optimally covered in two possible ways. By increasing the size of the gadget (to a $2k$-gon for $k \geq 2$) we can create more instances of x_i and $\overline{x_i}$ for a 3SAT variable x_i. This can also be achieved using a split gadget.

Wire gadget consists of a zig-zag corridor (Fig. 3), which can be optimally covered by placing the guards in every other corner. The wire thus has two settings, which depend on the initial corner in which the first guard is placed. Define the wire to be *active* for a given direction when guard are placed in every other corners starting from the second turn.

[1] `http://www.win.tue.nl/~kbuchin/proj/ruler/`

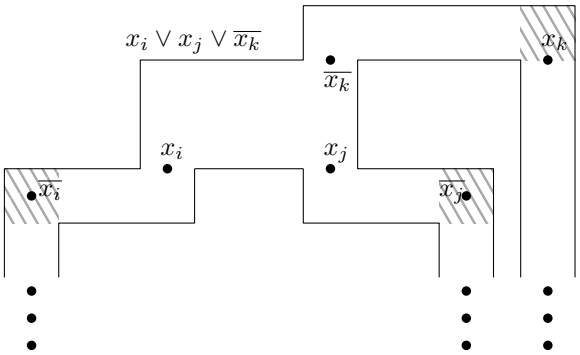


Figure 4 Clause gadget for 3SAT.

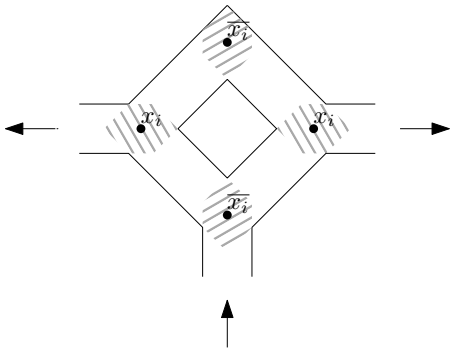


Figure 5 Split gadget for 3SAT.

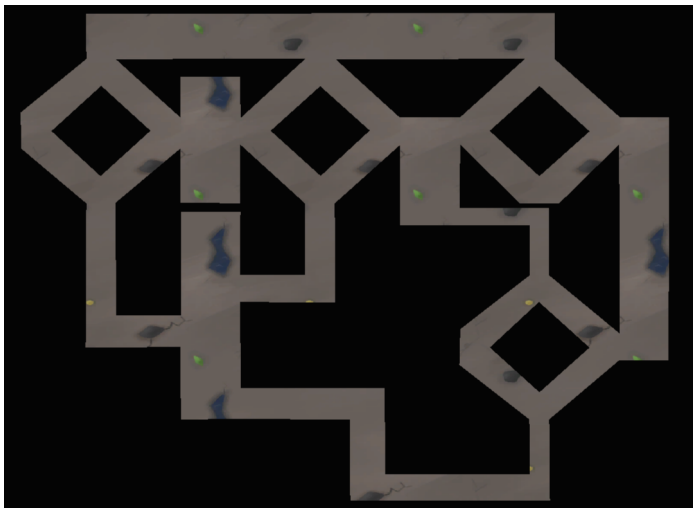


Figure 6 Art Gallery puzzle generated by reduction from planar 3SAT formula $(a \vee b \vee c) \wedge (\neg a \vee \neg b) \wedge (\neg b \vee \neg c \vee d) \wedge (\neg c \vee \neg d) \wedge (a \vee b \vee d)$; 13 guards are necessary for coverage ($K = 13$).

Clause gadget. In the clause gadget (Fig. 4) three corridors meet in a large room. It is covered if one of the wires is active, i.e., one of the wires has a guard in the corner closest to the room. The room can be of any size, as long as it cannot be covered by three inactive wires.

Split gadget. Though a split gadget (Fig. 5) is not technically necessary, for generating interesting puzzles we believe it might be better than a large variable gadget. For the split gadget we can reuse part of the variable gadget.

Puzzle level design. Figure 6 shows an example of an Art Gallery level generated from a small instance of planar 3SAT, with three variables and three clauses. Starting from a 3SAT instance, variables and clauses are replaced by gadgets and connected using corridors. The shape (e.g. variable-clause graph and corridors) for a level is designed by hand.

Planar Vertex Cover. Planar VC is another classic NP-complete problem which can be reduced to Art Gallery. For each vertex and edge, we build a *vertex-* and *edge-gadget*.

Vertex and edge gadgets. Figure 7 shows a vertex gadget. We represent the edges with positive width corridors which meet together in a crossing (highlighted in orange) corresponding to a vertex. The main property is that placing a guard at the crossing will cover all adjacent corridors. Additionally, if all corridors are covered by guards the crossing will also be covered.

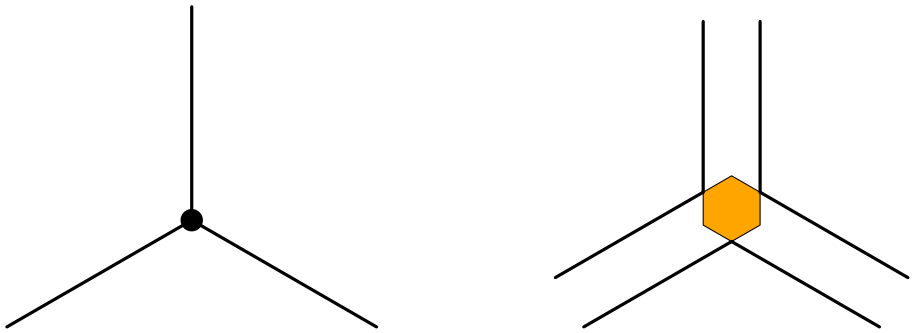


Figure 7 Vertex gadget. Left: original vertex. Right: simple vertex gadget.

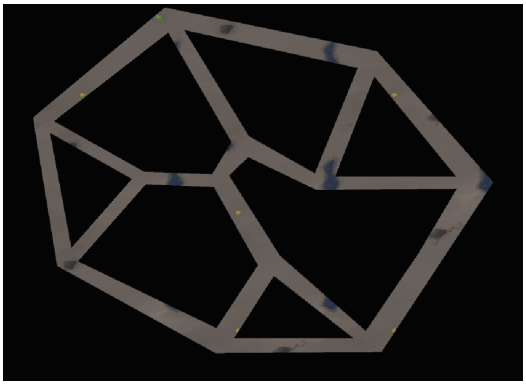


Figure 8 From Vertex Cover ($K = 7$).

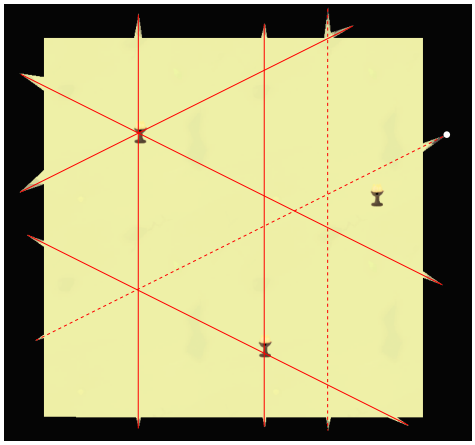


Figure 9 From Geometric Hitting Set ($K = 3$).

One subtlety for the vertex gadget is that corridors must not be collinear for the reduction to work. Potential issues may also arise from the interplay of the thickness of the corridors and the general layout of the construction. However, by carefully designing the layout of the graph and by adjusting the thickness of the corridors, these issues can be avoided.

Puzzle level design. In the final construction we use a planar embedding of the graph to create the Art Gallery level. Figure 8 shows an example of a generated level.

Geometric Hitting Set. A third NP-hardness reduction, used for level generation, is from hitting the lines of a line arrangement with a minimum number of intersection points, the so called "spike box" construction [1]. For a given arrangement of lines in the plane, construct a large box containing all the intersections of the lines. For each line attach two spikes sticking out of the box. Guarding the spike box then is equivalent to hitting all the lines in the line arrangement. A spike box construction is shown in Figure 9, where solid lines are covered by the two guards, and dashed lines are not covered.

3 Defining Interestingness

To be able to compare the three described methods, and to evaluate the quality of generated puzzles, we measure how interesting the resulting puzzles are to a human. It may occur that a certain puzzle is trivial to an algorithmic solver but a human may have a very hard time in finding the right step. We believe the quality of a puzzle to be a combination of the following factors: *niceness* of the polygon shape, *difficulty* of the puzzle, *fun* in solving the puzzle.

The intuition behind this is the following. A puzzle that is too easy is often not interesting, as the solution will be too obvious. A puzzle that is not aesthetically pleasing, for example containing too fine features, can be frustrating. A puzzle that is not fun, e.g., the logic behind which is obvious, is often not interesting, since players may get bored.

4 User Study

In a user study, participants ($n = 17$) were asked to solve puzzles and answer a set of questions. Specifically the flow for each art gallery was: *(i)* observe an Art Gallery puzzle level for five seconds, *(ii)* answer a pre-solve survey, *(iii)* attempt to solve the Art Gallery puzzle level, and *(iv)* answer a post-solve survey.

The conclusions we draw from the user study is that the puzzles generated based on the reduction from the planar 3SAT were the most difficult ones (indicated by high solve time and placement attempts), yet also the most enjoyable. The single most enjoyed puzzle is a 3SAT puzzle (a puzzle similar to Fig. 6). Vertex cover puzzles were the most logical to solve, yet also the easiest set of puzzles. The spike box puzzles were considered the least nice. Their too fine features resulted in significantly lower average scores for shape aesthetics, solving fun, logical approach, and shape uniqueness. The single least enjoyed puzzle is a spike box puzzle (one similar to Fig. 9), the main problem being the lack of clarity regarding what part is still not illuminated. As a consequence, we added a *feedback point* which highlights some point in the polygon (white dot in Fig. 9) that is not yet covered.

References

1 Yoav Amit, Joseph S.B. Mitchell, and Eli Packer. Locating guards for visibility coverage of polygons. *International Journal of Computational Geometry & Applications*, 20(05):601–630, 2010.

2 Sander Beekhuis, Kevin Buchin, Thom Castermans, Thom Hurks, and Willem Sonke. Ruler of the plane – games of geometry (multimedia contribution). In *Proc. 33rd International Symposium on Computational Geometry (SoCG)*, pages 63:1–63:5, 2017.

3 Der-Tsai Lee and Arthur K. Lin. Computational complexity of art gallery problems. *IEEE Transactions on Information Theory*, 32(2):276–282, 1986.

4 David Lichtenstein. Planar formulae and their uses. *SIAM Journal on Computing*, 11(2):329–343, 1982.

Plane-Filling Trails

Herman Haverkort
Universität Bonn, Germany
http://herman.haverkort.net/
cs.herman@haverkort.net

Abstract

The order in which plane-filling curves visit points in the plane can be exploited to design efficient algorithms. Typically, the curves are useful because they preserve locality: points that are close to each other along the curve tend to be close to each other in the plane, and vice versa. However, sketches of plane-filling curves do not show this well: they are hard to read on different levels of detail and it is hard to see how far apart points are along the curve. This paper presents a software tool to produce compelling visualisations that may give more insight in the structure of the curves.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases space-filling curve, plane-filling curve, spatial indexing

Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.81

Category Media Exposition

Related Version A full version of this paper is available at https://arxiv.org/abs/2003.12745.

Supplementary Material For software and additional figures visit http://spacefillingcurves.net.

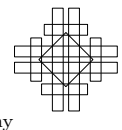
1 Plane-filling curves

A plane-filling curve is a continuous surjective mapping f from the unit interval to a subset of the plane that has positive area, that is, Jordan content. Although such a mapping cannot be one-to-one, an unambiguous inverse can be defined with a tie-breaking rule. Thus, the mapping provides an order in which to process points in the plane. Famous examples include Pólya's triangle-filling curve [12] and square-filling curves by Peano [11] and Hilbert [7]. Continuity of the mapping is not always required: if we drop this requirement, we speak of plane-filling *traversals*. Z-Order [9] is an example that is often applied in practice.

Plane-filling traversals and their inverses have been used to design efficient solutions for various applications, including indexing of points in the plane, geometric algorithms and data structures, finite element methods, load balancing in parallel computing, improving cache utilization in computations on large matrices or images, combinatorial optimization, image compression, information visualization, and sonification [1, 6]. It is therefore interesting to see the differences between the various plane-filling traversals that have been proposed.

2 Defining a plane-filling curve

Plane-filling traversals are usually visualised in a way that follows their definition. Consider Pólya's curve. To define it, we start with a single line segment (Figure 1a). We refine this simple drawing as follows. Let p and r be the end points of the original line segment. Imagine a circle with centre line pr and draw another point q halfway on the circle as we follow it clockwise from p to r. Erase the original line segment pr and replace it by two smaller segments pq and qr (Figure 1b). Next, refine the drawing again by applying the same refinement procedure to each segment, but this time changing the orientation: to find the new intermediate points, we now follow the circles in counterclockwise direction. To indicate this change in orientation, we add an arrow head to pr on the left side, and put the arrow heads for pq and qr on the right side. Thus, two line segments become four segments



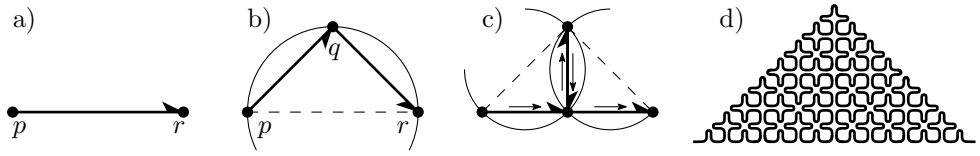


Figure 1 Pólya's triangle-filling curve.

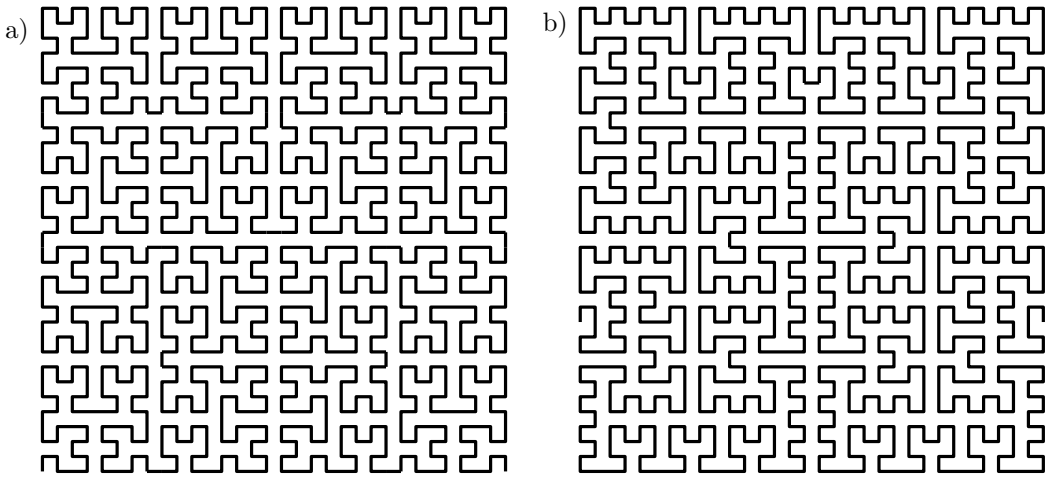


Figure 2 Sketches of a) the Hilbert curve [7] and b) an Ω section of the $\beta\Omega$-curve [16].

(Figure 1c). Note that the middle two segments lie on top of each other, but they have different directions. If we repeat this refinement process six more times, alternating clockwise and counterclockwise, and move all points slightly so that the curve does not back up on itself, we get Figure 1d. If we continue ad infinitum, the curve fills a right isosceles triangle.

3 The challenge of visualising plane-filling curves

Figures 1b and 1d are typical of the way in which plane-filling curves are usually sketched. Neither figure makes it clear in an instant in what order the curve fills what parts of the plane – not to mention showing the curve's locality-preserving properties and violations thereof. Try comparing, for example, Hilbert's curve in Figure 2a to the $\beta\Omega$-curve in Figure 2b (a promising alternative [17]). Furthermore, the impression one gets of the curve depends heavily on how one chooses to define it and on the details of how it is sketched. Figure 3a shows three sketches that all sketch the same curve, and Figure 3b shows a sketch of a trapezoid-filling curve that is nothing else than the first three quarters of Pólya's curve: none of this is visually obvious from the drawings.

4 Visualisation as three-dimensional landscapes

To visualise plane-filling curves and traversals more clearly, I developed a tool pftrail. The tool reads a definition of a plane-filling curve and produces a *plane-filling trail*, a model of the curve on a three-dimensional landscape, in which each point $f(t) = (x, y)$ of the curve is rendered as a point (x, y, t). Thus the curve becomes a steadily ascending path in the landscape, see Figure 4. At a low resolution, the concept can be seen in action in a Hilbert curve marble run design by Ortiz [10]. At higher resolutions, we obtain a clear visualisation of the locality-preserving and locality-violating properties of the curve that can be studied at different levels of detail. High, steep slopes reveal pairs of points that are close in the

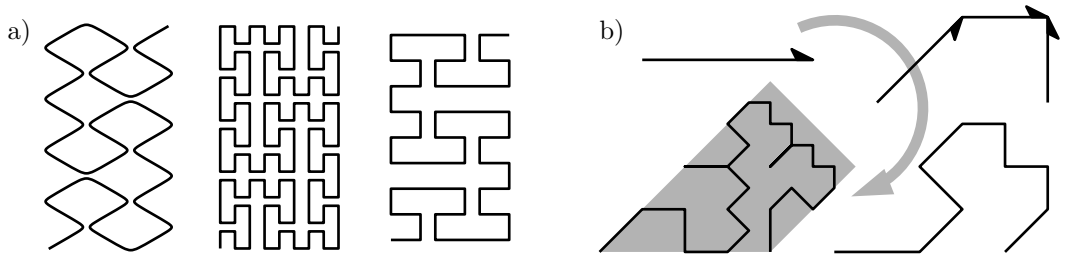


Figure 3 a) Three sketches of Peano's curve, mapped to a $\sqrt{3} : 1$ rectangle. b) A sketch of a trapezoid-filling curve.

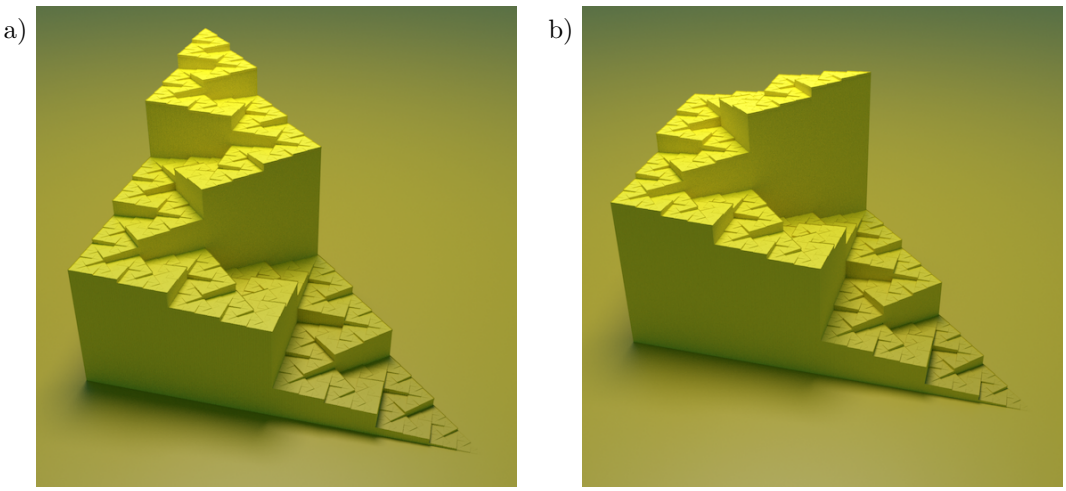


Figure 4 a) Pólya's curve. b) The trapezoid-filling curve sketched in Figure 3b.

plane but far apart along the curve. Narrow corridors reveal sections between points that are relatively close to each other along the curve, but far apart in the plane. Wide corridors show sections of the curve that have good locality-preserving properties in both directions. The global course of the curve is easy to follow, but the image also facilitates studying the curve in more detail. Moreover, the visualisation is independent of what definition of the curve is used, out of multiple equivalent definitions. For example, the fact that the trapezoid-filling curve is simply the first three quarters of the Pólya curve is now obvious, see Figure 4. The visualisation gives the user the possibility to study the curve without any bias towards an arbitrary underlying tessellation.

5 Alternative visualisations

Alternative methods that come closest to meeting the same goals render the t-coordinate as values on a grey or colour scale instead of elevation. Indeed, such renderings are quite common. However, in comparison to our perception of elevation in a landscape, our perception of colour is less precise, more context-dependent, and not invariant under translation. Another interesting alternative are the three-dimensional models by Irving and Segerman [8] that stack different refinement levels according to a definition of the curve. However, these models are hard to "read" when presented as a two-dimensional printed image, and they are inherently dependent on the chosen definition of the curve. That is fine if one wants to illustrate the definition, but it is a shortcoming if one wants to be able to reveal the equivalence of different definitions by producing the same image in such cases.

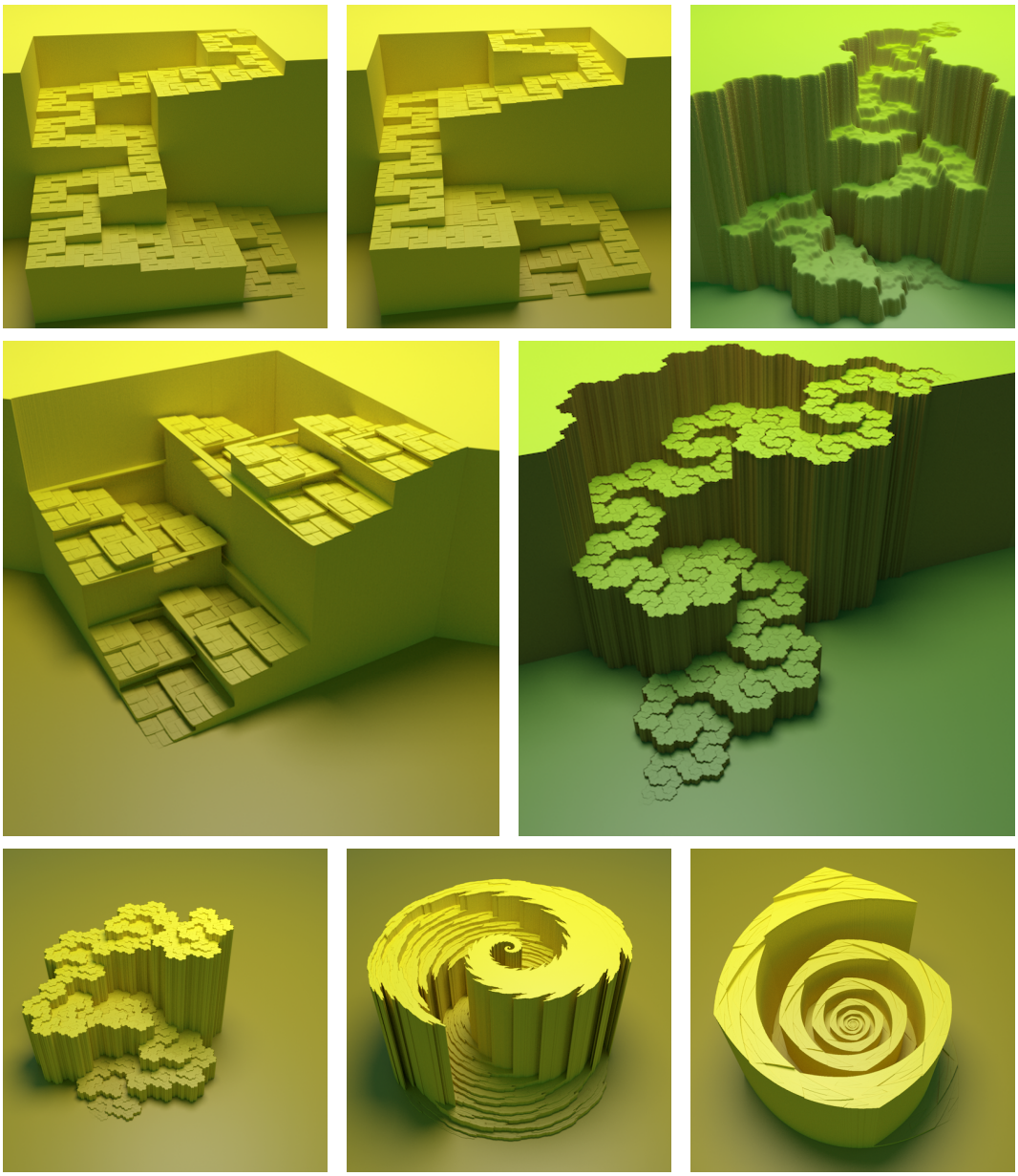


Figure 5 On top: Hilbert's curve [7], $\beta\Omega$ [16], and a curve from Ventrella [15] (p86) filling a fractal "pinwheel" tile [2], rendered "eroded". In the middle: Double-Gray-code [4] and a curve filling half of a Rauzy fractal [13]. Bottom: the Gosper curve [5], a close-up of the point at 2/7 of the curve where three tiles meet ($\zeta = 8$), and a close-up of the point at 1/3 of Pólya's curve ($\zeta = 5$).

6 The pftrail tool

The pftrail tool reads the definition of a plane-filling traversal in the format from Ventrella [15], extended to support discontinuities and multiple refinement rules (known as *generators*). Thus, the various traversals that have been proposed in the computer science literature [3, 14, 16] can all be rendered and it is easy to explore new designs. Traversals are not confined to an integer grid, so we can also visualise interesting traversals related to, for example, the Rauzy

fractal [13] (see Figure 5). For rendering, the traversal is sampled and drawn on a grid of hexagonal cells; thus pftrail operates without any knowledge of the shape that is filled by the traversal (which can be a complicated fractal). The tool offers various options to control parameters such as camera position, resolution of the rendering grid, visualisation style, and the height of "parapets" that accentuate steep drops to enhance the perception of depth. The output is a collada file that can be rendered with, for example, Blender; if the resolution is not too high, it can also be moved around in Blender in real time.

Special features include "polynomial" close-up: given a focus point p and a zoom parameter ζ, any point q at distance x from p is moved to the point at distance $x^{1/\zeta}$ on the ray from p through q. Elevation differences are modified in a similar way. This allows us to zoom in on features that remain invisible in normal close-up views. For example, the Gosper curve (Figure 5, bottom left) follows a tessellation with tiles arranged in a hexagonal grid pattern. At the vertices of this grid, the tiles wind around each other like logarithmic spirals that shrink by a factor of roughly $9 \cdot 10^7$ per revolution. No normal close-up view could show these spirals, but the polynomial close-up reveals them clearly, see Figure 5.

References

1. M. Bader. *Space-filling curves: an introduction with applications in scientific computing*. Springer, 2013.
2. C. Bandt, D. Mekhontsev, and A. Tetenov. A single fractal pinwheel tile. *Proc. Amer. Math,. Soc.*, 146:1271–1285, 2018.
3. C. Burstedde and J. Holke. A tetrahedral space-filling curve for nonconforming adaptive meshes. *SIAM J. Scientific Computing*, 38(5), 2016.
4. C. Faloutsos. Multiattribute hashing using Gray codes. In *Proc. 1986 Conf. ACM SIG Management of Data (SIGMOD 1986)*, pages 227–238, 1986.
5. M. Gardner. Mathematical games: in which "monster" curves force redefinition of the word "curve". *Scientific American*, 235:124–133, 1976.
6. H. Haverkort. Sixteen space-filling curves and traversals for d-dimensional cubes and simplices. *CoRR*, abs/1711.04473, 2017. `arXiv:1711.04473`.
7. D. Hilbert. Über die stetige Abbildung einer Linie auf ein Flächenstück. *Math. Ann.*, 38(3):459–460, 1891.
8. G. Irving and H. Segerman. Developing fractal curves. *J. of Mathematics and the Arts*, 7(3–4):103–121, 2013.
9. G. Morton. A computer oriented geodetic data base, and a new technique in file sequencing. Technical report, International Business Machines Co., Ottawa, Canada, 1966.
10. S. Ortiz. Hilbert curve marble run, 2018. Accessed 26 March 2020. URL: `https://www.thingiverse.com/thing:3031891`.
11. G. Peano. Sur une courbe, qui remplit toute une aire plane. *Math. Ann.*, 36(1):157–160, 1890.
12. G. Pólya. Über eine Peanosche Kurve. *Bull. Int. Acad. Sci. Cracovie, Ser. A*, pages 305–313, 1913.
13. G. Rauzy. Nombres algébriques et substitutions. *Bulletin Soc. Math. Fr.*, 110:147–178, 1982.
14. H. Samet. *Foundations of multidimensional and metric data structures*, page 199. Morgan Kaufmann, 2006.
15. J. Ventrella. *Brainfilling curves: a fractal bestiary*. Eyebrain books, 2012.
16. J.-M. Wierum. Definition of a new circular space-filling curve: $\beta\Omega$-indexing. Technical Report TR-001-02, Paderborn Center for Parallel Computing (PC^2), 2002.
17. S.-E. Yoon and P. Lindstrom. Mesh layouts for block-based caches. *IEEE Trans. on Visualization and Computer Graphics*, 12(5), 2006.

Visual Demo of Discrete Stratified Morse Theory

Youjia Zhou
University of Utah, Salt Lake City, UT, USA
zhou325@sci.utah.edu

Kevin Knudson
University of Florida, Gainesville, FL, USA
kknudson@ufl.edu

Bei Wang[1]
University of Utah, Salt Lake City, UT, USA
beiwang@sci.utah.edu

Abstract

Discrete stratified Morse theory, first introduced by Knudson and Wang, works toward a discrete analogue of Goresky and MacPherson's stratified Morse theory. It is inspired by the works of Forman on discrete Morse theory by generalizing stratified Morse theory to finite simplicial complexes. The class of discrete stratified Morse functions is much larger than that of discrete Morse functions. Any arbitrary real-valued function defined on a finite simplicial complex can be made into a discrete stratified Morse function with the proper stratification of the underlying complex. An algorithm is given by Knudson and Wang that constructs a discrete stratified Morse function on any finite simplicial complex equipped with an arbitrary real-valued function. Our media contribution is an open-sourced visualization tool that implements such an algorithm for 2-complexes embedded in the plane, and provides an interactive demo for users to explore the algorithmic process and to perform homotopy-preserving simplification of the resulting stratified complex.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Discrete Morse theory, stratified Morse theory, discrete stratified Morse theory, topological data analysis, data visualization

Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.82

Category Media Exposition

Related Version A paper describing the theoretical foundation for the visualization is available at `https://arxiv.org/abs/1801.03183`.

Supplementary Material The visualization tool (our media contribution) is available on GitHub: `https://github.com/tdavislab/VIS-DSMT` with a link to a visual demo.

Funding NSF IIS-1513616, NSF DBI-1661375.

Acknowledgements We thank Yulong Liang who worked on the first prototype of the visualization.

1 Discrete stratified Morse theory

We begin with a brief review of discrete stratified Morse theory (DSMT) [4, 5]. Several key components differentiate DSMT from previous theories [1, 2, 3]. First, we work with *open simplices*, which is not surprising, since we are developing a discrete version of the strata in stratified spaces, which are typically open manifolds. Second, we define a *stratified simplicial complex*, which mimics the frontier condition of a Whitney stratification in a discrete setting. Third, we derive the notion of a *discrete stratified Morse function* from the

[1] Corresponding author

definition of a stratified Morse function. Finally, we establish the theoretical foundations of DSMT regarding its ability to capture the shape of data (homotopy type and homology) and its relation to DMT (generality). In particular, DSMT is applicable to *any* function f defined on a simplicial complex K since f can be made into a discrete stratified Morse function with respect to some stratification $\mathcal{S}$.

▶ **Definition 1** (Definition 3.2, [5]). *Let K be a simplicial complex. A* stratification *of K, $\mathcal{S} = \{S_i\}$, is a locally finite collection of disjoint locally closed subsets called* strata*, $S_i \subset K$, such that $K = \bigcup S_i$ and S_i satisfies the condition of the frontier: $S_i \cap \overline{S}_j \neq \emptyset$ if and only if $S_i \subset \overline{S_j}$. Each S_i is a union of (open) simplices; its connected components are called* strata pieces*. Let $\overline{S}_i$ denote its closure, and $\mathring{S}_i$ its interior.*

A stratification gives an *assignment* $s : K \to \mathcal{S}$ taking each open simplex σ in K to a particular stratum $s(\sigma)$. Let K be a simplicial complex equipped with a stratification s and a function $f : K \to \mathbb{R}$. For a p-simplex α, we define

$$U_s(\alpha) = \{\beta^{(p+1)} > \alpha \mid s(\beta) = s(\alpha) \text{ and } f(\beta) \leq f(\alpha)\},$$
$$L_s(\alpha) = \{\gamma^{(p-1)} < \alpha \mid s(\gamma) = s(\alpha) \text{ and } f(\gamma) \geq f(\alpha)\}.$$

Similarly, $U(\alpha) = \{\beta^{(p+1)} > \alpha \mid f(\beta) \leq f(\alpha)\}$, and $L(\alpha) = \{\gamma^{(p-1)} < \alpha \mid f(\gamma) \geq f(\alpha)\}$.

▶ **Definition 2** (Definition 3.5, [5]). *A function $f : K \to \mathbb{R}$ (equipped with a stratification s) is a* discrete stratified Morse function *if for every p-dimensional simplex $\alpha^{(p)} \in K$, (i) $|U_s(\alpha)| \leq 1$, (ii) $|L_s(\alpha)| \leq 1$, and (iii) if one of these sets is nonempty, then the other must be empty.*

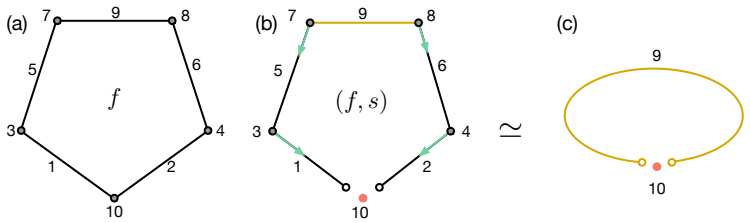


Figure 1 An example of a discrete stratified Morse function. Red are violators; yellow are critical simplicies. See **DSMT Exp. 1** in the demo.

▶ **Definition 3** (Definitions 3.6 and 3.7 [5]). *A simplex $\alpha^{(p)}$ is* globally critical *if $|U(\alpha)| = |L(\alpha)| = 0$. A simplex $\alpha^{(p)}$ is* locally critical *if it is not globally critical and if $|U_s(\alpha)| = |L_s(\alpha)| = 0$. A* critical value *of f is its value at a critical simplex. A simplex $\alpha^{(p)}$ is* globally noncritical *if $|U(\alpha)| + |L(\alpha)| = 1$. A simplex $\alpha^{(p)}$ is* locally noncritical *if it is not globally noncritical and 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 $|U_s(\alpha)| = 0$.*

▶ **Definition 4** (Definition 3.8, [5]). *A simplex $\alpha^{(p)}$ is a* violator *(of the conditions associated with a discrete Morse function) if it is neither critical nor noncritical.*

Violators are central to the algorithm in constructing a discrete stratified Morse function. See Figure 1 for an example: f in (a) is not a discrete stratified Morse function; however, it can be converted into one when it is equipped with an appropriate stratification s in (b).

Given a discrete stratified Morse function f equipped with a stratification s, f restricted to $S_i \in \mathcal{S}$, denoted as $f_i := f\,|_{S_i}$, is by definition a discrete Morse function. We may associate a discrete gradient vector field V_i to S_i as follows. Since any noncritical simplex $\alpha^{(p)} \in S_i$ has at most one of the sets $U_s(\alpha)$ and $L_s(\alpha)$ being nonempty, there is a unique face $\gamma^{(p-1)} < \alpha$ in S_i with $f(\gamma) \geq f(\alpha)$ or a unique coface $\beta^{(p+1)} > \alpha$ in S_i with $f(\beta) \leq f(\alpha)$. The pair $\{\alpha < \beta\}$ (or the pair $\{\gamma < \alpha\}$) is referred to as a *Morse pair*. Denote by V_i the collection of all such pairs. Such pairs are formed by (globally or locally) noncritical simplices. We visualize V_i by drawing an arrow from α to β for every Morse pair $\{\alpha < \beta\} \in V_i$. Such a discrete gradient provides a simplification (i.e., collapsing) order for the complex S_i, where Morse pairs can be removed to produce a reduced complex with the same homotopy type, see Figure 1(c). We have the following result [5, Theorem 3.1]:

▶ **Theorem 5** (Weak DSMT Theorem A, Theorem 3.3, [5]). *Given a discrete stratified Morse function (f, s), performing a collapse of either a global noncritical pair or a local noncritical pair is a stratum-preserving homotopy equivalence.*

2 Algorithm

Given a simplicial complex K, and any real-valued function $f : K \to \mathbb{R}$, we can construct a discrete stratified Morse function using the following algorithm described in [5, Section 3.5]:

1. Make a single pass of all simplices in K, and order the violators $\mathcal{V} = \{\sigma_1, \sigma_2, \ldots, \sigma_r\}$ by increasing dimension and by increasing function value within each dimension.
2. Initialize $\mathcal{S} = \emptyset$, $i = 1$.
3. Remove σ_i from $\mathcal{V}$ and add σ_i to $\mathcal{S}$.
4. Consider $K_i = K \setminus \{\sigma_1, \ldots, \sigma_i\}$:
   - If the restriction of f to K_i, $f|_{K_i}$, is a discrete Morse function, then let J denote the set of indices $k \leq i$ such that $\sigma_k \in \overline{K_i}$ and add the following strata to $\mathcal{S}$ (which may contain more than two strata pieces): the frontier $\overline{K_i} \setminus (\mathring{K}_i \cup \{\sigma_j\}_{j \in J})$ and $\mathring{K}_i$.
   - Otherwise, if $f|_{K_i}$ is not a discrete Morse function, then at least one σ_j with $j > i$ remains a violator.
5. Remove simplices that are no longer violators from the list and repeat steps 2-4 above until no violators are left.

The result of the algorithm is shown in Figure 1(b). We have shown the correctness of the algorithm in [5, Theorem 3.4].

3 Visualization design

We provide an interactive visualization tool that demonstrates the algorithmic process of DSMT as described in Section 2; see Figure 2 for its user interface and the video for a demo.

The tool constructs a discrete stratified Morse function from any real-valued function defined on a 1- or 2-complex embedded in the plane (e.g., a planar triangulation). The tool provides 6 examples for DSMT, 4 of which are described in [5, Section 4], and for comparison, 3 examples for DMT (since a discrete Morse function is a discrete stratified Morse function for the trivial stratification). The tool enables the random perturbation of function values for each example. It also allows the import of user-specified examples in an appropriate format (see the user's guide for details). Using such a tool, we visualize a simplicial complex with function values attached to each simplex. We explore various stages of the algorithm by

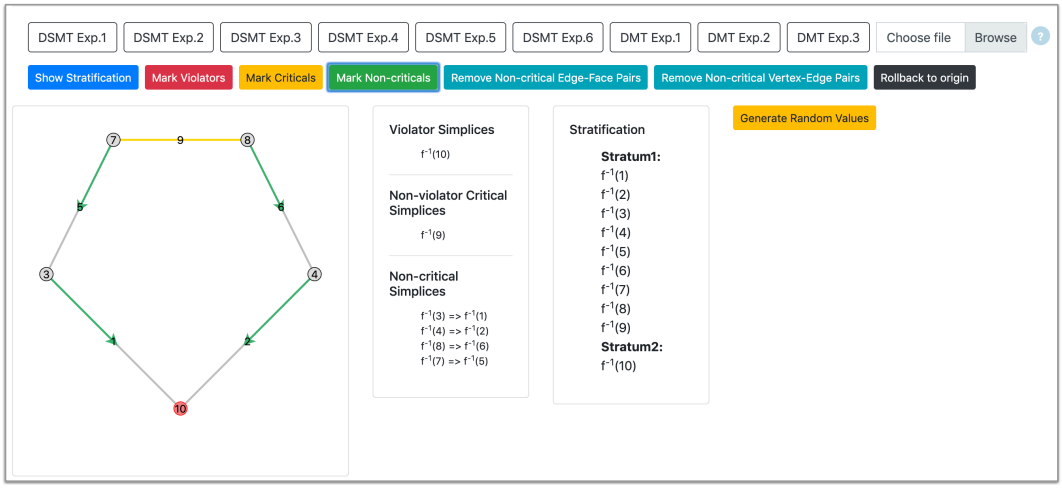


Figure 2 The user interface of our visualization tool.

marking violators, critical simplices, and noncritical simplices, which form Morse pairs that are visualized by green arrows. We also highlight the resulting stratification. We additionally perform homotopy-preserving simplification of the stratified simplicial complex by removing Morse pairs. Figure 3 illustrates such a process; see the visualization tool for more examples.

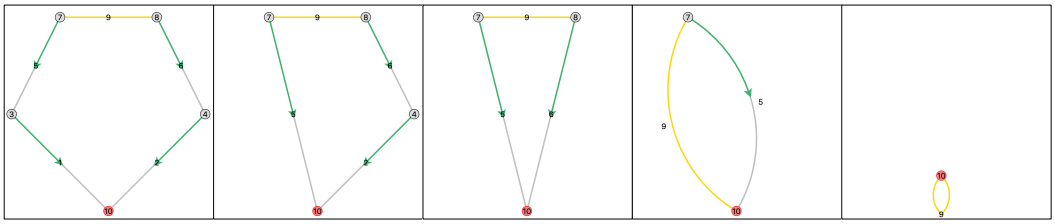


Figure 3 A homotopy-preserving simplification of a stratified simplicial complex.

References

1 Robin Forman. Morse theory for cell complexes. *Advances in Mathematics*, 134:90–145, 1998.
2 Robin Forman. A user's guide to discrete Morse theory. *Séminaire Lotharingien de Combinatoire*, 48, 2002.
3 Mark Goresky and Robert MacPherson. *Stratified Morse Theory*. Springer-Verlag, 1988.
4 Kevin Knudson and Bei Wang. Discrete stratified Morse theory: A user's guide. *International Symposium on Computational Geometry (SOCG)*, 2018.
5 Kevin Knudson and Bei Wang. Discrete stratified Morse theory: Algorithms and a user's guide, 2019. `arXiv:1801.03183`.

Computing Low-Cost Convex Partitions for Planar Point Sets with Randomized Local Search and Constraint Programming

Da Wei Zheng
Department of Computer Science, University of British Columbia, Vancouver, Canada
zhengdw@cs.ubc.ca

Jack Spalding-Jamieson
Department of Computer Science, University of British Columbia, Vancouver, Canada
jacketsj@alumni.ubc.ca

Brandon Zhang
Department of Computer Science, University of British Columbia, Vancouver, Canada
brandon.zhang@alumni.ubc.ca

Abstract

The Minimum Convex Partition problem (MCP) is a problem in which a point-set is used as the vertices for a planar subdivision, whose number of edges is to be minimized. In this planar subdivision, the outer face is the convex hull of the point-set, and the interior faces are convex. In this paper, we discuss and implement the approach to this problem using randomized local search, and different initialization techniques based on maximizing collinearity. We also solve small instances optimally using a SAT formulation. We explored this as part of the 2020 Computational Geometry Challenge, where we placed first as Team UBC.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases convex partition, randomized local search, planar point sets

Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.83

Category CG Challenge

Related Version A description of the 2020 CG:SHOP Challenge with related work and overall outcomes can be found at [4] `https://arxiv.org/abs/2004.04207`.

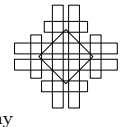
Acknowledgements We want to thank Sam Bayless for help with MonoSAT and constraint programming.

1 Introduction

For a point set P, a convex partition is a planar subdivison with vertex set P, including all the edges of the convex hull of P, such that all interior faces are convex. In the Minimum Convex Partition problem, the number of edges, or equivalently the number of faces, in a convex partition of P is minimized. When collinear points are allowed, this problem has been shown to be NP-hard [5].

One example of convex partitions is the family of triangulations. In fact this is the family of maximum convex partitions, so all other convex partitions can be compared to triangulations. We can also further conclude that any convex partition is a subset of some triangulation.

In this paper, we explore practical solutions to this problem, in the case of collinear points being allowed, and points having integer coordinates. We explored this during the 2020 Computational Geometry Challenge (CG:SHOP 2020), which was made to encourage the exploration of this problem [4]. We placed first in the challenge as Team UBC, and our experimental results in this paper are based on the instances provided.



We begin by briefly describing a simple constraint solving approach for small instances. We then discuss the details of a randomized local search approach. The details are divided into search operations, initialization techniques, and finally metaheuristics. Finally, we will discuss our experimental testing and results.

2 Algorithmic methods

2.1 Approaches for the Small Instances

We quickly solved the smallest instances (with ≤ 50 points) with a simple SAT formulation: For every pair of points in a point-set, create a variable representing whether the corresponding edge is enabled, i.e. it appears in the convex partition. In a convex partition, each interior vertex has what we refer to as the standard convexity constraint: For each edge coming into the vertex, and adjacent face, there should be another edge leaving the vertex touching the same face, such that the interior angle of the face at that vertex is at most 180 degrees. We can add this convexity constraint to the SAT formulation: For every enabled edge adjacent to an interior vertex, require that at least one of the edges within a 180 degree window in each direction is also enabled. Additionally, require that each vertex has degree at least 2, and that all convex hull edges are enabled. In addition to the encoded convexity constraint, we also require that for every two edges that intersect, at least one edge is disabled. Lastly, disable any edge that intersects a point in the point-set. The size of this formulation is $O(n^4)$ for a point-set with n vertices. The number of enabled edges in this formulation can be minimized using a MaxSAT solver. We used the MaxSAT solver UWrMaxSAT [2].

We also attempted a formulation with the geometry tools in MonoSAT [3], although it was not efficient enough to solve any instances.

2.2 Randomized local search

The main approach we used was a simple greedy local search starting from a valid convex partition. The search was designed for iterations to be as fast as possible. An edge that can be removed while preserving the convexity of its bordering faces is greedily removed. The removed edge is selected uniformly at random among all edges that can be removed. If no edge can be removed, a random edge is selected for a *rotation*. A rotation of an edge (u, v) around the point u can be performed when the edge is required to satisfy the vertex convexity constraint at u but not necessarily at v. This edge is removed and replaced with an edge (u, a) or (u, b) if these do not violate the convexity constraints at u (where a and b are the points adjacent to v on the same face as u). The direction of rotation is chosen at random. An edge for which this operation can be performed without violating the vertex convexity constraint is called *rotatable*.

2.3 Fast operations for high number of iterations

For this randomized local search strategy to be successful, we needed to be able to run many iterations quickly.

At each iteration, we sample at random a half-edge (an edge and a pivot vertex) that is rotatable in one or both directions. To sample valid edges quickly, we maintain a list of rotatable half-edges. After each rotation of an edge or deletion of an edge, we only change the rotation status of the neighbouring edges (sharing both a vertex and a face) of the rotation or deletion. This is because this operation only changes the local angular properties of the edges that are adjacent to the rotatable edge on the two faces that the edge was on.

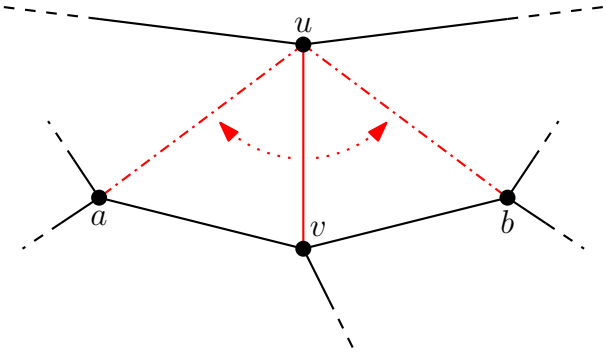


Figure 1 The edge (u, v) can rotate to either (u, a) or (u, b) in one iteration.

We managed to do an average of 20000 iterations per second on large instances. In our implementation, one iteration took $O(\log n)$ operations. It could have been written in expected $O(1)$ operations per iteration by making use of the expected (and practically) low degree of each vertex, but we did not explore this during the competition.

2.4 Initialization

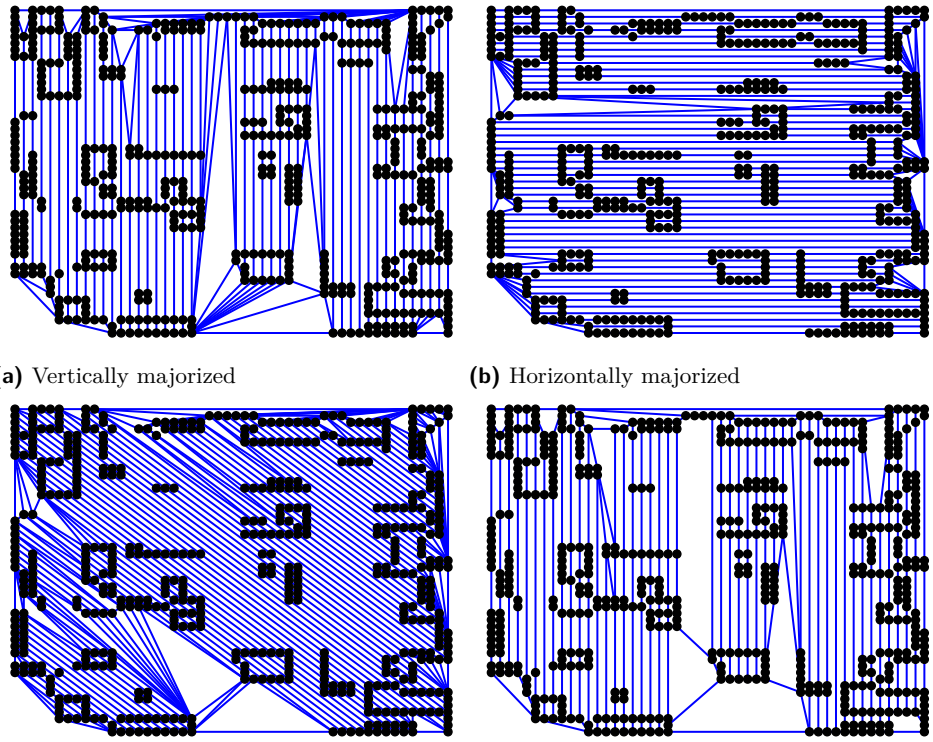
The local search strategy we employed is fairly sensitive to initialization and to the random choices that were made. To mitigate this, we spent some effort finding different initializations. We found that the best solutions for most instances were obtained by starting the local search from the Delaunay triangulation.

The second set of instances included dense point sets in a small grid, and had a large number of collinear points. With collinear points, it is possible to obtain convex partitions with a large number of degree two vertices that have edges on opposite sides, which is what we did. To do well on these instances we joined all vertices with the same x-coordinate in a chain. For the top and bottom ends of each chain we joined them with an algorithm similar to the monotone chain algorithm [1] for convex hulls where we add every edge the monotone chain algorithm considers to create a valid convex partition. Figure 2(a) is an example of such an instance.

We also generalized this to arbitrary slopes, by rotating the points before running the monotone chain subroutine. 2(b) and (c) are examples of this. We call this *majorization* with respect to a slope. We tried to initialize the local search with majorized initializations using commonly occuring slopes we found by sampling random pairs of points. However these other initializations did not improve our results.

2.5 Metaheuristics

To mitigate the sensitivity of the solver to the random choices it makes, we restarted an instance from a randomly chosen initial configuration if the objective value of the solution did not improve for $8n \log^2 n$ iterations, where n is the number of points in the instance. We chose a value that was $\Omega(n \log^2 n)$ to ensure that every edge would be moved at least once with high probability, as we sample the $O(n)$ rotatable half-edges with uniform probability.



(a) Vertically majorized

(b) Horizontally majorized

(c) Slope -1 majorized

(d) Final result after local search

Figure 2 Images of convex partitions of the `rop0000548` instance.

3 Practical computation

3.1 Computational envorinment

We performed computations on one of the shared UBC undergrad CS servers, which has two 32-core Intel Xeon E5-2698 v3 CPUs running at 2.30 GHz. After some initial testing of the algorithm, we ran the local search continuously on all instances for about 16 days, for a total of approximately 2.8 years of CPU time.

3.2 Experimental results

We tested our implementations using the instances provided for CG:SHOP 2020. Here we provide some analysis on the following groups of instances:

- `euro-night` and `us-night`: Randomly sampled points from an illumination map of Europe/the US at night.
- `uniform`: Uniformly randomly generated points.
- `rop`: Instances with many orthogonally collinear points in a bounded grid.

For each convex partition of an instance, a score can be computed according to the following formula:

$$\text{score} = \frac{\text{number of edges in convex partition}}{\text{number of edges in triangulation}}$$

As was mentioned in the introduction, the number of edges in the triangulation is maximum, so the score will be a value between 0 and 1. Smaller scores are better.

3.3 Performance on small instances

All the instances up to size 100 were run using the MaxSAT solver UWrMaxSAT [2], which allows the instances to be solved optimally. The largest instance to be completely solved within a 600 second time limit had 45 points. Most of the instances with at most 45 points completed in under a second. During the contest, there was no strict time limit for the MaxSAT solver, and a total of 70 instances were solved optimally with the SAT formulation before they were later matched by our local search methods. The largest of these had 100 points.

Our local search methods eventually matched the answers we obtained from running MaxSAT. The longest any of these took to match the answers was 65142 iterations, with 20 restarts. Most took under a second and less than 20000 iterations, and used no restarts.

3.4 Performance on uniform instances

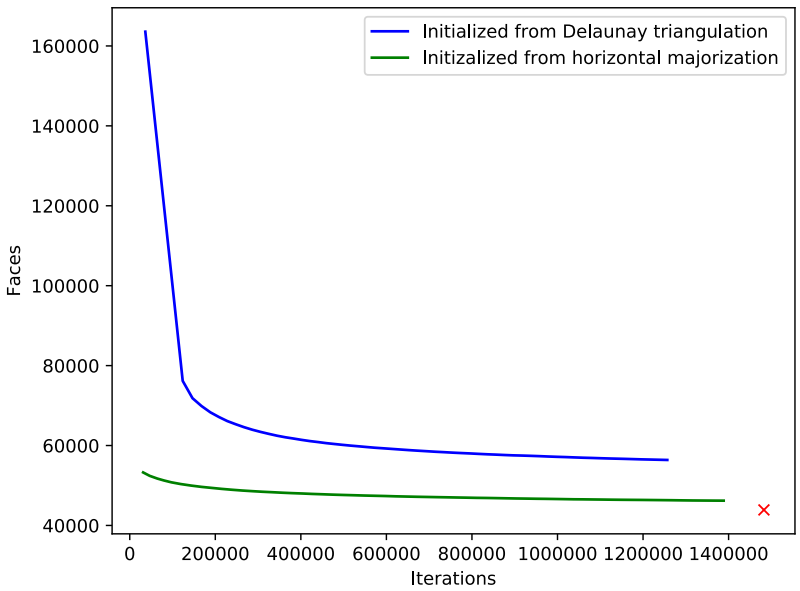


Figure 3 Performance of our algorithm with different initializations on the instance `euro-night-0100000`. Both lines depict 1 minute of runtime of our implementation. The red x-mark denotes our final score on the instance.

Uniform point sets should approximate general position point sets. For general position point sets it can be shown with Euler's characteristic that the maximum score achievable is asymptotically $\frac{1}{2}$. On large point sets, we are approaching this limit.

3.5 Performance on `rop`

We were able to achieve much better performance on `rop` instances by first horizontally or vertically majorizing the instance as initialization to the randomized local search. This took advantage of the many collinear points in the instance, as most of them had very small ranges of x and y coordinates.

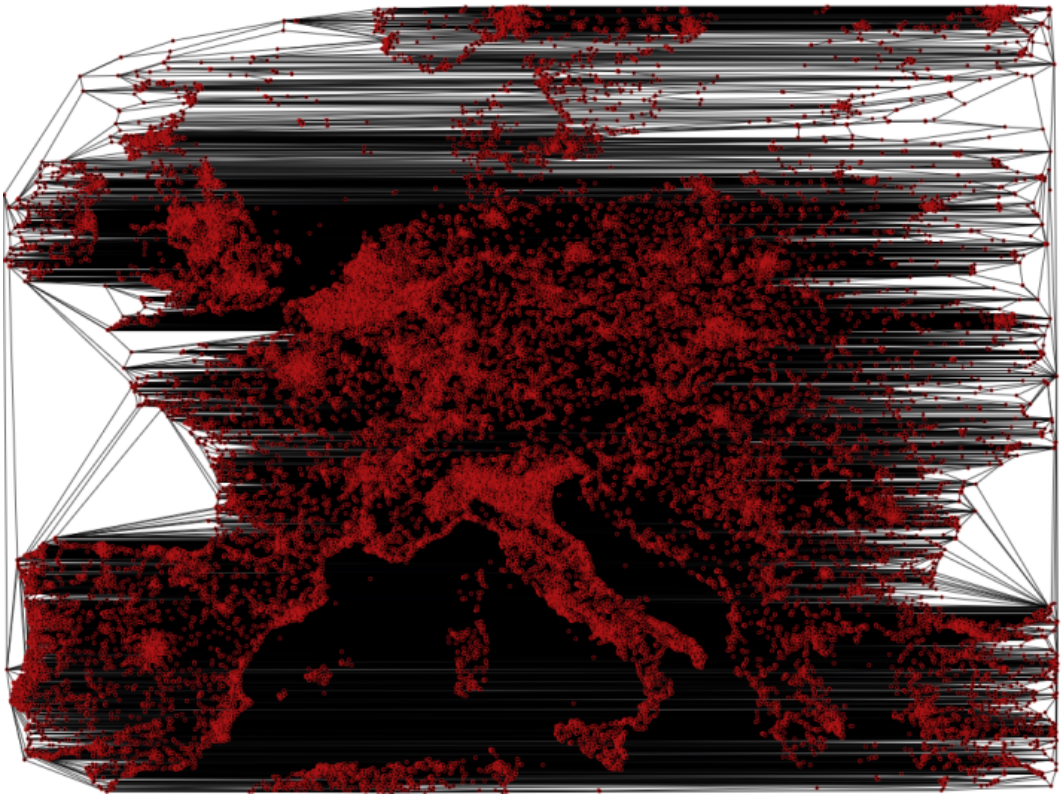


Figure 4 Our final solution to `euro-night-0040000` found by running randomized local search from horizontally majorized initialization.

3.6 Improved performance on large `euro-night` and `us-night`

We obtained better scores on large `euro-night` and `us-night` instances than on large `uniform` instances by initializing our randomized local search with horizontal majorization. This can be seen in Figure 5. This is an artifact of how the test data for the contest was created: The uniform instances have coordinates sampled from the range $[0, 6000000]$. However, the `us-night` and `euro-night` instances have coordinates in much smaller ranges: The `us-night-0100000` instance has y coordinates in the range $[0, 76956]$, and x coordinates in the range $[4, 136766]$, while the largest `euro-night` instance of 100000 points has y coordinates in the range $[0, 57598]$, and x coordinates in the range $[8, 102392]$. These instances were points sampled from a distribution based on an illumination map. As a result of the restriction to integer points, and these very small bounds for the integers, the large `us-night` and `euro-night` instances had lots of points with the same x or y coordinate, and hence lots of collinear points. By majorizing these instances, we were able to successfully take advantage of this. For example, our solution for `euro-night-0100000` has 36820 vertices with degree 2, for a total of 143883 edges and 43884 faces. We believe that with more clever methods to leverage collinear points, these solutions can improved even further.

4 Open questions

Are there other modifications of our local search strategy that can perform well, or even better? During the competition, we tried various methods of adding edges and simulated annealing based on total length of the edges in the convex partition, but we were unable to do any better with these methods.

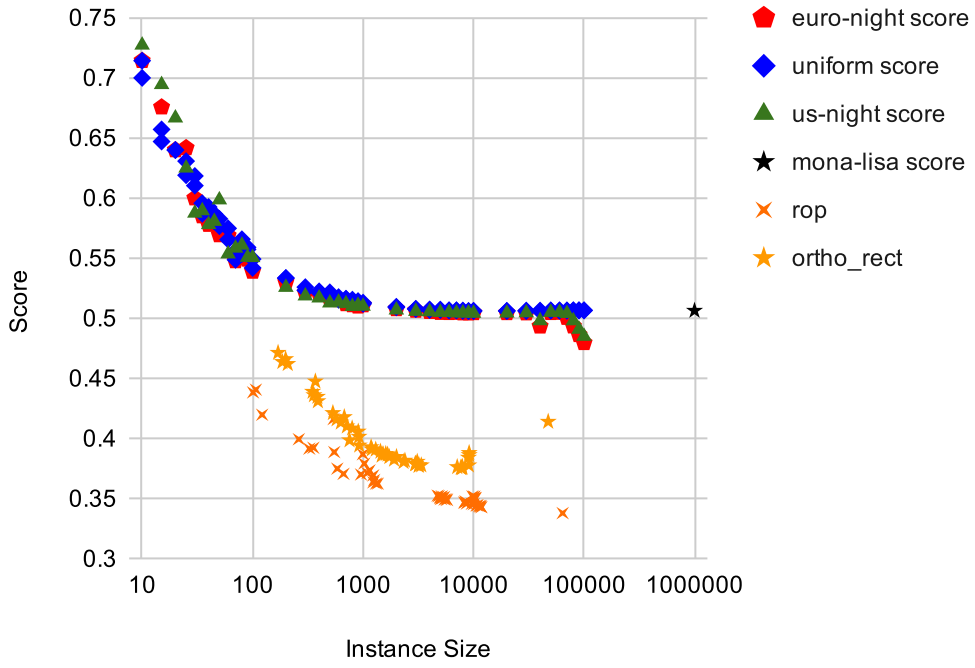


Figure 5 A plot of score vs number of points in the instance, where the score is defined as the number of edges over the number of edges in a triangulation.

Our algorithm local search strategy was very simple, yet it was able to achieve very good results on all instances. On almost uniform point sets, it was able to approach the theoretical limit of $\frac{1}{2}$. Is it possible to prove theoretical guarantees about the performance of this algorithm on random point sets or in general? We conjecture that on large random point sets, the performance of our local search algorithm approaches 0.5 in score.

References

1 A.M. Andrew. Another efficient algorithm for convex hulls in two dimensions. *Information Processing Letters*, 9(5):216–219, 1979. doi:10.1016/0020-0190(79)90072-3.

2 Fahiem Bacchus, Matti Järvisalo, and Ruben Martins, editors. *MaxSAT Evaluation 2019: Solver and Benchmark Descriptions*. Department of Computer Science Report Series B. Department of Computer Science, University of Helsinki, Finland, 2019.

3 Sam Bayless, Noah Bayless, Holger H. Hoos, and Alan J. Hu. SAT Modulo Monotonic Theories. *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, 2015.

4 Erik D. Demaine, Sándor P. Fekete, Phillip Keldenich, Joseph S. B. Mitchell, and Dominik Krupke. Computing convex partitions for point sets in the plane: The cg:shop challenge 2020, 2020. arXiv:2004.04207.

5 Nicolas Grelier. Minimum convex partition of point sets is np-hard, 2019. arXiv:1911.07697.

Computing Low-Cost Convex Partitions for Planar Point Sets Based on a Memetic Approach

Laurent Moalic
Université de Haute-Alsace, IRIMAS UR 7499, F-68100 Mulhouse, France
Université de Strasbourg, France
laurent.moalic@uha.fr

Dominique Schmitt
Université de Haute-Alsace, IRIMAS UR 7499, F-68100 Mulhouse, France
Université de Strasbourg, France
dominique.schmitt@uha.fr

Julien Lepagnot
Université de Haute-Alsace, IRIMAS UR 7499, F-68100 Mulhouse, France
Université de Strasbourg, France
julien.lepagnot@uha.fr

Julien Kritter
Université de Haute-Alsace, IRIMAS UR 7499, F-68100 Mulhouse, France
Université de Strasbourg, France
julien.kritter@uha.fr

Abstract

We present a memetic approach designed to tackle the 2020 Computational Geometry Challenge on the Minimum Convex Partition problem. It is based on a simple local search algorithm hybridized with a genetic approach. The population is brought down to its smallest possible size – only 2 individuals – for a very simple implementation. This algorithm was applied to all the instances, without any specific parameterization or adaptation.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases metaheuristics, memetic algorithms, convex partition optimization

Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.84

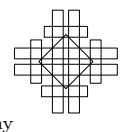
Category CG Challenge

Related Version A description of the 2020 CG:SHOP Challenge with related work and overall outcomes can be found at [2] `https://arxiv.org/abs/2004.04207`.

1 Introduction

Given a set P of points in the plane, the Minimum Convex Partition problem is that of identifying a partition of the convex hull of P into the smallest number of convex polygons whose vertices are the points of P. Finding the minimum convex partition of given instances of points with integer coordinates was the aim of the 2020 CG:SHOP Challenge [2].

It has recently be shown that the Minimum Convex Partition problem is NP-hard, when the point sets are not necessarily in general position [3]. Thus, simple local search algorithms, which are prone to be trapped in local optima, are not efficient enough and do not yield the best solutions for several instances of this problem. For these reasons, we propose to use a memetic approach, an effective class of metaheuristics commonly used to solve various combinatorial problems [5].



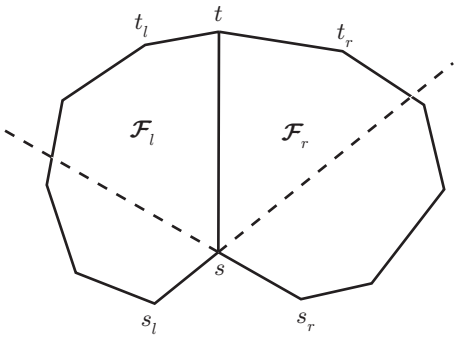


Figure 1 The edge st is rotatable at s.

2 Algorithmic Methods

2.1 A memetic approach

The main idea is to hybridize two mechanisms, a local search which intensifies the search (exploitation phase, which improves a solution by converging to a nearby local optimum), and a crossover which diversifies it (exploration phase, to escape from local optima and explore new areas of the search space). The goal is to cover all parts of the search space as well as possible and find the best local solution in each part. The idea of using only two individuals in the population was first successfully introduced in [4], for the graph coloring problem. It has the advantage of removing the selection phase, as well as the replacement strategy.

The memetic scheme starts here with two identical individuals which are the Delaunay triangulation of the point-set. In order to avoid wasting time with reparations, the individuals are kept legal, that is, convex partitions of the point-set. The fitness value of a solution is given by the number of its polygons, which we aim at minimizing.

2.2 A simple descent local search

Let P be a set of n points in the plane and let $\mathcal{P}$ be a convex partition of P, i.e., a partition of the convex hull of P into convex polygons whose vertices are the points of P.

The aim of the descent local search is to remove internal edges of $\mathcal{P}$, i.e. edges that do not belong to the boundary of the convex hull of P. Thereby, the two faces of $\mathcal{P}$ on each side of the removed edge are merged to become a single face. The descent local search never degrades the current convex partition.

Let st be an internal edge of $\mathcal{P}$ and let $\mathcal{F}_l$ and $\mathcal{F}_r$ be the faces of $\mathcal{P}$ that share the edge st, and that are respectively on the left side and on the right side of st (see Figure 1). Let s_l (resp., t_l) be the vertex of $\mathcal{F}_l$ that precedes s (resp., succeeds t) on the boundary of $\mathcal{F}_l$ in counterclockwise direction. Let s_r (resp., t_r) be the vertex of $\mathcal{F}_r$ that succeeds s (resp., precedes t) on the boundary of $\mathcal{F}_r$ in counterclockwise direction. Consider the two following conditions:

1. The point s_r is on the left side of or on the oriented straight line $(s_l s)$.
2. The point t_l is on the left side of or on the oriented straight line $(t_r t)$.

The edge st is said to be *immovable*, *rotatable*, or *removable* respectively, when 0, 1, or 2 of the conditions 1 and 2 are satisfied. When st is rotatable, we say that st is rotatable at t if condition 1 is satisfied, and we say that st is rotatable at s if condition 2 is satisfied.

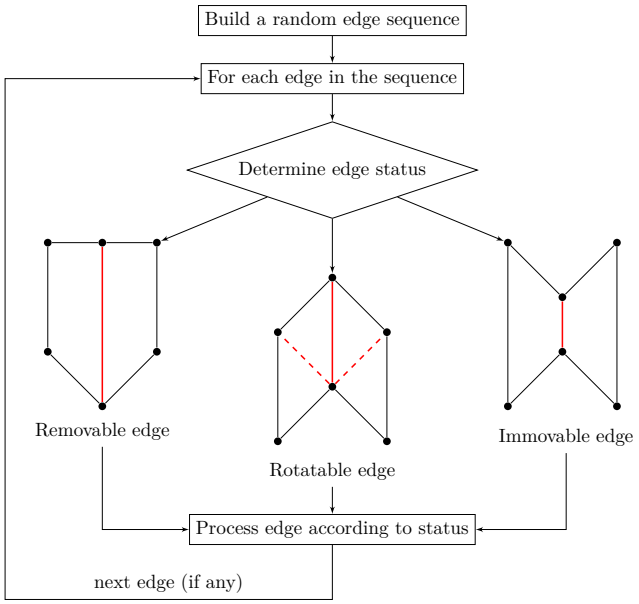


Figure 2 One step of the local search: a remove, move, unmove approach.

Let $\mathcal{F}$ be the face obtained by removing st and by merging $\mathcal{F}_l$ and $\mathcal{F}_r$. If the edge st is removable, $\mathcal{F}$ is obviously convex. If st is immovable, $\mathcal{F}$ is not convex, neither in s, nor in t. The only way to cut $\mathcal{F}$ into two convex faces is to put back the edge st.

If st is rotatable, suppose, without loss of generality, that it is rotatable at s. In this case, $\mathcal{F}$ is not convex in s but is convex in all other vertices. To cut $\mathcal{F}$ into two convex faces, we have to add an edge st' that connects s to a vertex t' of $\mathcal{F}$ other than s, s_l, and s_r. Whatever the choice of t', the two new faces are convex in all their vertices, except possibly in s. To ensure the convexity in s, t' must be chosen altogether on the left side of or on (ss_r), and on the left side of or on $(s_l s)$. The vertices t' that satisfy these conditions are called the *valid positions* for t (relatively to st in $\mathcal{P}$). The valid positions for t form a connected polyline on the boundary of $\mathcal{F}$. This polyline contains necessarily t and may be reduced to t.

The descent local search uses the above properties to improve the current convex partition $\mathcal{P}$. It proceeds by steps. In each step, a random sequence of internal edges of $\mathcal{P}$ is generated (see Figure 2). For every edge st in the sequence:

- if st is removable, it is removed from $\mathcal{P}$,
- if st is rotatable at s, the set of valid positions for t is generated, a point r in the set is randomly chosen, and st is replaced by sr (the processing is symetric if st is rotatable at t).

Clearly, every removable edge produces an improvement of the current convex partition. Rotatable edges give rise neither to improvements nor to degradations, but by moving a rotatable edge, another edge may become removable and may therefore be removed later on in the process.

This process is repeated until no more improvement seems to be possible, i.e. until the current convex partition is trapped in a local optimum. For all instances of the competition, the number of steps was fixed to 10,000.

Let us now consider the complexity of one step. The generation of a random sequence of edges, as well as the processing of all removable and immovable edges are linear in the number of edges. Thus, the complexity will be determined by the processing of the rotatable

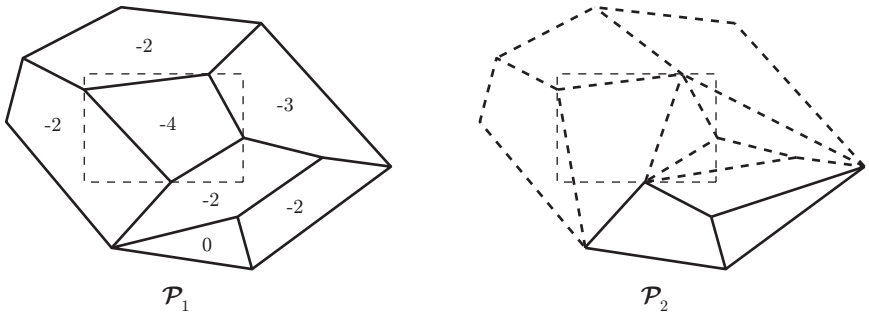


Figure 3 The numbers in the faces of $\mathcal{P}_1$ are the scores of theses faces with respect to $\mathcal{P}_2$. If the best face of $\mathcal{P}_1$ is transmitted to a child, then the faces of $\mathcal{P}_2$ with full edges are the only ones that can still be transmitted to that child.

edges. For each edge st rotatable at, say, s, all vertices of the two faces on both sides of st may be valid positions for t, except s and its two neighbors s_l and s_r. This leads to a complexity of $O(n)$ per edge, and thus to an overall complexity of $O(n^2)$ for one step of the algorithm.

In practice, this complexity is much lower. Consider, for example, the instances of the competition with 100,000 points. The largest set of valid positions encountered over 10 runs for each of these instances contained only 16 points. The average size of the set was about 2.64 points. The sets are larger for instances with large numbers of collinear points. For the instance "rop" with 64054 points, a set of 305 valid positions was found. The average size was 7.56 points.

2.3 A crossover which gets the best of the parents

Diversification starts with two given convex partitions $\mathcal{P}_1$ and $\mathcal{P}_2$ of P - the parents - generated by the descent local search. The crossover aims at getting part of each parent's "gene pool" to produce two new convex partitions of P - the children. To generate a child, the idea is to get some non-overlapping faces from each parent. The child is then, at first, a partial solution made of convex polygons and isolated points.

A "good" child is typically one which gathers "good" faces from its parents. Clearly, the optimal convex partition of P minimizes the sum of its vertices' degrees. We therefore compute a score for every face of each partition, which measures the attractiveness of the face relatively to the other partition. The score of a face of, say, $\mathcal{P}_1$ is obtained by summing up the degrees of its vertices and by subtracting the degrees of these same vertices in $\mathcal{P}_2$ (see Figure 3). Roughly speaking, if the score is negative, the environment of the face is better in $\mathcal{P}_1$ than it is in $\mathcal{P}_2$. The best face is the one with lowest score.

The crossover algorithm first sorts the faces of $\mathcal{P}_1$ and $\mathcal{P}_2$ independently by increasing scores. Then, it transmits alternatively one face from $\mathcal{P}_1$ and one face from $\mathcal{P}_2$ to one child, in order. Ties are broken randomly. This stochasticity helps to generate two different children. It is enhanced by the fact that the first face transmitted to the first child comes from $\mathcal{P}_1$, while the first face transmitted to the second child comes from $\mathcal{P}_2$.

So that the faces transmitted to a same child do not intersect, intersection tests between the faces in $\mathcal{P}_1$ and $\mathcal{P}_2$ have to be performed. To accelerate the intersection test when a face in, say, $\mathcal{P}_1$ is transmitted to a child, the axis-parallel bounding box of the face is computed. All faces in $\mathcal{P}_2$ which intersect the box are disabled, so that they cannot be transmitted to

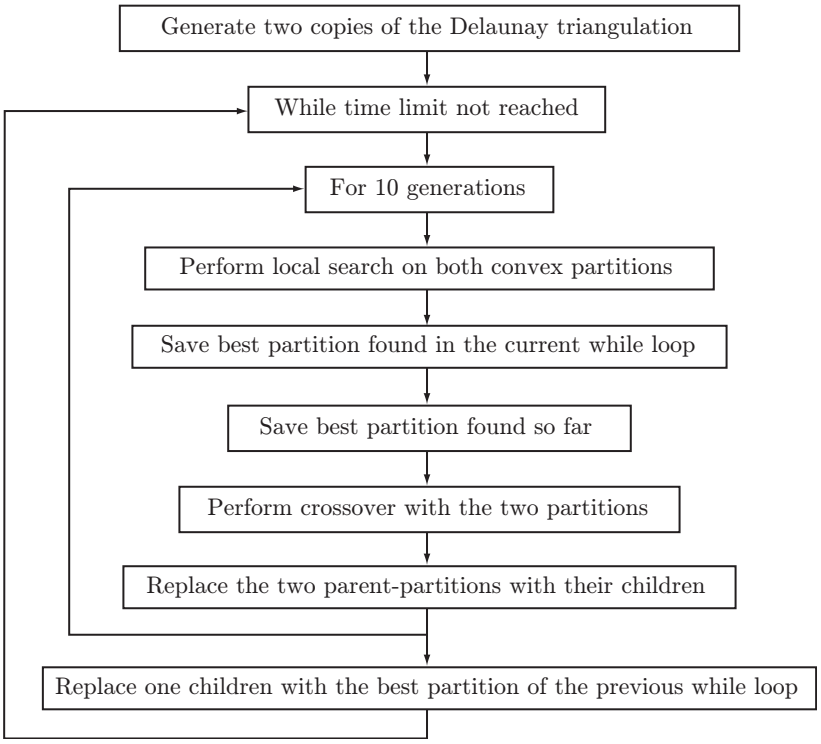


Figure 4 General scheme of the memetic approach.

that child. When no more faces in $\mathcal{P}_1$ and no more faces in $\mathcal{P}_2$ can be transmitted to a child, a constrained triangulation of P and of the set of transmitted faces is computed. The two child-partitions constructed that way replace their parents and are used in the next iteration of the algorithm (see Figure 4).

The algorithm alternates between the local search and the crossover phases until a time limit is reached. As the local search is a simple descent, the best encountered solution is recorded just before each crossover phase.

3 Practical Computation

3.1 Computational Environment

The program was written in C++ using data structures and functions from the CGAL library [1]. Several experiments were carried out on the Strasbourg high-performance computing cluster (HPC) using identical machines equipped with 2.6GHz Intel Xeon Gold 6126 CPUs. Some statistics on the results obtained by 10 executions of our algorithm are presented and discussed. To obtain these statistics, each execution was stopped after one hour. Note that the results submitted for the challenge were obtained without any information on running time. However, it turns out that, for 80% of the instances, the algorithm achieved the value submitted to the challenge in less than one hour.

3.2 Algorithm Engineering

It seems interesting to note that one of the strengths of the proposed approach is that there are no instance-specific settings. The same program is applied to all instances, regardless of their size or structure.

The only parameters are the local search duration between two crossovers, and the number of generations in one cycle. We have set the local search duration to 10,000 for all instances. That is to say that for each generation, each edge can move or be removed 10,000 times. The size of a cycle is set to 10 for all instances. That is, after 10 generations the best solution of the previous cycle is reintroduced. These values have been determined experimentally, and can be improved for a better behavior of the algorithm.

3.3 Experimental Results

For each instance of the problem, let p_{impr} be the improvement between an intermediary solution and a final solution (in percent). It is computed using the best fitness among the ones found by the first descent local search over the 10 runs (f_{first}), and the best fitness at the end of the one-hour execution over the 10 runs (f_{end}). It is given by $\frac{100\ (f_{first}-f_{end})}{f_{first}}$. For each class of instances, the evolution of this value is presented in Figure 5, over the number of points in the instance on which the algorithm is applied.

A similar percentage of improvement, denoted by p_{comp}, is computed between f_{end} and the fitness of the solution submitted for the competition.

One can see in Figure 5 that three different behaviors are adopted by the algorithm, depending on the class of instances on which it is applied:

1. On the class of "rop" instances, the values of p_{impr} are globally significantly greater than for any other class of instances. Hence, the exploration phase using a genetic crossover appears to be very useful for the "rop" instances. Among these instances, the one having the highest value of p_{comp} is "rop0010050", for which $p_{comp} = 25.74$.
2. On the "ortho_rect_union" instances, the opposite behavior is displayed by the algorithm, i.e. the values of p_{impr} are globally significantly lower than for the other classes. Let std_{first} be the standard deviation of the solutions found by the first descent for the 10 executions of our approach on a given instance. Among the "ortho_rect_union" instances, the one having the highest value of std_{first} is "ortho_rect_union_47381", for which $std_{first} = 4.93$. It is the lowest value of std_{first} compared to 50,000 and even 40,000 point instances of all other classes. Furthermore, among these instances, the one having the highest value of p_{comp} is "ortho_rect_union_7663", for which $p_{comp} = 0.10$. It could mean that a simple local search is sufficient to find good solutions for these instances.
3. On the other classes of instances except "mona-lisa", our approach appears to behave similarly. The values of p_{impr} are globally significantly greater than the ones of "ortho_rect_union" instances, but significantly lower than the ones of "rop" instances. Among the instances of all classes except the "ortho_rect_union", the "rop" and the "mona-lisa" ones, the one having the highest value of p_{comp} is "uniform-0090000-1", for which $p_{comp} = 0.13$.

For "mona-lisa", containing only one instance of 1,000,000 points, it is not possible to observe how p_{impr} would evolve over the number of points. However, p_{comp} equals 0.65, which is low compared to most "rop" instances. One can also notice significant fitness differences between instances belonging to the same class, which could be due either to a stability issue of the approach or to the nature of the instances.

The analyses presented in Figures 6, 7, and 8 are based on one of the 10 runs that leads to the median performance. It shows the evolution of the number of faces of the best solution found so far over the generations. These 3 figures correspond to a one hour run.

One can see that our approach is able to converge to a good solution in few generations.

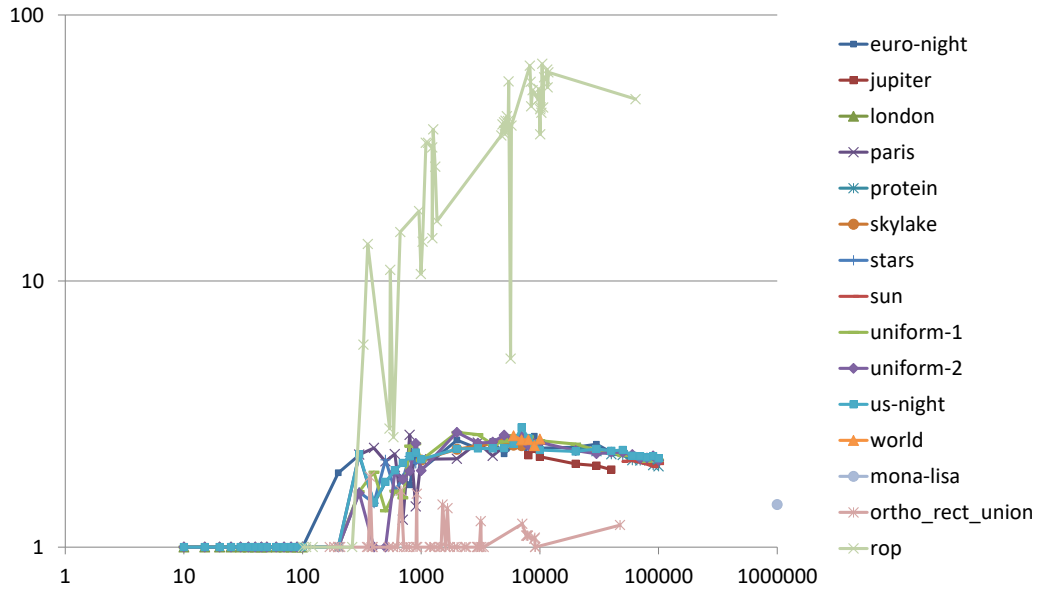


Figure 5 Evolution of $p_{impr} + 1$ (in ordinate) depending on the number of points of the instance (in abscissa). A logarithmic scale is used on both axes for clarity.

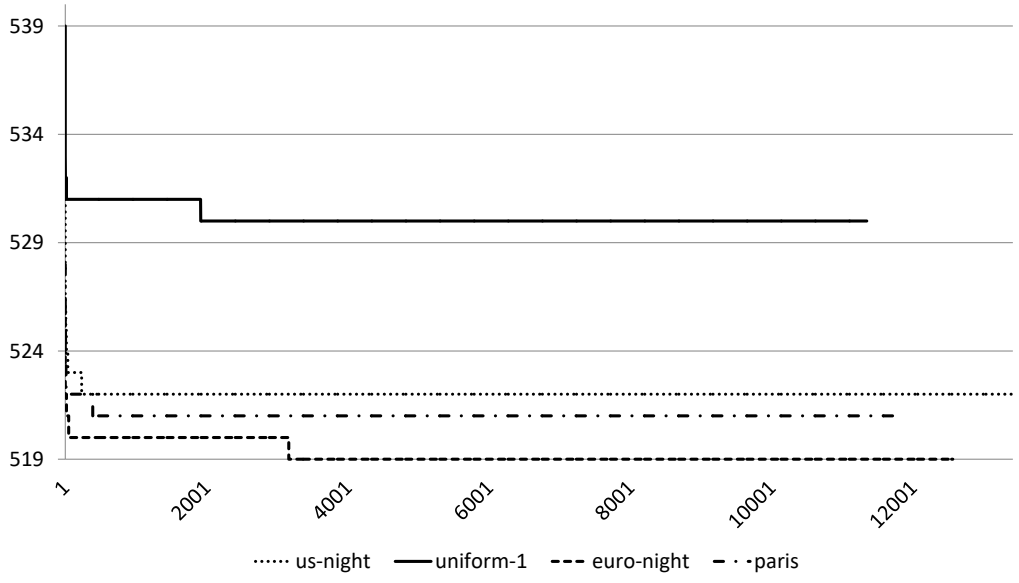


Figure 6 Convergence curve of the algorithm for 1,000 point instances over generations.

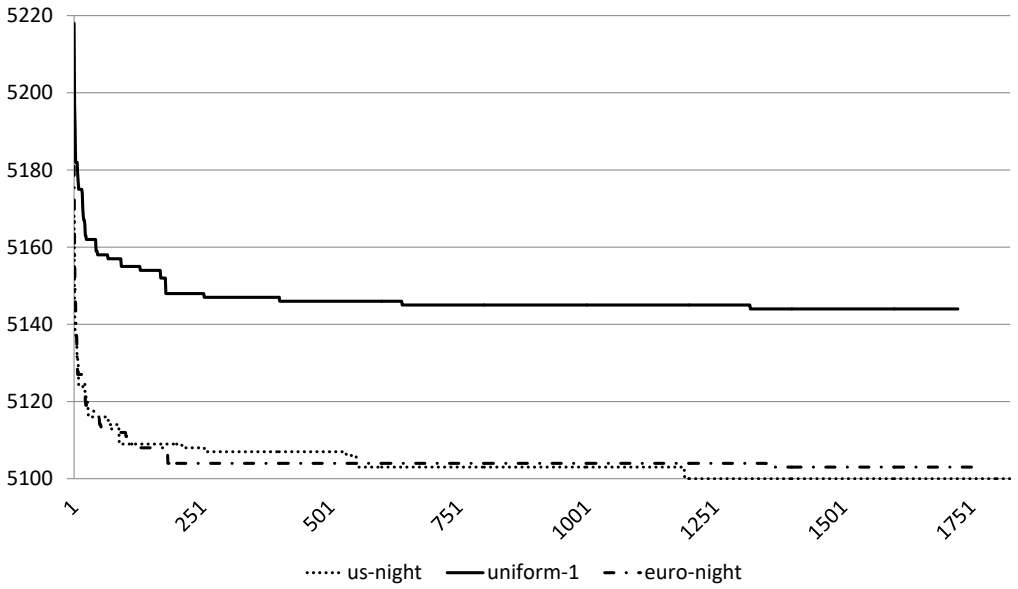


Figure 7 Convergence curve of the algorithm for 10,000 point instances over generations.

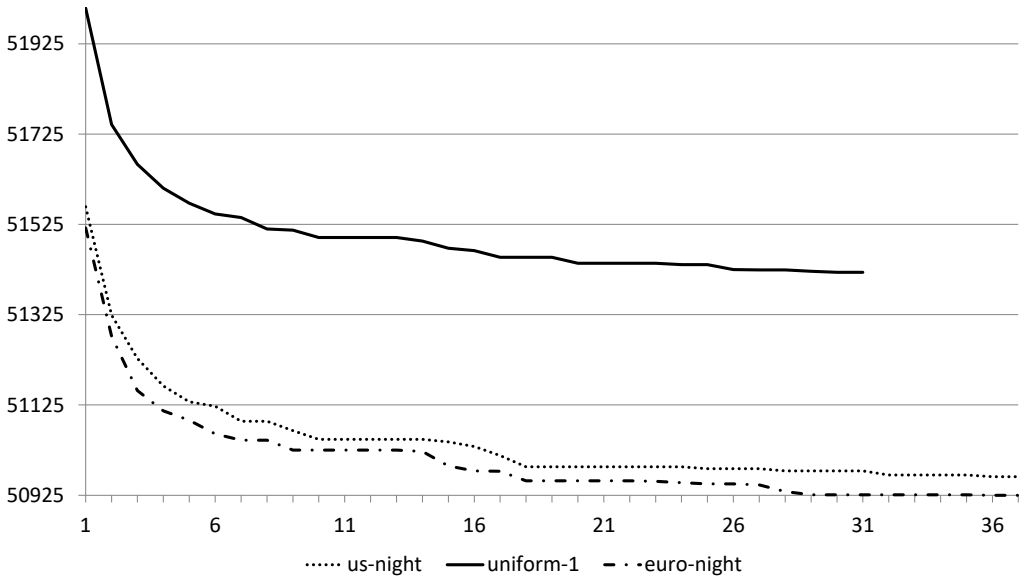


Figure 8 Convergence curve of the algorithm for 100,000 point instances over generations.

4 Conclusion

The proposed memetic algorithm has proven its overall effectiveness by ranking second among the best algorithms competing at the 2020 Computational Geometry Challenge on the Minimum Convex Partition problem. As pointed out in the analysis of section 3.3, significant fitness differences are observed between instances, belonging to the same class or not. In spite of this, the proposed algorithm does not have instance-specific parameter settings. These differences between instances should be studied, as well as the stability of the algorithm on each class of instances, to lead to an improved variant of our approach.

References

1 CGAL, Computational Geometry Algorithms Library. http://www.cgal.org.

2 Erik D. Demaine, Sándor P. Fekete, Phillip Keldenich, Dominik Krupke, and Joseph S. B. Mitchell. Computing convex partitions for point sets in the plane: The cg:shop challenge 2020, 2020. arXiv:2004.04207.

3 Nicolas Grelier. Minimum convex partition of point sets is NP-hard, 2019. arXiv:1911.07697.

4 Laurent Moalic and Alexandre Gondran. Variations on memetic algorithms for graph coloring problems. *Journal of Heuristics*, 24(1):1–24, 2018. doi:10.1007/s10732-017-9354-9.

5 Pablo Moscato and Carlos Cotta. A Gentle Introduction to Memetic Algorithms. In F. Glover and G. A. Kochenberger, editors, *Handbook of Metaheuristics*, pages 105–144. Springer, 2003.

Computing Low-Cost Convex Partitions for Planar Point Sets Based on Tailored Decompositions

Günther Eder
Universität Salzburg, FB Computerwissenschaften, Austria
geder@cs.sbg.ac.at

Martin Held
Universität Salzburg, FB Computerwissenschaften, Austria
held@cs.sbg.ac.at

Stefan de Lorenzo
Universität Salzburg, FB Computerwissenschaften, Austria
slorenzo@cs.sbg.ac.at

Peter Palfrader
Universität Salzburg, FB Computerwissenschaften, Austria
palfrader@cs.sbg.ac.at

Abstract

Our work on minimum convex decompositions is based on two key components: (1) different strategies for computing initial decompositions, partly adapted to the characteristics of the input data, and (2) local optimizations for reducing the number of convex faces of a decomposition. We discuss our main heuristics and show how they helped to reduce the face count.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Computational Geometry, geometric optimization, algorithm engineering, convex decomposition

Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.85

Category CG Challenge

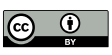
Supplementary Material The source code of our tools and heuristics is available at GitHub and can be used freely under the GPL(v3) license: `https://github.com/cgalab`.

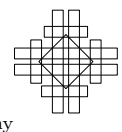
Funding Work supported by Austrian Science Fund (FWF): Grants ORD 53-VO and P31013-N31.

1 Introduction

The task of the 2020 Computational Geometry Challenge – called Challenge in the sequel for the sake of brevity – was to compute minimum convex decompositions (MCD) of point sets in the plane. We refer to the survey by Demaine et al. [2] for background information.

We employed several tools and heuristics to tackle the Challenge. All tools submitted their solutions to a central database of ours, such that tool A could query and then improve on solutions obtained by tool B, and vice versa. Most of our heuristics are based on local search: Begin with a convex decomposition and iteratively modify it locally to reduce the number of convex faces. The source code of our tools and heuristics is available at GitHub and can be used freely under the GPL(v3) license: `https://github.com/cgalab`.





2 Algorithmic methods

2.1 3-Approximation

Our tool 3Apx implements the 3-approximation algorithm by Knauer and Spillner [3]. Tests quickly showed that this approach generates decompositions that are clearly not optimal. Hence, we extended 3Apx by an approach based on onion layers [1]: We construct all onion layers and then find convex decompositions of the annuli between the layers. Contrary to [3], this approach does not modify the layer boundaries. See Figure 1 for sample decompositions obtained via 3-approximation and the onion layers. Experiments showed that computing a convex decomposition based on onion layers is superior to the 3-approximation algorithm even without merging convex faces across different onion layers, see the plot in Figure 7.

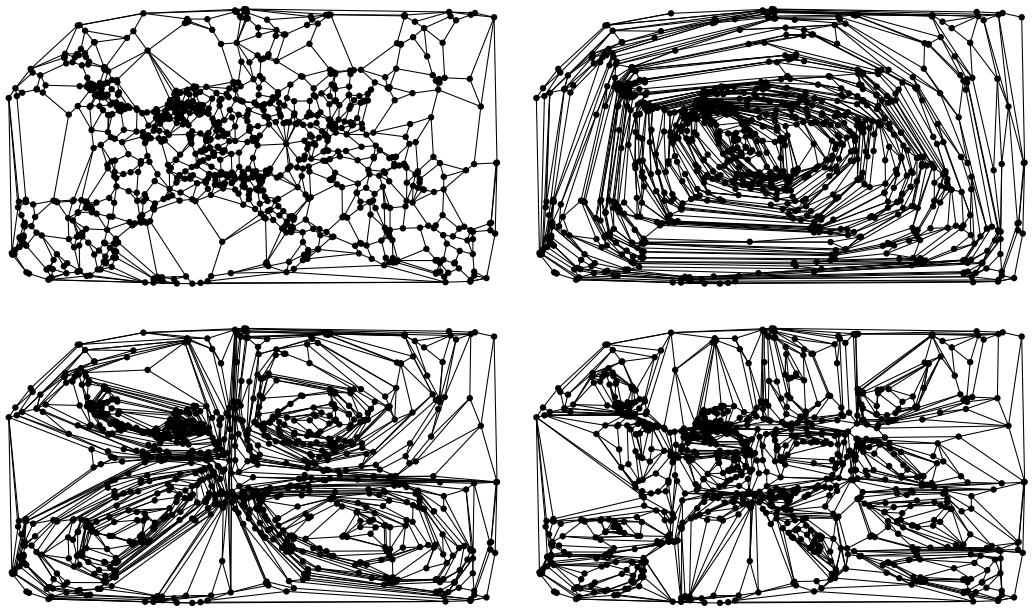


Figure 1 In reading order: When using the 3-approximation implemented in 3Apx for an instance with 1000 vertices we obtain a convex decomposition with 1350 faces; 1125 faces when using our approach based on onion layers without partitioning into cells; 1123 faces when partitioning into four cells and subsequent onion-layer based decomposition; and 1148 faces when using 16 cells.

A visual inspections of the results achieved by 3Apx quickly made it apparent that the decompositions computed contained lots of extremely long and thin triangles. Hence, we tried to partition a given input P into smaller "cells" and then run 3Apx on each cell individually. Then the individual decompositions are joined by triangulating the area between them and randomly dropping triangulation edges if this is possible without violating convexity. This produced visually nicer images such as the last two decompositions in Figure 1 but did not reduce the face counts substantially.

2.2 Merging faces

One of our earliest ideas was to do the obvious: Start with a triangulation of P and then merge adjacent faces by randomly dropping triangulation edges as long as faces remain convex. Tests with an initial straightforward implementation, MergeRefine, suggested that this is a promising approach, easily beating 3Apx (Figure 7). In order to be better prepared for refined heuristics we quickly re-implemented it in a new tool called Recursor. In particular, we resorted to a more advanced data structure for storing our decompositions.

RECURSOR keeps its state in a variant of a doubly-connected edge list (DCEL) or half-edge data structure. The base layer of our variant is a DCEL of a triangulation of P. Additionally, each half-edge pair is considered either *constrained* or not constrained, depending on whether the edge is part of our convex decomposition of P. As a layer on top of the base DCEL, each constrained half-edge, in addition to the pointers to the next triangulation edges encountered in clockwise (CW) or counter-clockwise (CCW) direction, also holds a reference to next CW and CCW constrained edges; cf. Figure 2. This enables constant-time testing whether an edge can be dropped, i.e., marked unconstrained, while keeping a fully fledged triangulation of P during the entire process. To obtain a decomposition, RECURSOR first uses Shewchuk's TRIANGLE [4] to construct a Delaunay triangulation of P, and then iterates over the edges in a random order, dropping every edge that can be dropped. This process continues until no further edge can be dropped, i.e., the decomposition is locally optimal.

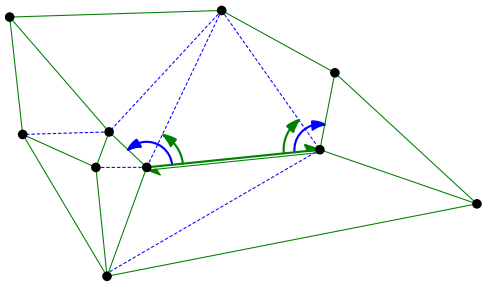


Figure 2 Our two-layer doubly-connected edge list stores two planar graphs simultaneously, with one planar graph being a subgraph of the other. A constrained half-edge h has references to its neighbors in the convex decomposition (green) and to the underlying triangulation (blue). The Challenge data set `euro-night-10` is shown.

Hole refinement. It is not surprising that a locally optimal decomposition may consist of many more faces than the true global optimum. Therefore, we worked on strategies that allow us to move away from local optima: RECURSOR picks a face f of the decomposition and a (random) number of its neighbors as a "hole" to work on. In general, it picks a face f that is incident to a high-degree vertex. We consider a vertex of the decomposition to be of high degree if its degree is larger than 3 or if its degree is equal to 3 and two incident edges span an angle of 180°. In other words, high-degree vertices are vertices whose degree could (locally) be reduced without violating convexity.

Once a hole has been selected, RECURSOR marks all its triangulation edges as constrained again. In the next step it tries to drop these edges in a (different) random order. If this results in a decomposition with no more faces than previously, we keep the new decomposition. Otherwise, we abandon the modifications and restore the old decomposition. See Figure 3 for a sample modification of a decomposition for the Challenge data set `euro-night-0000100`.

RECURSOR has several parameters to adjust, and we tried to fine-tune them "on the fly" as we applied it to the Challenge instances. Eventually we settled on hole sizes of $7 + P$ faces where P is a random number drawn from a geometric distribution with $p = 0.4$. In each hole, we try a number of decompositions that is equal to the number of triangulation edges in that hole.

Edge flips. Our initial decompositions were based on Delaunay triangulations of the input points. But there is no argument to justify why Delaunay edges were to be preferred over other triangulation edges. Hence, the next improvement of RECURSOR does a number of

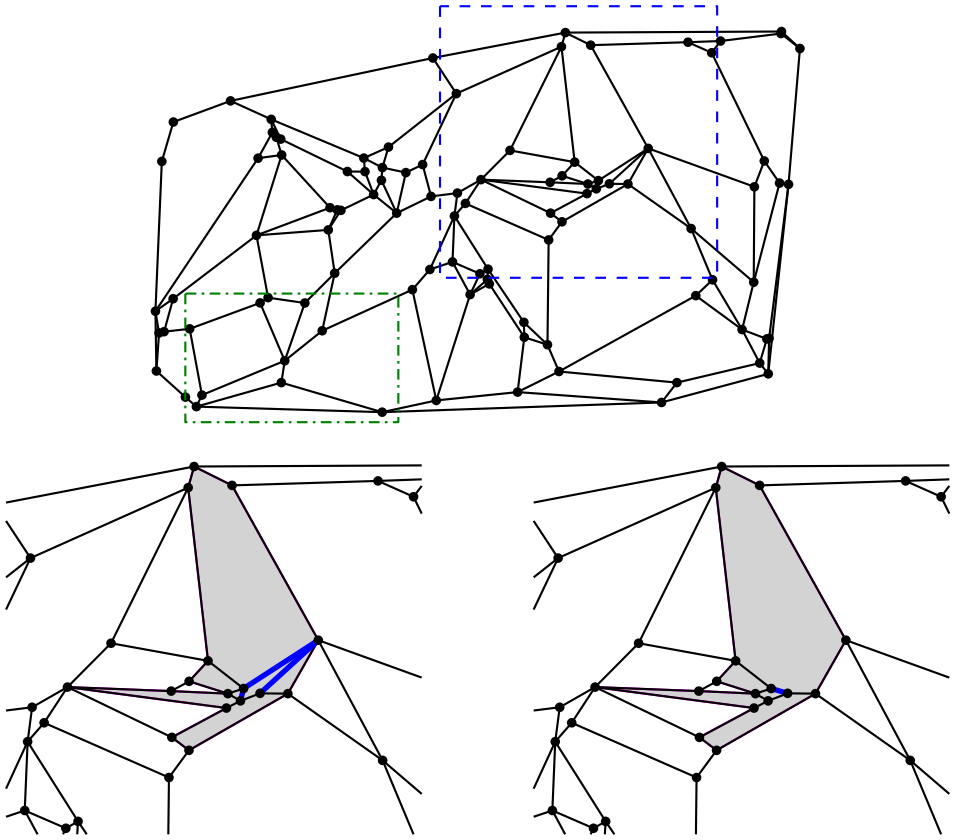


Figure 3 Top: An initial decomposition of `euro-night-0000100` by RECURSOR. Bottom: A detail of the initial decomposition (of the dashed blue frame in the full image), with those seven faces shaded in gray that were selected by the hole-refinement algorithm. The decomposition after one round of local optimization is shown on the right. The edges affected are shown in blue and bold. The improved variant has two faces less.

random edge flips on the triangulation of a hole before attempting to drop edges. The number of edge flips used by our code changed over time. After a series of quick experiments we ended up with using roughly $\sqrt[5]{t}$ edge flips, where t is the number of triangles in the hole.

Continuous refinement. So far, each run of RECURSOR always started from a triangulation of an input. We modified RECURSOR such that it could load a previous decomposition and work on it. This allowed us to run it on different instances whenever we had computational resources to spare, with no need to run it for long continuous stretches of time.

Parallel recursor. RECURSESPLIT is a wrapper around RECURSOR that partitions a given decomposition into a few dozen or a hundred non-overlapping sets of contiguous faces such that each set of faces forms a simply-connected region. Each such region is handed to a dedicated instance of RECURSOR which attempts to reduce the face count within that region. Note that RECURSOR does not require such a region to be convex. Since every individual run of RECURSOR operates strictly within its own region, the resulting decompositions can be merged trivially upon the completion of all runs of RECURSOR.

2.3 Flipper

Flipper was implemented relatively late during the time of the Challenge, not even a month prior to its end. It picks a point set and loads our currently best decomposition for that point set. Then it performs the following steps repeatedly: First, Flipper picks a high-degree vertex v and finds, if one exists, an incident edge (u, v) that can be rotated away from v without violating convexity at either u or v. That is, if u_0, u, u_1 is a CCW ordering of the vertices that share a decomposition edge with v then Flipper attempts to replace the edge (u, v) by either (u, u_0) or (u, u_1) if permissible. See the green edge in Figure 4, left. As shown in Figure 4, right, such a rotation may cause one of the edges incident at v or u to become unnecessary. In that case, we drop it. If, however, no edge can be removed, then the degree of v has decreased and the degree of either u_0 or u_1 has increased by one. Flipper then applies this process to u_0 or u_1.

Variations, added even later, try to pick a specific input point p at regular intervals. Then, with some probability, a rotation may only be carried out if the vertex whose degree is increased by one gets closer to p. The motivation for this decision was that finding edges that can be dropped gets easier if several vertices with higher degree are in close proximity.

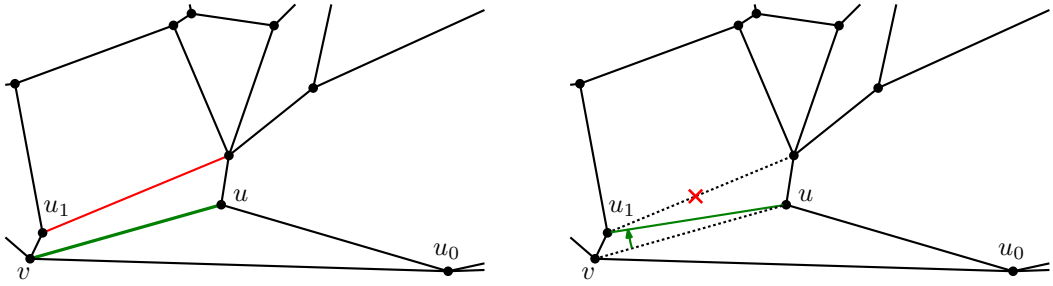


Figure 4 A detail of the initial decomposition (within the dash-dotted green frame of Figure 3): Rotating the green edge allows to drop the red edge while maintaining the convexity of all faces.

2.4 Orthogonal optimizer

Towards the end of the Challenge, a second batch of input instances was made available. While the organizers had warned a priori that the inputs may contain collinear points, the first batch of inputs contained relatively few subsets of collinear points per instance. In contrast, in the second batch of data, each input instance contained points sampled from a dense integer grid, resulting in every input instance containing many subsets of collinear points aligned along horizontal and vertical lines

A visual inspection quickly revealed that the approaches implemented so far did not generate decent decompositions for several inputs of the second batch. Therefore, we were forced to devise and implement a new heuristic. OrthoOpt generates initial convex decompositions geared towards this new type of input instances. It proceeds as follows: First, it sorts the input points of P lexicographically. Then it connects input points that share the same x-coordinate in order of increasing y-coordinates. Finally, it constructs a bottom bounding chain B and a top bounding chain T by linking the bottom-most (top-most, resp.) input points, and it triangulates all pockets between the convex hull of P and the current decomposition, as bounded by B and T. Of course, OrthoOpt can also proceed relative to y-coordinates rather than x-coordinates; see Figure 5. These initial decompositions were

passed to FLIPPER and RECURSOR for further optimization. In particular, these tools helped to get rid of unnecessary triangulation edges inside of the pockets formed by the convex hull of P and the two chains B and T.

3 Practical computation

3.1 Computational environment

Our tools were run on a diverse set of computers operated by our lab as well as by other groups at the University of Salzburg. We used a varying number of standard PCs plus some (rather small) compute servers, whenever a machine was available. (We did not have access to a genuine high-performance computer.) In particular, we used our own desktop machines whenever they were (partially) idle. One of them, an Intel Core i7-6700 CPU clocked at 3.40 GHz, was used to obtain the performance plot of Figure 6, which shows CPU-time consumptions of several of our tools for Challenge instances with different numbers of points.

Our low-profile way of accessing computers resulted in a highly non-uniform consumption of computational resources, which in turn had highly non-uniform performance levels, ranging from 15-year-old compute servers to machines acquired just a year ago. The availability of a particular machine or of some of its cores was discussed with the operator of that machine on a day-by-day or week-by-week basis. We set up a database and engineered some scripts that allowed all machines to fetch problem instances from and send results back to a home base.

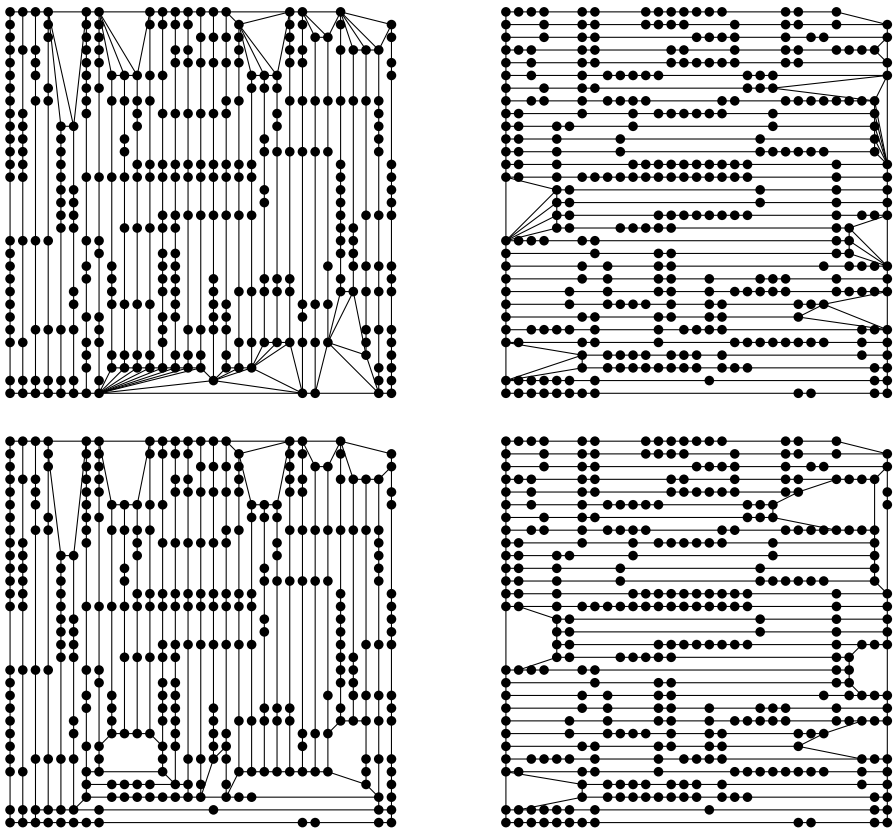


Figure 5 The two top figures show initial decompositions generated by ORTHOOPT for the 355-vertex instance `rop0000355`. The bottom left figure shows the best decomposition (with 44 faces) derived from an initial triangulation of `rop0000355`. The bottom right figure shows our overall best decomposition (with 36 faces) derived from an initial decomposition generated by ORTHOOPT.

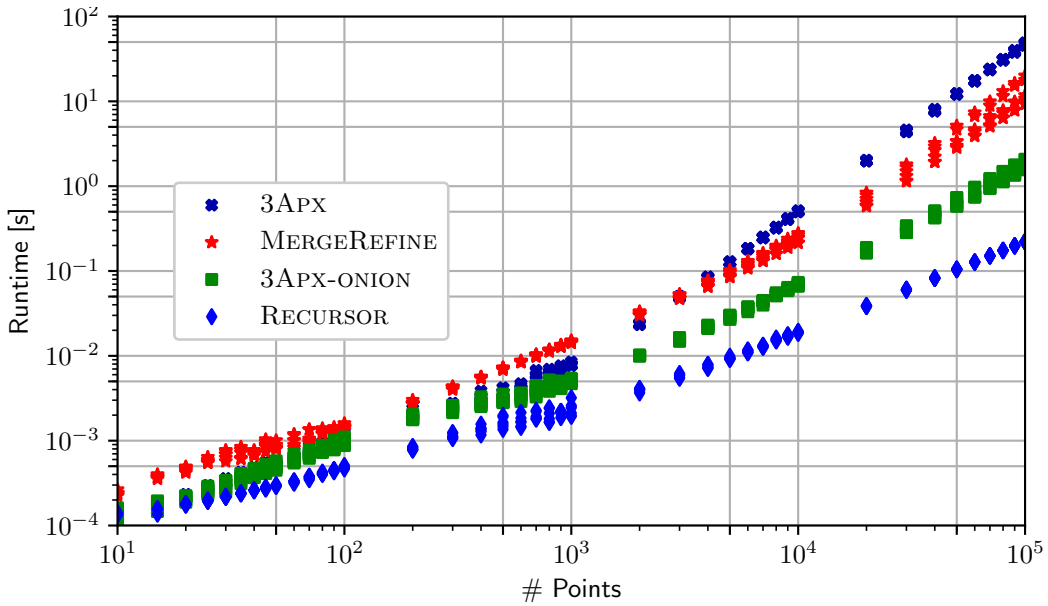


Figure 6 Time needed to obtain one initial decomposition for the competition inputs.

The heterogeneity of (our use of) the computational resources makes it very difficult to come up with a reliable ball-park figure of the total CPU time consumed. We estimate, though, that our tools would have kept a standard desktop machine busy for a few years.

3.2 Experimental results

The estimated quality of a specific convex decomposition is based upon its *score*, where

$$\text{score} := \frac{\text{number of edges in convex partition}}{\text{number of edges in triangulation}}.$$

Figure 7 plots the score for the Challenge instance `euro-night-0100000` over time. It reflects the improvements achieved by refining our tools. While we did not generate such a plot for each and every instance, we did compare sample plots for a few instances: No significant differences were observed. That is, the plot shown in Figure 7 can be regarded as representative for the progress that we made on the Challenge instances of the first batch. The plot shows nicely how Recursor and Flipper interacted. Near the end of the competition, Recursor and Flipper by themselves rarely found better decompositions. However, even when a tool did not reduce the total number of faces, it still restructured the decomposition and uploaded it to our central server, which in turn may have enabled another tool to find some small improvement. The plot also indicates that each new tool yielded a substantial improvement at the beginning, with the gains tapering off as time progressed. So, likely, investing drastically more computational resources than what we had at our disposal would have hardly led to truly substantial improvements. In our case, the availability of human resources for devising and implementing new tools was the decisive limiting factor.

The second batch of Challenge instances made it apparent that our heuristics had been (implicitly) geared towards the inputs that they had to handle. The `rop*` input class proved to be particularly challenging for our initial strategy. Therefore, we introduced OrthoOpt

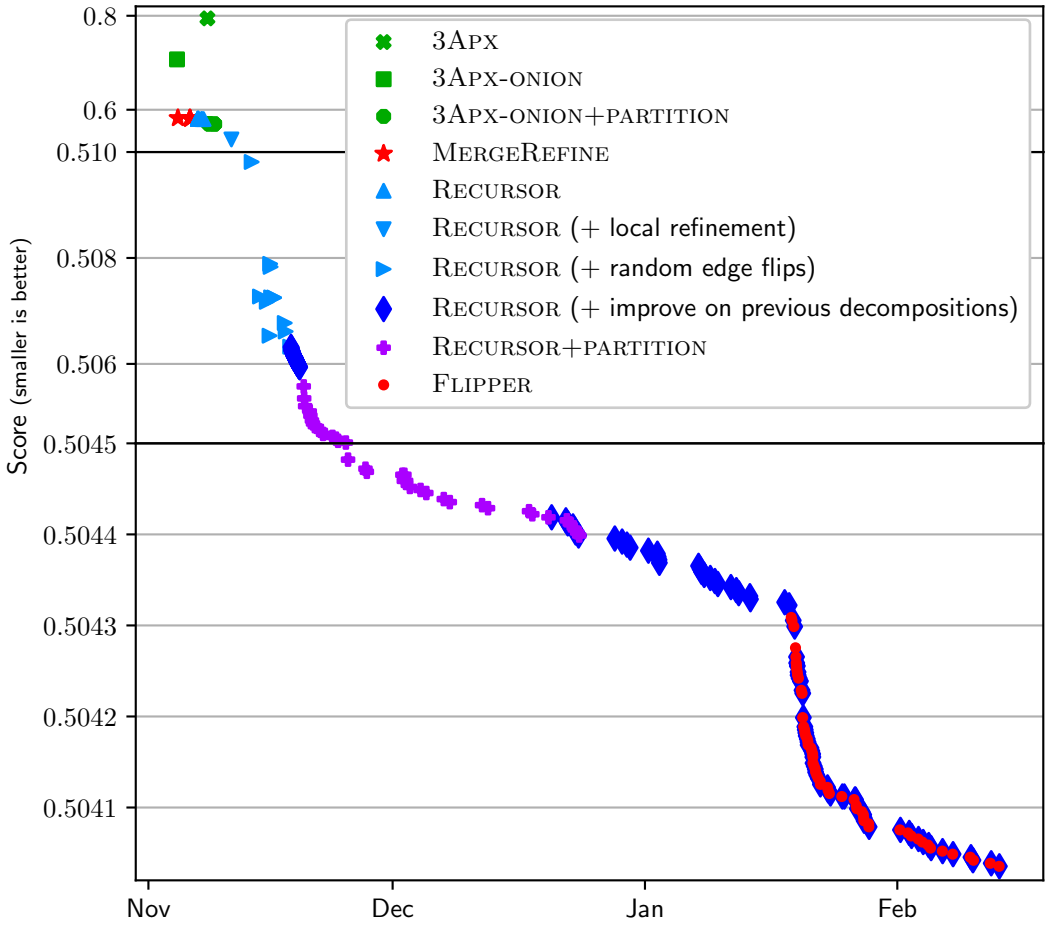


Figure 7 Score over time for `euro-night-0100000`. Note that the y-axis changes scale twice.

to generate initial decompositions that are tailored towards inputs with lots of dense, grid-aligned and, thus, collinear points. Figure 8 illustrates the score over time for `rop0064054` and `ortho_rect_union_47381`, which act as representatives for their corresponding input classes. Apparently, the introduction of OrthoOpt improved our solutions for the `rop*` instances, whereas it provided no improvement for the `ortho_rect_union*` input class.

In Figure 9, we show the scores of the overall best decompositions for various Challenge instances. Additionally, Figure 10 illustrates the development of the average score over time. Note that the significant improvement of the average score in mid January is due to the introduction of Flipper.

4 Conclusion

Our work makes it apparent that well-crafted heuristics run on moderate computing equipment are good enough to achieve decent minimum convex decompositions. But the second batch of Challenge instances made it also apparent that heuristics need not be universally applicable. Rather, they may require an adaption relative to the characteristics of the input data.

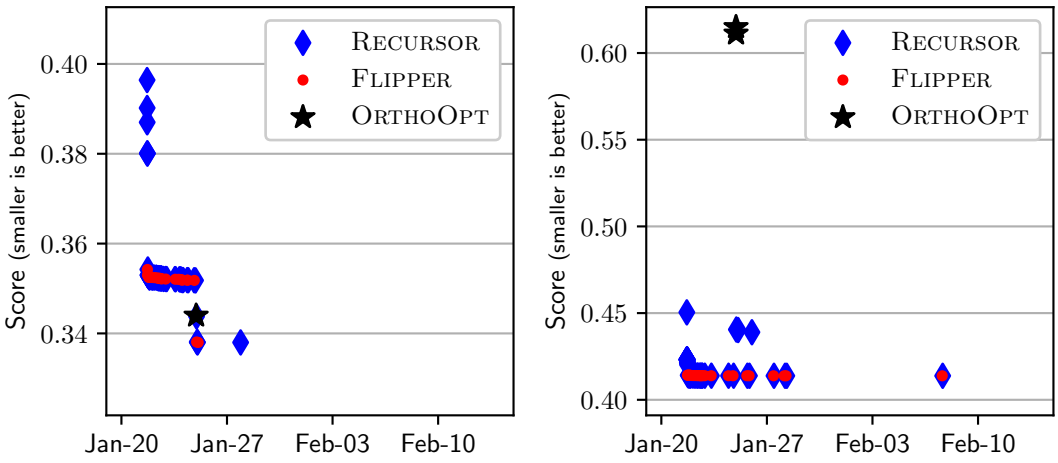


Figure 8 Score over time for `rop0064054` and `ortho_rect_union_47381`.

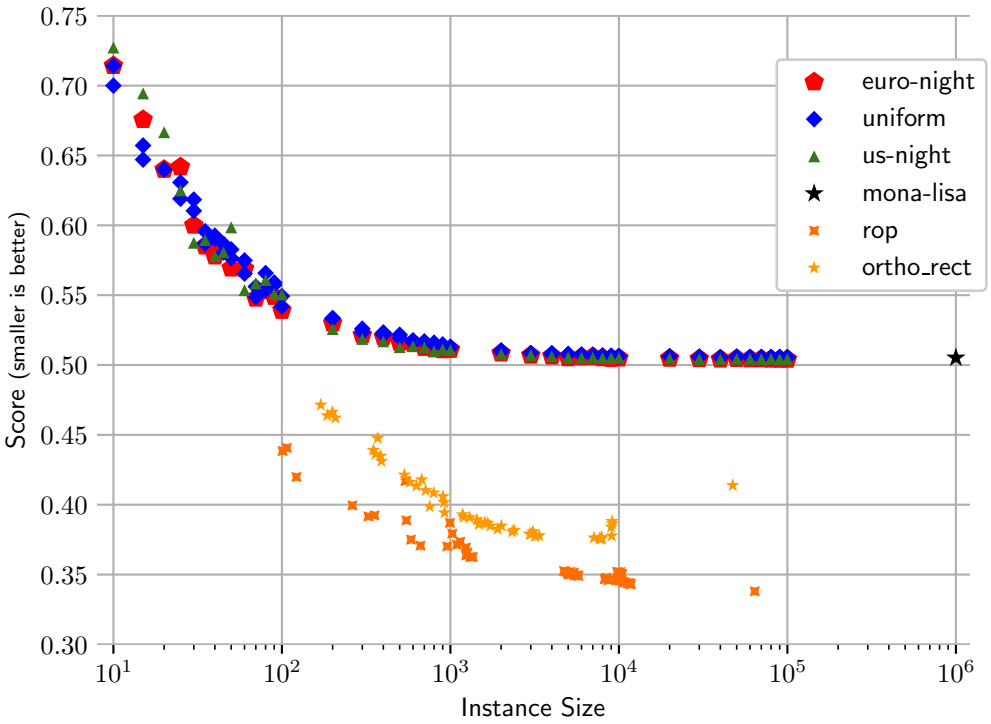


Figure 9 Score per instance.

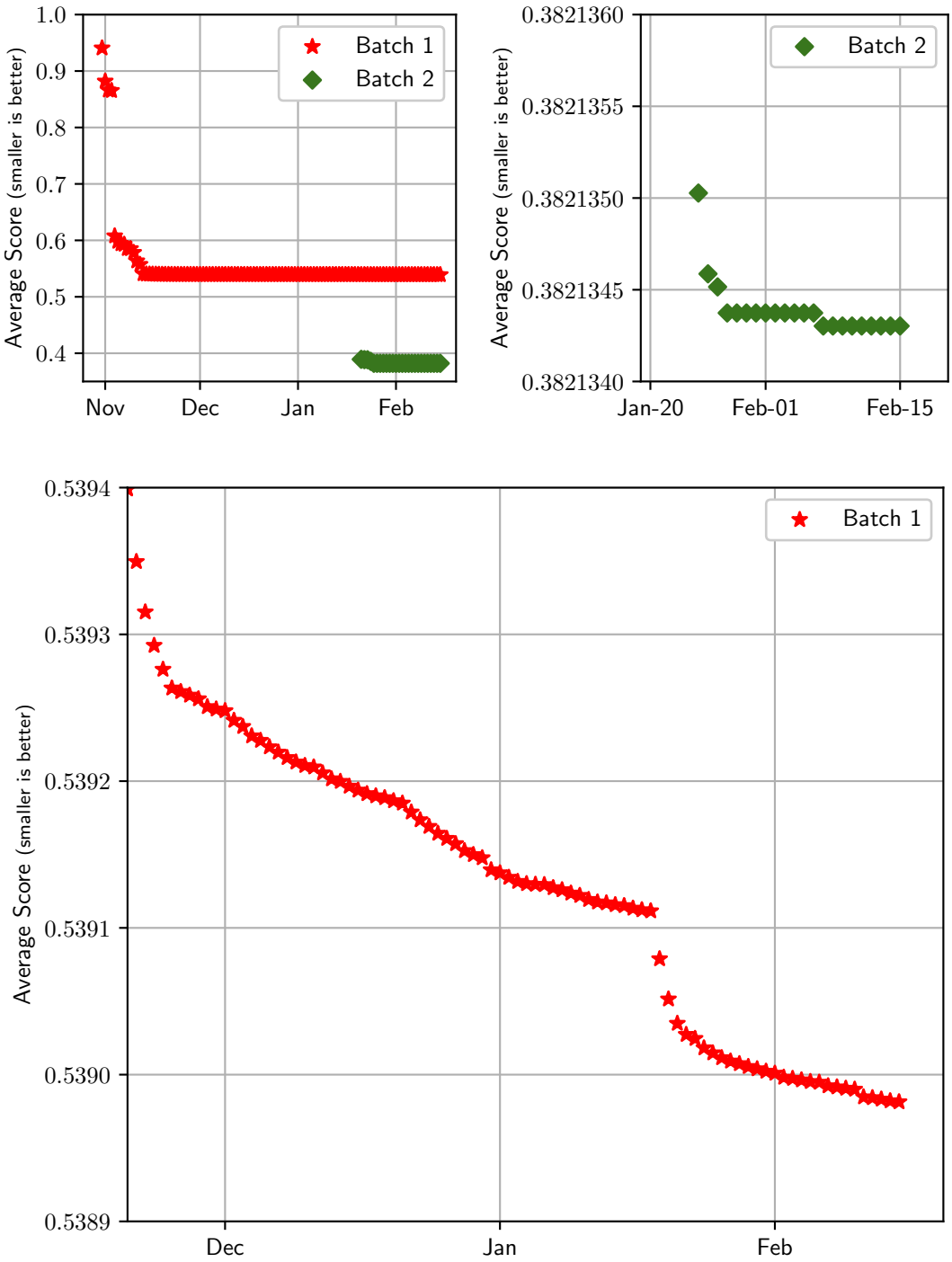


Figure 10 Average score over time.

References

1 Bernard Chazelle. On the Convex Layers of a Planar Set. *IEEE Transactions on Information Theory*, 31(4):509–517, July 1985. doi:10.1109/TIT.1985.1057060.

2 Erik D. Demaine, Sándor P. Fekete, Phillip Keldenich, Dominik Krupke, and Joseph S. B. Mitchell. Computing Convex Partitions for Point Sets in the Plane: The CG:SHOP Challenge 2020, 2020. arXiv:2004.04207.

3 Christian Knauer and Andreas Spillner. Approximation Algorithms for the Minimum Convex Partition Problem. In *Algorithm Theory – SWAT 2006*, pages 232–241, 2006.

4 Jonathan Richard Shewchuk. Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. In *Applied Computational Geometry: Towards Geometric Engineering*, volume 1148 of *Lecture Notes in Computer Science*, pages 203–222. Springer-Verlag, May 1996. ISBN 3-540-61785-X.