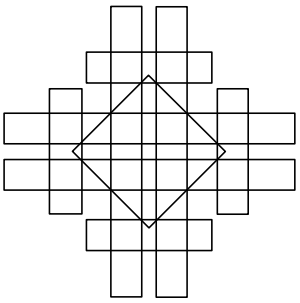
33rd International Symposium on Computational Geometry

SoCG 2017, July 4–7, 2017, Brisbane, Australia

Edited by

Boris Aronov

Matthew J. Katz



Editors

Boris Aronov
New York University
New York
USA
boris.aronov@nyu.edu

Matthew J. Katz
Ben-Gurion University
Beer-Sheva
Israel
matya@cs.bgu.ac.il

ACM Classification 1998
F.2.2 [Analysis of Algorithms and Problem Complexity] Nonnumerical Algorithms and Problems – Geometrical problems and computations, G.2.1 [Discrete Mathematics] Combinatorics, I.3.5 [Computer Graphics] Computational Geometry and Object Modeling

ISBN 978-3-95977-038-5

Published online and open access by
Schloss Dagstuhl – Leibniz-Zentrum für Informatik GmbH, Dagstuhl Publishing, Saarbrücken/Wadern, Germany. Online available at http://www.dagstuhl.de/dagpub/978-3-95977-038-5.

Publication date
June, 2017

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

Digital Object Identifier: 10.4230/LIPIcs.SoCG.2017.0

ISBN 978-3-95977-038-5 **ISSN 1868-8969** **http://www.dagstuhl.de/lipics**

LIPIcs – Leibniz International Proceedings in Informatics

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

ISSN 1868-8969

http://www.dagstuhl.de/lipics

Contents

Invited Talks

Regular SoCG Papers

33rd International Symposium on Computational Geometry (SoCG 2017).
Editors: Boris Aronov and Matthew J. Katz
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Multimedia Contributions

Foreword

Computational Geometry has a rich and successful history. For many years, the prime annual event of the community has been the Symposium on Computational Geometry (SoCG), which we are celebrating for the 33rd time (as part of CG Week 2017) at The University of Queensland, Brisbane, Australia, July 4–7, 2017. Herein, we are happy to present the contributions that were selected for SoCG'17, consisting of abstracts of invited talks, research papers, and descriptions of multimedia presentations.

There were 148 papers submitted to SoCG'17 via EasyChair. After a substantial review process, during which each paper received at least three reviews and which involved 250 external reviewers, the program committee has selected 59 papers for presentation and inclusion in the proceedings. A subset of these was also invited to forthcoming special issues of *Discrete & Computational Geometry* and the *Journal of Computational Geometry*, dedicated to the best papers of SoCG'17.

The Best Paper Award goes to the paper "Computing the Geometric Intersection Number of Curves" by Vincent Despré and Francis Lazarus. The Best Student Presentation Award will be determined and announced at the symposium, based on the input of the attendees.

In addition to the technical papers, there were six submissions to the multimedia exposition. All six were reviewed and accepted for presentation. The extended abstracts that describe these submissions are included in this proceedings volume. The final versions of the multimedia content are archived at `http://www.computational-geometry.org`.

We thank all authors of submitted papers and multimedia contributions. We also thank all the people who contributed time and expertise to the quality and success of this conference, especially the local organizers, the external reviewers, and the members of the program committees.

Boris Aronov
Program Committee co-chair
New York University

Matthew J. Katz
Program Committee co-chair
Ben-Gurion University

Matias Korman
Multimedia Committee chair
Tohoku University

33rd International Symposium on Computational Geometry (SoCG 2017).
Editors: Boris Aronov and Matthew J. Katz
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Conference Organization

SoCG Program Committee

Boris Aronov *(co-chair, New York University, USA)*
Matthew J. Katz *(co-chair, Ben-Gurion University, Israel)*
Peyman Afshani *(Madalgo, Denmark)*
Maike Buchin *(Ruhr-Universität Bochum, Germany)*
Danny Chen *(University of Notre Dame, USA)*
Sariel Har-Peled *(University of Illinois at Urbana-Champaign, USA)*
Michael Hoffmann *(ETH Zürich, Switzerland)*
Ravi Janardan *(University of Minnesota, USA)*
David Kirkpatrick *(The University of British Columbia, Canada)*
Sylvain Lazard *(INRIA Nancy – Grand Est, France)*
Maarten Löffler *(Utrecht University, The Netherlands)*
Anil Maheshwari *(Carleton University, Canada)*
Arnaud de Mesmay *(CNRS, Gipsa-lab, France)*
Pat Morin *(Carleton University, Canada)*
Yoshio Okamoto *(The University of Electro-Communications, Japan)*
Evanthia Papadopoulou *(Università della Svizzera italiana, Switzerland)*
Valentin Polishchuk *(Linköping University, Sweden)*
Günter Rote *(Freie Universität Berlin, Germany)*
Rodrigo I. Silveira *(Universitat Politècnica de Catalunya, Spain)*
Martin Tancer *(Charles University, Czech Republic)*

Multimedia Program Committee

Matias Korman *(chair, Tohoku University, Japan)*
Yoshio Okamoto *(The University of Electro-Communications, Japan)*
Alexander Pilz *(ETH Zürich, Switzerland)*
Rodrigo I. Silveira *(Universitat Politècnica de Catalunya, Spain)*
Darren Strash *(Colgate University, USA)*
Kevin Verbeek *(TU Eindhoven, The Netherlands)*
Sander Verdonschot *(University of Ottawa, Canada)*

Workshop Program Committee

John Hershberger *(chair, Mentor Graphics, USA)*
David Hsu *(National University of Singapore, Singapore)*
Donald Sheehy *(University of Connecticut, USA)*
Bettina Speckmann *(TU Eindhoven, The Netherlands)*

33rd International Symposium on Computational Geometry (SoCG 2017).
Editors: Boris Aronov and Matthew J. Katz
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Young Researchers Forum Program Committee

Therese Biedl *(chair, University of Waterloo, Canada)*
Arijit Bishnu *(Indian Statistical Institute, India)*
Stephane Durocher *(University of Manitoba, Canada)*
Seok-hee Hong *(University of Sydney, Australia)*
John Iacono *(New York University, USA)*
Clement Maria *(University of Queensland, Australia)*
David Mount *(University of Maryland, USA)*
Yoshio Okamoto *(The University of Electro-Communications, Japan)*

Local Organizer

Benjamin Burton *(The University of Queensland, Australia)*

Steering Committee (2016–)

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

Additional Reviewers

Mikkel Abrahamsen
Anna Adamaszek
Henry Adams
Pankaj Agarwal
Hee-Kap Ahn
Hugo Akitaya
Joshua Alman
Laurent Alonso
Helmut Alt
Alexandr Andoni
Sunil Arya
Yakov Babichenko
Arturs Backurs
Sang Won Bae
Martin Balko
Aritra Banik
Michael J. Bannister
Bahareh Banyassady
Imre Barany
Luis Barba
Gill Barequet
Ulrich Bauer
Mikhail Belkin
Sergey Bereg
Mark de Berg
Marshall Bern
Binay Bhattacharya
Therese Biedl
Davide Bilò
Ahmad Biniaz
Jean-Daniel Boissonnat
Marthe Bonamy
Peter Brass
David Bremner
Karl Bringmann
Mickaël Buchet
Kevin Buchin
Mark Bun
Sergio Cabello
Martin Cadek
Jean Cardinal
Paz Carmi
Erin Chambers
Timothy M. Chan
Steven Chaplick
Frédéric Chazal
Siu-Wing Cheng
Man Kwun Chiu
Sunghee Choi
Vincent Cohen-Addad
David Cohen-Steiner
David Conlon
Sabine Cornelsen
Ovidiu Daescu
Sandip Das
Minati De
William E. Devanny
Olivier Devillers
Tamal Dey
Claudia Dieckmann
Hu Ding
Vida Dujmović
Stephane Durocher
Alon Efrat
David Eppstein
Jeff Erickson
Louis Esperet
William Evans
Esther Ezra
Rolf Fagerberg
Mohammad Farshi
Brittany Terese Fasy
Sándor Fekete
Omrit Filtser
Vissarion Fisikopoulos
Guilherme D. da Fonseca
Fabrizio Frati
Florian Frick
Radoslav Fulek
Jie Gao
Alfredo Garcia
Delia Garijo
Bernd Gärtner
Joachim Giesen
David Glickenstein
Marc Glisse
Xavier Goaoc
Lee-Ad Gottlieb
Carsten Grimm
Allan Grønlund
Romain Grunert
Joachim Gudmundsson
Dan Halperin
John Hershberger
Frank Hoffmann
Ziyun Huang
Alfredo Hubard
John Iacono
Piotr Indyk
Hiro Ito
Iwan Jensen
Gwenaël Joret
Vojtěch Kaluža
Iyad Kanj
Haim Kaplan
Mark Keil
Elena Khramtcova
Philipp Kindermann
James King
Jun Kitagawa
Rolf Klein
Boris Klemz
Christian Knauer
Kolja Knauer
Irina Kostitsyna
Laszlo Kozma
Marc van Kreveld
Klaus Kriegel
Erik Krohn
Jason S. Ku
Nirman Kumar
Jan Kynčl
Abhiruk Lahiri
Stefan Langerman
Kasper Green Larsen
James Lee

33rd International Symposium on Computational Geometry (SoCG 2017).
Editors: Boris Aronov and Matthew J. Katz
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Erik Jan van Leeuwen
Tuomo Lempiäinen
David Letscher
Bruno Levy
Jian Li
Yuan Li
Bernard Lidicky
Andre Linhares
Giuseppe Liotta
Chih-Hung Liu
Paul Liu
Anna Lubiw
Isaac Mabillard
Sepideh Mahabadi
Clément Maria
Dániel Marx
Tyler Mayer
Marina Meila
Facundo Memoli
Manor Mendel
Wouter Meulemans
Malte Milatz
Tillmann Miltzow
Joseph Mitchell
Bojan Mohar
Guillaume Moroz
David Mount
Wolfgang Mulzer
Nabil Mustafa
Subhas Nandy
Assaf Naor
Amir Nayyeri
Ofer Neiman
Frank Nielsen
Aleksandar Nikolov
Bengt J. Nilsson
Jerri Nummenpalo
Eunjin Oh
Tim Ophelders
David Orden
Yota Otachi
Steve Oudot
Arnau Padrol
Leonidas Palios
Dömötör Pálvölgyi
Salman Parsa
Paul Pearson
Jeff Phillips
Michał Pilipczuk
Alexander Pilz
Marc Pouget
Kent Quanrud
Saladi Rahul
Rajiv Raman
Orit E. Raz
André van Renssen
Marcel Roeloffzen
Edgardo Roldán-Pensado
Sasanka Roy
Ignaz Rutter
Leonardo Ignacio Martínez Sandoval
Filippo Santambrogio
Jenya Sapir
Manfred Scheucher
Stefan Schirra
Jean-Marc Schlenker
Christiane Schmidt
Patrick Schnider
André Schulz
Matthias Schymura
Leonid Sedov
Raimund Seidel
Paul Seiferth
Micha Sharir
Don Sheehy
Jonathan Shewchuk
Dayu Shi
Anastasios Sidiropoulos
Michiel Smid
Shakhar Smorodinsky
Noam Solomon
Kiril Solovey
Joachim Spoerhase
Jonathan Spreer
Boris Springborn
Frank Staals
Ladislav Stacho
Yannik Stein
Milos Stojakovic
Darren Strash
Ileana Streinu
Andrew Suk
Dougal Sutherland
May Szedlak
Topi Talvitie
Shin-Ichi Tanigawa
Monique Teillaud
Pratap Tokekar
Csaba Toth
Torsten Ueckerdt
Ryuhei Uehara
Jonathan Ullman
Pavel Valtr
Kasturi Varadarajan
Mikael Vejdemo-Johansson
Kevin Verbeek
Sander Verdonschot
Antoine Vigneron
Haitao Wang
Yusu Wang
Carola Wenk
Manuel Wettstein
Max Willert
Mathijs Wintraecken
Steve Wismath
David R. Wood
David P. Woodruff
Jinhui Xu
Jie Xue
Frank de Zeeuw
Norbert Zeh
Rico Zenklusen
Piotr Zgliczyński
Xinhua Zhang

Sponsors

We are grateful to The University of Queensland for hosting and sponsoring CG week 2017, and to the National Science Foundation (NSF) for providing financial support.

The Geometry and Topology of Crystals: From Sphere-Packing to Tiling, Nets, and Knots*

Vanessa Robins

Department of Applied Mathematics, Research School of Physics and Engineering, The Australian National University, Canberra, Australia
vanessa.robins@anu.edu.au

Abstract

Crystal structures have inspired developments in geometry since the Ancient Greeks conceived of Platonic solids after observing tetrahedral, cubical and octahedral mineral forms in their local environment. The internal structure of crystals became accessible with the development of x-ray diffraction techniques just over 100 years ago, and a key step in developing this method was understanding the arrangement of atoms in the simplest crystals as close-packings of spheres. Determining a crystal structure via x-ray diffraction unavoidably requires prior models, and this has led to the intense study of sphere packing, atom-bond networks, and arrangements of polyhedra by crystallographers investigating ever more complex compounds. In the 21st century, chemists are exploring the possibilities of coordination polymers, a wide class of crystalline materials that self-assemble from metal cations and organic ligands into periodic framework materials. Longer organic ligands mean these compounds can form multi-component interwoven network structures where the "edges" are no longer constrained to join nearest-neighbour "nodes" as in simpler atom-bond networks. The challenge for geometers is to devise algorithms for enumerating relevant structures and to devise invariants that will distinguish between different modes of interweaving. This talk will survey various methods from computational geometry and topology that are currently used to describe crystalline structures and outline research directions to address some of the open questions suggested above.

1998 ACM Subject Classification I.3.5 Computational Geometry and Object Modelling, J.2 Physical Sciences and Engineering

Keywords and phrases Mathematical crystallography, Combinatorial tiling theory, Graphs and surfaces in the 3-torus

Digital Object Identifier 10.4230/LIPIcs.SoCG.2017.1

Category Invited Talk

* This work is supported by ARC Future Fellowship Grant FT140100604.

33rd International Symposium on Computational Geometry (SoCG 2017).
Editors: Boris Aronov and Matthew J. Katz; Article No. 1; pp. 1:1–1:1

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

The Algebraic Revolution in Combinatorial and Computational Geometry: State of the Art

Micha Sharir

Blavatnik School of Computer Science, Tel Aviv University, Tel Aviv, Israel
michas@tau.ac.il

Abstract

For the past 10 years, combinatorial geometry (and to some extent, computational geometry too) has gone through a dramatic revolution, due to the infusion of techniques from algebraic geometry and algebra that have proven effective in solving a variety of hard problems that were thought to be unreachable with more traditional techniques. The new era has begun with two groundbreaking papers of Guth and Katz, the second of which has (almost completely) solved the distinct distances problem of Erdős, open since 1946.

In this talk I will survey some of the progress that has been made since then, including a variety of problems on distinct and repeated distances and other configurations, on incidences between points and lines, curves, and surfaces in two, three, and higher dimensions, on polynomials vanishing on Cartesian products with applications, on cycle elimination for lines and triangles in three dimensions, on range searching with semialgebraic sets, and I will most certainly run out of time while doing so.

1998 ACM Subject Classification F.2.2 Geometrical Problems and Computations, G.2.1 Combinatorics

Keywords and phrases Combinatorial Geometry, Incidences, Polynomial method, Algebraic Geometry, Distances

Digital Object Identifier 10.4230/LIPIcs.SoCG.2017.2

Category Invited Talk

33rd International Symposium on Computational Geometry (SoCG 2017).
Editors: Boris Aronov and Matthew J. Katz; Article No. 2; pp. 2:1–2:1

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

Irrational Guards are Sometimes Needed*

Mikkel Abrahamsen[1], Anna Adamaszek[2], and Tillmann Miltzow[3]

1 **University of Copenhagen, Copenhagen, Denmark**
`miab@di.ku.dk`
2 **University of Copenhagen, Copenhagen, Denmark**
`anad@di.ku.dk`
3 **Institute for Computer Science and Control, Hungarian Academy of Sciences (MTA SZTAKI), Budapest, Hungary**
`t.miltzow@gmail.com`

Abstract

In this paper we study the *art gallery problem*, which is one of the fundamental problems in computational geometry. The objective is to place a minimum number of guards inside a simple polygon so that the guards together can see the whole polygon. We say that a guard at position x sees a point y if the line segment xy is contained in the polygon.

Despite an extensive study of the art gallery problem, it remained an open question whether there are polygons given by integer coordinates that require guard positions with irrational coordinates in any optimal solution. We give a positive answer to this question by constructing a *monotone* polygon with integer coordinates that can be guarded by three guards only when we allow to place the guards at points with irrational coordinates. Otherwise, four guards are needed. By extending this example, we show that for every n, there is a polygon which can be guarded by $3n$ guards with irrational coordinates but needs $4n$ guards if the coordinates have to be rational. Subsequently, we show that there are rectilinear polygons given by integer coordinates that require guards with irrational coordinates in any optimal solution.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases art gallery problem, computational geometry, irrational numbers

Digital Object Identifier 10.4230/LIPIcs.SoCG.2017.3

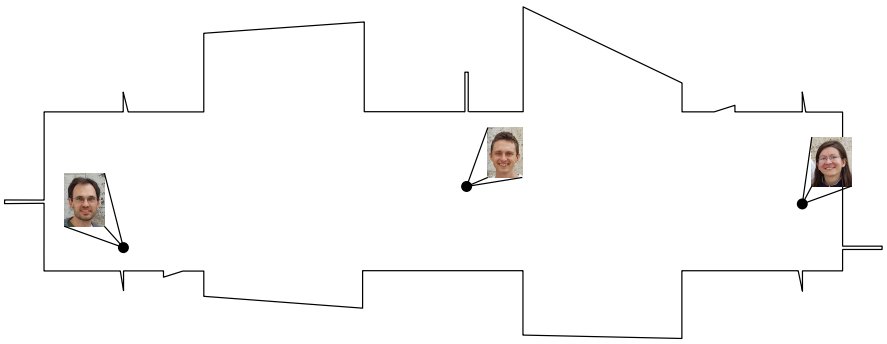


Figure 1 Till, Mikkel, and Anna are meticulously guarding the polygon. They are a little irrational, but pretty optimal.

* Research partially supported by Mikkel Thorup's Advanced Grant from the Danish Council for Independent Research under the Sapere Aude research career programme, by the Danish Council for Independent Research DFF-MOBILEX mobility grant, and by the ERC grant "PARAMTIGHT: Parameterized complexity and the search for tight complexity results" no. 280152.

1 Introduction

For a polygon $\mathcal{P}$ and points $x, y \in \mathcal{P}$, we say that x *sees* y if the line segment xy is contained in $\mathcal{P}$. A *guard set* S is a set of points in $\mathcal{P}$ such that every point in $\mathcal{P}$ is seen by some point in S. The points in S are called *guards*. The *art gallery problem* is to find a minimum cardinality guard set for a given simple polygon $\mathcal{P}$ on n vertices. Such a guard set is called *optimal*. The polygon $\mathcal{P}$ is considered to be filled, i.e., it consists of a closed, simple polygonal curve in the plane and the bounded region enclosed by this curve.

This classical version of the art gallery problem has been originally formulated in 1973 by Victor Klee (see the book of O'Rourke [20, page 2]). It is often referred to as the *interior-guard art gallery problem* or the *point-guard art gallery problem*, to distinguish it from other versions that have been introduced over the years.

Chvátal proved in 1975 that $\lfloor n/3 \rfloor$ guards are always sufficient and sometimes necessary to guard a polygon with n vertices [9]. A simpler proof was later found by Fisk [15]. Since then, the art gallery problem has been extensively studied, both from the combinatorial and the algorithmic perspective. Most of this research, however, is not focused directly on the classical art gallery problem, but on its numerous versions, including different definitions of visibility, restricted classes of polygons, restrictions on the positions of the guards, etc. For more detailed information we refer the reader to the surveys [26, 28, 20, 22].

Despite extensive research on the art gallery problem, no combinatorial algorithm for finding an optimal solution, or even for deciding whether a guard set of a given size k exists, is known. The only exact algorithm is attributed to Micha Sharir (see [12]), who has shown that in $n^{O(k)}$ time one can decide whether a guard set consisting of k guards exists. This result is obtained by using standard tools from real algebraic geometry [2], and it is not known how to find an optimal solution without using this powerful machinery (see [3] for an analysis of the very restricted case of $k = 2$). Some recent lower bounds [5] based on the exponential time hypothesis suggest that there might be no better exact algorithms than the one by Sharir.

To explain the difficulty in constructing exact algorithms, we want to emphasize that it is *not* known whether the decision version of the art gallery problem (i.e., the problem of deciding whether there is a guard set consisting of k guards, where k is a parameter) lies in the complexity class NP. While NP-hardness and APX-hardness of the art gallery problem have been shown for different versions of the problem [18, 25, 27, 6, 13, 21, 17], the question of whether the point-guard art gallery problem is in NP remains open. A simple way to show NP-membership would be to prove that there always exists an optimal set of guards with *rational* coordinates of polynomially bounded description.

Sándor Fekete posed at MIT in 2010 and at Dagstuhl in 2011 an open problem, asking whether there are polygons requiring irrational coordinates in an optimal guard set [14, 1]. The question has been raised again by Günter Rote at EuroCG 2011 [23]. It has also been mentioned by Rezende *et al.* [10]: "it remains an open question whether there are polygons given by rational coordinates that require optimal guard positions with irrational coordinates". A similar question has been raised by Friedrichs *et al.* [16]: "[...] it is a long-standing open problem for the more general Art Gallery Problem (AGP): For the AGP it is not known whether the coordinates of an optimal guard cover can be represented with a polynomial number of bits".

Our results. We answer the open question of Sándor Fekete by proving the following result. Recall that a polygon $\mathcal{P}$ is called *monotone* if there exists a line l such that the intersection between any line orthogonal to l and $\mathcal{P}$ is either empty or a single line segment.

▶ **Theorem 1.** *There is a simple monotone polygon $\mathcal{P}$ with integer vertex coordinates such that*
1. *$\mathcal{P}$ can be guarded by 3 guards, and*
2. *an optimal guard set of $\mathcal{P}$ with guards at points with rational coordinates has size 4.*

An interesting consequence of Theorem 1 is that there is no optimal guard set of $\mathcal{P}$ among a candidate set of guard positions consisting of intersections between extensions of chords and edges of $\mathcal{P}$. It does not help to expand the candidate set by adding a line through each pair of candidates, thus creating new intersections to be added to the set of candidates, or to repeat this procedure any finite number of iterations, since all candidate points created by such a process must inevitably have rational coordinates. This shows that algorithms based on this procedure, as well as other algorithms for the art gallery problem which consider only rational points as possible guard positions, will in general not find an optimal guard set.

We then extend Theorem 1 by providing a family of polygons for which the ratio between the size of an optimal rational guard set and the size of an optimal set with irrational guards allowed is $4/3$.

▶ **Theorem 2.** *There is a family of simple polygons $(\mathcal{P}_n)_{n\in\mathbb{Z}_+}$ with integer vertex coordinates such that*
1. *$\mathcal{P}_n$ can be guarded by $3n$ guards, and*
2. *an optimal guard set of $\mathcal{P}_n$ with guards at points with rational coordinates has size $4n$.*

Moreover, the coordinates of the points defining the polygons $\mathcal{P}_n$ are polynomial in n.

We show that the phenomenon with guards at irrational coordinates occurs already in the much simpler class of rectilinear polygons, i.e., polygons where each edge is parallel to the x-axis or to the y-axis.

▶ **Theorem 3.** *There is a rectilinear polygon $\mathcal{P}_R$ with vertices at integer coordinates satisfying the following properties.*
1. *$\mathcal{P}_R$ can be guarded by 9 guards.*
2. *An optimal guard set of $\mathcal{P}_R$ with guards at points with rational coordinates has size 10.*

The Structure of the Paper. Section 2 contains the description of a monotone polygon $\mathcal{P}$ with vertices at points with rational coordinates that can be guarded by three guards only if the guards are placed at points with irrational coordinates. In Section 3, we describe the intuition behind our construction, and explain how we have found the polygon $\mathcal{P}$. The formal proof of Theorems 1 and 2 is then provided in Section 4. In Section 5, we present the rectilinear polygon $\mathcal{P}_R$ from Theorem 3 requiring guards with irrational coordinates in an optimal guard set. Finally, in Section 6 we suggest some open problems for future research.

2 The Polygon

In Figure 2 we present the polygon $\mathcal{P}$. In Section 4 we will prove that $\mathcal{P}$ can be guarded by three guards only when we allow the guards to be placed at points with irrational coordinates.

The polygon $\mathcal{P}$ is constructed as follows. We start with a *basic rectangle* $[0,20]\times[0,4]\subset\mathbb{R}^2$. Then, we append to it six *triangular pockets* (colored with green in the figure), which are triangles defined by the following coordinates:

$$T_t^\ell:\ \{(2,4),(2,4.5),(2.1,4)\},\qquad T_b^\ell:\ \{(2,0),(2,-0.5),(1.9,0)\},$$
$$T_t^m:\ \{(16\tfrac{5}{6},4),(17\tfrac{2}{6},4.15),(17\tfrac{2}{6},4)\},\qquad T_b^m:\ \{(3.5,0),(3,-0.15),(3,0)\},$$
$$T_t^r:\ \{(19,4),(19,4.5),(19.1,4)\},\quad\text{and}\quad T_b^r:\ \{(19,0),(19,-0.5),(18.9,0)\}.$$

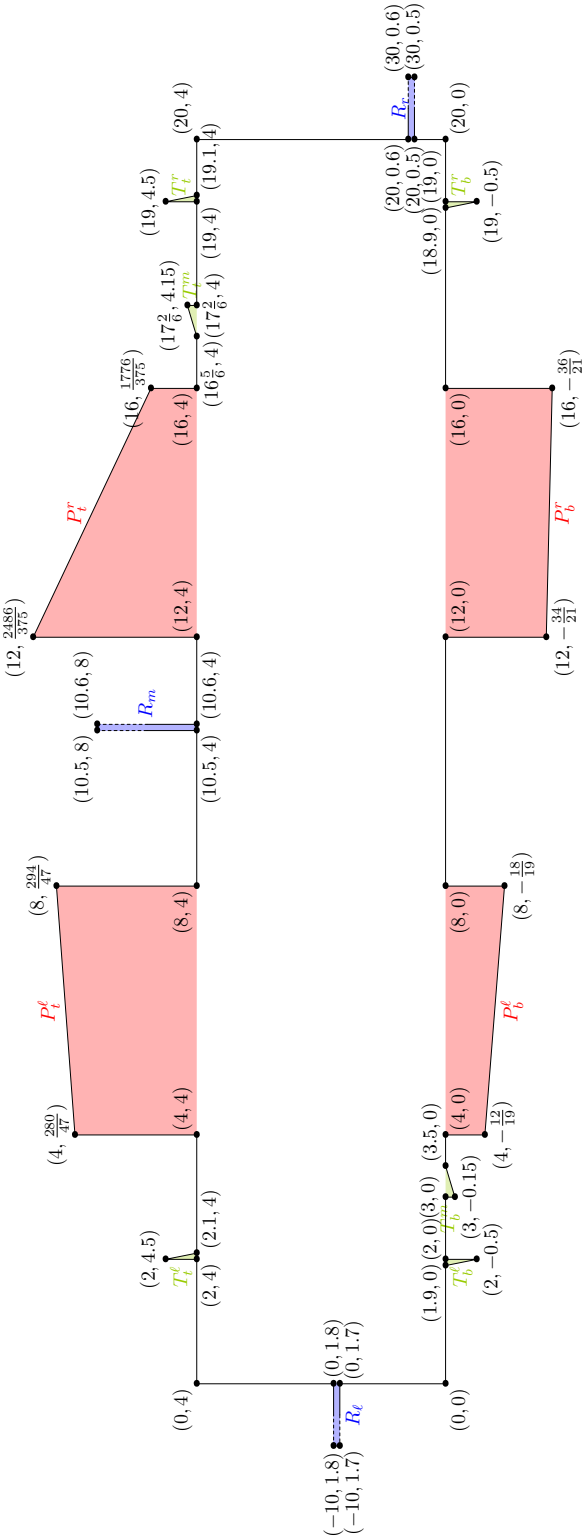
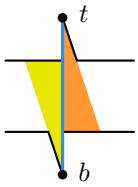
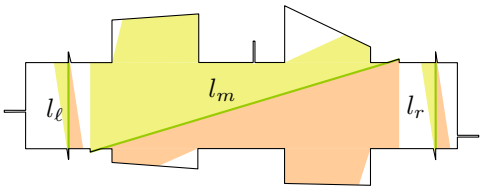


Figure 2 The polygon $\mathcal{P}$. We will show that $\mathcal{P}$ can be guarded by three guards only when we allow the guards to be placed at points with irrational coordinates. For practical reasons, the blue rectangular pockets are drawn shorter than they actually are.



(a) The only way that one guard can see both t and b is when the guard is on the blue line segment.



(b) The only way to guard the polygon with three guards requires one guard on each of the green line segments l_ℓ, l_m, l_r.

Figure 3 Forcing guards to lie on specific line segments.

Next, we append three *rectangular pockets* (colored with blue in the figure, for practical reasons these pockets are drawn in the figure shorter than they actually are), which are rectangles defined in the following way.

R_ℓ: $[-10, 0] \times [1.7, 1.8]$, R_r: $[20, 30] \times [0.5, 0.6]$, and R_m: $[10.5, 10.6] \times [4, 8]$.

Last, we append four *quadrilateral pockets* (colored with red in the figure), which are defined by points with the following coordinates:

Top-left pocket P_t^ℓ	$\{(4, 4),$	$(4, \frac{280}{47}),$	$(8, \frac{294}{47}),$	$(8, 4)\}$
Top-right pocket P_t^r	$\{(12, 4),$	$(12, \frac{2486}{375}),$	$(16, \frac{1776}{375}),$	$(16, 4)\}$
Bottom-left pocket P_b^ℓ	$\{(4, 0),$	$(4, -\frac{12}{19}),$	$(8, -\frac{18}{19}),$	$(8, 0)\}$
Bottom-right pocket P_b^r	$\{(12, 0),$	$(12, -\frac{34}{21}),$	$(16, -\frac{36}{21}),$	$(16, 0)\}$.

The polygon $\mathcal{P}$ is clearly monotone. We will denote by e_t^ℓ, e_t^r, e_b^ℓ, and e_b^r the non-axis-parallel edge within each of the four quadrilateral pockets, respectively.

3 Intuition

In this section, we explain the key ideas behind the construction of the polygon $\mathcal{P}$. Our presentation is informal, but it resembles the work process that lead to the construction of $\mathcal{P}$ more than the formal proof of Theorem 1 in Section 4 does. Here we omit all "scary" computations and focus on conveying the big picture. In the end of this section, we also explain how we actually constructed the polygon $\mathcal{P}$.

Define a *rational point* to be a point with two rational coordinates. An *irrational point* is a point that is not rational. A *rational line* is a line that contains two rational points. An *irrational line* is a line that is not rational.

Forcing a Guard on a Line Segment. Consider the drawing of the polygon $\mathcal{P}$ in Figure 2. We will now explain an idea of how three pairs of triangular pockets, (T_t^ℓ, T_b^ℓ), (T_t^m, T_b^m), and (T_t^r, T_b^r), can enforce three guards on three line segments within $\mathcal{P}$.

Consider the two triangular pockets in Figure 3a. The blue line segment contains one edge of each of these pockets, and the interiors of the pockets are at different sides of the line segment. A guard which sees the point t must be placed within the orange triangular region, and a guard which sees b must be placed within the yellow triangular region. Thus, a single guard can see both t and b only if it is on the blue line segment tb, which is the intersection of the two regions.

Consider now the case that we have k pairs of triangular pockets and no two regions corresponding to different pairs of pockets intersect. In order to guard the polygon with k guards, there must be one guard on the line segment corresponding to each pair. Our

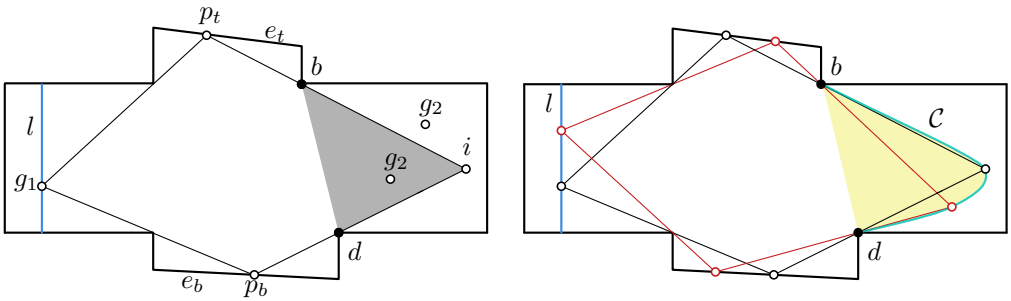


Figure 4 Left: The guard g_2 must be inside the triangular region (or to the left of it) in order to guard the entire part of the polygon that is not seen by g_1. Right: All possible positions of the point i define a simple curve $\mathcal{C}$.

polygon $\mathcal{P}$ has three such pairs of pockets (see Figure 3b), and it can be checked that the corresponding regions do not intersect. Note that in this way we can only enforce a guard to be on a rational line as the line contains vertices of the polygon, which are rational points.

Restricting a Guard to a Region Bounded by a Curve. For the following discussion, see Figure 4 and notation therein. We want to guard the polygon from Figure 4 using two guards, g_1 and g_2. We assume that g_1 is forced to lie on the blue vertical line segment l.

Consider some position of g_1 on l such that g_1 can see at least one point of the top edge e_t of the top quadrilateral pocket and at least one point of the bottom edge e_b of the bottom quadrilateral pocket. Let p_t and p_b denote the leftmost points seen by g_1 on e_t and e_b, respectively. Observe that p_t moves to the right if g_1 moves up and to the left if g_1 moves down. The point p_b behaves in the opposite way when g_1 is moved. Consider some fixed position of g_1 on the blue line segment, and the corresponding positions of p_t and p_b. Let b be the bottom right corner of the top pocket and d the top right corner of the bottom pocket. Let i be the intersection point of the line containing p_t and b with the line containing p_b and d. The points b, d, i define a triangular region Δ. It is clear that if we place the guard g_2 anywhere inside Δ, then g_1 and g_2 will together see the entire polygon. On the other hand, if we place g_2 to the right of Δ, then g_1 and g_2 will not see the entire polygon, as some part of the top or the bottom pocket will not be seen.

Now, let us move the guard g_1 along l. Each position of g_1 yields an intersection point i. We denote the union of all these intersection points by $\mathcal{C}$ (see the right picture in Figure 4). It is easy to see that $\mathcal{C}$ is a simple curve.

Note that g_2 sees a larger part of *both* pockets if it is moved horizontally to the left and a smaller part of *both* pockets if it is moved horizontally to the right. Consider a fixed position of g_2 on or to the right of the segment bd. Let g_2' be the horizontal projection of g_2 on $\mathcal{C}$. Let g_1 be the unique position on l such that g_1 and g_2' see all of the polygon. If g_2 is to the left of $\mathcal{C}$, g_2' sees less of the pockets than g_2, so g_1 and g_2 can together see everything. If g_2 is to the right of $\mathcal{C}$, g_2 sees less of the pockets than g_2' and neither the top nor the bottom pocket are completely guarded by g_1 and g_2. For any higher placement of g_1 even less of the top pocket is guarded and for any lower placement of g_1 even less of the bottom pocket is guarded. Thus, there exists no placement of g_1 such that both pockets are completely guarded by g_1 and g_2. We summarize our reasoning in the following observation.

▶ **Observation 4.** *Consider a fixed position of g_2 on or to the right of the segment bd. There exists a position of g_1 on l such that the entire polygon is seen by g_1 and g_2 if and only if g_2 lies on or to the left of the curve $\mathcal{C}$.*

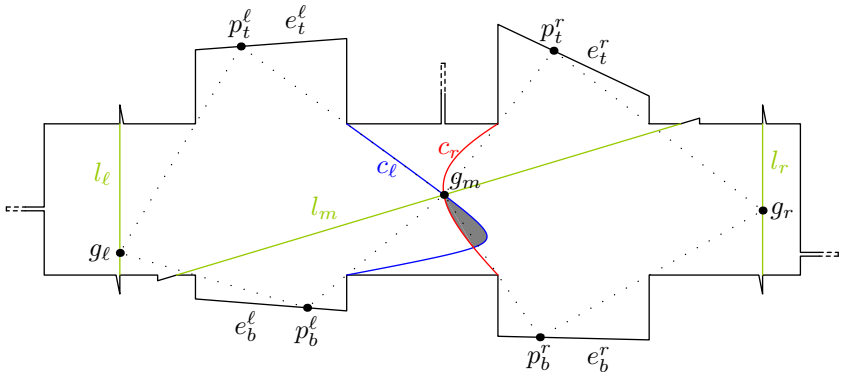


Figure 5 The polygon $\mathcal{P}$.

Restricting a Guard to a Single (Irrational) Point. For this paragraph, let us consider the polygon $\mathcal{P}$ introduced in Section 2, and consider a guard set for $\mathcal{P}$ consisting of three guards. The polygon $\mathcal{P}$ is drawn in Figure 5 with additional labels and information. The three guards g_ℓ, g_m, g_r are forced by the triangular pockets to lie on the three green line segments l_ℓ, l_m, l_r, respectively. Additionally, the three rectangular pockets R_ℓ, R_m, R_r force the guards to lie within one of two or three short intervals within each line segment. (These properties of our construction will be discussed in more detail in Section 4.) With these restrictions, we will show that for the three guards to see the whole polygon, it must hold that the guards g_ℓ and g_m can together see the left pockets P_t^ℓ and P_b^ℓ and the guards g_m and g_r can together see the right pockets P_t^r and P_b^r.

The curve c_ℓ bounds from the right the feasible region for the guard g_m such that g_ℓ and g_m can together see the left pockets P_t^ℓ and P_b^ℓ. Similarly, the curve c_r bounds from the left the feasible region for the guard g_m such that g_r and g_m can together see the right pockets P_t^r and P_b^r. Thus, the only way that g_ℓ, g_m, and g_r can see the whole polygon is when g_m is within the grey region between c_ℓ and c_r. Our idea is to define the line segment l_m so that it contains an intersection point of c_ℓ and c_r while not entering the interior of the grey region. A simple computation with sage [11] outputs equations defining the two curves:

$$c_\ell : 138x^2 - 568xy - 1071y^2 - 3018x + 8828y + 15312 = 0,$$

$$c_r : 138x^2 - 156xy - 356y^2 - 1791x + 3296y + 1620 = 0.$$

One can easily verify that the point $p = (3.5+5\sqrt{2}, 1.5\sqrt{2}) \approx (10.57, 2.12)$ lies on both curves and also on the line $l_m = \{\,(x,y) \,:\, y = 0.3x - 1.05\,\}$. Therefore, p is a feasible (and at the same time irrational) position for the guard g_m. Moreover, by plotting c_ℓ, c_r, and l_m in $\mathcal{P}$ as in Figure 5, we get an indication that as we traverse l_m from left to right, at the point p we exit the area where g_m and g_l can guard together the two left pockets and at the same time we enter the area where g_m and g_r can guard together the two right pockets. Thus, the only feasible position for the guard g_m is the irrational point p. A formal proof will be given in Section 4.

Searching for the Polygon. The simplicity of the ideas behind our construction does not reflect the difficulty of finding the exact coordinates for the polygon $\mathcal{P}$. The reader might for instance presume that most other choices of horizontal pockets would work if the line segment l_m is changed accordingly. However, this is not the case.

It is easy to construct the pockets so that the corresponding curves c_ℓ and c_r intersect at some point p. We expect p to be an irrational point in general since the curves c_ℓ and c_r are defined by two second degree polynomials, as indicated above. In our construction, we need to force g_m to be on a line segment l_m containing p, but we can only force g_m to be on a rational line. Hence, we require the existence of a rational line that contains p.

As any two rational lines intersect in a rational point, there can be at most one rational line containing the irrational point p. Moreover, there exists a rational line containing p if and only if $p = (r_1 + r_2\alpha, r_3 + r_4\alpha)$ for some $r_1, r_2, r_3, r_4 \in \mathbb{Q}$, where $\alpha \in \mathbb{R} \setminus \mathbb{Q}$ is an irrational number. The equation of the rational line containing p is then $y = \frac{r_4}{r_2} \cdot x + (r_3 - r_1 \cdot \frac{r_4}{r_2})$. We say that this line *supports* p. Therefore, we should not hope that the intersection point of the curves c_ℓ and c_r defined by arbitrarily chosen pockets will have a supporting line. Our main idea to overcome this problem has been to reverse-engineer the polygon, after having chosen the positions of the guards. We chose three irrational guards, all with supporting rational lines, and then defined the pockets so that g_m automatically became the intersection point between the curves c_ℓ and c_r associated with the pockets.

We chose all three guards to have coordinates of the form $(r_1 + r_2\sqrt{2}, r_3 + r_4\sqrt{2})$ for $r_1, r_2, r_3, r_4 \in \mathbb{Q}$. Assume, for the ease of presentation, that we already know that we can end up with a polygon described as follows. (In our initial attempts, our polygons were much less regular.) The polygon should consist of the rectangle $R = [0, 20] \times [0, 4]$ with some pockets added. We would like the pockets to extrude vertically from the horizontal edges of R such that the pockets meet R along the segments $(4, 0)(8, 0)$, $(12, 0)(16, 0)$, $(4, 4)(8, 4)$, and $(12, 4)(16, 4)$, respectively.

We now explain the technique for constructing the bottom pocket to the left which should extrude from R vertically downwards from the corners $(4, 0)$ and $(8, 0)$. We have to define the edge e_b^ℓ, which is the bottom edge in the pocket. We want p_b^ℓ to be a point on e_b^ℓ such that g_ℓ can only see the part of e_b^ℓ from p_b^ℓ and to the right, whereas g_m can only see the part of e_b^ℓ from p_b^ℓ and to the left. Therefore, we define p_b^ℓ to be the intersection point between the line containing g_ℓ and $(4, 0)$ and the line containing g_m and $(8, 0)$. It follows that p_b^ℓ is of the form $(r_1 + r_2\sqrt{2}, r_3 + r_4\sqrt{2})$ for some $r_1, r_2, r_3, r_4 \in \mathbb{Q}$. Hence, there is a unique rational line l supporting p_b^ℓ, and e_b^ℓ must be a segment on l. We therefore need that both of the points $(4, 0)$ and $(8, 0)$ are above l, since otherwise we do not get a meaningful polygon. However, this is not the case for arbitrary choices of the guards g_ℓ and g_m. The other pockets add similar restrictions to the positions of the guards.

In the construction we had to take care of other issues as well. In particular, the line l_m which supports the guard g_m cannot enter the grey region between the two curves c_ℓ and c_r, as otherwise the position of g_m would not be unique, and the guard could be moved to a rational point. Also, the three lines l_ℓ, l_m, l_r supporting the three guards g_ℓ, g_m, g_r cannot intersect within the polygon.

4 Proof of Theorems 1 and 2

Basic observations. Recall the construction of the polygon $\mathcal{P}$ as defined in Section 2, and consider a guard set of $\mathcal{P}$ of cardinality at most 3. Let l_ℓ, l_m, l_r, respectively, be the restrictions of the following lines to $\mathcal{P}$:

$$x = 2, \quad y = 0.3x - 1.05, \quad \text{and} \quad x = 19.$$

As argued in Section 3, the triangular pockets enforce a guard onto each of these lines.

▶ **Lemma 5.** *Consider any guard set S for $\mathcal{P}$ consisting of at most 3 guards. Then* (i) $|S| = 3$, *and* (ii) *there is one guard on each of the lines l_ℓ, l_m, l_r.*

Now, consider the intervals $i_1 = [0.5, 0.6]$ and $i_2 = [1.7, 1.8]$. Similarly as for the case of triangular pockets, we can show that the rectangular pockets R_ℓ, R_m, R_r enforce a guard with an x-coordinate in $[10.5, 10.6]$, and the two remaining guards with y-coordinates in i_1 and i_2, respectively.

▶ **Lemma 6.** *Consider any guard set for $\mathcal{P}$ consisting of 3 guards. Then one of the guards has an x-coordinate in $[10.5, 10.6]$. For the remaining two guards, one has a y-coordinate in i_1 and the other has one in i_2.*

Proof. From Lemma 5, there must be one guard g_ℓ on l_ℓ, one guard g_m on l_m, and the last guard g_r on l_r. Recall that the rectangular pockets are as follows R_ℓ: $[-10, 0] \times [1.7, 1.8]$, R_r: $[20, 30] \times [0.5, 0.6]$, and R_m: $[10.5, 10.6] \times [4, 8]$. It is straightforward to check that none of the guards g_ℓ, g_r can see the two top vertices of the pocket R_m. Therefore, the middle guard g_m has to see both of these vertices, so it must have an x-coordinate in $[10.5, 10.6]$.

Then, as $g_m \in l_m$, the y-coordinate of g_m is in $[2.1, 2.13]$. Therefore, g_m cannot see any of the left vertices of R_ℓ or any of the right vertices of R_r. These four vertices must be seen by the guards g_ℓ and g_r.

As some guard must see the bottom-left corner of the pocket R_ℓ, it must be placed at a height of at least 1.7. Then, this guard cannot see any of the right vertices of R_r. Therefore, the last guard must see both right vertices of R_r, and its height must be within $i_1 = [0.5, 0.6]$. Then, this guard cannot see any left vertex of the pocket R_ℓ, and the second guard must see both left vertices of the pocket, so its height must be within $i_2 = [1.7, 1.8]$. ◀

Dependencies between guard positions. Let $\{g_\ell, g_m, g_r\}$ be a guard set of $\mathcal{P}$ with $g_\ell \in l_\ell, g_m \in l_m$, and $g_r \in l_r$. We will now analyze dependencies between the positions of the guards that are caused by the quadrilateral pockets of $\mathcal{P}$. Recall that the non-axis-parallel edges of these pockets are denoted by e_t^ℓ, e_t^r, e_b^ℓ, and e_b^r.

We will first prove two technical lemmas.

▶ **Lemma 7.** *Let $h \in [0, 4]$ be the height of the guard g_ℓ. If $h > \frac{135}{47} \approx 2.87$ then g_ℓ cannot see any point on e_t^ℓ, and otherwise it can see a part of e_t^ℓ starting from the x-coordinate $\frac{908-188h}{181-47h}$ and to the right of it. If $h < \frac{9}{19} \approx 0.47$ then g_ℓ cannot see any point on e_b^ℓ, and otherwise it can see a part of e_b^ℓ starting from the x-coordinate $\frac{76h+12}{19h-3}$ and to the right of it.*

Proof. Consider the guard g_ℓ and the top-left pocket. The left-most point on e_t^ℓ that g_ℓ can see is at the intersection of the following two lines: the line containing g_ℓ and the bottom-left corner of the pocket (i.e., the point $(4, 4)$), and the line containing e_t^ℓ. If $g_\ell = (2, h)$, then the equation of the first line is $y = \frac{4-h}{2}x + (2h - 4)$. The second contains points $(4, \frac{280}{47})$ and $(8, \frac{294}{47})$, and its equation is $y = \frac{7}{94}x + \frac{266}{47}$. The x-coordinate of the intersection is $\frac{908-188h}{181-47h}$. It reaches a value of 8 (i.e., the point coincides with the right endpoint of e_t^ℓ) when $h = \frac{135}{47}$.

Now, consider the guard g_ℓ and the bottom-left pocket. The leftmost point on e_b^ℓ that g_ℓ can see is at the intersection of the following two lines: the line containing g_ℓ and the top-left corner of the pocket (i.e., the point $(4, 0)$), and the line containing e_b^ℓ. The first of these lines has equation $y = -\frac{h}{2}x + 2h$. The second line contains points $(4, -\frac{12}{19}), (8, -\frac{18}{19})$, and its equation is $y = -\frac{3}{38}x - \frac{6}{19}$. The x-coordinate of the intersection is $\frac{76h+12}{19h-3}$, which reaches 8 when $h = \frac{9}{19}$. ◀

▶ **Lemma 8.** *Let $h \in [0,4]$ be the height of the guard g_r. If $h > \frac{507}{250} = 2.028$ then g_ℓ cannot see any point on e_t^r, and otherwise it can see a part of e_t^r starting from the x-coordinate $\frac{4000h-9768}{250h-645}$ and to the left of it. If $h < \frac{17}{14} \approx 1.21$ then g_ℓ cannot see any point on e_b^r, and otherwise it can see a part of e_b^r starting from the x-coordinate $\frac{224h-56}{14h+1}$ and to the left of it.*

Proof. Consider the guard g_r and the top-right pocket. The right-most point on e_t^r that g_r can see is at the intersection of the following two lines: the line containing g_r and the bottom-right corner of the pocket (i.e., the point $(16,4)$), and the line containing e_t^r. If $g_r = (19,h)$, then the equation of the first line is $y = \frac{h-4}{3}x + \frac{76-16h}{3}$. The second contains points $(12, \frac{2486}{375})$ and $(16, \frac{1776}{375})$, and its equation is $y = -\frac{71}{150}x + \frac{4616}{375}$. The x-coordinate of the intersection is $\frac{4000h-9768}{250h-645}$. It reaches a value of 12 (i.e., the point coincides with the left endpoint of e_t^r) when $h = \frac{507}{250} = 2.028$.

Now, consider the guard g_r and the bottom-right pocket. The rightmost point on e_b^r that g_r can see is at the intersection of the following two lines: the line containing g_r and the top-right corner of the pocket (i.e., the point $(16,0)$), and the line containing e_b^r. The first of these lines has equation $y = \frac{h}{3}x - \frac{16h}{3}$. The second line contains points $(12, -\frac{34}{21}), (16, -\frac{36}{21})$, and its equation is $y = -\frac{1}{42}x - \frac{4}{3}$. The x-coordinate of the intersection is $\frac{224h-56}{14h+1}$, which reaches 12 when $h = \frac{17}{14} \approx 1.21$. ◀

We will now further restrict possible positions of the guards.

▶ **Lemma 9.** *The y-coordinate of the guard g_ℓ is in the interval $i_1 = [0.5, 0.6]$, and the y-coordinate of the guard g_r is in the interval $i_2 = [1.7, 1.8]$.*

Proof. As the guards g_ℓ and g_r lie on line segments l_ℓ and l_r, their x-coordinates are 2 and 19, respectively. From Lemma 6, the x-coordinate of g_m is in the interval $[10.5, 10.6]$. Also, one of the guards g_ℓ, g_r has a y-coordinate in i_1, and the other one in i_2.

Suppose that the y-coordinate of g_r is in i_1, i.e., it is at most 0.6. Let $v = (12, -\frac{34}{21})$ be the left endpoint of the edge e_b^r. We will show that none of the guards can see v. Clearly, as the x-coordinates of g_ℓ and g_m are smaller than 12, neither of them can see v. From Lemma 8, g_r cannot see v. Therefore, the y-coordinate of g_ℓ must be in i_1, and the y-coordinate of g_r in i_2. ◀

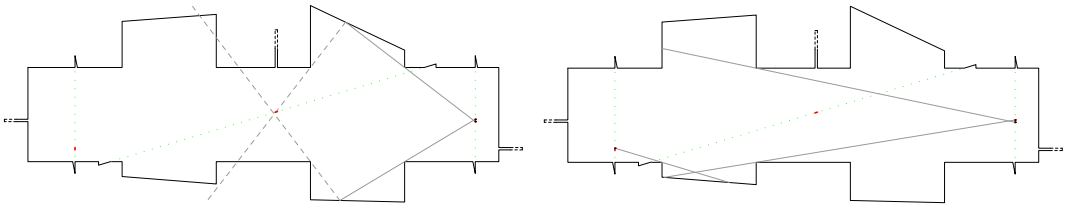
▶ **Lemma 10.** *The guards g_ℓ and g_m must together see all of e_t^ℓ and e_b^ℓ, and the guards g_m and g_r must together see all of e_t^r and e_b^r.*

Proof. By the construction of $\mathcal{P}$, it holds that if a guard sees a point on one of the edges e_t^ℓ, e_t^r, e_b^ℓ, and e_b^r, then the guard sees an interval of the edge containing an endpoint of the edge. It now follows that if three guards together see one of these edges, then two do as well. In order to prove the lemma, it thus suffices to prove that

- g_ℓ and g_r cannot together see any of the edges e_t^ℓ, e_b^ℓ, e_t^r, and e_b^r,
- g_ℓ and g_m cannot together see any of the right edges e_t^r and e_b^r, and
- g_m and g_r cannot together see any of the left edges e_t^ℓ and e_b^ℓ.

We now prove that g_ℓ and g_r cannot together see any of the right edges e_t^r and e_b^r (see Figure 6a). Since $h \in i_2$, Lemma 8 gives that g_r cannot see e_t^r to the right of the point $(\frac{742}{55}, \frac{1629}{275})$, and e_b^r to the right of the point $(\frac{1736}{131}, -\frac{216}{131})$. It is now easy to verify that no point on l_ℓ can see any of these two points. Hence, g_ℓ and g_r cannot together see any of the edges e_t^r and e_b^r.

We now prove that g_ℓ and g_r cannot together see e_t^ℓ (see Figure 6b). Since the y-coordinate of g_r is in i_2, it follows that g_r does not see any point on e_t^ℓ. Since the x-coordinate of g_ℓ is less than 4, neither g_ℓ nor g_r can see the left endpoint of e_t^ℓ.



(a) Guards g_ℓ and g_r cannot together see any of the right pockets.

(b) Guards g_ℓ and g_r cannot together see any of the left pockets.

Figure 6 Showing that guards g_ℓ and g_r cannot see together a whole pocket. Possible positions for the guards are pictured in red.

To show that g_ℓ and g_r cannot together see the edge e_b^ℓ, we argue as follows (see Figure 6b). The guard g_ℓ is placed at a height of at most 0.6, and g_r at a height of at most 1.8. It follows from Lemma 7 and from elementary computations that neither of the guards can see the interval of e_b^ℓ with x-coordinates between $\frac{2076}{507} < 4.1$ and $\frac{48}{7} > 6.8$.

As the x-coordinate of both g_ℓ and g_m is smaller than 12, none of these guards can see the left endpoint of the edges e_t^r, e_b^r. Therefore, g_ℓ and g_m cannot together see any of the edges e_t^r, e_b^r. Similarly, as the x-coordinates of g_m and g_r are greater than 8, g_m and g_r cannot together see e_t^ℓ or e_b^ℓ. This completes our proof. ◀

Computing the unique solution. We can now show that there is only one guard set for $\mathcal{P}$ consisting of three guards. Let us start by computing the right-most possible position of g_m such that g_ℓ and g_m can see together both left pockets.

▶ **Lemma 11.** *The maximum x-coordinate of g_m such that g_ℓ and g_m can together see e_t^ℓ and e_b^ℓ is $x = 3.5 + 5\sqrt{2}$. The corresponding position of g_ℓ is $(2, 2 - \sqrt{2})$.*

Proof. Consider the guard g_ℓ at position $(2, h)$. From Lemma 9, we know that $h \in [0.5, 0.6]$. If g_m and g_ℓ together see e_t^ℓ, we know from Lemma 7 that g_m has to be on or below the line containing the vertices $(8, 4)$ and $(\frac{908-188h}{181-47h}, \frac{7}{94} \cdot \frac{908-188h}{181-47h} + \frac{266}{47})$, i.e., the line with equation $y = \frac{92-23h}{-135+47h}x + \frac{-1276+372h}{-135+47h}$. As g_m is at the line $y = 0.3x - 1.05$, its x-coordinate satisfies $0.3x - 1.05 \leq \frac{92-23h}{-135+47h}x + \frac{-1276+372h}{-135+47h}$, i.e., $x \leq \frac{28355-8427h}{2650-742h}$.

If g_m and g_ℓ together see e_b^ℓ, then g_m has to be on or above the line containing the vertices $(8, 0)$ and $(\frac{76h+12}{19h-3}, -\frac{3}{38} \cdot \frac{76h+12}{19h-3} - \frac{6}{19})$, i.e., the line with equation $y = \frac{3h}{19h-9}x - \frac{24h}{19h-9}$. Hence, the x-coordinate of g_ℓ must satisfy $0.3x - 1.05 \geq \frac{3h}{19h-9}x - \frac{24h}{19h-9}$, i.e., $x(1-h) \leq \frac{81h+189}{54}$. Therefore, since $h < 1$, we must have $x \leq \frac{81h+189}{54-54h}$.

We now know that $x \leq \min\{\frac{28355-8427h}{2650-742h}, \frac{81h+189}{54-54h}\}$. The first of the two values decreases with h, and the second one increases with h. Therefore the maximum is obtained when $\frac{28355-8427h}{2650-742h} = \frac{81h+189}{54-54h}$, i.e., for $h = 2 - \sqrt{2}$. The value of x is then $3.5 + 5\sqrt{2}$. The corresponding position of the guard g_ℓ is $(2, h) = (2, 2 - \sqrt{2})$. ◀

Similarly, we can compute the left-most possible position of g_m such that g_m and g_r can see together both right pockets.

▶ **Lemma 12.** *The minimum x-coordinate of g_m such that g_r and g_m can see both e_t^r and e_b^r is $x = 3.5 + 5\sqrt{2}$. The corresponding position of g_r is $(19, 1 + \frac{\sqrt{2}}{2})$.*

Proof. Consider the guard g_r at position $(19, h)$. From Lemma 9, we know that $h \in [1.7, 1.8]$. If g_m and g_r together see e_t^r, we know from Lemma 8 that g_m has to be on or below the

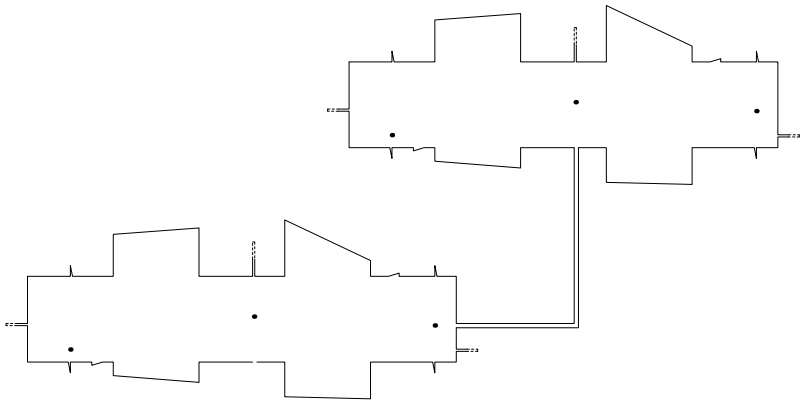


Figure 7 A sketch of a polygon that can be guarded by 6 guards when irrational coordinates are allowed, but needs 8 guards when only rational coordinates are allowed.

line containing the vertices $(12, 4)$ and $(\frac{4000h-9768}{250h-645}, -\frac{71}{150}\frac{4000h-9768}{250h-645} + \frac{4616}{375})$, i.e., the line with equation $y = \frac{46h-184}{250h-507}x + \frac{448h+180}{250h-507}$. As g_m is at the line $y = 0.3x - 1.05$, its x coordinate satisfies: $0.3x - 1.05 \leq \frac{46h-184}{250h-507}x + \frac{448h+180}{250h-507}$, i.e., $x \geq \frac{490h-243}{20h+22}$.

If g_m and g_r together see e_b^r, then g_m has to be on or above the line containing the vertices $(12, 0)$ and $(\frac{224h-56}{14h+1}, -\frac{1}{42}\frac{224h-56}{14h+1} - \frac{4}{3})$, i.e., the line with equation $y = \frac{6h}{17-14h}x - \frac{72h}{17-14h}$. Hence, the x-coordinate of g_r must satisfy $0.3x - 1.05 \geq \frac{6h}{17-14h}x - \frac{72h}{17-14h}$, i.e., $x \geq \frac{34h-7}{4h-2}$.

We have to minimize the value of $\max\{\frac{490h-243}{20h+22}, \frac{34h-7}{4h-2}\}$. When the value of h increases, the first of these two values increases, and the second one decreases. The minimum value is therefore obtained when $\frac{490h-243}{20h+22} = \frac{34h-7}{4h-2}$, i.e., for $h = 1 + \frac{\sqrt{2}}{2}$. The value of x is then $3.5 + 5\sqrt{2}$. ◀

We are now ready to prove our main theorems.

Proof of Theorem 1. Let $\mathcal{P}$ be the polygon constructed as in Section 2, and let S be a guard set for $\mathcal{P}$ consisting of at most 3 guards. From Lemma 5 we have $|S| = 3$, and there is one guard at each of the lines l_ℓ, l_m, l_r. Denote these guards by g_ℓ, g_m, g_r, respectively. From Lemma 10 we know that if g_ℓ, g_m, and g_r together see all of $\mathcal{P}$, then g_ℓ and g_m must see all of e_t^ℓ and e_b^ℓ, and g_m and g_r must see all of e_t^r and e_b^r. It then follows from Lemmas 11 and 12 that g_m must have coordinates $(3.5+5\sqrt{2}, 1.5\sqrt{2}) \approx (10.57, 2.12)$, $g_\ell = (2, 2-\sqrt{2}) \approx (2, 0.59)$, and $g_r = (19, 1 + \frac{\sqrt{2}}{2}) \approx (19, 1.71)$. Thus, indeed, the guards g_ℓ, g_m, and g_r see the entire polygon $\mathcal{P}$ and are the only three guards doing so.

By scaling $\mathcal{P}$ up by the least common multiple of the denominators in the coordinates of the corners of $\mathcal{P}$, we obtain a polygon with integer coordinates. This does not affect the number of guards required to see all of $\mathcal{P}$.

In order to guard $\mathcal{P}$ using four guards with rational coordinates, we choose two rational guards $g'_{m,1}$ and $g'_{m,2}$ on l_m a little bit to the left and to the right of g_m, respectively. The guard $g'_{m,1}$ sees a little more of both of the edges e_t^ℓ and e_b^ℓ than does g_m, whereas $g'_{m,2}$ sees a little more of e_t^r and e_b^r. Therefore, we can choose a rational guard g'_ℓ on l_ℓ close to g_ℓ such that g'_ℓ and $g'_{m,1}$ together see e_t^ℓ and e_b^ℓ, and a rational guard g'_r on l_r with analogous properties. Thus, $g'_\ell, g'_{m,1}, g'_{m,2}, g'_r$ guard $\mathcal{P}$. ◀

Proof of Theorem 2. We will now construct a polygon $\mathcal{P}_n$ that can be guarded by $3n$ guards placed at points with irrational coordinates, but such that when we restrict guard positions

to points with rational coordinates, the minimum number of guards becomes $4n$. We start by making n copies of the polygon $\mathcal{P}$ described above, which we denote by $\mathcal{P}^{(1)}, \ldots, \mathcal{P}^{(n)}$. We connect the copies into one polygon $\mathcal{P}_n$ as follows. Each consecutive pair $\mathcal{P}^{(i)}, \mathcal{P}^{(i+1)}$ is connected by a thin corridor consisting of a horizontal piece $H^{(i)}$ visible by the rightmost guard in $\mathcal{P}^{(i)}$, and a vertical piece $V^{(i)}$ visible to the middle guard in $\mathcal{P}^{(i+1)}$ (see Figure 7 for the case $n = 2$). We can then guard $\mathcal{P}_n$ using $3n$ guards, by placing three guards within each polygon $\mathcal{P}^{(i)}$ in the same way as for $\mathcal{P}$, i.e., at irrational points.

Now, assume that $\mathcal{P}_n$ can be guarded by at most $4n-1$ guards. We will show that at least one guard must be irrational. For formal reasons, we define $H^{(0)} = V^{(0)} = H^{(n)} = V^{(n)} = \emptyset$. The horizontal and vertical corridors $H^{(i)}$ and $V^{(i)}$, for $i \in \{0, \ldots, n\}$, intersect at a rectangular area $B^{(i)} = H^{(i)} \cap V^{(i)}$ which we call a *bend*. For $i \in \{1, \ldots, n-1\}$, the bend $B^{(i)}$ is non-empty and visible from both polygons $\mathcal{P}^{(i)}$ and $\mathcal{P}^{(i+1)}$. Define the *extension* of $\mathcal{P}^{(i)}$, denoted by $E(\mathcal{P}^{(i)})$, to be the union of $\mathcal{P}^{(i)}$ and the adjacent corridors excluding the bends, i.e., $E(\mathcal{P}^{(i)}) = \mathcal{P}^{(i)} \cup (V^{(i-1)} \setminus B^{(i-1)}) \cup (H^{(i)} \setminus B^{(i)})$. Since the extensions are pairwise disjoint, there is an extension $E(\mathcal{P}^{(i)})$ containing at most three guards. If there are no guards in any of the bends $B^{(i-1)}, B^{(i)}$ it follows from Theorem 1 that three guards must be placed inside $\mathcal{P}^{(i)}$ at irrational coordinates, so assume that there is a guard in one or both of the bends. If the adjacent corridors $V^{(i-1)}$ and $H^{(i)}$ are long enough and thin enough, a guard in the bends $B^{(i-1)}$ and $B^{(i)}$ cannot see any of the convex corners of $\mathcal{P}^{(i)}$ in the rectangular pockets, any point in a triangular pocket, or any point in a quadrilateral pocket. Hence, all the features of $\mathcal{P}^{(i)}$ that enforce the irrationality of the guards are unseen by the guards in the bends and it follows that there must be irrational guards in $\mathcal{P}^{(i)}$. Therefore, at least $4n$ guards are needed if we require them to be rational. Similarly as in the proof of Theorem 1, we can show that $4n$ rational guards are enough to guard $\mathcal{P}_n$. ◀

5 Rectilinear Polygon

Figure 8 depicts a rectilinear polygon $\mathcal{P}_R$ with corners at rational coordinates that can be guarded by 9 guards, but requires 10 guards if we restrict the guards to points with rational coordinates. The construction of $\mathcal{P}_R$ starts with the polygon $\mathcal{P}$ from Theorem 1. We extend the non-rectilinear parts by "equivalent" rectilinear parts, colored gray in the figure. The rectilinear pockets are constructed in such a way that each of them requires at least one guard in the interior. Additionally, if the interior of each pocket contains only one guard, then these guards must be placed at specific positions, making the area not seen by these six additional guards exactly the polygon $\mathcal{P}$ described in Section 2 (the white area in Figure 8). Thus, the remaining 3 guards must be placed at three irrational points by Theorem 1.

6 Future Work

One of the most prominent open questions related to the art gallery problem is whether the problem is in NP. Recently, some researchers popularized an interesting complexity class, called $\exists\mathbb{R}$, being somewhere between NP and PSPACE [8, 24, 7, 19]. Many geometric problems for which membership in NP is uncertain have been shown to be complete for the complexity class $\exists\mathbb{R}$. Famous examples are: order type realizability, pseudoline stretchability, recognition of segment intersection graphs, recognition of unit disk intersection graphs, recognition of point visibility graphs, minimizing rectilinear crossing number, linkage realizability. This suggests that there might indeed be no polynomial sized witness for any of these problems as this would imply NP = $\exists\mathbb{R}$. It is an interesting open problem whether the art gallery problem is $\exists\mathbb{R}$-complete or not.

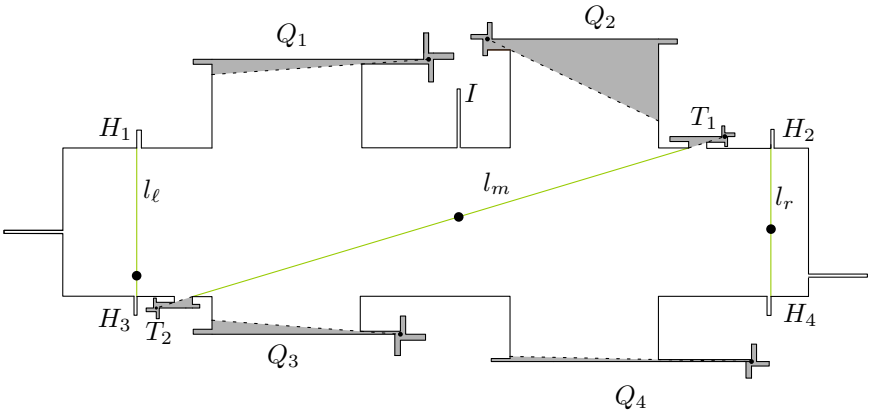


Figure 8 The rectilinear polygon $\mathcal{P}_R$ can be guarded with 9 guards only when we allow placing guards at irrational points.

The irrational coordinates of the guards in our examples are all of degree 2, i.e., they are roots in second-degree polynomials with integer coefficients. We would like to know if polygons exist where irrational numbers of higher degree are needed in the coordinates of an optimal solution.

We show that there exist polygons for which $|OPT_{\mathbb{Q}}| \geq \frac{4}{3}|OPT|$. It follows from the work by Bonnet and Miltzow [4] that it always holds that $|OPT_{\mathbb{Q}}| \leq 9|OPT|$. It is interesting to see if any of these bounds can be improved.

Acknowledgements. We want to thank Sándor Fekete, Frank Hoffmann, Udo Hoffmann, Linda Kleist, Péter Kutas, Günter Rote and Andrew Winslow for discussions on the problem and links to the literature. Special thanks goes to Michał Adamaszek for providing the sage code. We want to further thank the developers of the software GeoGebra. Being able to do computations and visualize parameter changes in real time facilitated our search tremendously.

References

1 Pankaj Kumar Agarwal, Kurt Mehlhorn, and Monique Teillaud. Dagstuhl Seminar 11111, Computational Geometry, March 13-18 , 2011.
2 Saugata Basu, Richard Pollack, and Marie-Françoise Roy. *Algorithms in real algebraic geometry*. Springer-Verlag Berlin Heidelberg, 2006.
3 Patrice Belleville. Computing two-covers of simple polygons. Master's thesis, McGill University, 1991.
4 Édouard Bonnet and Tillmann Miltzow. An approximation algorithm for the art gallery problem. *CoRR*, abs/1607.05527, 2016.
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 Björn Brodén, Mikael Hammar, and Bengt J. Nilsson. Guarding lines and 2-link polygons is APX-hard. In *Proceedings of the 13th Canadian Conference on Computational Geometry (CCCG)*, pages 45–48, 2001.
7 John Canny. Some algebraic and geometric computations in PSPACE. In *Proceedings of the twentieth annual ACM symposium on Theory of computing (STOC)*, pages 460–467. ACM, 1988.

8 Jean Cardinal. Computational geometry column 62. *SIGACT News*, 46(4):69–78, December 2015. doi:10.1145/2852040.2852053.

9 Vasek Chvátal. A combinatorial theorem in plane geometry. *Journal of Combinatorial Theory, Series B*, 18(1):39–41, 1975.

10 Pedro Jussieu 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*, LNCS, pages 379–417. Springer, 2016.

11 The Sage Developers. *SageMath, the Sage Mathematics Software System (Version 7.4)*, 2016. http://www.sagemath.org.

12 Alon Efrat and Sariel Har-Peled. Guarding galleries and terrains. *Inf. Process. Lett.*, 100(6):238–245, 2006.

13 Stephan Eidenbenz, Christoph Stamm, and Peter Widmayer. Inapproximability results for guarding polygons and terrains. *Algorithmica*, 31(1):79–113, 2001.

14 Sándor Fekete. Private communication.

15 Steve Fisk. A short proof of Chvátal's watchman theorem. *J. Comb. Theory, Ser. B*, 24(3):374, 1978.

16 Stephan Friedrichs, Michael Hemmer, James King, and Christiane Schmidt. The continuous 1.5D terrain guarding problem: Discretization, optimal solutions, and PTAS. *Journal of Computational Geometry*, 7(1):256–284, 2016.

17 Erik Krohn and Bengt J. Nilsson. Approximate guarding of monotone and rectilinear polygons. *Algorithmica*, 66(3):564–594, 2013.

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

19 Jiří Matoušek. Intersection graphs of segments and $\exists\mathbb{R}$. *CoRR*, abs/1406.2636, 2014.

20 Joseph O'Rourke. *Art Gallery Theorems and Algorithms*. Oxford University Press, 1987.

21 Joseph O'Rourke and Kenneth Supowit. Some NP-hard polygon decomposition problems. *IEEE Transactions on Information Theory*, 29(2):181–190, 1983.

22 Joseph O'Rourke. Visibility. In Jacob E. Goodman and Joseph O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, chapter 28. Chapman & Hall/CRC, second edition, 2004.

23 Günter Rote. EuroCG open problem session, 2011. See the personal webpage of Günter Rote: http://page.mi.fu-berlin.de/rote/Papers/slides/Open-Problem_artgallery-Morschach-EuroCG-2011.pdf.

24 Marcus Schaefer. Complexity of some geometric and topological problems. In *International Symposium on Graph Drawing*, pages 334–344. Springer, 2009.

25 Dietmar Schuchardt and Hans-Dietrich Hecker. Two NP-hard art-gallery problems for ortho-polygons. *Math. Log. Q.*, 41:261–267, 1995.

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

27 Ana Paula Tomás. Guarding thin orthogonal polygons is hard. In *Fundamentals of Computation Theory*, pages 305–316. Springer, 2013.

28 Jorge Urrutia. Art gallery and illumination problems. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, pages 973–1027. Elsevier, 2000.

Minimum Perimeter-Sum Partitions in the Plane*†

Mikkel Abrahamsen[1], Mark de Berg[2], Kevin Buchin[3], Mehran Mehr[4], and Ali D. Mehrabi[5]

1 **Department of Computer Science, University of Copenhagen, Copenhagen, Denmark**
miab@di.ku.dk
2 **Department of Computer Science, TU Eindhoven, Eindhoven, The Netherlands**
mdberg@win.tue.nl
2 **Department of Computer Science, TU Eindhoven, Eindhoven, The Netherlands**
k.a.buchin@tue.nl
2 **Department of Computer Science, TU Eindhoven, Eindhoven, The Netherlands**
m.mehr@tue.nl
2 **Department of Computer Science, TU Eindhoven, Eindhoven, The Netherlands**
amehrabi@win.tue.nl

Abstract

Let P be a set of n points in the plane. We consider the problem of partitioning P into two subsets P_1 and P_2 such that the sum of the perimeters of $\text{CH}(P_1)$ and $\text{CH}(P_2)$ is minimized, where $\text{CH}(P_i)$ denotes the convex hull of P_i. The problem was first studied by Mitchell and Wynters in 1991 who gave an $O(n^2)$ time algorithm. Despite considerable progress on related problems, no subquadratic time algorithm for this problem was found so far. We present an exact algorithm solving the problem in $O(n \log^4 n)$ time and a $(1+\varepsilon)$-approximation algorithm running in $O(n + 1/\varepsilon^2 \cdot \log^4(1/\varepsilon))$ time.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Computational geometry, clustering, minimum-perimeter partition, convex hull

Digital Object Identifier 10.4230/LIPIcs.SoCG.2017.4

1 Introduction

The clustering problem is to partition a given data set into clusters (that is, subsets) according to some measure of optimality. We are interested in clustering problems where the data set is a set P of points in Euclidean space. Most of these clustering problems fall into one of two categories: problems where the maximum cost of a cluster is given and the goal is to find a clustering consisting of a minimum number of clusters, and problems where the number of clusters is given and the goal is to find a clustering of minimum total cost. In this paper we consider a basic problem of the latter type, where we wish to find a bipartition (P_1, P_2) of a planar point set P. Bipartition problems are not only interesting in their own right, but also because bipartition algorithms can form the basis of hierarchical clustering methods.

* A full version of the paper is available at http://arxiv.org/abs/1703.05549.

† MA is partly supported by Mikkel Thorup's Advanced Grant from the Danish Council for Independent Research under the Sapere Aude research career programme. MdB, KB, MM, and AM are supported by the Netherlands' Organisation for Scientific Research (NWO) under project no. 024.002.003, 612.001.207, 022.005025, and 612.001.118 respectively.

There are many possible variants of the bipartition problem on planar point sets, which differ in how the cost of a clustering is defined. A variant that received a lot of attention is the 2-center problem [8, 11, 12, 15, 20], where the cost of a partition (P_1, P_2) of the given point set P is defined as the maximum of the radii of the smallest enclosing disks of P_1 and P_2. Other cost functions that have been studied include the maximum diameter of the two point sets [3] and the sum of the diameters [14]; see also the survey by Agarwal and Sharir [2] for some more variants.

A natural class of cost function considers the size of the convex hulls $\text{CH}(P_1)$ and $\text{CH}(P_2)$ of the two subsets, where the size of $\text{CH}(P_i)$ can either be defined as the area of $\text{CH}(P_i)$ or as the perimeter $\text{per}(P_i)$ of $\text{CH}(P_i)$. (The perimeter of $\text{CH}(P_i)$ is the length of the boundary $\partial\,\text{CH}(P_i)$.) This class of cost functions was already studied in 1991 by Mitchell and Wynters [17]. They studied four problem variants: minimize the sum of the perimeters, the maximum of the perimeters, the sum of the areas, or the maximum of the areas. In three of the four variants the convex hulls $\text{CH}(P_1)$ and $\text{CH}(P_2)$ in an optimal solution may intersect [17, full version] – only in the *minimum perimeter-sum problem* the optimal bipartition is guaranteed to be a so-called *line partition*, that is, a solution with disjoint convex hulls. For each of the four variants they gave an $O(n^3)$ algorithm that uses $O(n)$ storage and that computes computes an optimal line partition; for all except the minimum area-maximum problem they also gave an $O(n^2)$ algorithm that uses $O(n^2)$ storage. Note that (only) for the minimum perimeter-sum problem the computed solution is an optimal bipartition. Around the same time, the minimum-perimeter sum problem was studied for partitions into k subsets for $k > 2$; for this variant Capoyleas *et al.* [7] presented an algorithm with running time $O(n^{6k})$. Mitchell and Wynters mentioned the improvement of the space requirement of the quadratic-time algorithm as an open problem, and they stated the existence of a subquadratic algorithm for any of the four variants as the most prominent open problem.

Rokne *et al.* [18] made progress on the first question, by presenting an $O(n^2 \log n)$ algorithm that uses only $O(n)$ space for the line-partition version of each of the four problems. Devillers and Katz [10] gave algorithms for the min-max variant of the problem, both for area and perimeter, which run in $O((n+k)\log^2 n)$ time. Here k is a parameter that is only known to be in $O(n^2)$, although Devillers and Katz suspected that k is subquadratic. They also gave linear-time algorithms for these problems when the point set P is in convex position and given in cyclic order. Segal [19] proved an $\Omega(n \log n)$ lower bound for the min-max problems. Very recently, and apparently unaware of some of the earlier work on these problems, Bae *et al.* [4] presented an $O(n^2 \log n)$ time algorithm for the minimum-perimeter-sum problem and an $O(n^4 \log n)$ time algorithm for the minimum-area-sum problem (considering all partitions, not only line partitions). Despite these efforts, the main question is still open: is it possible to obtain a subquadratic algorithm for any of the four bipartition problems based on convex-hull size?

1.1 Our contribution

We answer the question above affirmatively by presenting a subquadratic algorithm for the minimum perimeter-sum bipartition problem in the plane.

As mentioned, an optimal solution (P_1, P_2) to the minimum perimeter-sum bipartition problem must be a line partition. A straightforward algorithm would generate all $\Theta(n^2)$ line partitions and compute the value $\text{per}(P_1)+\text{per}(P_2)$ for each of them. If the latter is done from scratch for each partition, the resulting algorithm runs in $O(n^3 \log n)$ time. The algorithms by Mitchell and Wynters [17] and Rokne *et al.* [18] improve on this by using that the different

line bipartitions can be generated in an ordered way, such that subsequent line partitions differ in at most one point. Thus the convex hulls do not have to be recomputed from scratch, but they can be obtained by updating the convex hulls of the previous bipartition. To obtain a subquadratic algorithm a fundamentally new approach is necessary: we need a strategy that generates a subquadratic number of candidate partitions, instead considering all line partitions. We achieve this as follows.

We start by proving that an optimal bipartition (P_1, P_2) has the following property: either there is a set of $O(1)$ canonical orientations such that P_1 can be separated from P_2 by a line with a canonical orientation, or the distance between $\textsc{ch}(P_1)$ and $\textsc{ch}(P_2)$ is $\Omega(\min(\text{per}(P_1), \text{per}(P_2)))$. There are only $O(1)$ bipartitions of the former type, and finding the best among them is relatively easy. The bipartitions of the second type are much more challenging. We show how to employ a compressed quadtree to generate a collection of $O(n)$ canonical 5-gons – intersections of axis-parallel rectangles and canonical halfplanes – such that the smaller of $\textsc{ch}(P_1)$ and $\textsc{ch}(P_2)$ (in a bipartition of the second type) is contained in one of the 5-gons.

It then remains to find the best among the bipartitions of the second type. Even though the number of such bipartitions is linear, we cannot afford to compute their perimeters from scratch. We therefore design a data structure to quickly compute $\text{per}(P \cap Q)$, where Q is a query canonical 5-gon. Brass *et al.* [6] presented such a data structure for the case where Q is an axis-parallel rectangle. Their structure uses $O(n \log^2 n)$ space and has $O(\log^5 n)$ query time; it can be extended to handle canonical 5-gons as queries, at the cost of increasing the space usage to $O(n \log^3 n)$ and the query time to $O(\log^7 n)$. Our data structure improves upon this: it has $O(\log^4 n)$ query time for canonical 5-gons (and $O(\log^3 n)$ for rectangles) while using the same amount of space. Using this data structure to find the best bipartition of the second type we obtain our main result: an exact algorithm for the minimum perimeter-sum bipartition problem that runs in $O(n \log^4 n)$ time. As our model of computation we use the real RAM (with the capability of taking square roots) so that we can compute the exact perimeter of a convex polygon – this is necessary to compare the costs of two competing clusterings. We furthermore make the (standard) assumption that the model of computation allows us to compute a compressed quadtree of n points in $O(n \log n)$ time; see footnote 2 on page 10.

Besides our exact algorithm, we present a linear-time $(1+\varepsilon)$-approximation algorithm. Its running time is $O(n + T(1/\varepsilon^2)) = O(n + 1/\varepsilon^2 \cdot \log^4(1/\varepsilon))$, where $T(1/\varepsilon^2)$ is the running time of an exact algorithm on an instance of size $1/\varepsilon^2$.

Some arguments are omitted due to limited space. See the full version [1] for the details.

2 The exact algorithm

In this section we present an exact algorithm for the minimum-perimeter-sum partition problem. We first prove a separation property that an optimal solution must satisfy, and then we show how to use this property to develop a fast algorithm.

Let P be the set of n points in the plane for which we want to solve the minimum-perimeter-sum partition problem. An optimal partition (P_1, P_2) of P has the following two basic properties: P_1 and P_2 are non-empty, and the convex hulls $\textsc{ch}(P_1)$ and $\textsc{ch}(P_2)$ are disjoint [17, full version]. In the remainder, whenever we talk about a partition of P, we refer to a partition with these two properties.

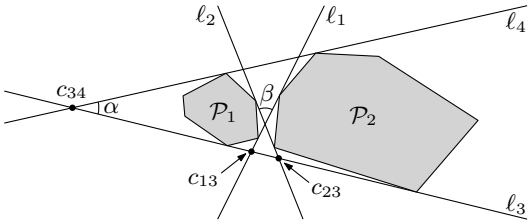


Figure 1 The angles α and β.

2.1 Geometric properties of an optimal partition

Consider a partition (P_1, P_2) of P. Define $\mathcal{P}_1 := \text{CH}(P_1)$ and $\mathcal{P}_2 := \text{CH}(P_2)$ to be the convex hulls of P_1 and P_2, respectively, and let ℓ_1 and ℓ_2 be the two inner common tangents of $\mathcal{P}_1$ and $\mathcal{P}_2$. The lines ℓ_1 and ℓ_2 define four wedges: one containing P_1, one containing P_2, and two empty wedges. We call the opening angle of the empty wedges the *separation angle* of P_1 and P_2. Furthermore, we call the distance between $\mathcal{P}_1$ and $\mathcal{P}_2$ the *separation distance* of P_1 and P_2.

▶ **Theorem 1.** *Let P be a set of n points in the plane, and let (P_1, P_2) be a partition of P that minimizes $\text{per}(P_1) + \text{per}(P_2)$. Then the separation angle of P_1 and P_2 is at least $\pi/6$ or the separation distance is at least $c_{\text{sep}} \cdot \min(\text{per}(P_1), \text{per}(P_2))$, where $c_{\text{sep}} := 1/250$.*

The remainder of this section is devoted to proving Theorem 1. To this end let (P_1, P_2) be a partition of P that minimizes $\text{per}(P_1) + \text{per}(P_2)$. Let ℓ_3 and ℓ_4 be the outer common tangents of $\mathcal{P}_1$ and $\mathcal{P}_2$. We define α to be the angle between ℓ_3 and ℓ_4. More precisely, if ℓ_3 and ℓ_4 are parallel we define $\alpha := 0$, otherwise we define α as the opening angle of the wedge defined by ℓ_3 and ℓ_4 containing $\mathcal{P}_1$ and $\mathcal{P}_2$. We denote the separation angle of P_1 and P_2 by β; see Fig. 1.

The idea of the proof is as follows. Suppose that the separation distance and the separation angle β are both relatively small. Then the region A in between $\mathcal{P}_1$ and $\mathcal{P}_2$ and bounded from the bottom by ℓ_3 and from the top by ℓ_4 is relatively narrow. But then the left and right parts of ∂A (which are contained in $\partial\mathcal{P}_1$ and $\partial\mathcal{P}_2$) would be longer than the bottom and top parts of ∂A (which are contained in ℓ_3 and ℓ_4), thus contradicting that (P_1, P_2) is an optimal partition. To make this idea precise, we first prove that if the separation angle β is small, then the angle α between ℓ_3 and ℓ_4 must be large. Second, we show that there is a value $f(\alpha)$ such that the distance between $\mathcal{P}_1$ and $\mathcal{P}_2$ is at least $f(\alpha) \cdot \min(\text{per}(P_1), \text{per}(P_2))$. Finally we argue that this implies that if the separation angle is smaller than $\pi/6$, then (to avoid the contradiction mentioned above) the separation distance must be relatively large. Next we present our proof in detail.

Let c_{ij} be the intersection point between ℓ_i and ℓ_j, where $i < j$. If ℓ_3 and ℓ_4 are parallel, we choose c_{34} as a point at infinity on ℓ_3. Assume without loss of generality that neither ℓ_1 nor ℓ_2 separate $\mathcal{P}_1$ from c_{34}, and that ℓ_3 is the outer common tangent such that $\mathcal{P}_1$ and $\mathcal{P}_2$ are to the left of ℓ_3 when traversing ℓ_3 from c_{34} to an intersection point in $\ell_3 \cap \mathcal{P}_1$. Assume furthermore that c_{13} is closer to c_{34} than c_{23}.

For two lines, rays, or segments r_1, r_2, let $\angle(r_1, r_2)$ be the angle we need to rotate r_1 in counterclockwise direction until r_1 and r_2 are parallel. For three points a, b, c, let $\angle(a, b, c) := \angle(ba, bc)$. For $i = 1, 2$ and $j = 1, 2, 3, 4$, let s_{ij} be a point in $P_i \cap \ell_j$. Let $\partial\mathcal{P}_i$ denote the boundary of $\mathcal{P}_i$ and $\text{per}(\mathcal{P}_i)$ the perimeter of $\mathcal{P}_i$. Furthermore, let $\partial\mathcal{P}_i(x, y)$ denote the portion of $\partial\mathcal{P}_i$ from $x \in \partial\mathcal{P}_i$ counterclockwise to $y \in \partial\mathcal{P}_i$, and $\text{length}(\partial\mathcal{P}_i(x, y))$ denote the length of $\partial\mathcal{P}_i(x, y)$.

▶ **Lemma 2.** *We have* $\alpha + 3\beta \geqslant \pi$.

Proof. Since $\mathrm{per}(\mathcal{P}_1) + \mathrm{per}(\mathcal{P}_2)$ is minimum, we know that

$$\mathrm{length}(\partial\mathcal{P}_1(s_{13}, s_{14})) + \mathrm{length}(\partial\mathcal{P}_2(s_{24}, s_{23})) \;\leqslant\; \Psi,$$

where $\Psi := |s_{13}s_{23}| + |s_{14}s_{24}|$. Furthermore, we know that $s_{11}, s_{12} \in \partial\mathcal{P}_1(s_{13}, s_{14})$ and $s_{21}, s_{22} \in \partial\mathcal{P}_1(s_{24}, s_{23})$. We thus have

$$\mathrm{length}(\partial\mathcal{P}_1(s_{13}, s_{14})) + \mathrm{length}(\partial\mathcal{P}_2(s_{24}, s_{23})) \;\geqslant\; \Phi,$$

where $\Phi := |s_{13}s_{11}| + |s_{11}s_{12}| + |s_{12}s_{14}| + |s_{24}s_{21}| + |s_{21}s_{22}| + |s_{22}s_{23}|$. Hence, we must have

$$\Phi \;\leqslant\; \Psi. \tag{1}$$

Now assume that $\alpha + 3\beta < \pi$. We will show that this assumption, together with inequality (1), leads to a contradiction, thus proving the lemma. To this end we will argue that if (1) holds, then it must also hold when (i) s_{21} or s_{22} coincides with c_{12}, and (ii) s_{11} or s_{12} coincides with c_{12}. To finish the proof it then suffices to observe that that if (i) and (ii) hold, then $\mathcal{P}_1$ and $\mathcal{P}_2$ touch in c_{12} and so (1) contradicts the triangle inequality.

It remains to argue that if (1) holds, then we can create a situation where (1) holds and (i) and (ii) hold as well. To this end we ignore that the points s_{ij} are specific points in the set P and allow the point s_{ij} to move on the tangent ℓ_j, as long as the movement preserves (1). Moving s_{13} along ℓ_3 away from s_{23} increases Ψ more than it increases Φ, so (1) is preserved. Similarly, we can move s_{14} away from s_{24}, s_{23} away from s_{13}, and s_{24} away from s_{14}.

We first show how to create a situation where (i) holds, and (1) still holds as well. Let $\gamma_{ij} := \angle(\ell_i, \ell_j)$. We consider two cases.

- *Case* (A)*:* $\gamma_{32} < \pi - \beta$.
  Note that $\angle(xs_{23}, \ell_2) \geqslant \gamma_{32}$ for any $x \in s_{22}c_{12}$. However, by moving s_{23} sufficiently far away we can make $\angle(xs_{23}, \ell_2)$ arbitrarily close to γ_{32}, and we can ensure that $\angle(xs_{23}, \ell_2) < \pi - \beta$ for any point $x \in s_{22}c_{12}$. We now let the point x move at unit speed from s_{22} towards c_{12}. To be more precise, let $T := |s_{22}c_{12}|$, let $\mathbf{v}$ be the unit vector with direction from c_{23} to c_{12}, and for any $t \in [0, T]$ define $x(t) := s_{22} + t \cdot \mathbf{v}$. Note that $x(0) = s_{22}$ and $x(T) = c_{12}$.
  Let $a(t) := |x(t)s_{23}|$ and $b(t) := |x(t)s_{21}|$. In the full version [1] we show that

  $$a'(t) = -\cos(\angle(x(t)s_{23}, \ell_2)) \text{ and } b'(t) = \cos(\angle(\ell_2, x(t)s_{21})).$$

  Since $\angle(x(t)s_{23}, \ell_2) < \pi - \beta$ for any value $t \in [0, T]$, we get $a'(t) < -\cos(\pi - \beta)$. Furthermore, we have $\angle(\ell_2, x(t)s_{21}) \geqslant \pi - \beta$ and hence $b'(t) \leqslant \cos(\pi - \beta)$. Therefore, $a'(t) + b'(t) < 0$ for any t and we conclude that $a(T) + b(T) \leqslant a(0) + b(0)$. This is the same as $|s_{21}c_{12}| + |c_{12}s_{23}| \leqslant |s_{21}s_{22}| + |s_{22}s_{23}|$, so (1) still holds when we substitute s_{22} by c_{12}.
- *Case* (B)*:* $\gamma_{32} \geqslant \pi - \beta$.
  Using our assumption $\alpha + 3\beta < \pi$ we get $\gamma_{32} > \alpha + 2\beta$. Note that $\gamma_{14} = \pi - \gamma_{32} + \alpha + \beta$. Hence, $\gamma_{14} < \pi - \beta$. By moving s_{24} and s_{21}, we can in a similar way as in Case (A) argue that (1) still holds when we substitute s_{21} by c_{12}.

We conclude that in both cases we can ensure (i) without violating (1).

Since $\gamma_{42} \leqslant \gamma_{32}$ and $\gamma_{13} \leqslant \gamma_{14}$, we likewise have $\gamma_{42} < \pi - \beta$ or $\gamma_{13} < \pi - \beta$. Hence, we can substitute s_{11} or s_{12} by c_{12} without violating (1), thus ensuring (ii) and finishing the proof. ◀

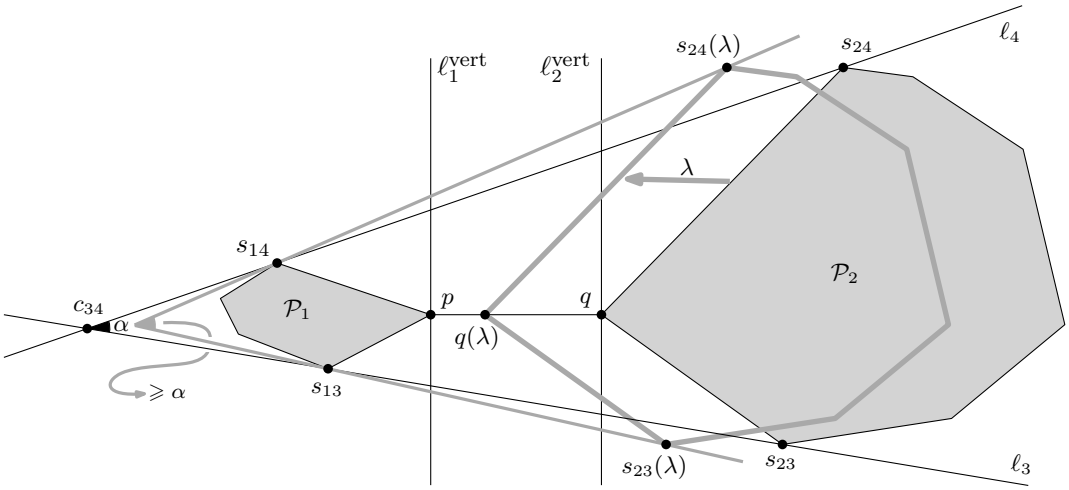


Figure 2 Illustration for the proof of Lemma 3.

Let $\text{dist}(\mathcal{P}_1, \mathcal{P}_2) := \min_{(p,q)\in\mathcal{P}_1\times\mathcal{P}_2}|pq|$ denote the separation distance between $\mathcal{P}_1$ and $\mathcal{P}_2$. Recall that α denotes the angle between the two common outer tangents of $\mathcal{P}_1$ and $\mathcal{P}_2$; see Fig. 1

▶ **Lemma 3.** *We have*

$$\textit{dist}(\mathcal{P}_1, \mathcal{P}_2) \;\geqslant\; f(\alpha)\cdot \text{per}(\mathcal{P}_1), \tag{2}$$

where $f\colon [0,\pi] \longrightarrow \mathbb{R}$ *is the increasing function*

$$f(\varphi) \;:=\; \frac{\sin(\varphi/4)}{1+\sin(\varphi/4)}\cdot\frac{\sin(\varphi/2)}{1+\sin(\varphi/2)}\cdot\frac{1-\cos(\varphi/4)}{2}.$$

Proof. The statement is trivial if $\alpha = 0$ so assume $\alpha > 0$. Let $p \in \mathcal{P}_1$ and $q \in \mathcal{P}_2$ be points so that $|pq| = \text{dist}(\mathcal{P}_1, \mathcal{P}_2)$ and assume without loss of generality that pq is a horizontal segment with p being its left endpoint. Let ℓ_1^{vert} and ℓ_2^{vert} be vertical lines containing p and q, respectively. Note that $\mathcal{P}_1$ is in the closed half-plane to the left of ℓ_1^{vert} and $\mathcal{P}_2$ is in the closed half-plane to the right of ℓ_2^{vert}. Recall that s_{ij} denotes a point on $\partial\mathcal{P}_i \cap \ell_j$.

▶ **Claim 4.** *There exist two convex polygons* $\mathcal{P}_1'$ *and* $\mathcal{P}_2'$ *satisfying the following conditions:*

1. $\mathcal{P}_1'$ *and* $\mathcal{P}_2'$ *have the same outer common tangents as* $\mathcal{P}_1$ *and* $\mathcal{P}_2$*, namely* ℓ_3 *and* ℓ_4*.*
2. $\mathcal{P}_1'$ *is to the left of* ℓ_1^{vert} *and* $p \in \partial\mathcal{P}_1'$*; and* $\mathcal{P}_2'$ *is to right of* ℓ_2^{vert} *and* $q \in \partial\mathcal{P}_2'$*.*
3. $\text{per}(\mathcal{P}_1') = \text{per}(\mathcal{P}_1)$*.*
4. $\text{per}(\mathcal{P}_1') + \text{per}(\mathcal{P}_2') \leqslant \text{per}(\text{CH}(\mathcal{P}_1' \cup \mathcal{P}_2'))$*.*
5. *There are points* $s_{ij}' \in \mathcal{P}_i' \cap \ell_j$ *for all* $i \in \{1,2\}$ *and* $j \in \{3,4\}$ *such that* $\partial\mathcal{P}_1'(s_{13}', p)$*,* $\partial\mathcal{P}_1'(p, s_{14}')$*,* $\partial\mathcal{P}_2'(s_{24}', q)$*, and* $\partial\mathcal{P}_2'(q, s_{23}')$ *each consist of a single line segment.*
6. *Let* $s_{2j}'(\lambda) := s_{2j}' - (\lambda, 0)$ *and let* $\ell_j'(\lambda)$ *be the line through* s_{1j}' *and* $s_{2j}'(\lambda)$ *for* $j \in \{3,4\}$*. Then* $\angle(\ell_3'(|pq|), \ell_4'(|pq|)) \geqslant \alpha/2$*.*

Proof of the Claim. Let $\mathcal{P}_1' := \mathcal{P}_1$ and $\mathcal{P}_2' := \mathcal{P}_2$, and let s_{ij}' be a point in $\mathcal{P}_i' \cap \ell_j$ for all $i \in \{1,2\}$ and $j \in \{3,4\}$. We show how to modify $\mathcal{P}_1'$ and $\mathcal{P}_2'$ until they have all the required conditions. Of course, they already satisfy conditions 1–4. We first show how to obtain condition 5, namely that $\partial\mathcal{P}_1'(s_{13}', p)$ and $\partial\mathcal{P}_1'(p, s_{14}')$ – and similarly $\partial\mathcal{P}_2'(s_{24}', q)$ and

$\partial\mathcal{P}_1'(q, s_{23}')$ – each consist of a single line segment, as depicted in Fig. 2. To this end, let v_{ij} be the intersection point $\ell_i^{\text{vert}} \cap \ell_j$ for $i \in \{1,2\}$ and $j \in \{3,4\}$. Let $s' \in s_{14}'v_{14}$ be the point such that $\text{length}(\partial\mathcal{P}_1'(p, s_{14}')) = |ps'| + |s's_{14}'|$. Such a point exists since

$$|ps_{14}'| \leqslant \text{length}(\partial\mathcal{P}_1'(p, s_{14}')) \leqslant |pv_{14}| + |v_{14}s_{14}'|.$$

We modify $\mathcal{P}_1'$ by substituting $\partial\mathcal{P}_1'(p, s_{14}')$ with the segments ps' and $s's_{14}'$. We can now redefine $s_{14}' := s'$ so that $\partial\mathcal{P}_1'(p, s_{14}') = ps_{14}'$ is a line segment. We can modify $\mathcal{P}_1'$ in a similar way to ensure that $\partial\mathcal{P}_1'(s_{13}', p) = s_{13}'p$, and we can modify $\mathcal{P}_2'$ to ensure $\partial\mathcal{P}_2'(s_{24}', q) = s_{24}'q$ and $\partial\mathcal{P}_2'(q, s_{23}') = qs_{23}'$. Note that these modifications preserve conditions 1–4 and that condition 5 is now satisfied.

The only condition that $(\mathcal{P}_1', \mathcal{P}_2')$ might not satisfy is condition 6. Let $s_{2j}'(\lambda) := s_{2j}' - (\lambda, 0)$ and let $\ell_j(\lambda)$ be the line through $s_{2j}'(\lambda)$ and s_{1j}' for $j \in \{3,4\}$. Clearly, if the slopes of ℓ_3 and ℓ_4 have different signs (as in Fig. 2), the angle $\angle(\ell_3(\lambda), \ell_4(\lambda))$ is increasing for $\lambda \in [0, |pq|]$, and condition 6 is satisfied. However, if the slopes of ℓ_3 and ℓ_4 have the same sign, the angle might decrease.

Consider the case where both slopes are positive – the other case is analogous. Changing $\mathcal{P}_2'$ by substituting $\partial\mathcal{P}_2'(s_{23}', s_{24}')$ with the line segment $s_{23}'s_{24}'$ makes $\text{per}(\mathcal{P}_1') + \text{per}(\mathcal{P}_2')$ and $\text{per}(\text{CH}(\mathcal{P}_1' \cup \mathcal{P}_2'))$ decrease equally much and hence condition 4 is preserved. This clearly has no influence on the other conditions. We thus assume that $\mathcal{P}_2'$ is the triangle $qs_{23}'s_{24}'$. Consider what happens if we move s_{23}' along the line ℓ_3 away from c_{34} with unit speed. Then $|s_{13}'s_{23}'|$ grows with speed exactly 1 whereas $|qs_{23}'|$ grows with speed at most 1. We therefore preserve condition 4, and the other conditions are likewise not affected.

We now move s_{23}' sufficiently far away so that $\angle(\ell_3, \ell_3(|pq|)) \leqslant \alpha/4$. Similarly, we move s_{24}' sufficiently far away from c_{34} along ℓ_4 to ensure that $\angle(\ell_4, \ell_4(|pq|)) \leqslant \alpha/4$. It then follows that $\angle(\ell_3(|pq|), \ell_4(|pq|)) \geqslant \angle(\ell_3, \ell_4) - \alpha/2 = \alpha/2$, and condition 6 is satisfied. ◀

Note that condition 2 in the claim implies that $\text{dist}(\mathcal{P}_1', \mathcal{P}_2') = \text{dist}(\mathcal{P}_1, \mathcal{P}_2) = |pq|$, and hence inequality (2) follows from condition 3 if we manage to prove $\text{dist}(\mathcal{P}_1', \mathcal{P}_2') \geqslant f(\alpha) \cdot \text{per}(\mathcal{P}_1')$. Therefore, with a slight abuse of notation, we assume from now on that $\mathcal{P}_1$ and $\mathcal{P}_2$ satisfy the conditions in the claim, where the points s_{ij} play the role as s_{ij}' in conditions 5 and 6.

We now consider a copy of $\mathcal{P}_2$ that is translated horizontally to the left over a distance λ; see Fig. 2. Let $s_{24}(\lambda)$, $s_{23}(\lambda)$, and $q(\lambda)$ be the translated copies of s_{24}, s_{23}, and q, respectively, and let $\ell_j(\lambda)$ be the line through s_{1j} and $s_{2j}(\lambda)$ for $j \in \{3,4\}$. Furthermore, define

$$\Phi(\lambda) := |s_{13}p| + |s_{14}p| + |s_{23}(\lambda)q(\lambda)| + |s_{24}(\lambda)q(\lambda)|$$

and

$$\Psi(\lambda) := |s_{13}s_{23}(\lambda)| + |s_{14}s_{24}(\lambda)|.$$

Note that $\Phi(\lambda) = \Phi$ is constant. By conditions 4 and 5, we know that

$$\Phi \leqslant \Psi(0). \tag{3}$$

Note that $q(|pq|) = p$. In the full version [1] we show that

$$\Phi - \Psi(|pq|) \geqslant \sin(\delta/2) \cdot \frac{1 - \cos(\delta/2)}{1 + \sin(\delta/2)} \cdot (|s_{13}p| + |s_{14}p|), \tag{4}$$

where $\delta := \angle(\ell_3(|pq|), \ell_4(|pq|))$. By condition 6, we know that $\delta \geqslant \alpha/2$. The function $\delta \longmapsto \sin(\delta/2) \cdot \frac{1-\cos(\delta/2)}{1+\sin(\delta/2)}$ is increasing for $\delta \in [0, \pi]$ and hence inequality (4) also holds for $\delta = \alpha/2$.

When λ increases from 0 to $|pq|$ with unit speed, the value $\Psi(\lambda)$ decreases with speed at most 2, i.e., $\Psi(\lambda) \geqslant \Psi(0) - 2\lambda$. Using this and inequalities (3) and (4), we get

$$2|pq| \;\geqslant\; \Psi(0) - \Psi(|pq|) \;\geqslant\; \Phi - \Phi + \sin(\alpha/4) \cdot \frac{1-\cos(\alpha/4)}{1+\sin(\alpha/4)} \cdot (|s_{13}p| + |s_{14}p|),$$

and we conclude that

$$|pq| \;\geqslant\; \frac{1}{2} \cdot \sin(\alpha/4) \cdot \frac{1-\cos(\alpha/4)}{1+\sin(\alpha/4)} \cdot (|s_{13}p| + |s_{14}p|). \tag{5}$$

By the triangle inequality, $|s_{13}p| + |s_{14}p| \geqslant |s_{13}s_{14}|$. Furthermore, for a given length of $s_{13}s_{14}$, the fraction $|s_{13}s_{14}|/(|s_{14}c_{34}| + |c_{34}s_{13}|)$ is minimized when $s_{13}s_{14}$ is perpendicular to the angular bisector of ℓ_3 and ℓ_4. (Recall that c_{34} is the intersection point of the outer common tangents ℓ_3 and ℓ_4; see Fig. 2.) Hence

$$|s_{13}s_{14}| \;\geqslant\; \sin(\alpha/2) \cdot (|s_{14}c_{34}| + |c_{34}s_{13}|)\,. \tag{6}$$

We now conclude

$$\begin{aligned}
|s_{13}p| + |s_{14}p| &= \tfrac{\sin(\alpha/2)}{1+\sin(\alpha/2)} \cdot \left(\tfrac{|s_{13}p|+|s_{14}p|}{\sin(\alpha/2)} + |s_{13}p| + |s_{14}p| \right) \\
&\geqslant \tfrac{\sin(\alpha/2)}{1+\sin(\alpha/2)} \cdot \left(\tfrac{|s_{13}s_{14}|}{\sin(\alpha/2)} + |s_{13}p| + |s_{14}p| \right) \qquad \text{by the triangle inequality} \\
&\geqslant \tfrac{\sin(\alpha/2)}{1+\sin(\alpha/2)} \cdot \left(|s_{14}c_{34}| + |c_{34}s_{13}| + |s_{13}p| + |s_{14}p| \right) \qquad \text{by (6)} \\
&\geqslant \tfrac{\sin(\alpha/2)}{1+\sin(\alpha/2)} \cdot \mathrm{per}(\mathcal{P}_1),
\end{aligned}$$

where the last inequality follows because $\mathcal{P}_1$ is fully contained in the quadrilateral $s_{14}, c_{34}, x_{13}, p$. The statement (2) in the lemma now follows from (5). ◀

We are now ready to prove Theorem 1.

Proof of Theorem 1. If the separation angle of P_1 and P_2 is at least $\pi/6$, we are done. Otherwise, Lemma 2 gives that $\alpha > \pi/2$, and Lemma 3 gives that $\mathrm{dist}(\mathcal{P}_1, \mathcal{P}_2) \;\geqslant\; f(\pi/2) \cdot \mathrm{per}(\mathcal{P}_1) \geqslant (1/250) \cdot \min(\mathrm{per}(\mathcal{P}_1), \mathrm{per}(\mathcal{P}_2))$. ◀

2.2 The algorithm

Theorem 1 suggests to distinguish two cases when computing an optimal partition: the case where the separation angle is large (namely at least $\pi/6$) and the case where the separation distance is large (namely at least $c_{\mathrm{sep}} \cdot \min(\mathrm{per}(P_1), \mathrm{per}(P_2))$). As we will see, the first case can be handled in $O(n \log n)$ time and the second case in $O(n \log^4 n)$ time, leading to the following theorem.

▶ **Theorem 5.** *Let P be a set of n points in the plane. Then we can compute a partition (P_1, P_2) of P that minimizes $\mathrm{per}(P_1) + \mathrm{per}(P_2)$ in $O(n \log^4 n)$ time using $O(n \log^3 n)$ space.*

To find the best partition when the separation angle is at least $\pi/6$, we observe that in this case there is a separating line whose orientation is $j \cdot \pi/7$ for some $0 \leqslant j < 7$. For each of these orientations we can scan over the points with a line ℓ of the given orientation, and maintain the perimeters of the convex hulls on both sides. This takes $O(n \log n)$ time in total; see the full version [1].

Next we show how to compute the best partition with large separation distance. We assume without loss of generality that $\mathrm{per}(P_2) \leqslant \mathrm{per}(P_1)$. It will be convenient to treat the case where P_2 is a singleton separately.

▶ **Lemma 6.** *The point $p \in P$ minimizing $\mathrm{per}(P \setminus \{p\})$ can be computed in $O(n \log n)$ time.*

Proof. The point p we are looking for must be a vertex of $\mathrm{CH}(P)$. First we compute $\mathrm{CH}(P)$ in $O(n \log n)$ time [5]. Let $v_0, v_1, \ldots, v_{m-1}$ denote the vertices of $\mathrm{CH}(P)$ in counterclockwise order. Let Δ_i be the triangle with vertices $v_{i-1}v_iv_{i+1}$ (with indices taken modulo m) and let P_i denote the set of points lying inside Δ_i, excluding v_i but including v_{i-1} and v_{i+1}. Note that any point $p \in P$ is present in at most two sets P_i. Hence, $\sum_{i=0}^{m} |P_i| = O(n)$. It is not hard to compute the sets P_i in $O(n \log n)$ time in total. After doing so, we compute all convex hulls $\mathrm{CH}(P_i)$ in $O(n \log n)$ time in total. Since

$$\mathrm{per}(P \setminus \{v_i\}) = \mathrm{per}(P) - |v_{i-1}v_i| - |v_iv_{i+1}| + \mathrm{per}(P_i) - |v_{i-1}v_{i+1}|,$$

we can now find the point p minimizing $\mathrm{per}(P \setminus \{p\})$ in $O(n)$ time. ◀

It remains to compute the best partition (P_1, P_2) with $\mathrm{per}(P_2) \leqslant \mathrm{per}(P_1)$ whose separation distance is at least $c_{\mathrm{sep}} \cdot \mathrm{per}(P_2)$ and where P_2 is not a singleton. Let (P_1^*, P_2^*) denote this partition. Define the *size* of a square[1] σ to be its edge length. A square σ is a *good square* if (i) $P_2^* \subset \sigma$, and (ii) $\mathrm{size}(\sigma) \leqslant c^* \cdot \mathrm{per}(P_2^*)$, where $c^* := 18$. Our algorithm globally works as follows.

1. Compute a set S of $O(n)$ squares such that S contains a good square.
2. For each square $\sigma \in S$, construct a set H_σ of $O(1)$ halfplanes such that the following holds: if $\sigma \in S$ is a good square then there is a halfplane $h \in H_\sigma$ such that $P_2^* = P(\sigma \cap h)$, where $P(\sigma \cap h) := P \cap (\sigma \cap h)$.
3. For each pair (σ, h) with $\sigma \in S$ and $h \in H_\sigma$, compute $\mathrm{per}(P \setminus P(\sigma \cap h)) + \mathrm{per}(P(\sigma \cap h))$, and report the partition $(P \setminus P(\sigma \cap h), P(\sigma \cap h))$ that gives the smallest sum.

Step 1: Finding a good square. To find a set S that contains a good square, we first construct a set S_{base} of so-called *base squares*. The set S will then be obtained by expanding the base squares appropriately.

We define a base square σ to be *good* if (i) σ contains at least one point from P_2^*, and (ii) $c_1 \cdot \mathrm{diam}(P_2^*) \leqslant \mathrm{size}(\sigma) \leqslant c_2 \cdot \mathrm{diam}(P_2^*)$, where $c_1 := 1/4$ and $c_2 := 4$ and $\mathrm{diam}(P_2^*)$ denotes the diameter of P_2^*. Note that $2 \cdot \mathrm{diam}(P_2^*) \leqslant \mathrm{per}(P_2^*) \leqslant 4 \cdot \mathrm{diam}(P_2^*)$. For a square σ, define $\overline{\sigma}$ to be the square with the same center as σ and whose size is $(1 + 2/c_1) \cdot \mathrm{size}(\sigma)$.

▶ **Lemma 7.** *If σ is a good base square then $\overline{\sigma}$ is a good square.*

Proof. The distance from any point in σ to the boundary of $\overline{\sigma}$ is at least

$$\frac{\mathrm{size}(\overline{\sigma}) - \mathrm{size}(\sigma)}{2} \geqslant \mathrm{diam}(P_2^*).$$

Since σ contains a point from P_2^*, it follows that $P_2^* \subset \overline{\sigma}$. Since $\mathrm{size}(\sigma) \leqslant c_2 \cdot \mathrm{diam}(P_2^*)$, we have

$$\mathrm{size}(\overline{\sigma}) \leqslant (2/c_1 + 1) \cdot c_2 \cdot \mathrm{diam}(P_2^*) = 36 \cdot \mathrm{diam}(P_2^*) \leqslant c^* \cdot \mathrm{per}(P_2^*).$$ ◀

To obtain S it thus suffices to construct a set S_{base} that contains a good base square. To this end we first build a compressed quadtree for P. For completeness we briefly review the definition of compressed quadtrees; see also Fig. 3 (left).

[1] Whenever we speak of squares, we always mean axis-parallel squares.

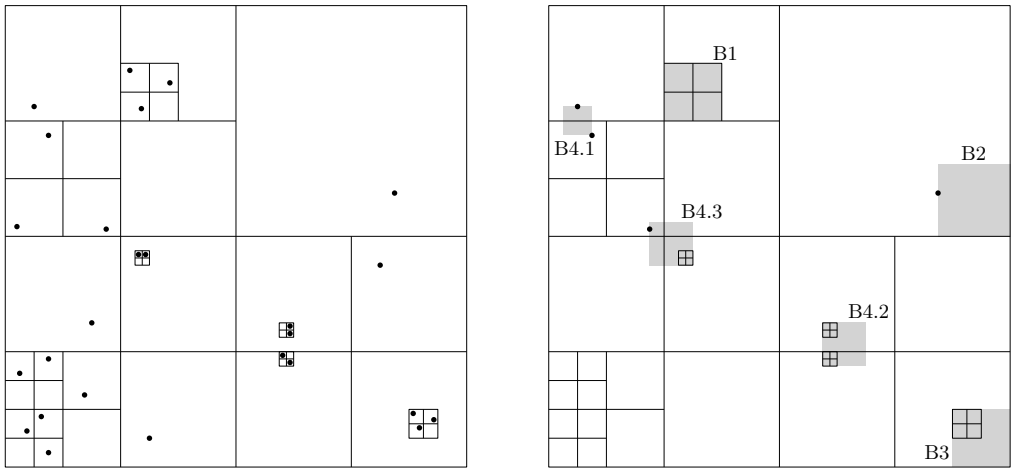


Figure 3 A compressed quadtree and some of the base squares generated from it. In the right figure, only the points are shown that are relevant for the shown base squares.

Assume without loss of generality that P lies in the interior of the unit square $U := [0,1]^2$. Define a *canonical square* to be any square that can be obtained by subdividing U recursively into quadrants. A *compressed quadtree* [13] for P is a hierarchical subdivision of U, defined as follows. In a generic step of the recursive process we are given a canonical square σ and the set $P(\sigma) := P \cap \sigma$ of points inside σ. (Initially $\sigma = U$ and $P(\sigma) = P$.)

- If $|P(\sigma)| \leqslant 1$ then the recursive process stops and σ is a square in the final subdivision.
- Otherwise there are two cases. Consider the four quadrants of σ. The first case is that at least two of these quadrants contain points from $P(\sigma)$. (We consider the quadrants to be closed on the left and bottom side, and open on the right and top side, so a point is contained in a unique quadrant.) In this case we partition σ into its four quadrants – we call this a *quadtree split* – and recurse on each quadrant. The second case is that all points from $P(\sigma)$ lie inside the same quadrant. In this case we compute the smallest canonical square, σ', that contains $P(\sigma)$ and we partition σ into two regions: the square σ' and the so-called *donut region* $\sigma \setminus \sigma'$. We call this a *shrinking step*. After a shrinking step we only recurse on the square σ', not on the donut region.

A compressed quadtree for a set of n points can be computed in $O(n \log n)$ time in the appropriate model of computation[2] [13]. The idea is now as follows. Let $p, p' \in P_2^*$ be a pair of points defining $\mathrm{diam}(P_2^*)$. The compressed quadtree hopefully allows us to zoom in until we have a square in the compressed quadtree that contains p or p' and whose size is roughly equal to $|pp'|$. Such a square will be then a good base square. Unfortunately this does not always work since p and p' can be separated too early. We therefore have to proceed more carefully: we need to add five types of base squares to S_{base}, as explained next and illustrated in Fig. 3 (right).

(B1) Any square σ that is generated during the recursive construction – note that this not only refers to squares in the final subdivision – is put into S_{base}.

(B2) For each point $p \in P$ we add a square σ_p to S_{base}, as follows. Let σ be the square of the final subdivision that contains p. Then σ_p is a smallest square that contains p and that shares a corner with σ.

[2] In particular we need to be able to compute the smallest canonical square containing two given points in $O(1)$ time. See the book by Har-Peled [13] for a discussion.

(B3) For each square σ that results from a shrinking step we add an extra square σ' to S_{base}, where σ' is the smallest square that contains σ and that shares a corner with the parent square of σ.

(B4) For any two regions in the final subdivision that touch each other – we also consider two regions to touch if they only share a vertex – we add at most one square to S_{base}, as follows. If one of the regions is an empty square, we do not add anything for this pair. Otherwise we have three cases.

- **(B4.1)** If both regions are non-empty squares containing points p and p', respectively, then we add a smallest enclosing square for the pair of points p, p' to S_{base}.
- **(B4.2)** If both regions are donut regions, say $\sigma_1 \setminus \sigma_1'$ and $\sigma_2 \setminus \sigma_2'$, then we add a smallest enclosing square for the pair σ_1', σ_2' to S_{base}.
- **(B4.3)** If one region is a non-empty square containing a point p and the other is a donut region $\sigma \setminus \sigma'$, then we add a smallest enclosing square for the pair p, σ' to S_{base}.

▶ **Lemma 8.** *The set S_{base} has size $O(n)$ and contains a good base square. Furthermore, S_{base} can be computed in $O(n \log n)$ time.*

Proof. A compressed quadtree has size $O(n)$ so we have $O(n)$ base squares of type (B1) and (B3). Obviously there are $O(n)$ base squares of type (B2). Finally, the number of pairs of final regions that touch is $O(n)$ – this follows because we have a planar rectilinear subdivision of total complexity $O(n)$ – and so the number of base squares of type (B4) is $O(n)$ as well. The fact that we can compute S_{base} in $O(n \log n)$ time follows directly from the fact that we can compute the compressed quadtree in $O(n \log n)$ time [13].

It remains to prove that S_{base} contains a good base square. We call a square σ *too small* when $\text{size}(\sigma) < c_1 \cdot \text{diam}(P_2^*)$ and *too large* when $\text{size}(\sigma) > c_2 \cdot \text{diam}(P_2^*)$; otherwise we say that σ has the *correct size*. Let $p, p' \in P_2^*$ be two points with $|pp'| = \text{diam}(P_2^*)$, and consider a smallest square $\sigma_{p,p'}$, in the compressed quadtree that contains both p and p'. Note that $\sigma_{p,p'}$ cannot be too small, since $c_1 = 1/4 < 1/\sqrt{2}$. If $\sigma_{p,p'}$ has the correct size, then we are done since it is a good base square of type (B1). So now suppose $\sigma_{p,p'}$ is too large.

Let $\sigma_0, \sigma_1, \ldots, \sigma_k$ be the sequence of squares in the recursive subdivision of $\sigma_{p,p'}$ that contain p; thus $\sigma_0 = \sigma_{p,p'}$ and σ_k is a square in the final subdivision. Define $\sigma_0', \sigma_1', \ldots, \sigma_{k'}'$ similarly, but now for p' instead of p. Suppose that none of these squares has the correct size – otherwise we have a good base square of type (B1). There are three cases.

- *Case* (i)*: σ_k and $\sigma_{k'}'$ are too large.*
  We claim that σ_k touches $\sigma_{k'}'$. To see this, assume without loss of generality that $\text{size}(\sigma_k) \leqslant \text{size}(\sigma_{k'}')$. If σ_k does not touch $\sigma_{k'}'$ then $|pp'| \geqslant \text{size}(\sigma_k)$, which contradicts that σ_k is too large. Hence, σ_k indeed touches $\sigma_{k'}'$. But then we have a base square of type (B4.1) for the pair p, p' and since $|pp'| = \text{diam}(P_2^*)$ this is a good base square.
- *Case* (ii)*: σ_k and $\sigma_{k'}'$ are too small.*
  In this case there are indices $0 < j \leqslant k$ and $0 < j' \leqslant k'$ such that σ_{j-1} and $\sigma_{j'-1}'$ are too large and σ_j and $\sigma_{j'}'$ are too small. Note that this implies that both σ_j and $\sigma_{j'}'$ result from a shrinking step, because $c_1 < c_2/2$ and so the quadrants of a too-large square cannot be too small. We claim that σ_{j-1} touches $\sigma_{j'-1}'$. Indeed, similarly to Case (i), if σ_{j-1} and $\sigma_{j'-1}'$ do not touch then $|pp'| > \min(\text{size}(\sigma_{j-1}), \text{size}(\sigma_{j'-1}'))$, contradicting that both σ_{j-1} and $\sigma_{j'-1}'$ are too large. We now have two subcases.
  - The first subcase is that the donut region $\sigma_{j-1} \setminus \sigma_j$ touches the donut region $\sigma_{j'-1}' \setminus \sigma_{j'}'$. Thus a smallest enclosing square for σ_j and $\sigma_{j'}'$ has been put into S_{base} as a base square of type (B4.2). Let σ^* denote this square. Since the segment pp' is contained

in σ^* we have

$$c_1 \cdot \mathrm{diam}(P_2^*) \;<\; \mathrm{diam}(P_2^*)/\sqrt{2} \;=\; |pp'|/\sqrt{2} \;\leqslant\; \mathrm{size}(\sigma^*).$$

Furthermore, since σ_j and $\sigma'_{j'}$ are too small we have

$$\mathrm{size}(\sigma^*) \;\leqslant\; \mathrm{size}(\sigma_j) + \mathrm{size}(\sigma'_{j'}) + |pp'| \;\leqslant\; 3 \cdot \mathrm{diam}(P_2^*) \;<\; c_2 \cdot \mathrm{diam}(P_2^*), \tag{7}$$

and so σ^* is a good base square.

- The second subcase is that $\sigma_{j-1} \setminus \sigma_j$ does not touch $\sigma'_{j'-1} \setminus \sigma_{j'}$. This can only happen if σ_{j-1} and $\sigma'_{j'-1}$ just share a single corner, v. Observe that σ_j must lie in the quadrant of σ_{j-1} that has v as a corner, otherwise $|pp'| \geqslant \mathrm{size}(\sigma_{j-1})/2$ and σ_{j-1} would not be too large. Similarly, $\sigma'_{j'}$ must lie in the quadrant of $\sigma'_{j'-1}$ that has v as a corner. Thus the base squares of type (B3) for σ_j and $\sigma'_{j'}$ both have v as a corner. Take the largest of these two base squares, say σ_j. For this square σ^* we have

  $$c_1 \cdot \mathrm{diam}(P_2^*) \;<\; \mathrm{diam}(P_2^*)/2\sqrt{2} \;=\; |pp'|/2\sqrt{2} \;\leqslant\; \mathrm{size}(\sigma^*),$$

  since $|pp'|$ is contained in a square of twice the size of σ^*. Furthermore, since σ_j is too small and $|pv| < |pp'|$ we have

  $$\mathrm{size}(\sigma^*) \;\leqslant\; \mathrm{size}(\sigma_j) + |pv| \;\leqslant\; (c_1 + 1) \cdot \mathrm{diam}(P_2^*) \;<\; c_2 \cdot \mathrm{diam}(P_2^*). \tag{8}$$

  Hence, σ^* is a good base square.
- *Case* (iii)*: neither* (i) *nor* (ii) *applies.*
  In this case σ_k is too small and $\sigma'_{k'}$ is too large (or vice versa). Thus there must be an index $0 < j \leqslant k$ such that σ_{j-1} is too large and σ_j is too small. We can now follow a similar reasoning as in Case (ii): First we argue that σ_j must have resulted from a shrinking step and that σ_{j-1} touches $\sigma'_{k'}$. Then we distinguish two subcases, namely where the donut region $\sigma_j \setminus \sigma_{j-1}$ touches $\sigma'_{k'}$ and where it does not touch $\sigma'_{k'}$. The arguments for the two subcases are similar to the subcases in Case (ii), with the following modifications. In the first subcase we use base squares of type (B4.3) and in (7) the term $\mathrm{size}(\sigma'_{j'})$ disappears; in the second subcase we use a type (B3) base square for σ_j and a type (B2) base square for p', and when the base square for p' is larger than the base square for σ_j then (8) becomes $\mathrm{size}(\sigma^*) \leqslant 2\,|p'v| < c_2 \cdot \mathrm{diam}(P_2^*)$. ◀

Step 2: Generating halfplanes. Consider a good square $\sigma \in S$. Let Q_σ be a set of $4 \cdot c^*/c_{\mathrm{sep}} + 1 = 18001$ points placed equidistantly around the boundary of σ. Note that the distance between two neighbouring points in Q_σ is less than $c_{\mathrm{sep}}/c^* \cdot \mathrm{size}(\sigma)$. For each pair q_1, q_2 of points in Q_σ, add to H_σ the two halfplanes defined by the line through q_1 and q_2.

▶ **Lemma 9.** *For any good square $\sigma \in S$, there is a halfplane $h \in H_\sigma$ such that $P_2^* = P(\sigma \cap h)$.*

Proof. In the case where $\sigma \cap P_1^* = \emptyset$, two points in Q_σ from the same edge of σ define a half-plane h such that $P_2^* = P(\sigma \cap h)$, so assume that σ contains one or more points from P_1^*.

We know that the separation distance between P_1^* and P_2^* is at least $c_{\mathrm{sep}} \cdot \mathrm{per}(P_2^*)$. Moreover, $\mathrm{size}(\sigma) \leqslant c^* \cdot \mathrm{per}(P_2^*)$. Hence, there is an empty open strip O with a width of at least $c_{\mathrm{sep}}/c^* \cdot \mathrm{size}(\sigma)$ separating P_2^* from P_1^*. Since σ contains a point from P_1^*, we know that $\sigma \setminus O$ consists of two pieces and that the part of the boundary of σ inside O consists of two disjoint portions B_1 and B_2 each of length at least $c_{\mathrm{sep}}/c^* \cdot \mathrm{size}(\sigma)$. Hence the sets $B_1 \cap Q_\sigma$ and $B_2 \cap Q_\sigma$ contain points q_1 and q_2, respectively, that define a half-plane h as desired. ◀

Step 3: Evaluating candidate solutions. In this step we need to compute for each pair (σ, h) with $\sigma \in S$ and $h \in H_\sigma$, the value $\mathrm{per}(P \setminus P(\sigma \cap h)) + \mathrm{per}(P(\sigma \cap h))$. We do this by preprocessing P into a data structure that allows us to quickly compute $\mathrm{per}(P \setminus P(\sigma \cap h))$ and $\mathrm{per}(P(\sigma \cap h))$ for a given pair (σ, h). Recall that the bounding lines of the halfplanes h we must process have $O(1)$ different orientations. We construct a separate data structure for each orientation.

Consider a fixed orientation ϕ. We build a data structure $\mathcal{D}_\phi$ for range searching on P with ranges of the form $\sigma \cap h$, where σ is a square and h is halfplane whose bounding line has orientation ϕ. Since the edges of σ are axis-parallel and the bounding line of the halfplanes h have a fixed orientation, we can use a standard three-level range tree [5] for this. Constructing this tree takes $O(n \log^2 n)$ time and the tree has $O(n \log^2 n)$ nodes.

Each node ν of the third-level trees in $\mathcal{D}_\phi$ is associated with a *canonical subset* $P(\nu)$, which contains the points stored in the subtree rooted at ν. We preprocess each canonical subset $P(\nu)$ as follows. First we compute the convex hull $\textsc{ch}(P(\nu))$. Let $v_1, \ldots, v_k$ denote the convex-hull vertices in counterclockwise order. We store these vertices in order in an array, and we store for each vertex v_i the value $\mathrm{length}(\partial P(v_1, v_i))$, that is, the length of the part of $\partial\,\textsc{ch}(P(\nu))$ from v_1 to v_i in counterclockwise order. Note that the convex hull $\textsc{ch}(P(\nu))$ can be computed in $O(|P(\nu)|)$ from the convex hulls at the two children of ν. Hence, the convex hulls $\textsc{ch}(P(\nu))$ (and the values $\mathrm{length}(\partial P(v_1, v_i))$) can be computed in $\sum_{\nu \in \mathcal{D}_\phi} O(|P(\nu)|) = O(n \log^3 n)$ time in total, in a bottom-up manner.

Now suppose we want to compute $\mathrm{per}(P(\sigma \cap h))$, where the orientation of the bounding line of h is ϕ. We perform a range query in $\mathcal{D}_\phi$ to find a set $N(\sigma \cap h)$ of $O(\log^3 n)$ nodes such that $P(\sigma \cap h)$ is equal to the union of the canonical subsets of the nodes in $N(\sigma \cap h)$. Standard range-tree properties guarantee that the convex hulls $\textsc{ch}(P(\nu))$ and $\textsc{ch}(P(\mu))$ of any two nodes $\nu, \mu \in N(\sigma \cap h)$ are disjoint. Note that $\textsc{ch}(P(\sigma \cap h))$ is equal to the convex hull of the set of convex hulls $\textsc{ch}(P(\nu))$ with $\nu \in N(\sigma \cap h)$. In the full version [1] we show that we can compute $\mathrm{per}(P(\sigma \cap h))$ in $O(\log^4 n)$ time.

Observe that $P \setminus P(\sigma \cap h)$ can also be expressed as the union of $O(\log^3 n)$ canonical subsets with disjoint convex hulls, since $\mathbb{R}^2 \setminus (\sigma \cap h)$ is the disjoint union of $O(1)$ ranges of the right type. Hence, we can compute $\mathrm{per}(P \setminus P(\sigma \cap h))$ in $O(\log^4 n)$ time. We thus obtain the following result, which finishes the proof of Theorem 5.

▶ **Lemma 10.** *Step 3 can be performed in $O(n \log^4 n)$ time and using $O(n \log^3 n)$ space.*

3 The approximation algorithm

▶ **Theorem 11.** *Let P be a set of n points in the plane and let (P_1^*, P_2^*) be a partition of P minimizing $\mathrm{per}(P_1^*) + \mathrm{per}(P_2^*)$. Suppose we have an exact algorithm for the minimum perimeter-sum problem running in $T(k)$ time for instances with k points. Then for any given $\varepsilon > 0$ we can compute a partition (P_1, P_2) of P such that $\mathrm{per}(P_1) + \mathrm{per}(P_2) \leqslant (1+\varepsilon) \cdot \big(\mathrm{per}(P_1^*) + \mathrm{per}(P_2^*)\big)$ in $O(n + T(1/\varepsilon^2))$ time.*

Proof. Consider the axis-parallel bounding box B of P. Let w be the width of B and let h be its height. Assume without loss of generality that $w \geqslant h$. Our algorithm works in two steps.

Step 1: Check if $\mathrm{per}(P_1^*) + \mathrm{per}(P_2^*) \leqslant w/16$. If so, compute the exact solution. We partition B vertically into four strips with width $w/4$, denoted B_1, B_2, B_3, and B_4 from left to right. If B_2 or B_3 contains a point from P, we have $\mathrm{per}(P_1^*) + \mathrm{per}(P_2^*) \geqslant w/2 > w/16$ and we go to Step 2. If B_2 and B_3 are both empty, we consider two cases.

Case (i)*:* $h \leqslant w/8$. In this case we simply return the partition $(P \cap B_1, P \cap B_4)$. To see that this is optimal, we first note that any subset $P' \subset P$ that contains a point from B_1 as well as a point from B_4 has $\mathrm{per}(P') \geqslant 2 \cdot (3w/4) = 3w/2$. On the other hand, $\mathrm{per}(P \cap B_1) + \mathrm{per}(P \cap B_4) \leqslant 2 \cdot (w/2 + 2h) \leqslant 3w/2$.

Case (ii)*:* $h > w/8$. We partition B horizontally into four rows with height $h/4$, numbered R_1, R_2, R_3, and R_4 from bottom to top. If R_2 or R_3 contains a point from P, we have $\mathrm{per}(P_1^*) + \mathrm{per}(P_2^*) \geqslant h/2 > w/16$, and we go the Step 2. If R_2 and R_3 are both empty, we overlay the vertical and the horizontal partitioning of B to get a 4×4 grid of cells $C_{ij} := B_i \cap R_j$ for $i, j \in \{1, \ldots, 4\}$. We know that only the corner cells $C_{11}, C_{14}, C_{41}, C_{44}$ contain points from P. If three or four corner cells are non-empty, $\mathrm{per}(P_1^*) + \mathrm{per}(P_2^*) \geqslant 6h/4 > w/16$. Hence, we may without loss of generality assume that any point of P is in C_{11} or C_{44}. We now return the partition $(P \cap C_{11}, P \cap C_{44})$, which is easily seen to be optimal.

Step 2: Handle the case where $\mathrm{per}(P_1^*) + \mathrm{per}(P_2^*) > w/16$.

The idea is to compute a subset $\widehat{P} \subset P$ of size $O(1/\varepsilon^2)$ such that an exact solution to the minimum perimeter-sum problem on $\widehat{P}$ can be used to obtain a $(1+\varepsilon)$-approximation for the problem on P.

We subdivide B into $O(1/\varepsilon^2)$ rectangular cells of width and height at most $c := \varepsilon w/(64\pi\sqrt{2})$. For each cell C where $P \cap C$ is non-empty we pick an arbitrary point in $P \cap C$, and we let $\widehat{P}$ be the set of selected points. For a point $p \in \widehat{P}$, let $C(p)$ be the cell containing p. Intuitively, each point $p \in \widehat{P}$ represents all the points $P \cap C(p)$. Let $(\widehat{P}_1, \widehat{P}_2)$ be a partition of $\widehat{P}$ that minimizes $\mathrm{per}(\widehat{P}_1) + \mathrm{per}(\widehat{P}_2)$. We assume we have an algorithm that can compute such an optimal partition in $T(|\widehat{P}|)$ time. For $i = 1, 2$, define

$$P_i := \bigcup_{p \in \widehat{P}_i} P \cap C(p).$$

Our approximation algorithm returns the partition (P_1, P_2). (Note that the convex hulls of P_1 and P_2 are not necessarily disjoint.) It remains to prove the approximation ratio. First, note that $\mathrm{per}(\widehat{P}_1) + \mathrm{per}(\widehat{P}_2) \leqslant \mathrm{per}(P_1^*) + \mathrm{per}(P_2^*)$ since $\widehat{P} \subseteq P$. For $i = 1, 2$, let $\widetilde{P}_i$ consist of all points in the plane (not only points in P) within a distance of at most $\sqrt{2}c$ from $\mathrm{CH}(\widehat{P}_i)$. In other words, $\widetilde{P}_i$ is the Minkowksi sum of $\mathrm{CH}(\widehat{P}_i)$ with a disk D of radius $c\sqrt{2}$ centered at the origin. Note that if $p \in \widehat{P}_i$, then $q \in \widetilde{P}_i$ for any $q \in P \cap C(p)$, since any two points in $C(p)$ are at most $\sqrt{2}c$ apart from each other. Therefore $P_i \subset \widetilde{P}_i$ and hence $\mathrm{per}(P_i) \leqslant \mathrm{per}(\widetilde{P}_i)$. Note also that $\mathrm{per}(\widetilde{P}_i) = \mathrm{per}(\widehat{P}_i) + 2c\pi\sqrt{2}$. These observations yield

$$\begin{aligned}
\mathrm{per}(P_1) + \mathrm{per}(P_2) &\leqslant \mathrm{per}(\widetilde{P}_1) + \mathrm{per}(\widetilde{P}_2) \\
&= \mathrm{per}(\widehat{P}_1) + \mathrm{per}(\widehat{P}_2) + 4c\pi\sqrt{2} \leqslant \mathrm{per}(P_1^*) + \mathrm{per}(P_2^*) + 4c\pi\sqrt{2} \\
&= \mathrm{per}(P_1^*) + \mathrm{per}(P_2^*) + 4\pi\sqrt{2} \cdot \big(\varepsilon w/(64\pi\sqrt{2})\big) \\
&\leqslant \mathrm{per}(P_1^*) + \mathrm{per}(P_2^*) + \varepsilon w/16 \leqslant (1+\varepsilon) \cdot (\mathrm{per}(P_1^*) + \mathrm{per}(P_2^*)).
\end{aligned}$$

As all the steps can be done in linear time, the time complexity of the algorithm is $O(n + T(n_\varepsilon))$ for some $n_\varepsilon = O(1/\varepsilon^2)$. ◀

Acknowledgements. This research was initiated when the first author visited the Department of Computer Science at TU Eindhoven during the winter 2015–2016. He wishes to express his gratitude to the other authors and the department for their hospitality.

References

1 M. Abrahamsen, M. de Berg, K. Buchin, M. Mehr, A. D. Mehrabi. Minimum Perimeter-Sum Partitions in the Plane. Preprint, http://arxiv.org/abs/1703.05549 (2017).

2 P. K. Agarwal, M. Sharir. Efficient algorithms for geometric optimization. *ACM Comput. Surv.* 30(4):412–458 (1998).

3 T. Asano, B. Bhattacharya, M. Keil, and F. Yao. Clustering algorithms based on minimum and maximum spanning trees. In *Proc. 4th ACM Symp. Comput. Geom. (SoCG)*, pages 252–257, 1988.

4 S. W. Bae, H.-G. Cho, W. Evans, N. Saeedi, and C.-S. Shin. Covering points with convex sets of minimum size. *Theor. Comput. Sci.*, in press (2016).

5 M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars. *Computational Geometry: Algorithms and Applications (3rd edition)*. Springer-Verlag, 2008.

6 P. Brass, C. Knauer, C.-S. Shin, M. Smid, and I. Vigan. Range-aggregate queries for geometric extent problems. In *Proc. 19th Computing: Australasian Theory Symp. (CATS)*, pages 3–10, 2013.

7 V. Capoyleas, G. Rote, G. Woeginger. Geometric clusterings. *J. Alg.* 12(2):341–356 (1991).

8 T. M. Chan. More planar two-center algorithms. *Comput. Geom. Theory Appl.* 13(2):189–198 (1999).

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

10 O. Devillers and M. J. Katz. Optimal line bipartitions of point sets. *Int. J. Comput. Geom. Appl.* 9(1):39–51 (1999).

11 Z. Drezner. The planar two-center and two-median problems. *Transportation Science* 18(4):351–361 (1984).

12 D. Eppstein. Faster construction of planar two-centers. In *Proc. 8th ACM-SIAM Symp. Discr. Alg. (SODA)*, pages 131–138 (1997).

13 S. Har-Peled. *Geometric approximation algorithms*. Mathematical surveys and monographs, Vol. 173. American Mathematical Society, 2011.

14 J. Hershberger. Minimizing the sum of diameters efficiently. *Comput. Geom. Theory Appl.* 2(2):111–118 (1992).

15 J. W. Jaromczyk and M. Kowaluk. An efficient algorithm for the Euclidean two-center problem. In *Proc. 10th ACM Symp. Comput. Geom. (SoCG)*, pages 303–311 (1994).

16 D. Kirkpatrick and J. Snoeyink. Computing common tangents without a separating line. In *Proc. 4th Workshop Alg. Data Struct. (WADS)*, LNCS 955, pages 183–193, 1995.

17 J. S. B. Mitchell and E.L. Wynters. Finding optimal bipartitions of points and polygons. In *Proc. 2nd Workshop Alg. Data Struct. (WADS)*, LNCS 519, pages 202–213, 1991. Full version available at http://www.ams.sunysb.edu/~jsbm/.

18 J. Rokne, S. Wang, and X. Wu. Optimal bipartitions of point sets. In *Proc. 4th Canad. Conf. Comput. Geom. (CCCG)*, pages 11–16, 1992.

19 M. Segal. Lower bounds for covering problems. *J. Math. Modelling Alg.* 1(1):17–29 (2002).

20 M. Sharir. A near-linear algorithm for the planar 2-center problem. *Discr. Comput. Geom.* 18(2):125–134 (1997).

Range-Clustering Queries*†

Mikkel Abrahamsen[1], Mark de Berg[2], Kevin Buchin[3], Mehran Mehr[4], and Ali D. Mehrabi[5]

1 **Department of Computer Science, University of Copenhagen, Copenhagen, Denmark**
miab@di.ku.dk
2 **Department of Computer Science, TU Eindhoven, Eindhoven, The Netherlands**
mdberg@win.tue.nl
2 **Department of Computer Science, TU Eindhoven, Eindhoven, The Netherlands**
k.a.buchin@tue.nl
2 **Department of Computer Science, TU Eindhoven, Eindhoven, The Netherlands**
m.mehr@tue.nl
2 **Department of Computer Science, TU Eindhoven, Eindhoven, The Netherlands**
amehrabi@win.tue.nl

Abstract

In a geometric k-clustering problem the goal is to partition a set of points in $\mathbb{R}^d$ into k subsets such that a certain cost function of the clustering is minimized. We present data structures for orthogonal *range-clustering queries* on a point set S: given a query box Q and an integer $k \geqslant 2$, compute an optimal k-clustering for $S \cap Q$. We obtain the following results.

- We present a general method to compute a $(1+\varepsilon)$-approximation to a range-clustering query, where $\varepsilon > 0$ is a parameter that can be specified as part of the query. Our method applies to a large class of clustering problems, including k-center clustering in any L_p-metric and a variant of k-center clustering where the goal is to minimize the sum (instead of maximum) of the cluster sizes.
- We extend our method to deal with capacitated k-clustering problems, where each of the clusters should not contain more than a given number of points.
- For the special cases of rectilinear k-center clustering in $\mathbb{R}^1$, and in $\mathbb{R}^2$ for $k = 2$ or 3, we present data structures that answer range-clustering queries exactly.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Geometric data structures, clustering, k-center problem

Digital Object Identifier 10.4230/LIPIcs.SoCG.2017.5

1 Introduction

Motivation

The range-searching problem is one of the most important and widely studied problems in computational geometry. In the standard setting one is given a set S of points in $\mathbb{R}^d$, and a query asks to report or count all points inside a geometric query range Q. In many

* A full version of the paper is available at http://arxiv.org/abs/1705.06242.
† MA is partly supported by Mikkel Thorup's Advanced Grant from the Danish Council for Independent Research under the Sapere Aude research career programme. MdB, KB, MM, and AM are supported by the Netherlands Organisation for Scientific Research (NWO) under project no. 024.002.003, 612.001.207, 022.005025, and 612.001.118 respectively.

applications, however, one would like to perform further analysis on the set $S \cap Q$, to obtain more information about its structure. Currently one then has to proceed as follows: first perform a range-reporting query to explicitly report $S \cap Q$, then apply a suitable analysis algorithm to $S \cap Q$. This two-stage process can be quite costly, because algorithms for analyzing geometric data sets can be slow and $S \cap Q$ can be large. To avoid this we would need data structures for what we call *range-analysis queries*, which directly compute the desired structural information about $S \cap Q$. In this paper we develop such data structures for the case where one is interested in a cluster-analysis of $S \cap Q$.

Clustering is a fundamental task in data analysis. It involves partitioning a given data set into subsets called *clusters*, such that similar elements end up in the same cluster. Often the data elements can be viewed as points in a geometric space, and similarity is measured by considering the distance between the points. We focus on clustering problems of the following type. Let S be a set of n points in $\mathbb{R}^d$, and let $k \geqslant 2$ be a natural number. A *k-clustering* of S is a partitioning $\mathcal{C}$ of S into at most k clusters. Let $\Phi(\mathcal{C})$ denote the *cost* of $\mathcal{C}$. The goal is now to find a clustering $\mathcal{C}$ that minimizes $\Phi(\mathcal{C})$. Many well-known geometric clustering problems are of this type. Among them is the k-center problem. In the *Euclidean k-center problem* $\Phi(\mathcal{C})$ is the maximum cost of any of the clusters $C \in \mathcal{C}$, where the cost of C is the radius of its smallest enclosing ball. Hence, in the Euclidean k-center problem we want to cover the point set S by k congruent balls of minimum radius. The *rectilinear k-center problem* is defined similarly except that one considers the L_∞-metric; thus we want to cover S by k congruent axis-aligned cubes[1] of minimum size. The k-center problem, including the important special case of the 2-center problem, has been studied extensively, both for the Euclidean case (e.g. [2, 8, 12, 17, 16, 22]) and for the rectilinear case (e.g. [7, 23]).

All papers mentioned above – in fact, all papers on clustering that we know of – consider clustering in the single-shot version. We are the first to study *range-clustering queries* on a point set S: given a query range Q and a parameter k, solve the given k-clustering problem on $S \cap Q$. We study this problem for the case where the query range is an axis-aligned box.

Related work

Range-analysis queries can be seen as a very general form of range-aggregate queries. In a range-aggregate query, the goal is to compute some aggregate function $F(S \cap Q)$ over the points in the query range. The current state of the art typically deals with simple aggregate functions of the following form: each point $p \in S$ has a *weight* $w(p) \in \mathbb{R}$, and $F(S \cap Q) := \bigoplus_{p \in S \cap Q} w(p)$, where $\oplus$ is a semi-group operation. Such aggregate functions are *decomposable*, meaning that $F(A \cap B)$ can be computed from $F(A)$ and $F(B)$, which makes them easy to handle using existing data structures such as range trees.

Only some, mostly recent, papers describe data structures supporting non-decomposable analysis tasks. Several deal with finding the closest pair inside a query range (e.g. [1, 10, 13]). However, the closest pair does not give information about the global shape or distribution of $S \cap Q$, which is what our queries are about. The recent works by Brass *et al.* [5] and by Arya *et al.* [4] are more related to our paper. Brass *et al.* [5] present data structures for finding extent measures, such the width, area or perimeter of the convex hull of $S \cap Q$, or the smallest enclosing disk. (Khare *et al.* [18] improve the result on smallest-enclosing-disk queries.) These measures are strictly speaking not decomposable, but they depend only on

[1] Throughout the paper, when we speak of cubes (or squares, or rectangles, or boxes) we always mean axis-aligned cubes (or squares, or rectangles, or boxes).

the convex hull of $S \cap Q$ and convex hulls are decomposable. A related result is by Nekrich and Smid [20], who present a data structure that returns an ε-coreset inside a query range. The measure studied by Arya *et al.* [4], namely the length of the MST of $S \cap Q$, cannot be computed form the convex hull either: like our range-clustering queries, it requires more information about the structure of the point set. Thus our paper continues the direction set out by Arya *et al.*, which is to design data structures for more complicated analysis tasks on $S \cap Q$.

Our contribution

Our main result is a general method to answer *approximate* orthogonal range-clustering queries in $\mathbb{R}^d$. Here the query specifies (besides the query box Q and the number of clusters k) a value $\varepsilon > 0$; the goal then is to compute a k-clustering $\mathcal{C}$ of $S \cap Q$ with $\Phi(\mathcal{C}) \leqslant (1+\varepsilon) \cdot \Phi(\mathcal{C}_{\text{opt}})$, where $\mathcal{C}_{\text{opt}}$ is an optimal clustering for $S \cap Q$. Our method works by computing a sample $R \subseteq S \cap Q$ such that solving the problem on R gives us the desired approximate solution. We show that for a large class of cost functions Φ we can find such a sample of size only $O(k(f(k)/\varepsilon)^d)$, where $f(k)$ is a function that only depends on the number of clusters. This is similar to the approach taken by Har-Peled and Mazumdar [15], who solve the (single-shot) approximate k-means and k-median problem efficiently by generating a coreset of size $O((k/\varepsilon^d) \cdot \log n)$. A key step in our method is a procedure to efficiently compute a lower bound on the value of an optimal solution within the query range. The class of clustering problems to which our method applies includes the k-center problem in any L_p-metric, variants of the k-center problem where we want to minimize the sum (rather than maximum) of the cluster radii, and the 2-dimensional problem where we want to minimize the maximum or sum of the perimeters of the clusters. Our technique allows us, for instance, to answer rectilinear k-center queries in the plane in $O((1/\varepsilon)\log n + 1/\varepsilon^2)$ for $k = 2$ or 3, in $O((1/\varepsilon)\log n + (1/\varepsilon^2)\text{polylog}(1/\varepsilon))$ for $k = 4$ or 5, and in $O((k/\varepsilon)\log n + (k/\varepsilon)^{O(\sqrt{k})})$ time for $k > 3$. We also show that for the rectilinear (or Euclidean) k-center problem, our method can be extended to deal with the capacitated version of the problem. In the capacitated version each cluster should not contain more than $\alpha \cdot (|S \cap Q|/k)$ points, for a given $\alpha > 1$.

In the second part of the paper we turn our attention to exact solutions to range-clustering queries. Here we focus on rectilinear *k-center queries* – that is, range-clustering queries for the rectilinear k-center problem – in $\mathbb{R}^1$ and $\mathbb{R}^2$. We present two linear-size data structures for queries in $\mathbb{R}^1$; one has $O(k^2 \log^2 n)$ query time, the other has $O(3^k \log n)$ query time. For queries in $\mathbb{R}^2$ we present a data structure that answers 2-center queries in $O(\log n)$ time, and one that answers 3-center queries in $O(\log^2 n)$ time. Both data structures use $O(n \log^\varepsilon n)$ storage, where $\varepsilon > 0$ is an arbitrary small (but fixed) constant.

2 Approximate Range-Clustering Queries

In this section we present a general method to answer approximate range-clustering queries. We start by defining the class of clustering problems to which it applies.

Let S be a set of n points in $\mathbb{R}^d$ and let $\text{Part}(S)$ be the set of all partitions of S. Let $\text{Part}_k(S)$ be the set of all partitions into at most k subsets, that is, all k-clusterings of S. Let $\Phi : \text{Part}(S) \mapsto \mathbb{R}_{\geqslant 0}$ be the cost function defining our clustering problem, and define

$$\textsc{Opt}_k(S) := \min_{\mathcal{C} \in \text{Part}_k(S)} \Phi(\mathcal{C})$$

to be the minimum cost of any k-clustering. Thus the goal of a range-clustering query with query range Q and parameter $k \geqslant 2$ is to compute a clustering $\mathcal{C} \in \text{Part}_k(S_Q)$ such that

$\Phi(\mathcal{C}) = \textsc{Opt}_k(S_Q)$, where $S_Q := S \cap Q$. From now on we use S_Q as a shorthand for $S \cap Q$. The method presented in this section gives an approximate answer to such a query: for a given constant $\varepsilon > 0$, which can be specified as part of the query, the method will report a clustering $\mathcal{C} \in \text{Part}_k(S_Q)$ with $\Phi(\mathcal{C}) \leqslant (1+\varepsilon) \cdot \textsc{Opt}_k(S_Q)$.

To define the class of clusterings to which our method applies, we will need the concept of r-packings [14]. Actually, we will use a slightly weaker variant, which we define as follows. Let $|pq|$ denote the Euclidean distance between two points p and q. A subset $R \subseteq P$ of a point set P is called a *weak r-packing* for P, for some $r > 0$, if for any point $p \in P$ there exists a *packing point* $q \in R$ such that $|pq| \leqslant r$. (The difference with standard r-packings is that we do not require that $|qq'| > r$ for any two points $q, q' \in R$.) The clustering problems to which our method applies are the ones whose cost function is *regular*, as defined next.

▶ **Definition 1.** A cost function $\Phi : \text{Part}(S) \mapsto \mathbb{R}_{\geqslant 0}$ is called $(c, f(k))$-*regular*, if there is a constant c and a function $f : \mathbb{N}_{\geqslant 2} \mapsto \mathbb{R}_{\geqslant 0}$ such that the followings hold.

- For any clustering $\mathcal{C} \in \text{Part}(S)$, we have

  $$\Phi(\mathcal{C}) \geqslant c \cdot \max_{C \in \mathcal{C}} \text{diam}(C),$$

  where $\text{diam}(C) = \max_{p,q \in C} |pq|$ denotes the Euclidean diameter of the cluster C. We call this the *diameter-sensitivity* property.
- For any subset $S' \subseteq S$, any weak r-packing R of S', and any $k \geqslant 2$, we have that

  $$\textsc{Opt}_k(R) \leqslant \textsc{Opt}_k(S') \leqslant \textsc{Opt}_k(R) + r \cdot f(k).$$

  Moreover, given a k-clustering $\mathcal{C} \in \text{Part}_k(R)$ we can compute a k-clustering $\mathcal{C}^* \in \text{Part}_k(S')$ with $\Phi(\mathcal{C}^*) \leqslant \Phi(\mathcal{C}) + r \cdot f(k)$ in time $T_{\text{expand}}(n, k)$. We call this the *expansion* property.

Examples

Many clustering problems have regular cost functions, in particular when the cost function is the aggregation – the sum, for instance, or the maximum – of the costs of the individual clusters. Next we give some examples.

Rectilinear and other k-center problems. For a cluster C, let $\text{radius}_p(C)$ denote the radius of the minimum enclosing ball of C in the L_p-metric. In the L_∞-metric, for instance, $\text{radius}_p(C)$ is half the edge length of a minimum enclosing axis-aligned cube of C. Then the cost of a clustering $\mathcal{C}$ for the k-center problem in the L_p-metric is $\Phi_p^{\max}(\mathcal{C}) = \max_{C \in \mathcal{C}} \text{radius}_p(C)$. One can easily verify that the cost function for the rectilinear k-center problem is $(1/(2\sqrt{d}), 1)$-regular, and for the Euclidean k-center problem it is $(1/2, 1)$-regular. Moreover, $T_{\text{expand}}(n, k) = O(k)$ for the k-center problem, since we just have to scale each ball by adding r to its radius.[2] (In fact $\Phi_p^{\max}(\mathcal{C})$ is regular for any p.)

Min-sum variants of the k-center problem. In the k-center problem the goal is to minimize $\max_{C \in \mathcal{C}} \text{radius}_p(C)$. Instead we can also minimize $\Phi_p^{\text{sum}}(\mathcal{C}) := \sum_{C \in \mathcal{C}} \text{radius}_p(C)$, the sum of the cluster radii. Also these costs functions are regular; the only difference is that the expansion property is now satisfied with $f(k) = k$, instead of with $f(k) = 1$. Another interesting variant is to minimize $\left(\sum_{C \in \mathcal{C}} \text{radius}_2(C)^2\right)^{1/2}$, which is $(1/(2\sqrt{d}), \sqrt{k})$-regular.

[2] This time bound only accounts for reporting the set of balls that define the clustering. If we want to report the clusters explicitly, we need to add an $O(nk)$ term.

Minimum perimeter k-clustering problems. For a cluster C of points in $\mathbb{R}^2$, define $\mathrm{per}(C)$ to be the length of the perimeter of the convex hull of C. In the minimum perimeter-sum clustering problem the goal is to compute a k-clustering $\mathcal{C}$ such that $\Phi_{\mathrm{per}} := \sum_{C\in\mathcal{C}} \mathrm{per}(C)$ is minimized [6]. This cost function is $(2, 2\pi k)$-regular. Indeed, if we expand the polygons in a clustering $\mathcal{C}$ of a weak r-packing R by taking the Minkowski sum with a disk of radius r, then the resulting shapes cover all the points in S. Each perimeter increases by $2\pi r$ in this process. To obtain a clustering, we then assign each point to the cluster of its closest packing point, so $T_{\mathrm{expand}}(n, k) = O(n \log n)$.

Non-regular costs functions. Even though many clustering problems have regular costs functions, not all clustering problems do. For example, the k-means problem does not have a regular cost function. Minimizing the the max or sum of the areas of the convex hulls of the clusters is not regular either.

Our data structure and query algorithm

We start with a high-level overview of our approach. Let S be the given point set on which we want to answer range-clustering queries, and let Q be the query range. We assume we have an algorithm SingleShotClustering(P, k) that computes an optimal solution to the k-clustering problem (for the given cost function Φ) on a given point set P. (Actually, it is good enough if SingleShotClustering(P, k) gives a $(1 + \varepsilon)$-approximation.) Our query algorithm proceeds as follows.

ClusterQuery(k, Q, ε)

1: Compute a lower bound LB on $\mathrm{Opt}_k(S_Q)$.
2: Set $r := \varepsilon \cdot \mathrm{LB}/f(k)$ and compute a weak r-packing R_Q on S_Q.
3: $\mathcal{C} :=$ SingleShotClustering(R_Q, k).
4: Expand $\mathcal{C}$ into a k-clustering $\mathcal{C}^*$ of cost at most $\Phi(\mathcal{C}) + r \cdot f(k)$ for S_Q.
5: **return** $\mathcal{C}^*$.

Note that Step 4 is possible because Φ is $(c, f(k))$-regular. The following lemma is immediate.

▶ **Lemma 2.** $\Phi(\mathcal{C}^*) \leqslant (1 + \varepsilon) \cdot \mathrm{Opt}_k(S_Q)$.

Next we show how to perform Steps 1 and 2: we will describe a data structure that allows us to compute a suitable lower bound LB and a corresponding weak r-packing, such that the size of the r-packing depends only on ε and k but not on $|S_Q|$.

Our lower bound and r-packing computations are based on so-called cube covers. A *cube cover* of S_Q is a collection $\mathcal{B}$ of interior-disjoint cubes that together cover all the points in S_Q and such that each $B \in \mathcal{B}$ contains at least one point from S_Q (in its interior or on its boundary). Define the size of a cube B, denoted by $\mathrm{size}(B)$, to be its edge length. The following lemma follows immediately from the fact that the diameter of a cube B in $\mathbb{R}^d$ is $\sqrt{d} \cdot \mathrm{size}(B)$.

▶ **Lemma 3.** *Let $\mathcal{B}$ be a cube cover of S_Q such that $\mathrm{size}(B) \leqslant r/\sqrt{d}$ for all $B \in \mathcal{B}$. Then any subset $R \subseteq S_Q$ containing a point from each cube $B \in \mathcal{B}$ is a weak r-packing for S.*

Our next lemma shows we can find a lower bound on $\mathrm{Opt}_k(S_Q)$ from a suitable cube cover.

▶ **Lemma 4.** *Suppose the cost function Φ is $(c, f(k))$-regular. Let $\mathcal{B}$ be a cube cover of S_Q such that $|\mathcal{B}| > k2^d$. Then $\mathrm{Opt}_k(S_Q) \geqslant c \cdot \min_{B\in\mathcal{B}} \mathrm{size}(B)$.*

Proof. For two cubes B, B' such that the maximum x_i-coordinate of B is at most the minimum x_i-coordinate of B', we say that B is *i-below* B' and B' is *i-above* B. We denote this relation by $B \prec_i B'$. Now consider an optimal k-clustering $\mathcal{C}_{\text{opt}}$ of S_Q. By the pigeonhole principle, there is a cluster $C \in \mathcal{C}_{\text{opt}}$ containing points from at least $2^d + 1$ cubes. Let $\mathcal{B}_C$ be the set of cubes that contain at least one point in C.

Clearly, if there are cubes $B, B', B'' \in \mathcal{B}_C$ such that $B' \prec_i B \prec_i B''$ for some $1 \leqslant i \leqslant d$, then the cluster C contains two points (namely from B' and B'') at distance at least $\text{size}(B)$ from each other. Since Φ is $(c, f(k))$-regular this implies that $\Phi(\mathcal{C}_{\text{opt}}) \geqslant c \cdot \text{size}(B)$, which proves the lemma.

Now suppose for a contradiction that such a triple B', B, B'' does not exist. Then we can define a characteristic vector $\Gamma(B) = (\Gamma_1(B), \ldots, \Gamma_d(B))$ for each cube $B \in \mathcal{B}_C$, as follows:

$$\Gamma_i(B) = \begin{cases} 0 & \text{if no cube } B' \in \mathcal{B}_C \text{ is } i\text{-below } B \\ 1 & \text{otherwise} \end{cases}$$

Since the number of distinct characteristic vectors is $2^d < |B_C|$, there must be two cubes $B_1, B_2 \in B_C$ with identical characteristic vectors. However, any two interior-disjoint cubes can be separated by an axis-parallel hyperplane, so there is at least one $i \in \{1, \ldots, d\}$ such that $B_1 \prec_i B_2$ or $B_2 \prec_i B_1$. Assume w.l.o.g. that $B_1 \prec_i B_2$, so $\Gamma_i(B_2) = 1$. Since $\Gamma(B_1) = \Gamma(B_2)$ there must be a cube B_3 with $B_3 \prec_i B_1$. But then we have a triple $B_3 \prec_i B_1 \prec_i B_2$, which is a contradiction. ◀

Next we show how to efficiently perform Steps 1 and 2 of ClusterQuery. Our algorithm uses a compressed octree $\mathcal{T}(S)$ on the point set S, which we briefly describe next.

For an integer s, let G_s denote the grid in $\mathbb{R}^d$ whose cells have size 2^s and for which the origin O is a grid point. A *canonical cube* is any cube that is a cell of a grid G_s, for some integer s. A *compressed octree* on a point set S in $\mathbb{R}^d$ contained in a canonical cube B is a tree-like structure defined recursively, as follows.

- If $|S| \leqslant 1$, then $\mathcal{T}(S)$ consists of a single leaf node, which corresponds to the cube B.
- If $|S| > 1$, then consider the cubes $B_1, \ldots, B_{2^d}$ that result from cutting B into 2^d equal-sized cubes.
  - If at least two of the cubes B_i contain at least one point from S then $\mathcal{T}(S)$ consists of a root node with 2^d children $v_1, \ldots, v_{2^d}$, where v_i is the root of a compressed octree for[3] $B_i \cap S$.
  - If all points from S lie in the same cube B_i, then let $B_{\text{in}} \subseteq B_i$ be the smallest canonical cube containing all points in S. Now $\mathcal{T}(S)$ consists of a root node with two children: one child v which is the root of a compressed octree for S inside B_{in}, and one leaf node w which represents the donut region $B \setminus B_{\text{in}}$.

A compressed octree for a set S of n points in any fixed dimension can be computed in $O(n \log n)$ time, assuming a model of computation where the smallest canonical cube of two points can be computed in $O(1)$ time [14, Theorem 2.23]. For a node $v \in \mathcal{T}(S)$, we denote the cube or donut corresponding to v by B_v, and we define $S_v := B_v \cap S$. It will be convenient to slightly modify the compressed quadtree by removing all nodes v such that $S_v = \emptyset$. (These nodes must be leaves.) Note that this removes all nodes v such that B_v is a donut. As a result, the parent of a donut node now has only one child. The modified tree $\mathcal{T}(S)$ – with a slight abuse of terminology we still refer to $\mathcal{T}(S)$ as a compressed octree

[3] Here we assume that points on the boundary between cubes are assigned to one of these cubes in a consistent manner.

Algorithm 1 Algorithm for Steps 1 and 2 of ClusterQuery, for a $(c, f(k))$-regular cost function.

1: $\mathcal{B}_{\text{inner}} := B_{\text{root}(\mathcal{T}(S))}$ and $\mathcal{B}_{\text{leaf}} := \emptyset$.
2: ▷ Phase 1: Compute a lower bound on $\text{Opt}_k(S_Q)$.
3: **while** $|\mathcal{B}_{\text{inner}} \cup \mathcal{B}_{\text{leaf}}| \leqslant k2^{2d}$ and $\mathcal{B}_{\text{inner}} \neq \emptyset$ **do**
4: Remove a largest cube B_v from $\mathcal{B}_{\text{inner}}$. Let v be the corresponding node.
5: **if** $B_v \not\subseteq Q$ **then**
6: Compute $\text{bb}(S_Q \cap B_v)$, the bounding box of $S_Q \cap B_v$.
7: Find the deepest node u such that $\text{bb}(S_Q \cap B_v) \subseteq B_u$ and set $v := u$.
8: **end if**
9: For each child w of v such that $B_w \cap S_Q \neq \emptyset$, insert B_w into $\mathcal{B}_{\text{inner}}$ if w is an internal node and insert B_w into $\mathcal{B}_{\text{leaf}}$ if w is a leaf node.
10: **end while**
11: $\text{LB} := c \cdot \max_{B_v \in \mathcal{B}_{\text{inner}}} \text{size}(B_v)$.
12: ▷ Phase 2: Compute a suitable weak r-packing.
13: $r := \varepsilon \cdot \text{LB}/f(k)$.
14: **while** $\mathcal{B}_{\text{inner}} \neq \emptyset$ **do**
15: Remove a cube B_v from $\mathcal{B}_{\text{inner}}$ and handle it as in lines 5–9, with the following change: if $\text{size}(B_w) \leqslant r/\sqrt{d}$ then always insert B_w into $\mathcal{B}_{\text{leaf}}$ (not into $\mathcal{B}_{\text{inner}}$).
16: **end while**
17: For each cube $B_v \in \mathcal{B}_{\text{leaf}}$ pick a point in $S_Q \cap B_v$ and put it into R_Q.
18: **return** R_Q.

– has the property that any internal node has at least two children. We augment $\mathcal{T}(S)$ by storing at each node v an arbitrary point $p \in B_v \cap S$.

Our algorithm descends into $\mathcal{T}(S)$ to find a cube cover $\mathcal{B}$ of S_Q consisting of canonical cubes, such that $\mathcal{B}$ gives us a lower bound on $\text{Opt}_k(S_Q)$. In a second phase, the algorithm then refines the cubes in the cover until they are small enough so that, if we select one point from each cube, we get a weak r-packing of S_Q for the appropriate value of r. The details are described in Algorithm 1, where we assume for simplicity that $|S_Q| > 1$. (The case $|S_Q| \leqslant 1$ is easy to check and handle.)

Note that we continue the loop in lines 3–9 until we collect $k2^{2d}$ cubes (and not $k2^d$, as Lemma 4 would suggest) and that in line 11 we take the maximum cube size (instead of the minimum, as Lemma 4 would suggest).

▶ **Lemma 5.** *The value* LB *computed by Algorithm 1 is a correct lower bound on* $\text{Opt}_k(S_Q)$*, and the set* R_Q *is a weak* r*-packing for* $r = \varepsilon \cdot \text{LB}/f(k)$ *of size* $O(k(f(k)/(c\,\varepsilon))^d)$.

Proof. As the first step to prove that LB is a correct lower bound, we claim that the loop in lines 3–9 maintains the following invariant: (i) $\bigcup(\mathcal{B}_{\text{inner}} \cup \mathcal{B}_{\text{leaf}})$ contains all points in S_Q, and (ii) each $B \in \mathcal{B}_{\text{inner}}$ contains at least two points from S_Q and each $B \in \mathcal{B}_{\text{leaf}}$ contains exactly one point from S_Q. This is trivially true before the loop starts, under our assumption that $|S_Q| \geqslant 2$. Now suppose we handle a cube $B_v \in \mathcal{B}_{\text{inner}}$. If $B_v \subseteq Q$ then we insert the cubes B_w of all children into $\mathcal{B}_{\text{inner}}$ or $\mathcal{B}_{\text{leaf}}$, which restores the invariant. If $B_v \not\subseteq Q$ then we first replace v by u. The condition $\text{bb}(S_Q \cap B_v) \subseteq B_u$ guarantees that all points of S_Q in B_v are also in B_u. Hence, if we then insert the cubes B_w of u's children into $\mathcal{B}_{\text{inner}}$ or $\mathcal{B}_{\text{leaf}}$, we restore the invariant. Thus at any time, and in particular after the loop, the set $\mathcal{B}_{\text{inner}} \cup \mathcal{B}_{\text{leaf}}$ is a cube cover of S_Q.

To complete the proof that LB is a correct lower bound we do not work with the set $\mathcal{B}_{\text{inner}} \cup \mathcal{B}_{\text{leaf}}$ directly, but we work with a set $\mathcal{B}$ defined as follows. For a cube $B_v \in \mathcal{B}_{\text{inner}} \cup \mathcal{B}_{\text{leaf}}$, define $\text{parent}(B_v)$ to be the cube B_u corresponding to the parent node u of v. For each cube $B_v \in \mathcal{B}_{\text{inner}} \cup \mathcal{B}_{\text{leaf}}$ we put one cube into $\mathcal{B}$, as follows. If there is another cube $B_w \in \mathcal{B}_{\text{inner}} \cup \mathcal{B}_{\text{leaf}}$ such that $\text{parent}(B_w) \subsetneq \text{parent}(B_v)$, then we put B_v itself into $\mathcal{B}$, and otherwise we put $\text{parent}(B_v)$ into $\mathcal{B}$. Finally, we remove all duplicates from $\mathcal{B}$. Since $\mathcal{B}_{\text{inner}} \cup \mathcal{B}_{\text{leaf}}$ is a cube cover for S_Q – that is, the cubes in $\mathcal{B}_{\text{inner}} \cup \mathcal{B}_{\text{leaf}}$ are disjoint and they cover all points in S_Q – the same is true for $\mathcal{B}$. Moreover, the only duplicates in $\mathcal{B}$ are cubes that are the parent of multiple nodes in $\mathcal{B}_{\text{inner}} \cup \mathcal{B}_{\text{leaf}}$, and so $|\mathcal{B}| \geqslant |\mathcal{B}_{\text{inner}} \cup \mathcal{B}_{\text{leaf}}|/2^d > k2^d$. By Lemma 4 we have $\textsc{Opt}_k(S_Q) \geqslant c \cdot \min_{B_v \in \mathcal{B}} \text{size}(B_v)$.

It remains to argue that $\min_{B_v \in \mathcal{B}} \text{size}(B_v) \geqslant \max_{B_v \in \mathcal{B}_{\text{inner}}} \text{size}(B_v)$. We prove this by contradiction. Hence, we assume $\min_{B_v \in \mathcal{B}} \text{size}(B_v) < \max_{B_v \in \mathcal{B}_{\text{inner}}} \text{size}(B_v)$ and we define $B := \arg\min_{B_v \in \mathcal{B}} \text{size}(B_v)$ and $B' := \arg\max_{B_v \in \mathcal{B}_{\text{inner}}} \text{size}(B_v)$. Note that for any cube $B_v \in \mathcal{B}$ either B_v itself is in $\mathcal{B}_{\text{inner}} \cup \mathcal{B}_{\text{leaf}}$ or $B_v = \text{parent}(B_w)$ for some cube $B_w \in \mathcal{B}_{\text{inner}} \cup \mathcal{B}_{\text{leaf}}$. We now make the following case distinction.

Case I: $B = \text{parent}(B_w)$ for some cube $B_w \in \mathcal{B}_{\text{inner}} \cup \mathcal{B}_{\text{leaf}}$. But this is an immediate contradiction since Algorithm 1 would have to split B' before splitting B.

Case II: $B \in \mathcal{B}_{\text{inner}} \cup \mathcal{B}_{\text{leaf}}$. Because B itself was put into $\mathcal{B}$ and not $\text{parent}(B)$, there exists a cube $B_w \in \mathcal{B}_{\text{inner}} \cup \mathcal{B}_{\text{leaf}}$ such that $\text{parent}(B) \supsetneq \text{parent}(B_w)$, which means $\text{size}(\text{parent}(B_w)) < \text{size}(\text{parent}(B))$. In order to complete the proof, it suffices to show that $\text{size}(\text{parent}(B_w)) \leqslant \text{size}(B)$. Indeed, since B' has not been split by Algorithm 1 (because $B' \in \mathcal{B}_{\text{inner}}$) we know that $\text{size}(B') \leqslant \text{size}(\text{parent}(B_w))$. This inequality along with the inequality $\text{size}(\text{parent}(B_w)) \leqslant \text{size}(B)$ imply that $\text{size}(B') \leqslant \text{size}(B)$ which is in contradiction with $\text{size}(B) < \text{size}(B')$. To show that $\text{size}(\text{parent}(B_w)) \leqslant \text{size}(B)$ we consider the following two subcases. (i) $\text{parent}(B)$ is a degree-1 node. This means that $\text{parent}(B)$ corresponds to a cube that was split into a donut and the cube corresponding to B. Since the cube corresponding to B_w must be completely inside the cube corresponding to $\text{parent}(B)$ (because $\text{size}(\text{parent}(B_w)) < \text{size}(\text{parent}(B))$) and a donut is empty we conclude that the cube corresponding to B_w must be completely inside the cube corresponding to B. Hence, $\text{size}(\text{parent}(B_w)) \leqslant \text{size}(B)$. (ii) $\text{parent}(B)$ is not a degree-1 node. The inequality $\text{size}(\text{parent}(B_w)) < \text{size}(\text{parent}(B))$ along with the fact that $\text{parent}(B)$ is not a degree-1 node imply that $\text{size}(\text{parent}(B_w)) \leqslant \text{size}(B)$.

This completes the proof that LB is a correct lower bound. Next we prove that R_Q is a weak r-packing for $r = \varepsilon \cdot \text{LB}/f(k)$. Observe that after the loop in lines 14–16, the set $\mathcal{B}_{\text{leaf}}$ is still a cube cover of S_Q. Moreover, each cube $B_v \in \mathcal{B}_{\text{leaf}}$ either contains a single point from S_Q or its size is at most $r/\sqrt{d}$. Lemma 3 then implies that R_Q is a weak r-packing for the desired value of r.

It remains to bound the size of R_Q. To this end we note that at each iteration of the loop in lines 3–9 the size of $\mathcal{B}_{\text{inner}} \cup \mathcal{B}_{\text{leaf}}$ increases by at most $2^d - 1$, so after the loop we have $|\mathcal{B}_{\text{inner}} \cup \mathcal{B}_{\text{leaf}}| \leqslant k2^{2d} + 2^d - 1$. The loop in lines 14–16 replaces each cube $B_v \in \mathcal{B}_{\text{inner}}$ by a number of smaller cubes. Since $\text{LB} = c \cdot \max_{B_v \in \mathcal{B}_{\text{inner}}} \text{size}(B_v)$ and $r = \varepsilon \cdot \text{LB}/f(k)$, each cube B_v is replaced by only $O((f(k)2^d\sqrt{d}/(c\,\varepsilon))^d)$ smaller cubes. Since d is a fixed constant, the total number of cubes we end up with (which is the same as the size of the r-packing) is $O(k(f(k)/(c\,\varepsilon))^d)$. ◀

Lemma 5, together with Lemma 2, establishes the correctness of our approach. To achieve a good running time, we need a few supporting data structures.

- We need a data structure that can answer the following queries: given a query box Z, find the deepest node u in $\mathcal{T}(S)$ such that $Z \subseteq B_u$. With a *centroid-decomposition tree* $\mathcal{T}_{\text{cd}}$ we can answer such queries in $O(\log n)$ time; see the full version for details.
- We need a data structure $\mathcal{D}$ that can answer the following queries on S: given a query box Z and an integer $1 \leqslant i \leqslant d$, report a point in $S \cap Z$ with maximum x_i-coordinate, and one with minimum x_i-coordinate. It is possible to answer such queries in $O(\log^{d-1} n)$ time with a range tree (with fractional cascading), which uses $O(n \log^{d-1} n)$ storage. Note that this also allows us to compute the bounding box of $S \cap Z$ in $O(\log^{d-1} n)$ time. (In fact slightly better bounds are possible [19], but for simplicity we stick to using standard data structures.)

▶ **Lemma 6.** *Algorithm 1 runs in* $O(k\,(f(k)/(c\,\varepsilon))^d + k\,((f(k)/(c\,\varepsilon))\log n)^{d-1})$ *time.*

Proof. The two key observations in the proof are the following. First, when $B_v \not\subseteq Q$ we replace v by the deepest node u such that $\text{bb}(S_Q \cap B_v) \subseteq B_u$, which implies at least two of the children of u must contain a point in S_Q. From this we conclude that the number of iterations in Phase 1 is bounded by $k2^{2d}$. Second, we use the fact that in Phase 2 the computation of $\text{bb}(S_Q \cap B_v)$ is only needed when $B_v \not\subseteq Q$. The complete proof is in the full version. ◀

This leads to the following theorem.

▶ **Theorem 7.** *Let S be a set of n points in $\mathbb{R}^d$ and let Φ be a $(c, f(k))$-regular cost function. Suppose we have an algorithm that solves the given clustering problem on a set of m points in $T_{\text{ss}}(m, k)$ time. Then there is a data structure that uses $O(n \log^{d-1} n)$ storage such that, for a query range Q and query values $k \geqslant 2$ and $\varepsilon > 0$, we can compute a $(1+\varepsilon)$-approximate answer to a range-clustering query in time*

$$O\left(k\left(\frac{f(k)}{c\,\varepsilon}\cdot\log n\right)^{d-1} + T_{\text{ss}}\left(k\left(\frac{f(k)}{c\,\varepsilon}\right)^{d}, k\right) + T_{\text{expand}}(n,k)\right).$$

As an example application we consider k-center queries in the plane. (The result for rectilinear 2-center queries is actually inferior to the exact solution presented later.)

▶ **Corollary 8.** *Let S be a set of n points in $\mathbb{R}^2$. There is a data structure that uses $O(n \log n)$ storage such that, for a query range Q and query values $k \geqslant 2$ and $\varepsilon > 0$, we can compute a $(1+\varepsilon)$-approximate answer to a k-center query within the following bounds:*

(i) *for the rectilinear case with $k = 2, 3$, the query time is $O((1/\varepsilon)\log n + 1/\varepsilon^2)$;*
(ii) *for the rectilinear case with $k = 4, 5$, the query time is $O((1/\varepsilon)\log n + (1/\varepsilon^2)\text{polylog}(1/\varepsilon))$;*
(iii) *for the Euclidean case with $k = 2$, the expected query time is $O((1/\varepsilon)\log n + (1/\varepsilon^2)\cdot \log^2(1/\varepsilon))$;*
(iv) *for the rectilinear case with $k > 5$ and the Euclidean case with $k > 2$ the query time is $O((k/\varepsilon)\log n + (k/\varepsilon)^{O(\sqrt{k})})$.*

Proof. Recall that the cost function for the k-center problem is $(1/(2\sqrt{d}), 1)$-regular for the rectilinear case and $(1/2, 1)$-regular for the Euclidean case. We now obtain our results by plugging in the appropriate algorithms for the single-shot version. For (i) we use the linear-time algorithm of Hoffmann [16], for (ii) we use the $O(n \cdot \text{polylog} n)$ algorithm of Sharir and Welzl [23], for (iii) we use the $O(n \log^2 n)$ randomized algorithm of Eppstein [12], for (iv) we use the $n^{O(\sqrt{k})}$ algorithm of Agarwal and Procopiuc [3]. ◀

3 Approximate Capacitated k-Center Queries

In this section we study the capacitated variant of the rectilinear k-center problem in the plane. In this variant we want to cover a set S of n points in $\mathbb{R}^2$ with k congruent squares of minimum size, under the condition that no square is assigned more than $\alpha \cdot n/k$ points, where $\alpha > 1$ is a given constant. For a capacitated rectilinear k-center query this means we want to assign no more than $\alpha \cdot |S_Q|/k$ points to each square. Our data structure will report a $(1+\varepsilon, 1+\delta)$-approximate answer to capacitated rectilinear k-center queries: given a query range Q, a natural number $k \geqslant 2$, a constant $\alpha > 1$, and real numbers $\varepsilon, \delta > 0$, it computes a set $\mathcal{C} = \{b_1, \ldots, b_k\}$ of congruent squares such that:

- Each b_i can be associated to a subset $C_i \subseteq S_Q \cap b_i$ such that $\{C_1, \ldots, C_k\}$ is a k-clustering of S_Q and $|C_i| \leqslant (1+\delta)\alpha \cdot |S_Q|/k$; and
- The size of the squares in $\mathcal{C}$ is at most $(1+\varepsilon)\cdot \textsc{Opt}_k(S_Q, \alpha)$, where $\textsc{Opt}_k(S_Q, \alpha)$ is the value of an optimal solution to the problem on S_Q with capacity upper bound $U_Q := \alpha \cdot |S_Q|/k$.

Thus we allow ourselves to violate the capacity constraint by a factor $1+\delta$.

To handle the capacity constraints, it is not sufficient to work with r-packings – we also need δ-approximations. Let P be a set of points in $\mathbb{R}^2$. A *δ-approximation* of P with respect to rectangles is a subset $A \subseteq P$ such that for any rectangle σ we have

$$\big|\ |P \cap \sigma|/|P| \ - \ |A \cap \sigma|/|A|\big| \leqslant \delta.$$

From now on, whenever we speak of δ-approximations, we mean δ-approximations with respect to rectangles. Our method will use a special variant of the capacitated k-center problem, where we also have points that must be covered but do not count for the capacity:

▶ **Definition 9.** Let $R \cup A$ be a point set in $\mathbb{R}^2$, $k \geqslant 2$ a natural number, and U a capacity bound. The *0/1-weighted capacitated k-center problem* in $\mathbb{R}^2$ is to compute a set $\mathcal{C} = \{b_1, \ldots, b_k\}$ of congruent squares of minimum size where each b_i is associated to a subset $C_i \subseteq (R \cup A) \cap b_i$ such that $\{C_1, \ldots, C_k\}$ is a k-clustering of $R \cup A$ and $|C_i \cap A| \leqslant U$.

For a square b, let $\text{expand}(b, r)$ denote the square b expanded by r on each side (so its radius in the L_∞-metric increases by r). Let $0/1$-WeightedKCenter be an algorithm for the single-shot capacitated rectilinear k-center problem. Our query algorithm is as follows.

CapacitatedKCenterQuery$(k, Q, \alpha, \varepsilon, \delta)$

1: Compute a lower bound LB on $\textsc{Opt}_k(S_Q)$.
2: Set $r := \varepsilon \cdot \text{LB}/f(k)$ and compute a weak r-packing R_Q on S_Q.
3: Set $\delta_Q := \delta/16k^3$ and compute a δ_Q-approximation A_Q on S_Q.
4: Set $U := (1+\delta/2)\cdot\alpha\cdot|A_Q|/k$ and $\mathcal{C} := 0/1\text{-WeightedKCenter}(R_Q \cup A_Q, k, U)$.
5: $\mathcal{C}^* := \{\text{expand}(b, r) : b \in \mathcal{C}\}$.
6: **return** $\mathcal{C}^*$.

Note that the lower bound computed in Step 1 is a lower bound on the uncapacitated problem (which is also a lower bound for the capacitated problem). Hence, for Steps 1 and 2 we can use the algorithm from the previous section. How Step 3 is done will be explained later. First we show that the algorithm gives a $(1+\varepsilon, 1+\delta)$-approximate solution. We start by showing that we get a valid solution that violates the capacity constraint by at most a factor $(1+\delta)$. (See the full version for a proof.)

▶ **Lemma 10.** *Let $\{b_1, \ldots, b_k\} := \mathcal{C}^*$ be the set of squares computed in Step 5. There exists a partition $\{C_1, \ldots, C_k\}$ of S_Q such that $C_i \subseteq b_i$ and $|C_i| \leqslant (1+\delta)\cdot U_Q$ for each $1 \leqslant i \leqslant k$, and such a partition can be computed in $O(k^2 + n \log n)$ time.*

We also need to prove that we get a $(1+\varepsilon)$-approximate solution. For this it suffices to show that an optimal solution $\mathcal{C}_{\text{opt}}$ to the problem on S_Q is a valid solution on $R_Q \cup A_Q$. We can prove this by a similar approach as in the proof of the previous lemma.

▶ **Lemma 11.** *The size of the squares in $\mathcal{C}^*$ is at most $(1+\varepsilon)\cdot \textsc{Opt}_k(S_Q, \alpha)$.*

To make CapacitatedKCenterQuery run efficiently, we need some more supporting data structures. In particular, we need to quickly compute a δ_Q-approximation within our range Q. We use the following data structures.

- We compute a collection $A_1, \ldots, A_{\log n}$ where A_i is a $(1/2^i)$-approximation on S using the algorithm of Phillips [21]. This algorithm computes, given a planar point set P of size m and a parameter δ, a δ-approximation of size $O((1/\delta)\log^4(1/\delta)\cdot \text{polylog}(\log(1/\delta)))$ in time $O((n/\delta^3)\cdot \text{polylog}(1/\delta))$. We store each A_i in a data structure for orthogonal range-reporting queries. If we use a range tree with fractional cascading, the data structure uses $O(|A_i|\log|A_i|)$ storage and we can compute all the points in $A_i \cap Q$ in time $O(\log n + |A_i \cap Q|)$.
- We store S in a data structure for orthogonal range-counting queries. There is such a data structure that uses $O(n)$ storage and such that queries take $O(\log n)$ time [9].

We can now compute a δ_Q-approximation for S_Q as follows.

DeltaApprox(Q, δ_Q)

1: Find the smallest value for i such that $\frac{1}{2^i} \leqslant \frac{\delta_Q}{4}\frac{|S_Q|}{|S|}$, and compute $A := Q \cap A_i$.
2: Compute a $(\delta_Q/2)$-approximation A_Q on A using the algorithm by Phillips [21].
3: **return** A_Q.

▶ **Lemma 12.** DeltaApprox(Q, δ_Q) *computes a δ_Q-approximation of size $O\left(\frac{1}{\delta_Q}\text{polylog}\frac{1}{\delta_Q}\right)$ on S_Q in time $O\big((\log n/\delta_Q)^4 \text{polylog}(\log n/\delta_Q)\big)$.*

The only thing left is now an algorithm 0/1-WeightedKCenter$(R_Q \cup A_Q, k, U)$ that solves the 0/1-weighted version of the capacitated rectilinear k-center problem. Here we use the following straightforward approach. Let $m := |R_Q \cup A_Q|$. First we observe that at least one square in an optimal solution has points on opposite edges. Hence, to find the optimal size we can do a binary search over $O(m^2)$ values, namely the horizontal and vertical distances between any pair of points. Moreover, given a target size s we can push all squares such that each has a point on its bottom edge and a point on its left edge. Hence, to test if there is a solution of a given target size s, we only have to test $O(m^{2k})$ sets of k squares. To test such a set $\mathcal{C} = \{b_1, \ldots, b_k\}$ of squares, we need to check if the squares cover all points in $R_Q \cup A_Q$ and if we can assign the points to squares such that the capacity constraint is met. For the latter we need to solve a flow problem, which can be done in $O(m^2 k)$ time; see the full version. Thus each step in the binary search takes $O(m^{2k+2}k)$, leading to an overall time complexity for 0/1-WeightedKCenter$(R_Q \cup A_Q, k, U)$ of $O(m^{2k+2}k\log m)$, where $m = |R_Q \cup A_Q| = O\left(k/\varepsilon^2 + (1/\delta_Q)\text{polylog}(1/\delta_Q)\right)$, where $\delta_Q = \Theta(\delta/k^3)$.

The following theorem summarizes the results in this section.

▶ **Theorem 13.** *Let S be a set of n points in $\mathbb{R}^2$. There is a data structure that uses $O(n\log n)$ storage such that, for a query range Q and query values $k \geqslant 2$, $\varepsilon > 0$ and $\delta > 0$, we can compute a $(1+\varepsilon, 1+\delta)$-approximate answer to a rectilinear k-center query in $O^*((k/\varepsilon)\log n + ((k^3/\delta)\log n)^4 + (k/\varepsilon^2 + (k^3/\delta))^{2k+2})$ time, where the O^*-notation hides $O(\text{polylog}(k/\delta))$ factors.*

Note that for constant k and $\varepsilon = \delta$ the query time simplifies to $O^*((1/\varepsilon^4)\log^4 n + (1/\varepsilon)^{4k+4})$. Also note that the time bound stated in the theorem only includes the time to compute the set of squares defining the clustering. If we want to also report an appropriate assignment of points to the squares, we have to add an $O(k^2 + |S_Q| \log |S_Q|)$ term; see Lemma 10.

▶ Remark. The algorithm can be generalized to the rectilinear k-center problem in higher dimensions, and to the Euclidean k-center problem; we only need to plug in an appropriate δ-approximation algorithm and an appropriate algorithm for the 0/1-weighted version of the problem.

4 Exact k-Center Queries in $\mathbb{R}^1$

In this section we consider k-center queries in $\mathbb{R}^1$. Here we are given a set S of n points in $\mathbb{R}^1$ that we wish to preprocess into a data structure such that, given a query interval Q and a natural number $k \geqslant 2$, we can compute a set $\mathcal{C}$ of at most k intervals of the same length that together cover all points in $S_Q := S \cap Q$ and whose length is minimum. We obtain the following result (complete proof in the full version).

▶ **Theorem 14.** *Let S be a set of n points in $\mathbb{R}^1$. There is a data structure that uses $O(n)$ storage such that, for a query range Q and a query value $k \geqslant 2$, we can answer a rectilinear k-center query in $O(\min\{k^2 \log^2 n, 3^k \log n\})$ time.*

Proof Sketch. We present two approaches, one with $O(k^2 \log^2 n)$ query time and one with $O(3^k \log n)$ query time. Let $p_1, \ldots, p_n$ be the points in S, sorted from left to right.

The first approach uses a subroutine Decider which, given an interval Q', a length L and an integer $\ell \leqslant k$, can decide in $O(\ell \log n)$ time if all points in $S \cap Q'$ can be covered by ℓ intervals of length L. The global query algorithm then performs a binary search, using Decider as subroutine, to find a pair of points $p_i, p_{i+1} \in S_Q$ such that the first interval in an optimal solution covers p_i but not p_{i+1}. Then an optimal solution is found recursively for $k-1$ clusters within the query interval $Q \cap [p_{i+1}, \infty)$.

The second approach searches for a point $q \in Q$ and value ℓ such that there is an optimal solution where $S_Q \cap (-\infty, q]$ is covered by ℓ intervals and $S_Q \cap [q, \infty)$ is covered by $k - \ell$ intervals. The efficiency depends on the interesting fact that there must be a *fair split point*, that is, a pair (q, ℓ) such that ℓ/k (the fraction of intervals used to the left of q) is proportional to the fraction of the length of Q to the left of q. ◀

5 Exact Rectilinear 2- and 3-Center Queries in $\mathbb{R}^2$

Suppose we are given a set $S = \{p_1, p_2, \ldots, p_n\}$ of n points in $\mathbb{R}^2$ and an integer k. In this section we build a data structure $\mathcal{D}$ that stores the set S and, given an orthogonal query rectangle Q, can be used to quickly find an optimal solution for the k-center problem on S_Q for $k = 2$ or 3, where $S_Q := S \cap Q$.

2-center queries

Our approach for the case of $k = 2$ is as follows. We start by shrinking the query range Q such that each edge of Q touches at least one point of S. (The time for this step is subsumed by the time for the rest of the procedure.) It is well known that if we want to cover S_Q by two squares σ, σ' of minimum size, then σ and σ' both share a corner with Q and these corners are opposite corners of Q. We say that σ and σ' are *anchored* at the corner they share with Q. Thus we need to find optimal solutions for two cases – σ and σ' are anchored at the

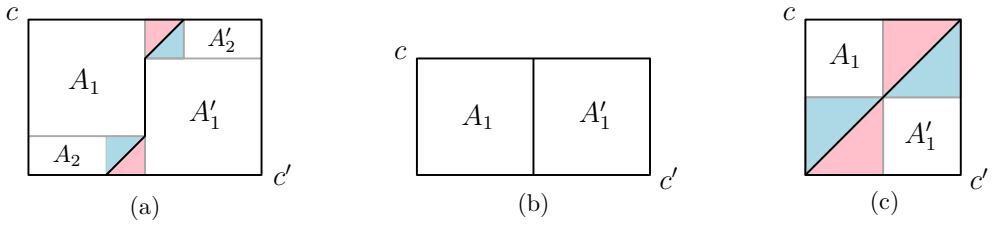


Figure 1 Various types of L_∞-bisectors. The bisectors and the boundaries of query regions are shown in black. (a): Q is "fat". The regions A_j, A'_j for $j = 1, 2$ are shown with text. (b): Q is "thin". The regions A_j and A'_j for $j = 2, 3, 4$ are empty. (c): Q is a square. The regions A_j and A'_j for $j = 2$ are empty. In both (a) and (c) regions A_3, A'_3 are colored in blue and A_4, A'_4 are colored in pink.

topleft and bottomright corner of Q, or at the topright and bottomleft corner – and return the better one. Let c and c' be the topleft and the bottomright corners of Q, respectively. In the following we describe how to compute two squares σ and σ' of minimum size that are anchored at c and c', respectively, and whose union covers S_Q. The topright/bottomleft case can then be handled in the same way.

First we determine the L_∞-bisector of c and c' inside Q; see Figure 1. The bisector partitions Q into regions A and A', that respectively have c and c' on their boundary. Obviously in an optimal solution (of the type we are focusing on), the square σ must cover $S_Q \cap A$ and the square σ' must cover $S_Q \cap A'$. To compute σ and σ', we thus need to find the points $q \in A$ and $q' \in A'$ with maximum L_∞-distance to the corners c and c' respectively. To this end, we partition A and A' into subregions such that in each of the subregions the point with maximum L_∞-distance to its corresponding corner can be found quickly via appropriate data structures discussed below. We assume w.l.o.g. that the x-span of Q is at least its y-span. We begin by presenting the details of such a partitioning for Case (a) of Figure 1 – Cases (b) and (c) can be seen as special cases of Case (a).

As Figure 1 suggests, we partition A and A' into subregions. We denote these subregions by A_j and A'_j, for $1 \leqslant j \leqslant 4$. From now on we focus on reporting the point $q \in S$ in A with maximum L_∞-distance to c; finding the furthest point from c' inside A' can be done similarly. Define four points $p(A_j) \in S$ for $1 \leqslant j \leqslant 4$ as follows.

- The point $p(A_1)$ is the point of S_Q with maximum L_∞-distance to c in A_1. Note that this is either the point with maximum x-coordinate in A_1 or the point with minimum y-coordinate.
- The point A_2 is a bottommost point in A_2.
- The point A_3 is a bottommost point in A_3.
- The point A_4 is a rightmost point in A_4.

Clearly

$$q = \arg \max_{1 \leqslant j \leqslant 4} \{d_\infty(p(A_j), c)\}, \tag{1}$$

where $d_\infty(.)$ denotes the L_∞-distance function.

Data structure. Our data structure now consists of the following components.

- We store S in a data structure $\mathcal{D}_1$ that allows us to report the extreme points in the x-direction and in the y-direction inside a rectangular query range. For this we use the structure by Chazelle [9], which uses $O(n \log^\varepsilon n)$ storage and has $O(\log n)$ query time.
- We store S in a data structure $\mathcal{D}_2$ with two components. The first component should answer the following queries: given a 45° query cone whose top bounding line is horizontal

and that is directed to the left – we obtain such a cone when we extend the region A_4 into an infinite cone –, report the rightmost point inside the cone. The second component should answer similar queries for cones that are the extension of A_3.
In the full version we describe a linear-size data structure for such queries that has $O(\log n)$ query time.

Query procedure. Given an axis-aligned query rectangle Q, we first (as already mentioned) shrink the query range so that each edge of Q contains at least on point of S. Then compute the L_∞-bisector of Q. Query $\mathcal{D}_1$ with A_1 and A_2, respectively, to get the points $p(A_1)$ and $p(A_2)$. Then query $\mathcal{D}_2$ with u and u' to get the points $p(A_3)$ and $p(A_4)$, where u and u' are respectively the bottom and the top intersection points of L_∞-bisector of Q and the boundary of Q. Among the at most four reported points, take the one with maximum L_∞-distance the corner c. This is the point $q \in S_Q \cap A$ furthest from c.

Compute the point $q' \in S_Q \cap A'$ furthest from c' in a similar fashion. Finally, report two minimum-size congruent squares σ and σ' anchored at c and c' and containing q and q', respectively.

Putting everything together, we end up with the following theorem.

▶ **Theorem 15.** *Let S be a set of n points in $\mathbb{R}^2$. For any fixed $\varepsilon > 0$, there is a data structure using $O(n \log^\varepsilon n)$ storage that can answer rectilinear 2-center queries in $O(\log n)$ time.*

▶ Remark. We note that the query time in Theorem 15 can be improved in the word-RAM model to $O(\log \log n)$ by using the range successor data structure of Zhou [24], and the point location data structure for orthogonal subdivisions by de Berg *et al.* [11].

3-center queries

Given a (shrunk) query range Q, we need to compute a set $\{\sigma_1, \sigma_2, \sigma_3\}$ of (at most) three congruent squares of minimal size whose union covers S_Q. It is easy to verify (and is well-known) that at least one of the squares in an optimal solution must be anchored at one of the corners of Q. Hence and w.l.o.g. we assume that σ_1 is anchored at one of the corners of Q. We try placing σ_1 in each corner of Q and select the placement resulting in the best overall solution. Next we briefly explain how to find the best solution subject to placing σ_1 in the leftbottom corner of Q. The other cases are symmetric. We perform two separate binary searches; one will test placements of σ_1 such that its right side has the same x-coordinate as a point in S, the other will be on possible y-coordinates for the top side. During each of the binary searches, we compute the smallest axis-parallel rectangle $Q' \subseteq Q$ containing the points of $Q \backslash \sigma_1$ (by covering $Q \backslash \sigma_1$ with axis-parallel rectangles and querying for extreme points in these rectangles). We then run the algorithm for $k = 2$ on Q'. We need to ensure that this query ignores the points already covered by σ_1. For this, recall that for $k = 2$ we covered the regions A and A' by suitable rectangular and triangular ranges. We can now do the same, but we cover $A \setminus \sigma_1$ and $A' \setminus \sigma_1$ instead.

After the query on Q', we compare the size of the resulting squares with the size of σ_1 to guide the binary search. The process stops as soon as the three sizes are the same or no further progress in the binary search can be made. Putting everything together, we end up with the following theorem.

▶ **Theorem 16.** *Let S be a set of n points in $\mathbb{R}^2$. For any fixed $\varepsilon > 0$, there is a data structure using $O(n \log^\varepsilon n)$ storage that can answer rectilinear 3-center queries in $O(\log^2 n)$ time.*

▶ Remark. Similar to Theorem 15, the query time in Theorem 16 can be improved in the word-RAM model of computation to $O(\log n \log\log n)$ time.

Acknowledgements. This research was initiated when the first author visited the Department of Computer Science at TU Eindhoven during the winter 2015–2016. He wishes to express his gratitude to the other authors and the department for their hospitality. The last author wishes to thank Timothy Chan for valuable discussions about the problems studied in this paper.

References

1 M. Abam, P. Carmi, M. Farshi, and M. Smid. On the power of the semi-separated pair decomposition. *Compututational Geometry: Theory and Applications*, 46:631–639, 2013.

2 P. K. Agarwal, R. Ben Avraham, and M. Sharir. The 2-center problem in three dimensions. *Compututational Geometry: Theory and Applications*, 46:734–746, 2013.

3 P. K. Agarwal and Cecilia M. Procopiuc. Exact and approximation algorithms for clustering. *Algorithmica*, 33:201–226, 2002.

4 Sunil Arya, David M. Mount, and Eunhui Park. Approximate geometric MST range queries. In *Proc. 36th International Symposium on Computational Geometry (SoCG)*, pages 781–795, 2015.

5 Peter Brass, Christian Knauer, Chan-Su Shin, Michiel H. M. Smid, and Ivo Vigan. Range-aggregate queries for geometric extent problems. In *Computing: The Australasian Theory Symposium 2013, CATS'13*, pages 3–10, 2013.

6 V. Capoyleas, G. Rote, and G. Woeginger. Geometric clusterings. *Journal of Algorithms*, 12:341–356, 1991.

7 T. M. Chan. Geometric applications of a randomized optimization technique. *Discrete & Compututational Geometry*, 22:547–567, 1999.

8 T. M. Chan. More planar two-center algorithms. *Compututational Geometry: Theory and Applications*, 13:189–198, 1999.

9 Bernard Chazelle. A functional approach to data structures and its use in multidimensional searching. *SIAM Journal on Computing*, 17:427–462, 1988.

10 A. W. Das, P. Gupta, K. Kothapalli, and K. Srinathan. On reporting the L_1-metric closest pair in a query rectangle. *Information Processing Letters*, 114:256–263, 2014.

11 Mark de Berg, Marc van Kreveld, and Jack Snoeyink. Two- and three-dimensional point location in rectangular subdivisions. *Journal of Algorithms*, 18:256–277, 1995.

12 D. Eppstein. Faster construction of planar two-centers. In *Proc. 8th Annual ACM-SIAM Symposiun on Discrete Algorithms (SODA)*, pages 131–138, 1997.

13 P. Gupta, R. Janardan, Y. Kumar, and M. Smid. Data structures for range-aggregate extent queries. *Compututational Geometry: Theory and Applications*, 47:329–347, 2014.

14 Sariel Har-Peled. *Geometric Approximation Algorithms*, volume 173 of *Mathematical surveys and monographs*. American Mathematical Society, 2011.

15 Sariel Har-Peled and Soham Mazumdar. On coresets for k-means and k-median clustering. In *Proc. 36th Annual ACM Symposium on Theory of Computing (STOC)*, pages 291–300, 2004.

16 M. Hoffmann. A simple linear algorithm for computing rectilinear 3-centers. *Compututational Geometry: Theory and Applications*, 31:150–165, 2005.

17 R. Z. Hwang, R. Lee, and R. C. Chang. The generalized searching over separators strategy to solve some NP-hard problems in subexponential time. *Algorithmica*, 9:398–423, 1993.

18 S. Khare, J. Agarwal, N. Moidu, and K. Srinathan. Improved bounds for smallest enclosing disk range queries. In *Proc. 26th Canadian Conference on Computational Geometry (CCCG)*, 2014.

19 H.-P. Lenhof and M. H. M. Smid. Using persistent data structures for adding range restrictions to searching problems. *Theoretical Informatics and Applications*, 28:25–49, 1994.

20 Yakov Nekrich and Michiel H. M. Smid. Approximating range-aggregate queries using coresets. In *Proc. 22nd Canadian Conference on Computational Geometry (CCCG)*, pages 253–256, 2010.

21 Jeff M. Phillips. Algorithms for ε-approximations of terrains. In *Proc. 35th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 447–458, 2008.

22 M. Sharir. A near-linear time algorithm for the planar 2-center problem. *Discrete & Compututational Geometry*, 18:125–134, 1997.

23 M. Sharir and E. Welzl. Rectilinear and polygonal p-piercing and p-center problems. In *Proc. 12th International Symposium on Computational Geometry (SoCG)*, pages 122–132, 1996.

24 Gelin Zhou. Two-dimensional range successor in optimal time and almost linear space. *Information Processing Letters*, 116:171–174, 2016.

Best Laid Plans of Lions and Men

Mikkel Abrahamsen[*1], **Jacob Holm**[2], **Eva Rotenberg**[3], **and Christian Wulff-Nilsen**[4]

1 **Department of Computer Science, University of Copenhagen, Copenhagen, Denmark**
miab@di.ku.dk
2 **Department of Computer Science, University of Copenhagen, Copenhagen, Denmark**
jaho@di.ku.dk
3 **Department of Computer Science, University of Copenhagen, Copenhagen, Denmark**
roden@di.ku.dk
4 **Department of Computer Science, University of Copenhagen, Copenhagen, Denmark**
koolooz@di.ku.dk

Abstract

We answer the following question dating back to J. E. Littlewood (1885–1977): Can two lions catch a man in a bounded area with rectifiable lakes? The lions and the man are all assumed to be points moving with at most unit speed. That the lakes are rectifiable means that their boundaries are finitely long. This requirement is to avoid pathological examples where the man survives forever because any path to the lions is infinitely long. We show that the answer to the question is not always "yes" by giving an example of a region R in the plane where the man has a strategy to survive forever. R is a polygonal region with holes and the exterior and interior boundaries are pairwise disjoint, simple polygons. Our construction is the first truly two-dimensional example where the man can survive.

Next, we consider the following game played on the entire plane instead of a bounded area: There is any finite number of unit speed lions and one fast man who can run with speed $1 + \varepsilon$ for some value $\varepsilon > 0$. Can the man always survive? We answer the question in the affirmative for any constant $\varepsilon > 0$.

1998 ACM Subject Classification I.3.5 Computational Geometry and Object Modeling

Keywords and phrases Lion and man game, Pursuit evasion game, Winning strategy

Digital Object Identifier 10.4230/LIPIcs.SoCG.2017.6

1 Introduction

'A lion and a man in a closed circular arena have equal maximum speeds. What tactics should the lion employ to be sure of his meal?'[1] These words (including the footnote) introduce the now famous lion and man problem, invented by R. Rado in the late thirties, in Littlewood's Miscellany [15]. It was for a long time believed that in order to avoid the lion, it was optimal for the man to run on the boundary of the arena. A simple argument then shows that the

* Research partly supported by Mikkel Thorup's Advanced Grant from the Danish Council for Independent Research under the Sapere Aude research career programme.

[1] The curve of pursuit (L running always straight at M) takes infinite time, so the wording has its point.

lion could always catch the man by staying on the radius OM defined by the man while approaching him as much as possible. However, A.S. Besicovitch proved in 1952 that the man has a very simple strategy (following which he will approach but not reach the boundary) that enables him to avoid capture forever no matter what the lion does. See [15] for details.

Throughout this paper, all men, lions, and other animals are assumed to be points. One can prove that two lions are enough to catch the man in a circular arena, and Croft [8] proves that in general a necessary and sufficient number of birds to catch a fly inside an n-dimensional spherical cage is just n (again, we assume that the fly and the birds have equal maximum speeds).

A well-known related discrete game is the *cop and robber game*: Let G be a finite connected undirected graph. Two players called cop C and robber R play a game on G according to the following rules: First C and then R occupy some vertex of G. After that they move alternately along edges of G. The cop C wins if at some point in time C and R are on the same vertex. If the robber R can prevent this situation forever, then R wins. The robber has a winning strategy on many graphs including all cycles of length at least 4. Therefore, the cop player C can be given a better chance by allowing him, say, k cops $C_1, \ldots, C_k$. At every turn C moves any non-empty subset of $\{C_1, \ldots, C_k\}$. Now, the *cop-number of* G is the minimal number of cops needed for C to win. Aigner and Fromme [2] observes that the cop-number of the dodecahedron graph is at least 3, since if there are only 2 cops, the robber can always move to a vertex not occupied by a cop and not in the neighbourhood of any. Furthermore, they prove that the cop-number of any planar graph is at most 3. Thus, the cop-number of the dodecahedron is exactly 3.

Returning to the lion and man game, Bollobás [6] writes that the following open problem was already mentioned by J.E. Littlewood (1885–1977): Can two lions catch a man in a bounded (planar) area with rectifiable lakes? An informal definition of a rectifiable curve is that it has finite length. We require that the boundaries of the lakes and the exterior boundary are all rectifiable curves to avoid pathological examples where the man survives forever because any path to the lions is infinite. Bollobás mentions the same problem in a comment in his edition of Littlewood's Miscellany [15] and in [7]. The problem is also stated by Fokkink et al. [11]. Berarducci and Intrigila [4] prove that the man can survive forever (for some initial positions of the man and lions) if the area is a planar embedding of the dodecahedron graph where each edge is a curve with the same length, say length 1. The proof is essentially the same as the proof by Aigner and Fromme [2] that the cop-number of the dodecahedron is at least 3: When the man is standing at a vertex, there will always be a neighbouring vertex with distance more than 1 to the nearest lion. It is thus safe for the man to run to that vertex. This, however, is a one-dimensional example. Berarducci and Intrigila raise the question whether it is possible to replace the one-dimensional edges by two-dimensional thin lines.

We present a truly two-dimensional region R in the plane where two lions are not enough to ever catch the man. We say that R is truly two-dimensional since R is a polygonal region with holes and the exterior and interior boundaries are all pairwise disjoint, simple polygons – in particular, they are clearly rectifiable. We were likewise inspired by the dodecahedron in the construction of our example. We explain the construction in Section 2.

Rado and Rado [16] and Janković [13] consider the problem where there are many lions and one man, but where the game is played in the entire unbounded plane. They prove that the lions can catch the man if and only if the man starts in the interior of the convex hull of the lions. Inspired by that problem, we ask the following question: What if the lions have maximum speed 1 and the man has maximum speed $1 + \varepsilon$ for some $\varepsilon > 0$? We prove

that for any constant ε and any finite number of lions, such a fast man can survive forever provided that he does not start at the same point as one of the lions. We explain a strategy in Section 3.

Other variants of the game with a faster man have been studied previously. Flynn [9, 10] and Lewin [14] study the problem where there is one lion and one fast man in a circular arena. The lion tries to get as close to the man as possible and the man tries to keep the distance as large as possible. Variants of the cop and robber game where the robber is faster than the cops have also been studied. See for instance [3, 12].

1.1 Definitions

We follow the conventions of Bollobás et al. [5]. Let $R \subseteq \mathbb{R}^2$ be a region in the plane on which the lion and man game is to be played, and assume that the lion starts at point l_0 and the man at point m_0. We define a *man path* as a function $m\colon [0,\infty) \longrightarrow R$ satisfying $m(0) = m_0$ and the Lipschitz condition $\|m(s) - m(t)\| \leq V \cdot |s - t|$, where V is the speed of the man. In our case, we either have $V = 1$ or, in the case of a fast man, $V = 1 + \varepsilon$ for some small constant $\varepsilon > 0$. Note that it follows from the Lipschitz condition that any man path is continuous. A *lion path* l is defined similarly, but the lions we consider always run with at most unit speed. Let $\mathcal{L}$ be the set of all lion paths and $\mathcal{M}$ be the set of all man paths. Then a *strategy* for the man is a function $M\colon \mathcal{L} \longrightarrow \mathcal{M}$ such that if $l, l' \in \mathcal{L}$ agree on $[0,t]$, then $M(l)$ and $M(l')$ also agree on $[0,t]$. This last condition is a formal way to describe that the man's position $M(l)(t)$, when he follows strategy M, depends only on the position of the lion at points in time before and including time t, i.e., he is not allowed to act based on the lion's future movements. (By the continuity of any man path, the man's position at time t is in fact determined by the lion's position at all times strictly before time t.) A strategy M for the man is *winning* if for any $l \in \mathcal{L}$ and any $t \in [0,\infty)$, it holds that $M(l)(t) \neq l(t)$. Similarly, a strategy for the lion $L\colon \mathcal{M} \longrightarrow \mathcal{L}$ is winning if for any $m \in \mathcal{M}$, it holds that $L(m)(t) = m(t)$ for some $t \in [0,\infty)$. These definitions are extended to games with more than one lion in the natural way.

It might seem unfair that the lion is not allowed to react on the man's movements when we evaluate whether a strategy M for the man is winning. However, we can give the lion full information about M and allow it to choose its path l depending on M *prior* to the start of the game. If M is a winning strategy, the man can also survive the lion running along l.

We call a man strategy M *locally finite* if it satisfies the following property: if l and l' are any two lion paths that agree on $[0,t]$ for some t then the corresponding man paths $M(l)$ and $M(l')$ agree on $[0,t+\delta]$ for some $\delta > 0$ (we allow that δ depends on $l|_{[0,t]}$). Thus, informally, the man commits to doing something for some positive amount of time dependent only on the situation so far. Bollobás et al. [5] prove that if the man has a locally finite winning strategy, then the lion does not have any winning strategy. The argument easily extends to games with multiple lions. At first sight, it might sound absurd to even consider the possibility that the lion has a winning strategy when the man also does. However, it does not follow from the definition that the existence of a winning strategy for the man implies that the lion does not also have a winning strategy. See the paper by Bollobás et al. [5] for a detailed discussion of this (including descriptions of natural variants of the lion and man game where both players have winning strategies). In each of the problems we describe, the winning strategy of the man is locally finite, so it follows that the lions do not have winning strategies. In fact, the strategies we describe satisfy the much stronger condition that they are *equitemporal*, i.e., there is a constant $\Delta > 0$ such that the man at any point in time $i \cdot \Delta$, for $i = 0, 1, \ldots$, decides where he wants to run until time $(i+1) \cdot \Delta$.

2 The Man Surviving Two Lions in a Bounded Area

In this section, we present a polygonal region R in the plane with 11 lakes. See [1] for an illustration of such a region. The exterior and interior boundaries of R are all pairwise disjoint simple polygons, and a man can survive forever in R against two lions provided that the lions are initially at a sufficient distance.

Consider a planar embedding $\mathcal{D}$ of the dodecahedron where each edge is a polygonal curve. We can obtain that all edges have the same length by prolonging some edges using a zig-zag pattern. This embedding corresponds to an area with 11 lakes and infinitely thin paths between the lakes, and as observed by Berarducci and Intrigila [4], the man can survive forever against two lions on such an embedding by deciding at each vertex which neighbouring vertex to visit next. First, we explain why it is not straight-forward to obtain the region R from $\mathcal{D}$, or, at least, why some natural initial attemps will not work.

We want to "thicken" each edge of $\mathcal{D}$ such that the boundaries of the lakes become disjoint, thus obtaining a truly two-dimensional region $\mathcal{D}'$ containing $\mathcal{D}$ as a subset. However, doing so, the point in $\mathcal{D}'$ corresponding to a vertex of $\mathcal{D}$ does not necessarily lie on the shortest path between its neighbours. We thus cannot simply employ the strategy from $\mathcal{D}$, roughly speaking, because the man must plan in advance which turn to take in the upcoming vertex. Thus, before he reaches the region R_v corresponding to a given vertex, he should already know which neighbouring vertex he will visit afterwards. Then, he can choose a path through R_v that makes the concatenated path shortest possible.

In order to carry out this idea, we first need to describe a winning strategy of the man on the dodecahedron graph with the special property that he does not make his decisions at the vertices. Let $\mathcal{G}$ be a planar embedding of the dodecahedron where all edges have length 4. The distance between two points in $\mathcal{G}$ is the length of a shortest path between the points. Let the *quarters* denote the points on the edges of $\mathcal{G}$ at distance 1 to the closest vertex. Consider a quarter x on the edge ab of $\mathcal{G}$. For a point $p \in \mathcal{G}$, $p \neq x$, let $d_a(x, p)$ be the length of a shortest *simple* path in $\mathcal{G}$ from x to p that initially follows the edge $\{a, b\}$ in direction towards a. Let $d_b(x, p)$ be defined similarly.

When the man is at a quarter x with distance 1 to the vertex a and 3 to the vertex b, we let d_{near} denote the distance from x to the closest lion with respect to d_a, and let d_{far} denote the distance from x to the closest lion with respect to d_b. To avoid confusion, we write them as $d_{\text{near}}(t)$ and $d_{\text{far}}(t)$ when x is the position of the man at the time t.

We will now show that if the lions are sufficiently far away in the initial situation, there exists a winning strategy for the man where he only takes stock of the situations in the quarters. That is, when he reaches a quarter, he must plan for the next 2 units of time where to run to, and then he has reached a quarter again, and so on.

▶ **Invariant 1.** *In the scenario described above:*

1. *The man is standing on a quarter.*
2. $\min\{d_{\text{near}}, d_{\text{far}}\} \geq 1$.
3. *At least one of the two following statements is true:*
   - $d_{\text{near}} \geq 3$
   - $d_{\text{far}} \geq 7$

▶ **Lemma 2.** *If Invariant 1 is satisfied initially, the man has a winning strategy by which he runs from quarter to quarter at unit speed so that Invariant 1 is true at any quarter. The strategy maintains Invariant 1 Point 2 at all times, that is, that the closest lion is always at least at distance* 1.

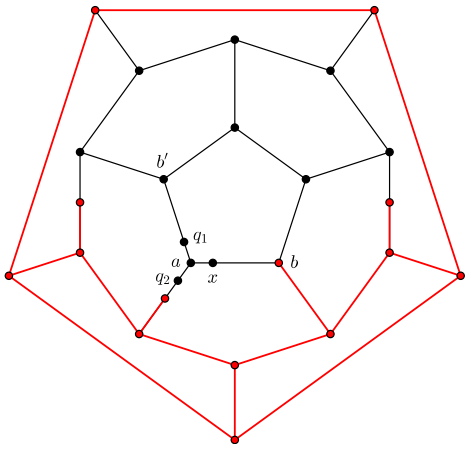


Figure 1 A situation from the proof of Lemma 2. Imagine that all edges have length 4. The lion l_{near} is in the red part.

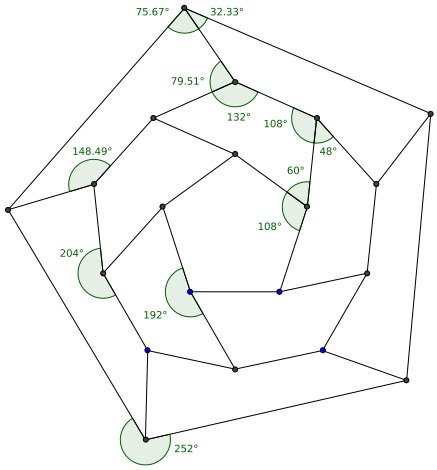


Figure 2 The embedding $\mathcal{D}$ of the dodecahedron. All edges have lengths 1 or 3.

Proof. Let x denote the position of the man at the time t, and assume the invariant holds. We prove that he can run to another quarter x' without getting caught such that the invariant again holds when he reaches x'.

The proof goes by inspecting cases. Let ab be the edge containing x and suppose a be the nearest vertex to x and b the furthest.

Case 1: $d_{\text{far}}(t) \geq 7$. Let y denote the other quarter on the same edge as x. We claim that the man can run to y without violating the invariant. We must thus argue that the invariants are satisfied at time $t+2$ for a man situated at y. First, note that he will not encounter any lion while running towards y because $d_{\text{far}}(t) > 4$. Note also that $d_{\text{far}}(t+2) \geq 1$, since $d_{\text{near}}(t) \geq 1$ and the worst case is that the lion follows the man. Furthermore, $d_{\text{near}}(t+2) \geq 7-4 = 3$, since $d_{\text{far}}(t) \geq 7$ and the worst case is that the man and lion have run towards each other. Thus, the invariant holds at the time $t+2$.

Case 2: $d_{\text{far}} < 7$, and thus $d_{\text{near}}(t) \geq 3$. In this case, we exploit the fact that d_{far} is so small that we can bound $d_{\text{far}}(t+2)$ from below. Let l_{far} denote the lion at distance d_{far} from x, and let l_{near} denote the other lion. Consider the two other quarters at distance 1 from a, call them q_1 and q_2. Assume without loss of generality that q_1 is furthest from l_{near}. The situation is sketched in Figure 1. We now argue that the man can choose to run towards q_1 without getting eaten, and while maintaining the invariant. Let b' denote the vertex at distance 3 to q_1. Note that $d_{b'}(q_1, l_{\text{far}}(t)) \geq 11$ and thus, $d_{b'}(q_1, l_{\text{far}}(t+2)) \geq 9$.

In Figure 1, the points that are both ≥ 3 from x, and (weakly) closer to q_2 than to q_1, are marked with red, and hence by our choice of q_1, l_{near} must be in the subset marked with red at time t. As is easily seen by inspection, $d_{b'}(q_1, l_{\text{near}}(t)) \geq 9$, and thus $d_{b'}(q_1, l_{\text{near}}(t+2)) \geq 7$. But then, $d_{\text{far}}(t+2) \geq \min\{9, 7\} = 7$, and Invariant 1.1 and 3 are maintained.

To see that Invariant 1.2 is still maintained, note that $d_a(q_1, l_{\text{near}}(t)) \geq 3$ and therefore $d_a(q_1, l_{\text{near}}(t+2)) \geq 1$. Similarly, since $d_b(x, l_{\text{far}}(t)) \geq 1$, we have $d_a(q_1, l_{\text{far}}(t)) \geq 3$ so that $d_a(q_1, l_{\text{far}}(t+2)) \geq 1$. Thus, $l_{\text{near}}(t+2) \geq 1$, and we are done. ◀

Our first goal is to find an embedding $\mathcal{G}$ of the dodecahedron in the plane with the properties described below, which will make it easier for us to construct the region R.

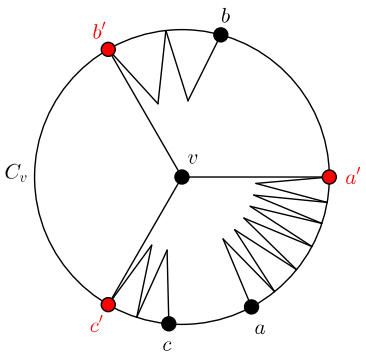


Figure 3 Regardless of angles between a, b, c, we can introduce bends to make the three edges meet at v in angles of size $\frac{3\pi}{2}$ and at the same time extend the lengths suitably.

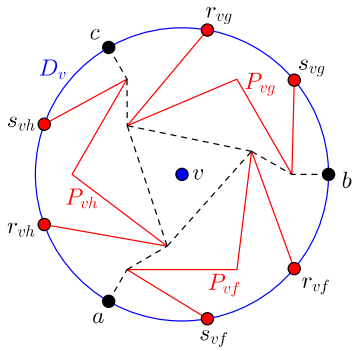


Figure 4 The shortest paths in the circle D_v between any two of a, b, c, that avoid crossing the polygonal curves P_{vf}, P_{vg}, P_{vh} all have length $1/8$.

▶ **Lemma 3.** *There exists a planar embedding $\mathcal{G}$ of the dodecahedron such that*

- *all edges have length 4,*
- *all edges consist of line segments with lengths being multiples of $\frac{1}{8}$,*
- *any pair of line segments from different edges that meet at a vertex each have length $1/4$ and form an angle of size $\frac{2\pi}{3}$, and*
- *for any vertex v, the circle D_v centered at v with radius $1/16$ only intersects the three edges incident to v.*

After proving this lemma, we derive from $\mathcal{G}$ a truly two-dimensional area R in the plane where the man can survive against two lions. Lemma 2 gives a winning strategy for the man in $\mathcal{G}$ where he runs from quarter to quarter. The paths along which he runs in R will be exactly the same as in $\mathcal{G}$ except for inside the circles D_v.

We first need the following elementary geometric observations:

▶ **Observation 4.** *There exists a planar embedding $\mathcal{D}$ of the dodecahedron such that all edges have length 1 or 3. $\mathcal{D}$ furthermore has the property that the circle of radius $\frac{1}{4}$ centered at any vertex v only intersects the three edges incident to v. (See Figure 2.)*

▶ **Lemma 5.** *For any three points a, b, c on a circle C, there exist a equilateral triangle with corners a', b', c' on C where $\{a, b, c\}$ and $\{a', b', c'\}$ are disjoint and such that, when considering the points a, b, c, a', b', c' all together, a is a neighbour of a', and b is a neighbour of b', and c is a neighbour of c'.*

Proof. See Figure 3. The points a, b, c divide C into three arcs. Clearly, we can choose an equilateral triangle with corners on C disjoint from $\{a, b, c\}$ so that not all three corners of the triangle are on the same arc. It is now easy to label the corners of the triangle with a', b', c' to satisfy the lemma. ◀

We are now ready to prove that a planar embedding $\mathcal{G}$ of the dodecahedron exists as stated in Lemma 3.

Proof of Lemma 3. Start with the embedding $\mathcal{D}$ shown in Figure 2, where all edges have length 1 or 3. Consider a vertex v and the circle C_v of radius $r = \frac{1}{4}$ centered at v. Assume the three edges incident to v enter C_v in the points a, b, c, and let u_a, u_b, u_c be the neighbouring

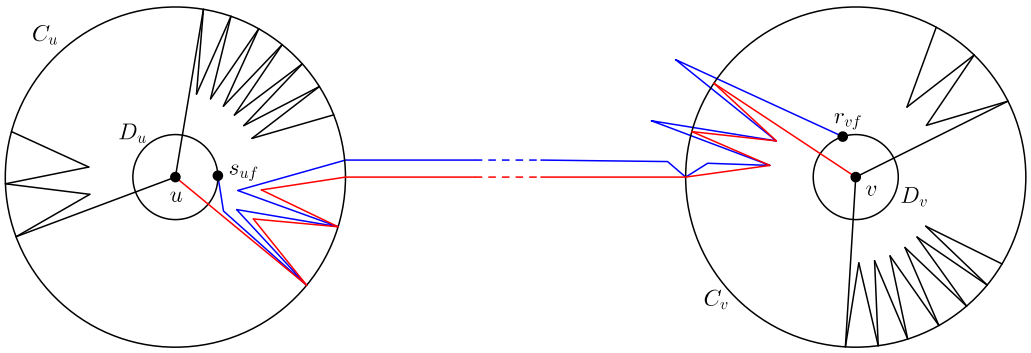


Figure 5 The edge e_{uv} of $\mathcal{G}$ is red and is one of the edges bounding the face f, which is above e_{uv}. The polygonal curve Q_{uv}, which is on the boundary of the lake L_f, is blue.

vertices of v such that a is a point on the edge $\{u_a, v\}$, b is a point on $\{u_b, v\}$, and c is a point on $\{u_c, v\}$. We now delete the segments va, vb, and vc, and therefore need to reconnect a, b, and c to v. We explain how to reconnect a to v; b and c are handled analogously. We find points a', b', c' on C_v as described in Lemma 5. See Figure 3. We first connect a' to v. We now need to connect a to a' using some bends. A *bend* is two segments xy and yz, each of length $r/2 = 1/8$, such that x and z are on C_v and y is in the interior of C_v. If the edge $\{u_a, v\}$ had length 3 in $\mathcal{D}$, we make two bends that together connect a and a'. We thus increase the length of the edge $\{u_a, v\}$ by $1/2$ in each end and the resulting edge has length 4. If the edge $\{u_a, v\}$ had length 1 in $\mathcal{D}$, we connect a and a' by 6 bends, corresponding to extending the length of the edge by 3. The result is a planar embedding $\mathcal{G}$ of the dodecahedron with the properties stated in the lemma. ◀

We now describe how to make the region R. We want each quarter of $\mathcal{G}$ to be a point in R and we want all pairs of quarters to have the same distances in $\mathcal{G}$ and R. It will then follow from Lemma 2 that the man has a winning strategy by running from quarter to quarter in R. We make one lake L_f corresponding to each face f of $\mathcal{G}$. Here, we also consider the outer boundary of R to be the boundary of an unbounded lake corresponding to the exterior face of $\mathcal{G}$. The shortest paths in R will be polygonal paths with corners at convex corners of the lakes. Outside the circles D_v, the paths along which the man will run are exactly the paths in $\mathcal{G}$. Inside a circle D_v, we need to take special care to ensure that the man can always run along an optimal path.

We now explain the construction of the lakes L_f corresponding to faces f of $\mathcal{G}$. Consider a vertex v of $\mathcal{G}$ and the faces f, g, h on which v is a vertex. We first describe how the boundaries of L_f, L_g, L_h look in the circle D_v of radius $1/16$ centered at v. See Figure 4. Let a, b, c be the points where the edges incident to v enter D_v. Suppose that the arc on D_v from a to b is in the face f, the arc from b to c is in g, and the arc from c to a is in h. We now create three polygonal curves P_{vf}, P_{vg}, P_{vh} inside D_v so that the shortest path between any two of a, b, c contained in D_v and not crossing any of P_{vf}, P_{vg}, P_{vh} has length $1/8$. The curve P_{vf} starts at a point r_{vf} on D_v and ends at a point s_{vf} on D_v, and the endpoints r_{vf}, s_{vf} are inside f, and similarly for the faces g, h. These properties are easy to obtain by a construction as shown in Figure 4. The curves P_{vf}, P_{vg}, P_{vh} will be part of the boundary of the lakes L_f, L_g, L_h, respectively.

We now explain how to construct the rest of the boundary of each lake L_f. Consider a face f of $\mathcal{G}$ and assume that the vertices on f are $uvxyz$ in that order on f. The curves $P_{uf}, P_{vf}, P_{xf}, P_{yf}, P_{zf}$ appear on the boundary of L_f in that order. In the following, we describe how to connect the end s_{uf} of P_{uf} with the start r_{vf} of P_{vf} – the other curves are connected in a completely analogous way. See Figure 5. Let e_{uv} be the edge of $\mathcal{G}$ between u and v, thus, e_{uv} is a polygonal curve. Let a *corner* of e_{uv} be a common point of two neighbouring segments of e_{uv}. We make a polygonal curve Q_{uv} corresponding to e_{uv}. Q_{uv} starts at s_{uf} and ends at r_{vf} so that it connects P_{uf} and P_{vf}. Q_{uv} stays near e_{uv} inside f and touches e_{uv} at the corners of e_{uv} which are convex corners of f. To summarize, Q_{uv} has the following properties:

1. Q_{uv} starts at s_{uf} and ends at r_{vf},
2. Q_{uv} is completely contained in f,
3. Q_{uv} is, except for the endpoints s_{uf}, r_{vf}, outside the circles D_u and D_v,
4. Q_{uv} and $Q_{u'v'}$ are completely disjoint for any ordered pair $(u'v') \neq (u, v)$ so that $\{u', v'\}$ is an edge of $\mathcal{G}$, and
5. Q_{uv} touches e_{uv} at a point p if and only if p is a corner of e_{uv} which is a convex corner of f.

Observe that Q_{vu} (note: not Q_{uv}!) touches e_{uv} at the corners which are concave corners of f, since those are convex corners of the neighbouring face on the other side of e_{uv}.

▶ **Theorem 6.** *There exists a polygonal region R in the plane with holes where the exterior and interior boundaries are all pairwise disjoint and such that the man has a winning strategy against two lions.*

Proof. R is the region that we get by removing from $\mathbb{R}^2$ the interior of each of the lakes L_f. Thus, the boundary of each lake is included in R, so that R is a closed set. R is also bounded because we remove the interior of the unbounded lake corresponding to the exterior face of $\mathcal{G}$. Note that any point on an edge e_{uv} of $\mathcal{G}$ which is outside the circles D_u and D_v is a point in R. Since the quarters of e_{uv} are outside the circles D_u and D_v, it follows that they are also points in R. Furthermore, our construction ensures that the distance in R between any two quarters is the same as in $\mathcal{G}$. Let $\mathcal{G}'$ be the points in R which are on some shortest path between two quarters in R. Thus, $\mathcal{G}'$ are the points that the man can possibly visit when running along shortest paths in R from quarter to quarter.

Let l_1 and l_2 be two lions in R. We define projections l_1' and l_2' of the lions l_1 and l_2 to be the closest points in $\mathcal{G}'$ (with respect to distances in R). We now define l_1'' and l_2'' to be projections of l_1' and l_2' in $\mathcal{G}$ in the following way. Outside the circles D_v, $\mathcal{G}$ and $\mathcal{G}'$ coincide, and here we simply define $l_i'' := l_i'$. Suppose now that l_i' is inside a circle D_v for some vertex v of $\mathcal{G}$. See Figure 6. Suppose that the three edges incident to v enter D_v at the points a, b, c. The projection l_i' is a point on one of the shortest paths between a pair of the points a, b, c. Recall that these shortest paths all have length $1/8$. Assume without loss of generality that l_i' is on the path from a to c. Let d be the distance from a to l_i' in R, so that $0 \leq d \leq 1/8$. If $d = 1/16$, we define $l_i'' := v$. Otherwise, if $d < 1/16$, we let l_i'' be the point on the segment av in $\mathcal{G}$ with distance d to a, i.e., $l_i'' \in av$ so that $\|al_i''\| = d$. Similarly, if $d > 1/16$, we let l_i'' be the point on bv with distance $1/8 - d$ to b.

We now prove that l_i'' moves with at most unit speed in $\mathcal{G}$. It will then follow from Lemma 2 that the man has a winning strategy.

$\mathcal{G}'$ subdivides R into some regions $R_1', \ldots, R_k'$, which are the connected components of $R \setminus \mathcal{G}'$. Let $R_i = \overline{R_i'}$ be the closure of R_i'. Now, $R = \bigcup_{i=1}^{k} R_i$. Inside each circle D_v, there is a *triangular* region bounded by three segments from $\mathcal{G}'$. All other regions are bounded by

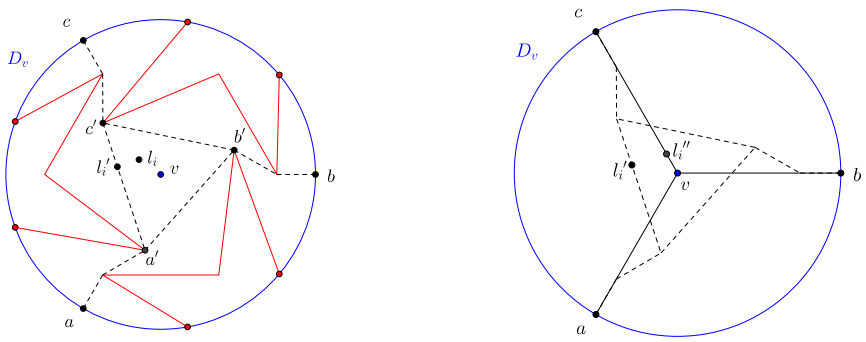


Figure 6 The projection of the lion's position l_i onto the point l_i' of $\mathcal{G}'$ (left), and the projection of l_i' onto the point l_i'' of $\mathcal{G}$ (right). The dashed lines illustrate $\mathcal{G}'$, and the solid lines illustrate $\mathcal{G}$. In the left figure, l_i' is the closest point on $\mathcal{G}'$ to l_i. In the right figure, the length of the segment cl_i'' equals the length of the dashed path from c to l_i'.

a polygonal curve $C \subset \partial L_f$ on the boundary of some lake L_f and a concave chain $\mathcal{H} \subset \mathcal{G}'$. Call such a region *normal*. If the lion l_i is in a normal region R_j with boundary $\partial R_j = C \cup \mathcal{H}$ as described before, the projection l_i' is on $\mathcal{H}$. It then follows from the concavity of $\mathcal{H}$ that l_i', and thus also l_i'', moves continuously and with at most unit speed.

However, when l_i is inside a triangular region in D_v, the projection l_i' might jump from one segment of the triangle to another. Suppose that the three edges incident to v enter D_v at the points a, b, c as in Figure 6. Let a' be the point where the shortest paths from a to b and c separate and define b' and c' similarly. Thus, the points $a'b'c'$ are the corners of the triangular region. Suppose that l_i' jumps from $a'b'$ to $a'c'$. Then, the distance from l_i to $a'b'$ and $a'c'$ is the same and the distance from a to l_i' before and after the jump is at most $1/16$, since otherwise, l_i would be closer to the segment $b'c'$ than to $a'b'$ and $a'c'$. It follows that l_i' jumps from one point to another which have the same projection l_i''. Thus, l_i'' moves continuously and with at most unit speed.

The man now employs the strategy from Lemma 2 in the following way. He imagines that he is playing in the dodecahedron $\mathcal{G}$ against the lions l_1'' and l_2''. Assume therefore that Invariant 1 holds initially. The strategy tells the man to which neighbouring quarter to run. That quarter also exists in $\mathcal{G}'$, and has the same distance, so the man runs to that quarter in $\mathcal{G}'$. Since l_1'' and l_2'' run with at most unit speed, the man can escape them forever. When the man is outside the circles D_v, it is a necessary condition for the lions to catch the man that l_1'' or l_2'' coincide with the man, so we conclude that they cannot catch him outside the circles. When the man is inside a circle D_v, we know from Lemma 2 that l_1'' and l_2'' are at least 1 away from the man. Therefore, l_1 and l_2 must be outside D_v, and hence they cannot catch him in that case either. Thus, the man survives forever in R. ◀

3 The Fast Man Surviving any Number of Lions in the Plane

Finally, we consider the case where the man is just slightly faster than the lions in the unbounded plane without obstacles. In this case, the man is able to escape arbitrarily many lions. The full proofs of some of the claims below can be found in [1].

▶ **Theorem 7.** *In the plane $\mathbb{R}^2$, for any $\varepsilon > 0$, a man able to run at speed $1 + \varepsilon$ has a locally finite strategy to escape the convex hull of any number $n \in \mathbb{N}$ of unit-speed lions, provided*

that the man does not start at the same point as a lion. Thus, the man has a locally finite winning strategy.

In fact, we prove that the man is able to keep some minimum distance $d_{\varepsilon,n}$ to any lion, where $d_{\varepsilon,n}$ only depends on ε, n, and the initial distances to the lions. Thus, if the n lions and man were disks with radius $< \frac{1}{2}d_{\varepsilon,n}$, the man is still able to escape.

We proceed by induction on the number n of lions. We define strategies M_j for the man to keep distance c_j to the first j lions. The j'th strategy yields a curve consisting of line segments all of the same length.

Inductively, the man can keep a safety distance c_{n-1} to the $n-1$ first lions by running at speed $1+\varepsilon_{n-1}$, where $\varepsilon_1 < \varepsilon_2 < \ldots < \varepsilon_n < \varepsilon$. The bends of the curve defined by strategy M_{n-1} are milestones that he runs towards when avoiding n lions. If the n'th lion ℓ_n is in the way, the man makes an *avoidance move*, keeping a much smaller safety distance c_n to ℓ_n and running slightly faster at speed ε_n (see Figure 9). Intuitively, when performing avoidance moves, the man runs counter-clockwise around a fixed-diameter circle centered at the lion.

After a limited number of avoidance moves, the man can make an *escape move*, where he simply runs towards the milestone defined by the strategy M_{n-1}.

By choosing c_n sufficiently small, we can make sure that the detour caused by the n'th lion is so small that it can only annoy the man once for each of the segments of the strategy M_{n-1}, and thus that he is ensured to have distance at least $c_{i-1}/2$ to the position defined by M_{n-1} and hence not in danger of the $(n-1)$'st lions.

▶ **Theorem 8.** *A man able to run at speed $1+\varepsilon$ for any $\varepsilon > 0$ has a locally finite strategy to escape the convex hull of any number $n \in \mathbb{N}$ of unit-speed lions, provided that the man does not start at the same point as a lion. Thus, the man has a locally finite winning strategy.*

Proof. We assume without loss of generality that $\varepsilon < 1$. Let $l_1, \ldots, l_n$ be n arbitrary lion paths and let the man start at position m_0 such that $m_0 \neq l_i(0)$ for all i. We show that the man has a strategy M_n with the following properties:

1. The man is always running at speed $1+\varepsilon_n$, where $\varepsilon_n := (1-2^{-n}) \cdot \varepsilon$.
2. The path defined by $M_n(l_1, \ldots, l_n)$ is a polygonal path with corners $m_0 m_1 \ldots$ and each segment $m_i m_{i+1}$ has the same length $\Delta_n \cdot (1+\varepsilon_n)$. Thus, the time it takes the man to run from m_i to m_{i+1} is Δ_n.
3. Let $t_i := i \cdot \Delta_n$ be the time where the man leaves m_i in order to run to m_{i+1}. The point m_{i+1} can be determined from the positions of the lions at time t_i.
4. There exists a *safety distance* $c_n > 0$ such that for any $i = 1, \ldots,$ any $t \in [t_i, t_{i+1}]$, and any point $x \in m_i m_{i+1}$, it holds that $\mathrm{dist}(x, \{l_1(t), \ldots, l_n(t)\}) \geq c_n$.
5. There is a corner $m_i = M_n(t_i)$ such that for all $t \geq t_i$,

$$M_n(l_1, \ldots, l_n)(t) \notin \mathcal{CH}\{l_1(t), \ldots, l_n(t)\}.$$

Clearly, it follows from the properties that M_n is a winning strategy for the man fulfilling the requirements in the theorem. We prove the statement by induction on n. If there is only one lion, the man will run on the same ray all the time with constant speed $1+\varepsilon_1 = 1+\varepsilon/2$. The man chooses the direction of the ray to be $m_0 - l_1(0)$. This strategy obviously satisfies the stated properties. Assume now that a strategy M_{n-1} with the stated properties exists for $n-1 \geq 1$ lions and consider a situation with n lions running along paths $l_1, \ldots, l_n$.

Assume without loss of generality that the lions are numbered according to their (increasing) distance to the man at time 0, i.e., $\|m_0 l_1(0)\| \leq \|m_0 l_2(0)\| \leq \cdots \leq \|m_0 l_n(0)\|$. For any $i \in \{1, \ldots, n\}$, let M_i be shorthand for $M_i(l_1, \ldots, l_i)$ and m shorthand for M_n.

At any time t, let the succeeding corner on the strategy M_{n-1} be

$$g(t) := M_{n-1}(\lfloor t/\Delta_{n-1} + 1 \rfloor \cdot \Delta_{n-1}).$$

By property 3, the man can always compute the point $g(t)$.

We first describe the intuition behind the man's strategy without specifying all details, and later give a precise description. In the situation with n lions, the man attempts to run according to the strategy for the $n-1$ first lions, i.e., the strategy M_{n-1}. Thus, at any time t, the man's goal is to run towards the point $g(t)$. However, the lion l_n might prevent him from doing so. Compared to the case with $n-1$ lions, the man has increased his speed by $1+\varepsilon_n - (1+\varepsilon_{n-1}) = 2^{-n}\varepsilon$, so he has time to take detours while still following the strategy M_{n-1} approximately.

Assume that we have defined the man's strategy up to time t. If he is close to the n'th lion, i.e., the distance $\|m(t)l_n(t)\|$ is close to r, for some small constant $r > 0$ to be specified later, he runs counterclockwise around the lion, maintaining approximately distance r to the lion. He does so until he gets to a point where running directly towards $g(t)$ will not decrease his distance to the lion. He then escapes from the lion, running directly towards $g(t)$. Doing so, he can be sure that the lion cannot disturb him anymore until he reaches $g(t)$ or $g(t)$ has changed.

We choose r so small that when the man is running around the lion, we are in one of the following cases:

- The lion is so close to $g(t)$ that the man is within the safety distance c_{n-1} from $g(t)$, and thus in no danger of the lions $l_1, \ldots, l_{n-1}$.
- After running around the lion in a period of time no longer than $12\pi r/\varepsilon_n$, the man escapes by running directly towards $g(t)$ without decreasing the distance to the lion. By choosing r sufficiently small, we can therefore limit the duration, and hence the length, of the detour that the lion can force the man to run, so that the man is ensured to be within the safety distance from the lions $l_1, \ldots, l_{n-1}$ during the detour.

We now describe the details that make this idea work. We define

$$r := \min\left\{\frac{\Delta_{n-1}\varepsilon_n(\varepsilon_n - \varepsilon_{n-1})}{2+2\varepsilon_n+18\pi(1+\varepsilon_n)}, \frac{\varepsilon_n c_{n-1}}{4+4\varepsilon_n+24\pi(1+\varepsilon_n)}\right\},$$
$$\rho := 2r/\varepsilon_n,$$
$$\theta := \arccos\frac{1}{1+\varepsilon_n},$$
$$\varphi \in (0, \pi/2] \quad \text{so that} \quad \tan\theta = \frac{\rho\sin\varphi}{\rho\cos\varphi - 2r}, \quad \text{and}$$
$$\Delta_n > 0 \text{ so that } 2\arcsin\frac{(1+\varepsilon_n)\Delta_n}{2(r-\Delta_n)} + \frac{\Delta_n}{\rho} \le \varphi,\ \Delta_n < \frac{r}{3+\varepsilon_n},\ \text{and } \Delta_{n-1}/\Delta_n \in \mathbb{N}.$$

We note that φ can be chosen since the function $x \longmapsto \frac{\rho\sin x}{\rho\cos x - 2r}$ is 0 for $x = 0$ and tends to $+\infty$ as $\rho\cos x$ decreases to $2r$. As for Δ_n, the function $x \longmapsto 2\arcsin\frac{(1+\varepsilon_n)x}{2(r-x)} + \frac{x}{\rho}$ is 0 for $x = 0$ and increases continuously, and hence Δ_n can be chosen.

Define a point in time t to be a *time of choice* if t has the form $t_i := i\Delta_n$ for $i \in \mathbb{N}_0$. At any time of choice t_i, the man chooses the point $m(t_{i+1})$ at distance $(1+\varepsilon_n)\Delta_n$ from his current position $m(t_i)$ by the following strategy (see Figures 7–9):

A. Suppose first that $\|m(t_i)l_n(t_i)\| \ge r + \Delta_n(1+\varepsilon_n)$. Then the man chooses the direction directly towards $g(t_i)$. In the exceptional case that $m(t_i) = g(t_i)$, he chooses an arbitrary direction.

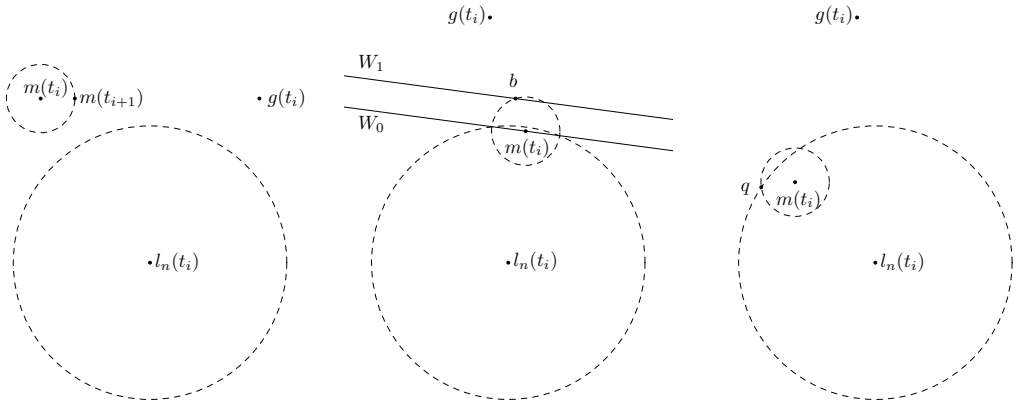


Figure 7 A free move. The circles with centers $m(t_i)$ and $l_n(t_n)$ have radii $(1+\varepsilon_n)\Delta_n$ and r, respectively.

Figure 8 An escape move. The man runs to b.

Figure 9 An avoidance move. The man runs to q.

B. Suppose now that $\|m(t_i)l_n(t_i)\| < r+\Delta_n(1+\varepsilon_n)$ and consider the case $m(t_i) \neq g(t_i)$. Let b be the point at distance $(1+\varepsilon_n)\Delta_n$ from $m(t_i)$ in the direction towards $g(t_i)$. If there exist two parallel lines W_0 and W_1 such that $m(t_i) \in W_0$, $b \in W_1$, $\text{dist}(l_n(t_i), W_0) \geq r - \Delta_n$, and $\text{dist}(l_n(t_i), W_1) \geq \text{dist}(l_n(t_i), W_0) + \Delta_n$, then the man runs to b.

C. In the remaining cases, the circles $C(m(t_i), \Delta_n(1+\varepsilon_n))$ and $C(l_n(t_i), r)$ intersect at two points p and q such that the arc on $C(l_n(t_i), r)$ from p counterclockwise to q is in the interior of $C(m(t_i), \Delta_n(1+\varepsilon_n))$. The man then runs towards the point q.

A move defined by case A, B, or C is called a *free move*, an *escape move*, or an *avoidance move*, respectively. Let *move* i be the move that the man does during the interval $[t_i, t_{i+1})$.

▶ **Claim 9.** *At any time of choice t_i, it holds that*

$$\|m(t_i)l_n(t_i)\| \geq r - \Delta_n$$

and if the preceding move was an avoidance move, it also holds that

$$\|m(t_i)l_n(t_i)\| \leq r + \Delta_n.$$

Furthermore, at an arbitrary point in time $t \in [t_{i-1}, t_i]$ and any point $m' \in m([t_{i-1}, t_i])$ it holds that

$$0 < r - (3+\varepsilon_n)\Delta_n \leq \|m'l_n(t)\|$$

and if move $i-1$ is an avoidance move then additionally

$$\|m'l_n(t)\| \leq r + (3+\varepsilon_n)\Delta_n.$$

Proof. (Sketch) The proof is by induction on i. The induction step for the first inequality follows easily from the speed of the man and the speed of the lion l_n and (in case of an escape or avoidance move) from considering the distance $\|m(t_i)l_n(t_{i-1})\|$. For the second inequality, note that $\|m(t_i)l_n(t_{i-1})\| = r$. The third inequality follows by considering the combined speed of the man and the lion l_n and by observing that $\|m(t_{i-1})l_n(t_{i-1})\| \geq r-\Delta_n$. A similar argument using the inequality $\|m(t_{i-1})l_n(t_{i-1})\| \leq r + \Delta_n$ shows the fourth inequality. ◀

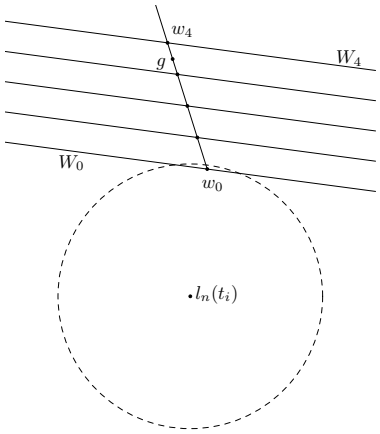


Figure 10 The distance between two consecutive of the parallel lines $W_0, \ldots, W_4$ is at least Δ_n, which proves that the man runs from $m(t_i) = w_0$ to w_4 unless g moves in the meantime.

▶ **Claim 10.** *An avoidance move is succeeded by an avoidance move or an escape move. When the man does an escape move, he will not do an avoidance move before he reaches $g(t)$ or $g(t)$ moves.*

Proof. Consider move i. We know from Claim 9 that if move $i-1$ was an avoidance move, then $\|m(t_i)l_n(t_i)\| \leq r + \Delta_n < r + (1+\varepsilon_n)\Delta_n$, so move i cannot be a free move.

For the second part of the statement, assume that move i is an escape move. Let $g := g(t_i)$. Let $w_0, \ldots, w_k$ be a sequence of points on the ray from $m(t_i)$ with direction to g such that $w_0 = m(t_i)$, $\|w_0 w_j\| = j(1+\varepsilon_n)\Delta_n$, and k is minimum such that either $g \in w_{k-1}w_k$ or $g(t') \neq g$ for some $t' \in [t_{i+k-1}, t_{i+k}]$. See Figure 10. Let W_0 and W_1 be the parallel lines defined in case B for move i. We define lines W_j for $j \geq 2$ to be parallel to W_0 and passing through w_j. We claim that for any $j \in \{0, \ldots, k-1\}$, the man moves from w_j to w_{j+1} during move $i+j$ using either an escape move or a free move. We prove this by induction on j. It holds for $j = 0$ by assumption, so assume it holds that $m(t_{i+j}) = w_j$ and that move $i+j-1$ was an escape move or a free move. Since the distance between consecutive lines W_j and W_{j+1} is at least Δ_n, it follows that $\text{dist}(l_n(t_i), W_j) \geq r + (j-1)\Delta_n$ and hence that $\text{dist}(l_n(t_{i+j}), W_j) \geq r - \Delta_n$. Now, if $\|m(t_{i+j})l_n(t_{i+j})\| < r + \Delta_n(1+\varepsilon_n)$, then the lines W_j and W_{j+1} are a witness that move $i+j$ is an escape move so that the man moves to w_{j+1}. Otherwise, move $i+j$ is a free move, in which case the man moves to w_{j+1}. Finally, since $g(t)$ moves or the man reaches g during move $i+k$, the statement holds. ◀

Define $\rho' := \rho + r + (3+\varepsilon_n)\Delta_n$ and $\tau := 6\pi r/\varepsilon_n$.

▶ **Claim 11.** *If move i is an avoidance move, one of the following events occurs before τ time has passed: (i) $g(t)$ moves, (ii) $\|m(t)g(t)\| < \rho'$, or (iii) the man makes an escape move.*

Proof Sketch. If the first two events do not occur, it follows from Claim 10 that the man keeps doing avoidance moves during this time. Let $\xi(t)$ resp. $\eta(t)$ denote the angle of the vector $\overrightarrow{l_n(t)m(t)}$ resp. $\overrightarrow{l_n(t)g(t_i)}$. A key observation is that if the difference in these angles is small, the man makes an escape move since then the lion and the goal g are roughly on opposite sides of the man. Showing that this difference eventually becomes small involves showing that η increases by at least 2π more than ξ after τ time so that at some point in time $t \in [t_i, t_i + \tau]$, vectors $\overrightarrow{l_n(t)m(t)}$ and $\overrightarrow{l_n(t)g(t_i)}$ have the same orientation. By Claim 9,

the lion l_n never gets closer than ρ to $g(t_i)$ which implies that the change in η is small in any time interval $[t_j, t_{j+1}]$. Since the man keeps a minimum distance to the lion, it similarly follows that the change in ξ is small in $[t_j, t_{j+1}]$. Picking j to be the maximum such that $t_j \leq t$ gives $t - t_j \leq \Delta_n$ which implies that the difference in the two angles is small at time t_j at which point the man makes an escape move. Since $t_j \leq t + \tau$, the lemma follows. ◀

For $i \in \mathbb{N}_0$, define the *canonical interval* I_i as $I_i := [i\Delta_{n-1}, (i+1)\Delta_{n-1})$, i.e., I_i is the interval of time where the man would run from the i'th to the $(i+1)$'st corner on the polygonal line defined by the strategy M_{n-1}. We say that I_i *ends* at time $t = (i+1)\Delta_{n-1}$. Note that if $t \in I_i$, then $g(t) = M_{n-1}((i+1)\Delta_{n-1})$ and $g(t)$ moves when I_i ends.

As a consequence of Claim 10 and Claim 11, we get the following.

▶ **Claim 12.** *If $t \in I_i$ and $\|m(t)g(t)\| \leq \rho'$, then for every $t' > t$, $t' \in I_i$, we have*

$$\|m(t')g(t)\| \leq \rho' + (1+\varepsilon_n)\tau.$$

▶ **Claim 13.** *For any $i \in \mathbb{N}_0$ and at any time during the canonical interval I_i, the man is at distance at most $\rho' + 2(1+\varepsilon_n)\tau$ away from the segment $M_{n-1}(I_i)$ and when I_i ends, the man is within distance $\rho' + (1+\varepsilon_n)\tau$ from the endpoint $M_{n-1}((i+1)\Delta_{n-1})$ of the segment.*

Proof. We prove the claim by induction on i. To easily handle the base-case, we introduce an auxiliary canonical interval $I_{-1} = [-\Delta_{n-1}, 0)$ and assume that the lions and the man are standing at their initial positions during all of I_{-1}. The statement clearly holds for $i = -1$.

Assume inductively that the statement holds for I_{i-1} and consider the interval I_i. Let $g := M_{n-1}((i+1)\Delta_{n-1})$. The additional distance that the man runs during I_i when his speed is $1+\varepsilon_n$ as compared to the speed $1+\varepsilon_{n-1}$ is $\Delta_{n-1}(\varepsilon_n - \varepsilon_{n-1})$. It follows from the definition of r that

$$\Delta_{n-1}(\varepsilon_n - \varepsilon_{n-1}) \geq \rho + 2r + 3(1+\varepsilon_n)\tau > \rho' + 3(1+\varepsilon_n)\tau.$$

By the induction hypothesis, the man is within a distance of $\rho' + (1+\varepsilon_n)\tau$ from $M_{n-1}(i\Delta_{n-1})$ at time $i\Delta_{n-1}$. Thus, his distance to g at the beginning of interval I_i is at most $\Delta_{n-1}(1+\varepsilon_{n-1}) + \rho' + (1+\varepsilon_n)\tau$, where $\Delta_{n-1}(1+\varepsilon_{n-1})$ is the length of the interval $M_{n-1}(I_i)$. If the man does not do any avoidance moves during I_i, he runs straight to g, so it follows that he reaches g at time $(i+1)\Delta_{n-1} - 2\tau$ at the latest. Therefore, the statement is clearly true in this case.

Otherwise, let $t \in I_i$ be the first time of choice at which he does an avoidance move during I_i. If he is at distance at most ρ' from g at time t, the statement follows from Claim 12. Therefore, assume that the distance is more than ρ'. Then, we must have that $t < (i+1)\Delta_{n-1} - 2\tau$, since, if t was larger, he would already have reached g by the above discussion. Hence, Claim 11 gives that at some time $t' \leq t + \tau$, either

1. the man gets within a distance of ρ' from g, or
2. he does an escape move.

We first prove that in the interval $[t, t']$, the distance from the man to the segment $M_{n-1}(I_i)$ is at most $\rho' + 2(1+\varepsilon_n)\tau$. To this end, note that his distance to the segment at time t is at most $\rho' + (1+\varepsilon_n)\tau$. Thus, since $t' \leq t + \tau$, his distance at time t' can be at most $\rho' + 2(1+\varepsilon_n)\tau$.

It remains to be proven that the man stays within distance $\rho' + 2(1+\varepsilon_n)\tau$ from $M_{n-1}(I_i)$ after time t' and that he is at distance at most $\rho' + (1+\varepsilon_n)\tau$ from g at time $(i+1)\Delta_{n-1}$. If we are in case 1, the statement follows from Claim 12, so assume case 2.

By Claim 10, the man will not do an avoidance move again after time t' until he reaches g or I_i ends. While he is running directly towards g, his distance to the segment $M_{n-1}(I_i)$ is decreasing, so it follows that the distance is always at most $\rho' + 2(1+\varepsilon_n)\tau$, as claimed. Since he was doing avoidance moves in a period of length at most τ before the escape move at time t', he can completely compensate for the delay caused by the avoidance moves in the same amount of time by running directly towards g. The total delay is therefore at most 2τ. Since he would reach g at time $(i+1)\Delta_{n-1} - 2\tau$ at the latest if he did not do any avoidance moves, it follows that he reaches g at time $(i+1)\Delta_{n-1}$ or earlier. The statement then follows from Claim 12. ◀

We are now ready to finish our proof of Theorem 8. Claim 13 implies that during interval I_i for any i, the distance from the man to any of the lions $l_1, \ldots, l_{n-1}$ is at most

$$\rho' + 2(1+\varepsilon_n)\tau < \rho + 2r + 2(1+\varepsilon_n)\tau \leq c_{n-1}/2.$$

Thus, these lions will not catch the man. By Claim 9, the distance to l_n is always at least $r - (3+\varepsilon_n)\Delta_n$. Therefore, we now define $c_n := \min\{c_{n-1}/2, r - (3+\varepsilon_n)\Delta_n\}$, and it holds that in the time interval $I := [i\Delta_n, (i+1)\Delta_n]$, the distance from any point on the segment $m(I)$ to any lion is at least c_n.

Claim 13 implies that at any time t and for any $i \in \{2, \ldots, n\}$, the distance $\|M_{i-1}(t)M_i(t)\|$ is bounded by some constant. It follows that $\|M_1(t)M_n(t)\| \leq d_n$ for some constant $d_n > 0$. Since $M_1(t)$ traverses a ray with constant speed $1 + \varepsilon/2 > 1$, the man eventually escapes the convex hull of the lions and that the distance diverges to ∞ as $t \to \infty$. This proves the theorem. ◀

References

1 M. Abrahamsen, J. Holm, E. Rotenberg, and C. Wulff-Nilsen. Best laid plans of lions and men. *CoRR*, abs/1703.03687, 2017. URL: `https://arxiv.org/abs/1703.03687`.

2 Martin Aigner and Michael Fromme. A game of cops and robbers. *Discrete Applied Mathematics*, 8(1):1–12, 1984.

3 Noga Alon and Abbas Mehrabian. Chasing a fast robber on planar graphs and random graphs. *Journal of Graph Theory*, 78(2):81–96, 2015.

4 Alessandro Berarducci and Benedetto Intrigila. On the cop number of a graph. *Advances in Applied Mathematics*, 14(4):389–403, 1993.

5 B. Bollobás, I. Leader, and M. Walters. Lion and man – can both win? *Israel Journal of Mathematics*, 189(1):267–286, 2012.

6 Béla Bollobás. *The Art of Mathematics: Coffee Time in Memphis*. Cambridge University Press, 2006.

7 Béla Bollobás. The lion and the christian, and other pursuit and evasion games. In Dierk Schleicher and Malte Lackmann, editors, *An Invitation to Mathematics: From Competitions to Research*, pages 181–193. Springer-Verlag Berlin Heidelberg, 2011.

8 Hallard T. Croft. "Lion and man": A postscript. *Journal of the London Mathematical Society*, 39:385–390, 1964.

9 James Flynn. Lion and man: The boundary constraint. *SIAM Journal on Control*, 11:397–411, 1973.

10 James Flynn. Lion and man: The general case. *SIAM Journal on Control*, 12:581–597, 1974.

11 Robbert Fokkink, Leonhard Geupel, and Kensaku Kikuta. Open problems on search games. In Steve Alpern, Robbert Fokkink, Leszek Antoni Gąsieniec, Roy Lindelauf, and V. S.

Subrahmanian, editors, *Search Theory: A Game Theoretic Perspective*, chapter 5, pages 181–193. Springer-Verlag New York, 2013.

12 Fedor V. Fomin, Petr A. Golovach, Jan Kratochvíl, Nicolas Nisse, and Karol Suchan. Pursuing a fast robber on a graph. *Theoretical Computer Science*, 411:1167–1181, 2010.

13 Vladimir Janković. About a man and lions. *Matematički Vesnik*, 2:359–361, 1978.

14 J. Lewin. The lion and man problem revisited. *Journal of Optimization Theory and Applications*, 49(3):411–430, 1986.

15 John Edensor Littlewood. *Littlewood's miscellany: edited by Béla Bollobás.* Cambridge University Press, 1986.

16 Peter A. Rado and Richard Rado. More about lions and other animals. *Mathematical Sprectrum*, 7(3):89–93, 1974/75.

Faster Algorithms for the Geometric Transportation Problem*

Pankaj K. Agarwal[1], Kyle Fox[2], Debmalya Panigrahi[3], Kasturi R. Varadarajan[4], and Allen Xiao[5]

1 **Duke University, Durham, NC, USA**
2 **Duke University, Durham, NC, USA**
3 **Duke University, Durham, NC, USA**
4 **University of Iowa, Iowa City, IA, USA**
5 **Duke University, Durham, NC, USA**

Abstract

Let $R, B \subset \mathbb{R}^d$, for constant d, be two point sets with $|R| + |B| = n$, and let $\lambda : R \cup B \to \mathbb{N}$ such that $\sum_{r\in R} \lambda(r) = \sum_{b\in B} \lambda(b)$ be demand functions over R and B. Let $\mathrm{d}(\cdot,\cdot)$ be a suitable distance function such as the L_p distance. The transportation problem asks to find a map $\tau : R \times B \to \mathbb{N}$ such that $\sum_{b\in B} \tau(r,b) = \lambda(r)$, $\sum_{r\in R} \tau(r,b) = \lambda(b)$, and $\sum_{r\in R, b\in B} \tau(r,b)\mathrm{d}(r,b)$ is minimized. We present three new results for the transportation problem when $\mathrm{d}(\cdot,\cdot)$ is any L_p metric:

- For any constant $\epsilon > 0$, an $O(n^{1+\epsilon})$ expected time randomized algorithm that returns a transportation map with expected cost $O(\log^2(1/\epsilon))$ times the optimal cost.
- For any $\epsilon > 0$, a $(1+\epsilon)$-approximation in $O(n^{3/2}\epsilon^{-d}\,\mathrm{polylog}(U)\,\mathrm{polylog}(n))$ time, where $U = \max_{p\in R\cup B} \lambda(p)$.
- An exact strongly polynomial $O(n^2\,\mathrm{polylog}\,n)$ time algorithm, for $d = 2$.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Transportation map, earth mover's distance, shape matching, approximation algorithms

Digital Object Identifier 10.4230/LIPIcs.SoCG.2017.7

1 Introduction

Let R and B be two point sets in $\mathbb{R}^d$ with $|R| + |B| = n$, where d is a constant, and let $\lambda : R \cup B \to \mathbb{N}$ be a function satisfying $\sum_{r\in R} \lambda(r) = \sum_{b\in B} \lambda(b)$. We denote $U := \max_{p\in R\cup B} \lambda(p)$. We call a function $\tau : R \times B \to \mathbb{N}$, a *transportation map* between R and B if $\sum_{b\in B} \tau(r,b) = \lambda(r)$ for all $r \in R$ and $\sum_{r\in R} \tau(r,b) = \lambda(b)$ for all $b \in B$. Informally, for a point $r \in R$, the value of $\lambda(r)$ represents the *supply* at r, while for a point $b \in B$, the value of $\lambda(b)$ represents the *demand* at b. A transportation map represents a plan for moving the supplies at points in R to meet the demands at points in B.

The cost of a transportation τ is defined as $\mu(\tau) = \sum_{(r,b)\in R\times B} \tau(r,b)\mathrm{d}(r,b)$, where $\mathrm{d}(\cdot,\cdot)$ is a suitable distance function such as the L_p distance. The *Hitchcock-Koopmans transportation problem* (or simply *transportation problem*) on $\Sigma = (R, B, \lambda)$ is to find the

* Work by Agarwal, Fox, and Xiao is supported by NSF under grants CCF-15-13816, CCF-15-46392, and IIS-14-08846, by ARO grant W911NF-15-1-0408, and by grant 2012/229 from the U.S.-Israel Binational Science Foundation. Work by Fox and Panigrahi is supported in part by NSF grants CCF-1527084 and CCF-1535972. Work by Varadarajan is supported by NSF awards CCF-1318996 and CCF-1615845

minimum-cost transportation map for Σ, denoted $\tau^* := \tau^*(\Sigma)$. The cost $\mu(\tau^*)$ is often referred to as the *transportation distance* or *earth mover's distance*.

The transportation problem is a discrete version of the so-called *optimal transport*, or *Monge-Kantarovich*, problem, originally proposed by the French mathematician Gaspard Monge in 1781. This latter problem has been extensively studied in mathematics since the early 20th century. See the book by Villani [27]. In addition to this connection, the (discrete) transportation problem has a wide range of applications, including similarity computation between a pair of images, shapes, and distributions, computing the barycenter of a family of distributions, finding common structures in a set of shapes, fluid mechanics, and partial differential equations. Motivated by these applications, this problem has been studied extensively in many fields including computer vision, computer graphics, machine learning, optimization, and mathematics. See e.g. [20, 14, 12, 23, 13] and references therein for a few examples.

The transportation problem can be formulated as an instance of the uncapacitated min-cost flow problem in a complete bipartite graph, in which edges have no capacity constraints. The min-cost flow problem has been widely studied; see [18] for a detailed review of known results. The uncapacitated min-cost flow problem in a graph with n vertices and m edges can be solved in $O((m + n \log n)n \log n)$ time using Orlin's algorithm [19] or $\tilde{O}(m\sqrt{n} \operatorname{polylog} U)$ time[1] using the algorithm by Lee and Sidford [18].

For transportation in geometric settings, Atkinson and Vaidya [8] adapted the Edmonds-Karp algorithm to exploit geometric properties, and obtained an $\tilde{O}(n^{2.5} \log U)$ time algorithm for any L_p-metric, and $\tilde{O}(n^2)$ for L_1, L_∞-metrics. The Atkinson-Vaidya algorithm was improved using faster data structures for dynamic nearest-neighbor searching, first by [1] and most recently by [17], for a running time of $\tilde{O}(n^2 \log U)$. Sharathkumar and Agarwal [21] designed a $(1 + \epsilon)$-approximation algorithm with a $\tilde{O}((n\sqrt{nU} + U \log U) \log(n/\epsilon))$ running time.

More efficient algorithms are known for estimating the the optimal cost (earth mover's distance) without actually computing the transportation, provided that $U = n^{O(1)}$. Indyk [16] gave an algorithm to find an $O(1)$-approximate estimate in $\tilde{O}(n)$ time with probability at least $1/2$. Cabello *et al.* [9] reduced the problem to min-cost flow using a geometric spanner, obtaining an $(1 + \epsilon)$-approximate estimate in $\tilde{O}(n^2)$ time. Andoni *et al.* [4] give a streaming algorithm that finds a $(1 + \epsilon)$-approximate estimate in $O(n^{1+o_\epsilon(1)})$ time. However, in many applications, one is interested in computing the map itself and not just the transportation distance [13, 12]. This is the problem that we address in this paper.

The special case of the transportation problem where every point has unit demand/supply is called the *geometric bipartite matching* problem. After a sequence of papers [25, 26, 21, 3], a near-linear $\tilde{O}(n)$ time $(1 + \epsilon)$-approximation was found by Agarwal and Sharathkumar [22] for this problem. On the other hand, before our work, no constant-factor approximation in subquadratic time was known for the transportation problem with arbitrary demands and supplies, even for the special case of $U = O(n^2)$.

Our results. We present three new results for the geometric transportation problem, for any L_p-metric.

Our first result (Section 2) is a randomized algorithm that for any $\epsilon > 0$, computes in $O(n^{1+\epsilon})$ expected time a transportation map whose expected cost is $O(\log^2(1/\epsilon))\mu(\tau^*)$. The

[1] We use $\tilde{O}(f(n))$ to denote $O(f(n) \operatorname{polylog}(n))$.

expected cost improves to $O(\log(1/\epsilon))\mu(\tau^*)$ if the spread[2] of $R \cup B$ is $n^{O(1)}$. The overall structure of our algorithm is a simpler version of the matching algorithm by Agarwal and Varadarajan [3], but several new ideas are needed to handle arbitrary demands and supplies.

We note that our algorithm can be extended to spaces with bounded doubling dimension. For example, suppose R, B lie in a subspace of $\mathbb{R}^d$ such that the doubling dimension with respect to $\mathrm{d}(\cdot,\cdot)$ is a constant, $\mathrm{d}(\cdot,\cdot)$ is computable in $O(1)$ time, and the spread of $R \cup B$ is $n^{O(1)}$. Then our algorithm can be adapted so that it computes in $O(n^{1+\epsilon})$ expected time a transportation map whose expected cost is $O(\log(1/\epsilon))\mu(\tau^*)$. Recall that Indyk's algorithm [16] estimates the cost of τ^* within an $O(1)$ factor assuming that $U = n^{O(1)}$. Using our ideas, his algorithm can be extended to arbitrary values of U. In particular, $\mu(\tau^*)$ can be estimated within an $O(1)$ factor in $\tilde{O}(n)$ time.

Our second result (Section 3) is a $(1+\epsilon)$-approximation algorithm to the transportation problem that runs in $\tilde{O}(n^{3/2}\epsilon^{-d}\,\mathrm{polylog}(U))$ time. Using a quad-tree based well-separated pair decomposition (WSPD) [11] of a point set, we construct a graph G with $O(n)$ vertices and $O(n/\epsilon^d)$ edges, and reduce the problem of computing a $(1+\epsilon)$-approximate transportation map to computing the min-cost flow in G. Next, we compute a min-cost flow f^* in G using the Lee-Sidford [18] algorithm. Finally, we recover in $O(n/\epsilon^d)$ time a transportation map $\tau : R \times B \to \mathbb{N}$ from f^* such that $\mu(\tau) \le (1+\epsilon)\mu(\tau^*)$. Our algorithm can be extended to spaces with bounded doubling dimension by using the appropriate WSPD construction [24] for such spaces. In particular, if the doubling dimension is D and the spread of $R \cup B$ is $n^{O(1)}$, then a $(1+\epsilon)$-approximate transportation map can be computed in time $\tilde{O}(n^{3/2}\epsilon^{-O(D)}\,\mathrm{polylog}(U))$.

Our third result is an exact, strongly polynomial $\tilde{O}(n^2)$ time algorithm for $d = 2$, thereby matching (up to poly-logarithmic factors) the best exact algorithm for geometric matching [17]. Our algorithm is an implementation of Orlin's strongly polynomial min-cost flow algorithm [19], an augmenting-paths algorithm with edge contractions. A naive application of Orlin's algorithm has a running time of $\tilde{O}(n^3)$. By exploiting the geometry of the underlying graph, we improve this running time to $\tilde{O}(n^2)$ in the plane.

Proofs of many lemmas, extensions of our approximation algorithms, and the details of our exact algorithm are omitted from this extended abstract due to space constraints. They will appear in the full version of the paper.

2 A Near-Linear Approximation

Let $\Sigma = (R, B, \lambda)$ be an instance of the transportation problem in $\mathbb{R}^d$. We say that Σ has *bounded spread* if the spread of $R \cup B$ is bounded by n^a for some constant $a > 0$. We present a randomized recursive algorithm that, given Σ and a parameter $\epsilon > 0$, returns a transportation map in $O(n^{1+\epsilon})$ expected time whose expected cost is $O(\log(1/\epsilon))\mu(\tau^*)$ if Σ has bounded spread, and $O(\log^2(1/\epsilon))\mu(\tau^*)$ otherwise (recall that τ^* is the optimal map). We assume that n is sufficiently large so that n^ϵ is at least a suitably large constant.

We first give a high-level description of the algorithm without describing how each step is implemented efficiently. Next, we analyze the cost of the transportation map computed by the algorithm. We then discuss an efficient implementation of the algorithm. For simplicity, we describe the algorithm and its analysis for dimension $d = 2$; the algorithm extends to $d > 2$ in a straightforward manner.

[2] The *spread* of a point set S is the ratio of the maximum and the minimum distance between a pair of points in S.

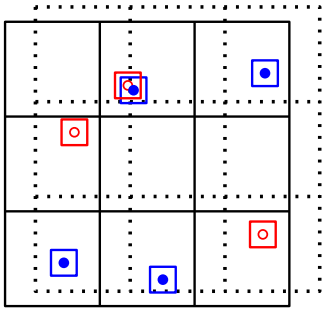


Figure 1 Moats, a safe grid (solid), an unsafe grid (dotted).

We need the notion of *randomly shifted grids*, as in [5, 3]. Formally, let $\square = [a-\ell, a] \times [b-\ell, b]$ be a square of side length ℓ with (a, b) as its top right corner. For a parameter $\Delta > 0$ (grid cell length), set $l = \lceil \log_2 \left(1 + \frac{\ell}{\Delta}\right) \rceil$, and $L = 2^{l+1}\Delta$. Let $\square_L = [a-L, a] \times [b-L, b]$ be the square of side length L with (a, b) as its top-right corner. We choose uniformly at random a point $\xi \in [0, \Delta)^2$ and set $\square_{shifted} := \square_L + \xi$. Note that $\square \subseteq \square_{shifted}$. Let $\mathbb{G}(\square, \Delta)$ be the partition of $\square_{shifted}$ into the uniform grid of side length Δ; $\mathbb{G}(\square, \Delta)$ has $2^{l+1} \times 2^{l+1}$ grid cells. $\mathbb{G}(\square, \Delta)$ is called the randomly shifted grid on $\square$.

2.1 A high-level description

A recursive subproblem $\Sigma = (\mathsf{R}, \mathsf{B}, \lambda)$ consists of point sets R and B, a demand function $\lambda : \mathsf{R} \cup \mathsf{B} \to \mathbb{N}$ such that $\lambda(\mathsf{R}) = \lambda(\mathsf{B})$. We denote $|\mathsf{R} \cup \mathsf{B}| = m$. If $m \leq n^{\epsilon/4}$, we call Σ a *base* subproblem and compute an optimal transportation using Orlin's algorithm. Thus, assume that $m > n^{\epsilon/4}$.

Let $\delta = 1/6$. Also, let $\square$ be the smallest axis-aligned square containing $\mathsf{R} \cup \mathsf{B}$, with its sidelength denoted ℓ. Set $\Delta = \ell/m^{\delta}$. The first step of the algorithm is to choose a randomly shifted grid $\mathbb{G} = \mathbb{G}(\square, \Delta)$ that has the following additional property: any two points in $\mathsf{R} \cup \mathsf{B}$ that are within a distance of ℓ/m^3 lie in the same grid cell. We call a grid $\mathbb{G}$ satisfying this property *safe*. Algorithmically, we place an axis-parallel square of side length $2\ell/m^3$ around every $p \in \mathsf{R} \cup \mathsf{B}$, called the *moat* of p; $\mathbb{G}$ is safe if none of its grid lines cross any moat (see Figure 1). If $\mathbb{G}$ is not safe, we sample a new random shift. It can be verified that $\mathbb{G}$ is safe with probability at least $1 - 1/m$.

Let $\Pi \subseteq \mathbb{G}$ be the set of nonempty grid cells, i.e., ones that contain at least one point of $\mathsf{R} \cup \mathsf{B}$. For each cell $\pi \in \Pi$, we create a recursive instance Σ_π, which we refer to as an *internal subproblem*. Each Σ_π aims to transport as much as possible within π. Whatever we are unable to transport locally within cells of Π, we transport globally with a single *external subproblem* $\Sigma_\square$. We now describe these subproblems in more detail.

For each cell $\pi \in \Pi$, we define the *excess* χ_π of π to be the absolute difference between the red and blue demand in π. Without loss of generality, assume $\lambda(\mathsf{R} \cap \pi) \geq \lambda(\mathsf{B} \cap \pi)$, i.e. that the excess of π is red. Roughly speaking, the entirety of $\mathsf{B} \cap \pi$ is used for the internal subproblem, while $\mathsf{R} \cap \pi$ is arbitrarily partitioned such that $\lambda(\mathsf{B} \cap \pi)$ red demand is used for the internal subproblem, and the remainder (of total demand $= \chi_\pi$) is used for the external subproblem. We pick an arbitrary maximal subset of points $(\mathsf{R}_{ex})_\pi \subseteq \mathsf{R} \cap \pi$ such that $\lambda((\mathsf{R}_{ex})_\pi) \leq \chi_\pi$. Let $\mathsf{R}_\pi = (\mathsf{R} \cap \pi) \setminus (\mathsf{R}_{ex})_\pi$, $\mathsf{B}_\pi = \mathsf{B} \cap \pi$, and $(\mathsf{B}_{ex})_\pi = \emptyset$. If $\lambda((\mathsf{R}_{ex})_\pi) < \chi_\pi$, we arbitrarily pick a point p in R_π, and split p into two copies, say p' and p'', with $\lambda(p') = \chi_\pi - \lambda((\mathsf{R}_{ex})_\pi)$ and $\lambda(p'') = \lambda(p) - \lambda(p')$. We then add p' to $(\mathsf{R}_{ex})_\pi$ and replace p with p'' in R_π. This step ensures that $\lambda((\mathsf{R}_{ex})_\pi) = \chi_\pi$. Let λ_π be the restriction

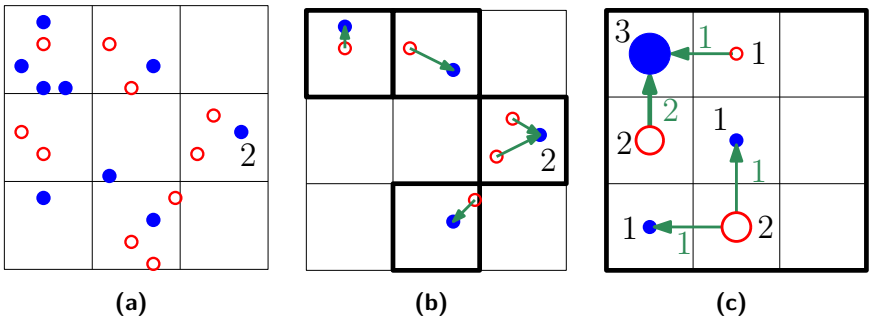


Figure 2 A subproblem (a) with its internal subproblems (b) and external subproblem (c).

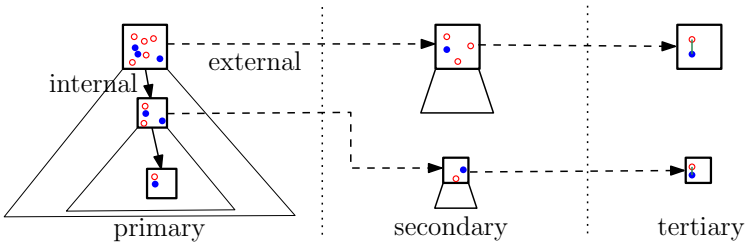


Figure 3 Primary-secondary classification of recursive problems.

of λ to $\mathsf{R}_\pi \cup \mathsf{B}_\pi$; by construction, $\lambda_\pi(\mathsf{R}_\pi) = \lambda_\pi(\mathsf{B}_\pi)$. The internal subproblem for π is $\Sigma_\pi = (\mathsf{R}_\pi, \mathsf{B}_\pi, \lambda_\pi)$.

We now describe the external subproblem. Let $\mathsf{R}_{ex} = \bigcup_{\pi\in\Pi}(\mathsf{R}_{ex})_\pi$, $\mathsf{B}_{ex} = \bigcup_{\pi\in\Pi}(\mathsf{B}_{ex})_\pi$, and set λ_{ex} as the restriction of λ to $\mathsf{R}_{ex} \cup \mathsf{B}_{ex}$. To solve the excess demand instance $\Sigma_{ex} = (\mathsf{R}_{ex}, \mathsf{B}_{ex}, \lambda_{ex})$, we merge the excess in each cell into a single artificial point at the center of the cell. The resulting transportation instance has relatively few points ($O(m^{2\delta})$) and distorts the "real" distances by an amount proportional to the side length of the cell. If $\lambda(\mathsf{R} \cap \pi) > \lambda(\mathsf{B} \cap \pi)$, we create a red point r_π at the center of π and define the demand of r_π, denoted $\lambda_\square(r_\pi)$, to be χ_π. Similarly, if $\lambda(\mathsf{B} \cap \pi) > \lambda(\mathsf{R} \cap \pi)$, we create a blue point b_π at the center of π with $\lambda_\square(b_\pi) = \chi_\pi$. Let $\mathsf{R}_\square$ (resp., $\mathsf{B}_\square$) be the set of red (resp., blue) points that were created at the centers of cells in Π. We create the external subproblem $\Sigma_\square = (\mathsf{R}_\square, \mathsf{B}_\square, \lambda_\square)$; $\Sigma_\square$ acts as an approximate view of the actual excess instance Σ_{ex}. See Figure 2.

For each cell $\pi \in \Pi$, we recursively compute a transportation map τ_π on the internal subproblem Σ_π. If the root instance – the original input to our transportation problem – has bounded spread, we compute an optimal solution $\tau_\square$ for the external subproblem $\Sigma_\square$ using Orlin's algorithm. If the root instance does not have bounded spread, then we recursively compute an approximately optimal solution $\tau_\square$ for the external subproblem $\Sigma_\square$. Note that irrespective of the spread of the original instance, every external subproblem $\Sigma_\square$ has spread bounded by $O(n^\delta)$, i.e., has bounded spread. We categorize subproblems by the number of external subproblems in the recursive chain leading to them: Σ is *primary* if there are none; *secondary* if there is exactly one; and *tertiary* if there are two. All tertiary problems are solved exactly using Orlin's algorithm, as are base subproblems in the primary and secondary recursion. See Figure 3 for a visualization of the recursion tree of the algorithm.

Finally, we construct a transportation map τ for Σ by combining the solutions to the internal and external subproblems. For a pair $(r, b) \in \mathsf{R}_\pi \times \mathsf{B}_\pi$, we simply set $\tau(r, b) = \tau_\pi(r, b)$. For the external subproblem, we first convert the transportation map $\tau_\square$ on $\Sigma_\square$ into a map

for Σ_{ex}, as follows: For each red point $r_\pi \in \mathsf{R}_\square$ (resp., blue point in $\mathsf{B}_\square$), at the center of a cell $\pi \in \Pi$, we "redistribute" the transport from r_π (resp., b_π) to the points of $(\mathsf{R}_{ex})_\pi$ (resp., $(\mathsf{B}_{ex})_\pi$) to compute a transportation map τ_{ex} of Σ_{ex}. That is, for any $r_\pi, b_\pi \in \mathsf{R}_\square \times \mathsf{B}_\square$, we assign the units of $\tau_\square(r_\pi, b_\pi)$ among the pairs in $(\mathsf{R}_{ex})_\pi \times (\mathsf{B}_{ex})_\pi$ in an arbitrary manner, while respecting the demands. We then set $\tau(r, b) = \tau_{ex}(r, b)$ for $(r, b) \in \mathsf{R}_{ex} \times \mathsf{B}_{ex}$. This completes the description of the algorithm.

2.2 Cost analysis

There are two sources of error in our algorithm: the distortion between $\Sigma_\square$ and Σ_{ex}, and the error from restricting the solution to the internal/external partitioning of demand.

δ-closeness. We first formalize the way that $\Sigma_\square$ approximates Σ_{ex} when it shifts demand to cell centers. We introduce a notion called δ-closeness between transportation instances: informally, two instances are δ-close if we can shift the demands of one to form the other, without moving any demand more than δ. We give a formal definition next.

Let $\Sigma = (R, B, \lambda)$ be an instance of the transportation problem and τ a transportation map for Σ. We define $\mu_\infty(\tau) = \max_{(r,b):\tau(r,b)>0} \mathrm{d}(r, b)$ as the maximum distance used in τ. Let $\Sigma' = (R', B', \lambda')$ be another instance of the transportation problem with $\lambda(R) = \lambda'(R')$. Consider the transportation instances $\Sigma_R = (R, R', \lambda_R)$ and $\Sigma_B = (B, B', \lambda_B)$ where λ_R (resp., λ_B) is the demand of points in R and R' (resp., B and B') in Σ and Σ' respectively. We call Σ and Σ' *δ-close* if there exist transportation maps τ_R and τ_B of Σ_R and Σ_B such that $\mu_\infty(\tau_R), \mu_\infty(\tau_B) \leq \delta$, i.e., demands of R (resp., B) (and therefore the units of τ) points of R' (resp., B') within distance δ. We then say that the resulting transportation τ' in Σ' is a map *derived* from τ in Σ. The next lemma follows immediately from definition of δ-closeness.

▶ **Lemma 1.** *Let Σ and Σ' be two δ-close instances of the transportation problem with U being the total demand of each. Let τ be a transportation map of Σ and let τ' be a transportation map of Σ' derived from τ. Then, $|\mu(\tau') - \mu(\tau)| \leq 2\delta U$.*

▶ **Observation 2.** *Any point in a grid cell of side length Δ is within $(\Delta/\sqrt{2})$ of the center; so $\Sigma_\square$ and Σ_{ex} are $(\Delta/\sqrt{2})$-close.*

Thus, the lemma relates the transportation map for Σ_{ex} to the solution for $\Sigma_\square$ produced by the external subproblem.

Partitioning of demand. Now that we have quantified the error between $\Sigma_\square$ and Σ_{ex}, we begin our analysis of the second source of error. First, we bound the error in a single subproblem (i.e. one subdividing grid), and then the error for a single pair $(r, b) \in R \times B$ across all subproblems. Eventually, we combine these two arguments to bound the expected error due to the partitioning across all subproblems and all pairs of points.

Fix a recursive problem $\Sigma = (R, B, \lambda)$, with cell side length Δ. Let χ_π, R_π, B_π, $(R_{ex})_\pi$, $(B_{ex})_\pi$ for $\pi \in \Pi$ and $R_\square$, $B_\square$, R_{ex}, B_{ex}, λ_{ex}, be as defined in Section 2.1 for the instance $\Sigma = (R, B, \lambda)$. Let $\mathcal{I} = \bigcup_{\pi\in\Pi} R_\pi \times B_\pi$ be the set of "local" point pairs, solved by the algorithm within internal subproblems. We refer to a pair $(r, b) \in (R \times B) \setminus \mathcal{I}$ as "non-local".

The next lemma outlines a method for deforming an arbitrary transportation map to one that respects the local/non-local partitioning used by the algorithm.

▶ **Lemma 3.** *Let $\Sigma = (R, B, \lambda)$ be a recursive subproblem with cell side length Δ, and let $\hat{\tau}$ be an arbitrary transportation map for Σ. Let $\hat{\mathsf{X}} = \sum_{(r,b)\notin\mathcal{I}} \hat{\tau}(r, b)$ be the total non-local*

transport in $\hat{\tau}$, and $\mathsf{X} = \sum_{\pi \in \Pi} \chi_\pi$ be the total excess of Σ. Then, there exists a transportation map $\tilde{\tau}$ comprising local solutions $\tilde{\tau}_\pi$ for each Σ_π and a non-local solution $\tilde{\tau}_{ex}$ for Σ_{ex}, such that the following properties hold:

(A) *The cost of the transportation map $\tilde{\tau}$ is given by*

$$\mu(\tilde{\tau}) = \sum_{\pi \in \Pi} \mu(\tilde{\tau}_\pi) + \mu(\tilde{\tau}_{ex}) \leq \mu(\hat{\tau}) + 8\sqrt{2}\Delta\hat{\mathsf{X}}.$$

(B) *The local transport in $\tilde{\tau}$ satisfies $\tilde{\tau}(r,b) \geq \hat{\tau}(r,b)$ for all $(r,b) \in \mathcal{I}$, and the difference $\sum_{(r,b)\in\mathcal{I}} (\tilde{\tau}(r,b) - \hat{\tau}(r,b)) \leq 3\hat{\mathsf{X}}$.*

(C) *The non-local transport in $\tilde{\tau}$ satisfies $\hat{\mathsf{X}} \geq \mathsf{X} = \tilde{\mathsf{X}} := \sum_{(r,b)\notin\mathcal{I}} .\tilde{\tau}(r,b)$.*

Error parameter η. In the previous lemma, we bounded the error due to a single subproblem. We now bound the error due to a single pair of points $(r,b) \in R \times B$, using a random variable $\eta(r,b)$, defined as the cell side length of the first recursive grid to split (r,b) into different cells.

Formally, recall that a recursive subproblem may split a point $p \in R \cup B$ into two copies p' and p'' with $\lambda(p') + \lambda(p'') = \lambda(p)$; one of them passed to the external subproblem, and the other passed down to an internal subproblem. Abusing notation slightly, we use R and B to denote the multisets that contain all copies of points that are split along with the updated demands. Hence, for any base subproblem (R_i, B_i, λ_i) (i.e., the recursive base case) every point $p \in R_i \cup B_i$ can be identified with a point $p \in R \cup B$ such that $\lambda_i(p) = \lambda(p)$. With this interpretation, we define a function $\eta : R \times B \to \mathbb{R}_{\geq 0}$ as follows: If there is a base subproblem (R_i, B_i, λ_i) such that $(r,b) \in R_i \times B_i$, we set $\eta(r,b) = 0$. Otherwise, there is a recursive subproblem such that $(r,b) \in \mathsf{R} \times \mathsf{B}$, and r and b are split into different cells of the randomly shifted grid. In this case, $\eta(r,b)$ denotes the side length of the grid cells, i.e. $\eta(r,b) = \ell/m^\delta$ where ℓ is the length of the smallest square containing $\mathsf{R} \cup \mathsf{B}$, and $m = |\mathsf{R} \cup \mathsf{B}|$.

The next lemma bounds the expected value of the error parameter $\eta(r,b)$ in terms of the distance $\mathrm{d}(r,b)$ for any pair of points $(r,b) \in R \times B$. Its proof uses our choice of safe grids to argue that, though the recursion depth can be large, the number of recursive subproblems that can potentially split (r,b) is small.

▶ **Lemma 4.** *There exists a constant $c_1 > 0$ such that for any $(r,b) \in R \times B$, the expectation $\mathsf{E}[\eta(r,b)] \leq c_1 \log_2(1/\epsilon)\mathrm{d}(r,b)$.*

Proof. Let n be the number of points in the input instance, and m be the number of points in the subproblem which splits (r,b), and ℓ the side length of the smallest orthogonal bounding square of that subproblem (i.e. $\eta(r,b) = \ell/m^\delta$). We can assume that $m > n^{\epsilon/4}$, since otherwise the subproblem is a base case and $\eta(r,b) = 0$.

Define $\underline{\ell} := \mathrm{d}(r,b)/\sqrt{2}$, clearly, $\ell \geq \underline{\ell}$. We partition the interval $[n^{\epsilon/4}, n]$ into $u = \lceil \log_2(4/\epsilon) \rceil$ intervals of the form $[n_j, n_j^2]$, where $n_j = n^{2^{-j}}$, for $1 \leq j \leq u$. There exists an index j^* where $m \in [n_{j^*}, n_{j^*}^2]$, and $\ell \leq \overline{\ell} := (n_{j^*}^2)^3 \mathrm{d}(r,b)$ because the grid is safe. Thus, (r,b) must be split by a grid with ℓ in the interval $[\underline{\ell}, \overline{\ell}]$, where $\overline{\ell}/\underline{\ell} = \sqrt{2}n_{j^*}^6$.

The interval $[\underline{\ell}, \overline{\ell}]$ can be covered by $7/\delta > (1/\delta)(6 + \log_2(\sqrt{2})/\log(n_{j^*}))$ intervals of the form $J_i = [n_{j^*}^{i\delta}\underline{\ell}, n_{j^*}^{(i+1)\delta}\underline{\ell}]$ for $0 \leq i \leq 7/\delta = O(1)$. For each value of i, algorithm produces at most one subproblem – whose bounding square contains both r, b – with side length in J_i. The total set of intervals, by enumeration, bounds the number of subproblems whose grids could possibly split (r,b) as $O(\log(1/\epsilon))$.

For our subproblem with m points and side length ℓ, the probability that a (safe) randomly shifted grid splits (r,b) is no more than $3\mathrm{d}(r,b)/(\ell/m^\delta)$. Recall that the value of $\eta(r,b)$, in

this case, is ℓ/m^δ. Summing over the $O(\log(1/\epsilon))$ subproblems where (r, b) can be split, the expected value of $\eta(r, b)$ is $O(\log(1/\epsilon))\mathrm{d}(r, b)$. ◀

Expected cost of algorithm. We are now ready to analyze the expected cost of τ, the algorithm's transportation map. First, we analyze the cost if Σ has bounded spread. Recall that, in this case, our algorithm computes an optimal solution for each external subproblem (using Orlin's algorithm).

▶ **Lemma 5.** *If Σ is a transportation instance with bounded spread, then there exists a constant $c_2 > 0$ such that for any transportation map $\hat{\tau}$ of Σ,*

$$\mu(\tau) \leq \mu(\hat{\tau}) + c_2 \sum_{(r,b)\in R\times B} \hat{\tau}(r, b)\eta(r, b).$$

An immediate corollary of Lemmas 4 and 5 is:

▶ **Corollary 6.** *If Σ has bounded spread, then $\mathsf{E}[\mu(\tau)] = O(\log(1/\epsilon))\mu(\tau^*)$.*

Proof of Lemma 5. We prove the lemma by induction on the number of points in the subproblem. If Σ is a base problem, then τ is an optimal transport of Σ and the lemma holds. Otherwise $\square$, the smallest square containing $R \cup B$, is split into a set of grid cells. Following the notation in Section 2.1, let Π be the set of non-empty cells and Δ the side length of each grid cell.

Recall that τ is the combination of solutions τ_π for the internal subproblems $\Sigma_\pi = (R_\pi, B_\pi, \lambda_\pi)$ of $\pi \in \Pi$, and the map τ_{ex} for $\Sigma_{ex} = (R_{ex}, B_{ex}, \lambda_{ex})$ derived from the solution $\tau_\square$ to the external subproblem $\Sigma_\square = (R_\square, B_\square, \lambda_\square)$. From Observation 2, $\Sigma_\square$ and Σ_{ex} are $(\Delta/\sqrt{2})$-close; thus Lemma 1 implies $\mu(\tau_\square) \leq \mu(\tau_{ex}) + \sqrt{2}\Delta\mathsf{X}$. We have

$$\mu(\tau) = \mu(\tau_{ex}) + \sum_{\pi\in\Pi} \mu(\tau_\pi) \leq \mu(\tau_\square) + \sqrt{2}\Delta\mathsf{X} + \sum_{\pi\in\Pi} \mu(\tau_\pi). \tag{1}$$

Thus, using (1), we can bound τ by bounding the local (τ_π) and non-local ($\tau_\square$) solutions individually.

Let $\tilde{\tau}$ be the transportation map created by deforming $\hat{\tau}$ in Lemma 3, with $\tilde{\tau}_\pi$ and $\tilde{\tau}_{ex}$ its restrictions to local and non-local pairs of points respectively. To bound $\tau_\square$, notice that that $\tilde{\tau}_{ex}$ solves $\Sigma_{ex} = (R_{ex}, B_{ex}, \lambda_{ex})$, and Σ_{ex} is $(\Delta/\sqrt{2})$-close to $\Sigma_\square$. We apply Lemma 1 and optimality of $\tau_\square$ to conclude,

$$\mu(\tau_\square) \leq \mu(\tilde{\tau}_{ex}) + \sqrt{2}\Delta\mathsf{X}. \tag{2}$$

We now bound the local solutions. Since $\tilde{\tau}_\pi$ is a transportation map of the internal subproblem Σ_π, by the induction hypothesis,

$$\mu(\tau_\pi) \leq \mu(\tilde{\tau}_\pi) + c_2 \sum_{(r,b)\in R_\pi\times B_\pi} \tilde{\tau}_\pi(r, b)\eta(r, b). \tag{3}$$

We can now combine (2) and (3) to bound τ in (1).

$$\begin{aligned}
\mu(\tau) &\leq \mu(\tau_\square) + \sum_{\pi\in\Pi} \mu(\tau_\pi) + \sqrt{2}\Delta\mathsf{X} \\
&\leq \mu(\tilde{\tau}_{ex}) + 2\sqrt{2}\Delta\mathsf{X} + \sum_{\pi\in\Pi}\left[\mu(\tilde{\tau}_\pi) + c_2 \sum_{\pi\in\Pi}\sum_{(r,b)\in R_\pi\times B_\pi} \tilde{\tau}_\pi(r,b)\eta(r,b)\right] && \text{(by (2), (3))} \\
&= \mu(\tilde{\tau}) + c_2 \sum_{(r,b)\in\mathcal{J}} \tilde{\tau}(r,b)\eta(r,b) + 2\sqrt{2}\Delta\mathsf{X} && \text{(Lem. 3(A))} \\
&\leq \mu(\hat{\tau}) + c_2 \sum_{(r,b)\in\mathcal{J}} \tilde{\tau}(r,b)\eta(r,b) + 10\sqrt{2}\Delta\hat{\mathsf{X}} && \text{(Lem. 3(A), 3(C))} \\
&= \mu(\hat{\tau}) + c_2 \sum_{(r,b)\in R\times B} \hat{\tau}(r,b)\eta(r,b) + \Gamma,
\end{aligned}$$

where $\Gamma = c_2 \sum_{(r,b)\in\mathcal{J}} (\tilde{\tau}(r,b) - \hat{\tau}(r,b))\,\eta(r,b) + 10\sqrt{2}\Delta\hat{\mathsf{X}} - c_2 \sum_{(r,b)\notin\mathcal{J}} \hat{\tau}(r,b)\eta(r,b)$.

By definition, $\eta(r,b) = \Delta$ for $(r,b) \notin \mathcal{J}$, $\eta(r,b) \leq \Delta/n^{\epsilon/4} \leq \Delta/4$ for $(r,b) \in \mathcal{J}$, and $\sum_{(r,b)\notin\mathcal{J}} \hat{\tau}(r,b) = \hat{\mathsf{X}}$. Therefore, using Lemma 3(B),

$$\Gamma \leq c_2\frac{\Delta}{4}\cdot 3\hat{\mathsf{X}} + 10\sqrt{2}\Delta\hat{\mathsf{X}} - c_2\Delta\hat{\mathsf{X}} = \left(10\sqrt{2} - \frac{c_2}{4}\right)\Delta\hat{\mathsf{X}} \leq 0.$$

provided that $c_2 \geq 40\sqrt{2}$. Hence, $\mu(\tau) \leq \mu(\hat{\tau}) + c_2 \sum_{(r,b)} \hat{\tau}(r,b)\eta(r,b)$. This completes the proof of the lemma. ◀

The general case. We now analyze the cost of the transportation map for the general case, when the spread of $R \cup B$ is arbitrary.

Recall Figure 3 and the categorization of recursive subproblems as primary, secondary, or tertiary based on the number of external subproblem invocations on its path in the recursion tree of the algorithm. We now introduce two functions $\eta_1, \eta_2 : R \times B \to \mathbb{R}_{\geq 0}$ corresponding to the errors introduced in the primary and secondary recursions. (Note that tertiary subproblems are solved exactly; hence, there is no error introduced in solving a tertiary subproblem.)

The function η_1 corresponds to the primary recursion and is the same as the η defined above, i.e., $\eta_1(r,b) = 0$ if (r,b) belongs to a primary base subproblem, otherwise it is the length of the grid cell at the subproblem which splits r and b. The function $\eta_2(r,b)$ corresponds to the secondary recursion for (r,b). If (r,b) belongs to a primary base problem (i.e. does not appear in any secondary recursion), then we set $\eta_2(r,b) = 0$. Otherwise, let $\bar{r} \in R_\square$ and $\bar{b} \in B_\square$ be the centers of the grid cells of the primary subproblem where r and b were split. Then, $\eta_2(r,b)$ is defined to be $\eta(\bar{r},\bar{b})$ for the secondary recursion on $(R_\square, B_\square)$.

We now state two lemmas that are counterparts of Lemmas 4 and 5 for the general case.

▶ **Lemma 7.** *For any pair $(r,b) \in R \times B$, $\mathsf{E}[\eta_2(r,b)] = O(\log^2(1/\epsilon))\mathrm{d}(r,b)$.*

▶ **Lemma 8.** *There exists a constant $c_3 > 0$ such that for any transportation map $\hat{\tau}$ of Σ,*

$$\mu(\tau) \leq \mu(\hat{\tau}) + c_3 \sum_{(r,b)\in R\times B} \hat{\tau}(r,b)\left[\eta_1(r,b) + \eta_2(r,b)\right].$$

The bound on the expected cost follows directly from the above two lemmas.

▶ **Corollary 9.** $\mathsf{E}[\mu(\tau)] = O(\log^2(1/\epsilon))\mu(\tau^*)$.

2.3 An efficient implementation

We now explain how the various steps of the algorithm are implemented to run in $O(n^{1+\epsilon})$ time. There are three main steps in the algorithm:

(i) partitioning a recursive subproblem $\Sigma = (\mathsf{R}, \mathsf{B}, \lambda)$ into internal subproblems and an external subproblem;

(ii) solving subproblems recursively;

(iii) recovering the transportation map τ of Σ from the internal and external solutions τ_π ($\pi \in \Pi$) and $\tau_\square$.

A recursive subproblem partitions its points into internal subproblems, but generates an additional set of points (at cell centers) for its external subproblem; let us call these *external points*. We bound the number of external points generated in the next lemma.

▶ **Lemma 10.** *The total number of external points over all recursive subproblems is $O(n)$.*

A base subproblem of size $n_i \leq n^{\epsilon/4}$ is solved in $O(n_i^3 \log n_i)$ time using Orlin's algorithm. We distribute this on the n_i points by charging $O(n_i^2 \log n_i) = O(n^{\epsilon/2} \log n)$ to each point. Note every point in $R \cup B$, as well as every external point, belongs to at most one base subproblem. Since, by Lemma 10, the number of external points is $O(n)$, it follows that the total time spent solving base subproblems is $O(n) \cdot O(n^{\epsilon/2} \log n) = O(n^{1+\epsilon})$.

The time spent in recovering the transportation map τ for Σ from its internal and external subproblems is proportional to the number of external points in Σ. Hence, by Lemma 10, step (iii) takes $O(n)$ time.

Finally, implementation of step (i) depends on whether the instance $\Sigma = (R, B, \lambda)$ has bounded spread.

The bounded spread case. In this case, step (i) is implemented naively. We choose a random shift, distribute the points of $\mathsf{R} \cup \mathsf{B}$ among the grid cells in $O(m \log m)$ time, where $|\mathsf{R} \cup \mathsf{B}| = m$, and check in additional $O(m)$ time whether the shift is safe. So step (i) can be implemented in $O(m \log m)$ expected time. We charge $O(\log m)$ time to each point of $\mathsf{R} \cup \mathsf{B}$. Since the spread is $n^{O(1)}$, the depth of recursion is $O(1/\epsilon)$. Therefore, each input point is charged $O(\frac{1}{\epsilon} \log n)$ units of time, implying that steps (i) over all subproblems take $O(n \log n)$ expected time (note that ϵ is a constant). As the size of the external subproblem at $\mathsf{R} \cup \mathsf{B}$ is $O(m^{2\delta})$, and $\delta = 1/6$, the time for solving it exactly is $O(m)$. Putting everything together and applying Corollary 6, we obtain the following.

▶ **Theorem 11.** *Let Σ be an instance of the transportation problem in $\mathbb{R}^d$, where d is a constant. Let Σ have size n and bounded spread, and let $\epsilon > 0$ be a constant. A transportation map of Σ can be computed in $O(n^{1+\epsilon})$ expected time whose expected cost is $O(\log(1/\epsilon))\mu(\tau^*)$, where τ^* is an optimal transport of Σ.*

The general case. Let $\Sigma = (\mathsf{R}, \mathsf{B}, \lambda)$ be a recursive subproblem, and $\square$ the smallest square containing $\mathsf{R} \cup \mathsf{B}$. As defined earlier, let $\Sigma_\square = (\mathsf{R}_\square, \mathsf{B}_\square, \lambda_\square)$ be the external subproblem generated by Σ using the points of $\mathsf{R}_{ex}, \mathsf{B}_{ex}$. Since $\Sigma_\square$ has bounded spread and is solved recursively, the running time to solve $\Sigma_\square$ is $O(|\mathsf{R}_\square \cup \mathsf{B}_\square|^{1+\epsilon}) = O(|\mathsf{R}_{ex} \cup \mathsf{B}_{ex}|^{1+\epsilon})$. Summing over all recursive problems, by Lemma 10, the total time spent in solving all external subproblems is $O(n^{1+\epsilon})$.

Step (i) is more challenging in this case because the depth of recursion can be as large as $\Omega(n)$. We can neither spend linear time at a recursive subproblem, nor can we afford to visit each cell of the randomly shifted grid $\mathbb{G}$ explicitly to compute the set Π of non-empty

cells. To avoid checking all cells of $\mathbb{G}$ explicitly, we (implicitly) construct a quad tree $\mathbb{T}$ on $\mathbb{G}$ – i.e. leaves of $\mathbb{T}$ are cells of $\mathbb{G}$ and the root of $\mathbb{T}$ is $\square$. The depth of $\mathbb{T}$ is $O(\log m)$. The role of $\mathbb{T}$ will be to guide the search for non-empty cells of $\mathbb{G}$. Again, we will not construct $\mathbb{T}$ explicitly.

To avoid spending $\Omega(m)$ time in step (i), we do not maintain the set $\mathsf{R} \cup \mathsf{B}$ explicitly. We build a 2D dynamic orthogonal range searching data structure that maintains a set $X \subset \mathbb{R}^2$ of weighted points, and supports the following operations:

- $\textsc{Wt}(\rho)$: Given a rectangle ρ, return $w(X \cap \rho)$.
- $\textsc{Report}(\rho, \Delta)$: Report a maximal subset Y of $X \cap \rho$ such that $w(Y) \leq \Delta$.
- $\textsc{Empty}(\rho)$: Return Yes if $X \cap \rho = \emptyset$ and No otherwise.
- $\textsc{Delete}(p)$: Delete p from X.
- $\textsc{ReduceWt}(p, \Delta)$: Update $w(p) := w(p) - \Delta$, assuming $w(p) \geq \Delta$.

Using a range-tree based data structure, each operation except for Report can be performed in $O(\log^2 m)$ time [2]. Report requires $O(\log^2 n + k)$ time, where k is the number of reported points.

We maintain two copies $\mathbb{D}_R, \mathbb{D}_B$ of this data structure. The first one is initialized with R and the supplies, and the second with B and the demands. We use $\mathcal{R}$ and $\mathcal{B}$ to denote the current sets in these data structures.

With each recursive subproblem $\Sigma = (\mathsf{R}, \mathsf{B}, \lambda)$ we associate a bounding rectangle ρ that contains $\mathsf{R} \cup \mathsf{B}$. For the root problem, ρ is the smallest square containing $R \cup B$; for others it is defined recursively. We maintain the invariant that when the subproblem $\Sigma = (\mathsf{R}, \mathsf{B}, \lambda)$ is being processed,

(i) $\mathcal{R} \cap \rho = \mathsf{R}$ and for any $r \in \mathcal{R} \cap \rho, w(r) = \lambda(r)$,
(ii) $\mathcal{B} \cap \rho = \mathsf{B}$ and for any $b \in \mathcal{B} \cap \rho, w(b) = \lambda(b)$.

We first compute Π, the set of non-empty cells of $\mathbb{G}$, using $\mathbb{T}$ and the data structures $\mathbb{D}_R, \mathbb{D}_B$. We visit $\mathbb{T}$ in a top-down manner. Suppose we are at a node $v \in \mathbb{T}$, and let $\square_v$ be the square associated with v. We call $\textsc{Empty}(\rho \cap \square_v)$ on both $\mathbb{D}_R$ and $\mathbb{D}_B$ to check whether $(\mathsf{R} \cup \mathsf{B}) \cap \square_v = \emptyset$. If Yes, we ignore the subtree rooted at v. If No and v is a leaf of $\mathbb{T}$, i.e., $\square_v$ is a nonempty cell of $\mathbb{G}$, we add $\square_v$ to Π. If v is an internal node and $(\mathsf{R} \cup \mathsf{B}) \cap \square_v \neq \emptyset$, we recursively search the children of v in $\mathbb{T}$. Since the depth of $\mathbb{T}$ is $O(\log n)$, the above procedure visits $O(|\Pi| \log n)$ nodes of $\mathbb{T}$. The total time spent in computing Π is thus $O(|\Pi| \log^3 n)$.

For each cell $\pi \in \Pi$, we can compute the total demands of $\lambda(\mathsf{R} \cap \pi)$ and $\lambda(\mathsf{B} \cap \pi)$ – and thus χ_π – using the $\textsc{Wt}(\rho \cap \pi)$ operations on $\mathbb{D}_R, \mathbb{D}_B$. Without loss of generality, assume $\lambda(\mathsf{R} \cap \pi) > \lambda(\mathsf{B} \cap \pi)$. Using $\textsc{Report}(\rho \cap \pi, \chi)$, we report a maximal subset of points of $\mathsf{R} \cap \pi$ whose total weight is at most χ. We then delete each of these points (by Delete) and reduce the weight (by Reduce) of one additional point in $\mathsf{R} \cap \pi$ or $\mathsf{B} \cap \pi$ if needed. Let $(R_\pi, B_\pi, \lambda_\pi)$ be the recursive (internal) subproblem generated for π with $\rho_\pi = \rho \cap \pi$ as the associated rectangle. Then the above update operation ensures that $\mathcal{R} \cap \rho_\pi = \mathsf{R}_\pi$, $\mathcal{B} \cap \rho_\pi = \mathsf{B}_\pi$, and their weights are consistent with λ_π.

$\mathbb{D}_R$ and $\mathbb{D}_B$ can also be used to test whether the random shift is safe: For each $\pi \in \Pi$, we check whether the moat of any point in $(\mathsf{R} \cup \mathsf{B}) \cap \pi$ intersects an edge of π. This is equivalent to checking whether $\square_\pi \cap (\mathsf{R} \cup \mathsf{B}) = \emptyset$, where $\square_\pi$ is the set of points that are within distance ℓ/m^3 from the boundary of π. This test can be done in $O(\log^2 n)$ time using the Empty procedure. The total expected time spent in generating internal subproblems Σ_π and external subproblems $\Sigma_\square$ of Σ is $O(|\Pi| \log^3 n + m_\square \log^2 n)$, where $m_\square$ is the total number of external points.

By Lemma 10, the total number of nonempty cells over all subproblems is $O(n)$, and the number of external points is $O(n)$. Thus, the expected time spent in step (i) overall is $O(n \log^3 n)$. Putting everything together, we obtain the main result of this section.

▶ **Theorem 12.** *Let Σ be an instance of the transportation problem in $\mathbb{R}^d$, where d is a constant. Let Σ have size n, and let $\epsilon > 0$ be a constant. A transportation map of Σ can be computed in $O(n^{1+\epsilon})$ expected time whose expected cost is $O(\log^2(1/\epsilon))\mu(\tau^*)$, where τ^* is an optimal transport of Σ.*

3 A $(1+\epsilon)$-Approximate Algorithm

In this section, we describe a $(1+\epsilon)$-approximation algorithm for the transportation problem, based on a reduction to min-cost flow. As in Section 2, we hierarchically cluster points, but this time for the purpose of approximately representing all $\Theta(n^2)$ pairwise distances between R and B compactly. At a high level, our algorithm is as follows:

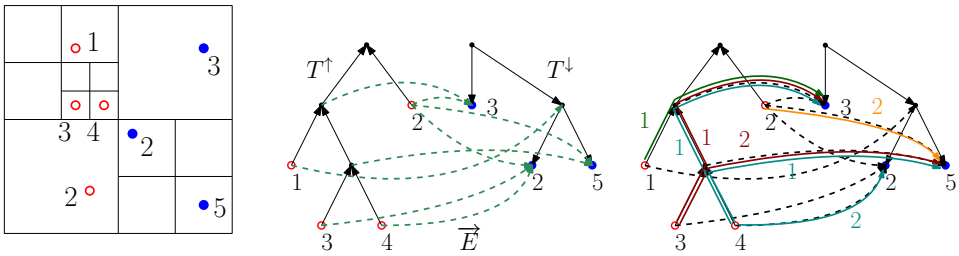
(i) Compute a hierarchical clustering of $R \cup B$, using a quadtree (input points are leaves).
(ii) Construct a sparse directed acyclic graph $G = (V, E)$ over the clusters with R as the set of source nodes, and B as the set of sink nodes with the following property: for every pair $(r, b) \in R \times B$, there is a unique path in G from r to b, with cost roughly $\mathrm{d}(r, b)$. Then, a minimum-cost flow from sources to sinks in G approximates the optimal transportation map on (R, B, λ).
(iii) Compute an optimal flow f^* in G using the algorithm by Lee and Sidford [18].
(iv) Recover a transportation map τ from f^*.

The hierarchical structure is important to us for two reasons: it is the foundation for the compact representation (well-separated pair decomposition), and enables a near-linear time procedure for recovering the transportation map (step 4). This fast recovery step is what distinguishes this algorithm from the similar reduction in Cabello *et al.* [9], and why *geometric spanners* [10, 7, 6], as black boxes, seem to be insufficient.

Construction of the graph. For simplicity, we describe the algorithm in $\mathbb{R}^2$ under Euclidean distance. Let $\square$ be the smallest orthogonal square containing $R \cup B$. We construct a *compressed* quad tree T on $R \cup B$ with $\square$ as the square associated with the root of T. A compressed quadtree prunes certain interior nodes of a standard quadtree, guaranteeing that T has $O(n)$ nodes. We can construct a compressed quadtree in $O(n \log n)$ time, see e.g. [15]. Each node v of T is associated with a square $\square_v$. For each node $v \in T$, let $R_v = R \cap \square_v$ and $B_v = B \cap \square_v$. The sets R_v, B_v form a hierarchical clustering of $R \cup B$.

To construct $G = (V, E)$, we make two copies of T: the *up-tree* $T^\uparrow = (V^\uparrow, E^\uparrow)$ and *down-tree* $T^\downarrow = (V^\downarrow, E^\downarrow)$. We orient the edges of $E^\uparrow$ upward – from a node to its parent, and orient the edges of $E^\downarrow$ downward – from a node to its child. We delete blue points from $T^\uparrow$ and red points from $T^\downarrow$, thus $T^\uparrow$ contains only R, and $T^\downarrow$ contains only B. We set $V = V^\uparrow \cup V^\downarrow$ and $E = E^\uparrow \cup E^\downarrow \cup \overrightarrow{E}$ where $\overrightarrow{E} \subseteq V^\uparrow \times V^\downarrow$ is a set of *cross edges* connecting $T^\uparrow$ to $T^\downarrow$ that we define below. See Figure 4.

Originally proposed by Callahan and Kosaraju [11], the notion of a *well-separated pair decomposition* (WSPD) of a point set S is widely used to represent the pairwise distances of S approximately in a compact manner. A simple WSPD construction using compressed quadtrees is described in Chapter 3 of [15]. Using this algorithm, we construct $\overrightarrow{E}$ as follows: Set $\delta = \frac{\epsilon}{4}$. For a pair of nodes $u, v \in T^\uparrow \times T^\downarrow$, we define $c(u, v) = \min\{\mathrm{d}(x, y) \mid x \in \square_u, y \in \square_v\}$ to be the minimum distance between the squares $\square_u$ and $\square_v$. Using the algorithm in [15], we compute a pair decomposition $D \subseteq V^\uparrow \times V^\downarrow$ with the following properties:



(a) Quadtree on $R \cup B$. **(b)** Composition of G. **(c)** Decomposition of flow.

Figure 4 Construction of G and recovering the transportation.

(W1) For every $(r, b) \in R \times B$, there is a unique pair $(u, v) \in D$ such that $r \in R_u, b \in B_v$.
(W2) For every $(u, v) \in D$, $\max\{\text{diam}(\square_u), \text{diam}(\square_v)\} \leq \delta \cdot c(u, v)$.
(W3) $|D| = O(n/\epsilon^2)$.
After having constructed $T^{\uparrow}$ and $T^{\downarrow}$, that algorithm constructs D in $O(n/\epsilon^2)$ time. We set $\overrightarrow{E} = D$ with each edge oriented from $T^{\uparrow}$ to $T^{\downarrow}$. The cost of each edge in $E^{\uparrow} \cup E^{\downarrow}$ is set to 0, and the cost of each $(u, v) \in \overrightarrow{E}$ is $c(u, v)$. Finally, we define the *excess* $\chi(v)$ at each node v of G: we set $\chi(v) = 0$ for all internal nodes v of $T^{\uparrow} \cup T^{\downarrow}$, $\chi(r) = \lambda(r)$ for $r \in R$, and $\chi(b) = -\lambda(b)$ for $b \in B$. The capacity of all edges in G is set to ∞. Let (G, c, χ) be the resulting min-cost flow instance. The total time spent in constructing G is $O(n \log n + n/\epsilon^2)$.

Cost analysis. Flow moves up from the leaves of $T^{\uparrow}$ through its interior, crosses along the cross edges into $T^{\downarrow}$, and finally descends to the sinks at leaves of $T^{\downarrow}$. By construction and 3, any pair $(r, b) \in R \times B$ has a unique path $p(r, b)$ from r to b in G, using a single cross edge. We can map any transport τ (injectively) to a feasible flow f_τ on G, by placing a flow of $\tau(r, b)$ on $p(r, b)$. Similarly, any flow f can be mapped to a feasible transportation τ_f by *decomposing* f into flow on source-sink paths: by the classical flow decomposition theorem, f can be decomposed into a set $\{f(p(r, b)\}$ of flows on the $p(r, b)$ (since G is a directed acyclic graph, any decomposition has no flow cycles, only paths). Then, setting $\tau_f(r, b) = f(p(r, b))$ for all $(r, b) \in R \times B$ is a feasible transportation.

By 3 and the triangle inequality, $\mu(f_\tau) \leq \mu(\tau)$ and $\mu(\tau_f) \leq (1 + \epsilon)\mu(f)$. We can apply these transformations to bound the approximation quality of a transportation recovered by decomposing the *optimal* flow f^* of G. We have

$$\mu(\tau^*) \leq \mu(\tau_{f^*}) \leq (1 + \epsilon)\mu(f^*) \leq (1 + \epsilon)\mu(f_{\tau^*}) \leq (1 + \epsilon)\mu(\tau^*).$$

If we compute the optimal f^* on G and construct some τ_{f^*} by a flow decomposition of f^*, then τ_{f^*} meets the claimed approximation quality – this is precisely what we do. Note that this cost analysis applies regardless of the specific flow decomposition of f^*. Our recovery procedure in 4 is a greedy decomposition.

Recovering a transportation map. Let f^* be the optimal flow from R to B in G. We use a two-part greedy algorithm to decompose f^*: assigning the flow from R to the cross edges through the up-tree, then claiming the assigned flow using the down-tree. Both steps amount to performing a flow decomposition on the portion of the flow lying in each tree, treating the cross edge endpoints as sinks (resp., sources) with demand equal to the sum of outgoing (resp., incoming) flow. Both are arborescences, so flow decomposition can be done with a

postorder traversal. After solving both trees, we combine the paths assigned into and out of each cross edge to find end-to-end path flows.

We only describe the decomposition for $T^{\uparrow}$ in detail; it is nearly identical for $T^{\downarrow}$. For each cross edge $(u, v) \in \overrightarrow{E}$, this produces lists $A_R(u, v)$ and $A_B(u, v)$ which hold pairs (p, F) indicating *point $p \in R \cup B$ contributes F units of the flow through (u, v)*. The tree is processed in postorder: a node is visited only after all its children. Denote the children of node u by children(u). Each node $u \in T^{\uparrow}$ maintains a list $L(u)$ of the positive-demand red points in its subtree, and a list $N(u)$ of the positive-flow cross edges leaving u. $L(u)$ is initialized by joining lists $L(w)$ from each $w \in$ children(u) (at the leaves, we initialize $L(r) = \{r\}$ for $r \in R$). While $N(u)$ is not empty, let $(u, v) \in N(u)$ with flow $f^*(u, v) > 0$. Take any point $r \in L(u)$ and add to $A_R(u, v)$ a pair (r, F), with $F = \min\{\lambda(r), f^*(u, v)\}$, also updating $\lambda(r) \leftarrow \lambda(r) - F$ and $f^*(u, v) \leftarrow f^*(u, v) - F$. This has the effect of removing either r from $L(u)$, (u, v) from $N(u)$, or both. Once $N(u) = \emptyset$, all cross edges leaving u have their flow assigned.

Finally, we complete the decomposition using $A_R(u, v)$ and $A_B(u, v)$, for each cross edge (u, v). While both $A_R(u, v)$ and $A_B(u, v)$ are nonempty, let $(r, F_r) \in A_R(u, v)$ and $(b, F_b) \in A_B(u, v)$. Output $\tau(r, b) := \min\{F_r, F_b\}$, update $F_r \leftarrow F_r - \tau(r, b)$ and $F_b \leftarrow F_b - \tau(r, b)$, and remove from the lists any pair (p, F) for which $F = 0$.

We charge the list union which constructs $L(u)$ to the children of u. Each iteration performs a constant number of list operations and either removes a node from $L(u)$, or a cross edge from $N(u)$. Each removal occurs exactly once for every $r \in R$ and $(u, v) \in \overrightarrow{E}$, so we charge iterations to the $r \in R$ or $(u, v) \in \overrightarrow{E}$ removed that iteration. The processing of $A_R(u, v)$ and $A_B(u, v)$ can also be charged per iteration to the pair (p, F) removed in that iteration, which is then charged back to the $p \in R \cup B$ or $(u, v) \in \overrightarrow{E}$ whose removal introduced (p, F). Thus, the total running time is $O(|V| + |\overrightarrow{E}|) = O(n/\epsilon^2)$.

We computed an optimal flow $f^* : E \to \mathbb{N}$ using the algorithm by Lee and Sidford [18]. Since $|E| = O(n/\epsilon^2)$, their algorithm takes $\tilde{O}(n^{3/2}\epsilon^{-2} \operatorname{polylog} U)$ time; recall, $U = \max_{p \in R \cup B} \lambda(p)$ is the maximum demand. This step dominates the running time compared to the construction of G and the recovery algorithm. We state the main theorem in terms of an arbitrary d.

▶ **Theorem 13.** *Let Σ be an instance of the transportation problem in $\mathbb{R}^d$ where d is a constant. Let Σ have size n, and let $\epsilon > 0$ be a constant. A transportation map τ for Σ can be computed in $\tilde{O}(n^{3/2}\epsilon^{-d} \operatorname{polylog} U)$ time with cost $\mu(\tau) \leq (1 + \epsilon)\mu(\tau^*)$.*

References

1. Pankaj K. Agarwal, Alon Efrat, and Micha Sharir. Vertical decomposition of shallow levels in 3-dimensional arrangements and its applications. *SIAM J. Comput.*, 29(3):912–953, 1999. doi:10.1137/S0097539795295936.
2. Pankaj K. Agarwal and Jeff Erickson. Geometric range searching and its relatives. *Contemporary Mathematics*, 223:1–56, 1999.
3. Pankaj K. Agarwal and Kasturi R. Varadarajan. A near-linear constant-factor approximation for Euclidean bipartite matching? In *Proc. of the 20th ACM Symp. on Comp. Geometry*, pages 247–252, 2004. doi:10.1145/997817.997856.
4. Alexandr Andoni, Aleksandar Nikolov, Krzysztof Onak, and Grigory Yaroslavtsev. Parallel algorithms for geometric graph problems. In *Proc. of the 46th Ann. ACM Symp. on Theory of Comp.*, pages 574–583, 2014. doi:10.1145/2591796.2591805.

5 Sanjeev Arora. Polynomial time approximation schemes for Euclidean TSP and other geometric problems. In *Proc. of the 37th Ann. IEEE Symp. on Found. of Comp. Sci.*, pages 2–11, 1996. doi:10.1109/SFCS.1996.548458.

6 Sunil Arya, Gautam Das, David M. Mount, Jeffrey S. Salowe, and Michiel H. M. Smid. Euclidean spanners: short, thin, and lanky. In *Proc. of the 27th Ann. ACM Symp. on Theory of Comp.*, pages 489–498, 1995. doi:10.1145/225058.225191.

7 Sunil Arya, David M. Mount, and Michiel H. M. Smid. Randomized and deterministic algorithms for geometric spanners of small diameter. In *Proc. of the 35th Ann. IEEE Symp. on Found. of Comp. Sci.*, pages 703–712, 1994. doi:10.1109/SFCS.1994.365722.

8 David S. Atkinson and Pravin M. Vaidya. Using geometry to solve the transportation problem in the plane. *Algorithmica*, 13(5):442–461, 1995.

9 Sergio Cabello, Panos Giannopoulos, Christian Knauer, and Günter Rote. Matching point sets with respect to the Earth Mover's Distance. *Comput. Geom.*, 39(2):118–133, 2008. doi:10.1016/j.comgeo.2006.10.001.

10 Paul B. Callahan and S. Rao Kosaraju. Faster algorithms for some geometric graph problems in higher dimensions. In *Proc. of the 4th Ann. ACM/SIGACT-SIAM Symp. on Discrete Algo.*, pages 291–300, 1993. URL: http://dl.acm.org/citation.cfm?id=313559.313777.

11 Paul B. Callahan and S. Rao Kosaraju. A decomposition of multidimensional point sets with applications to k-nearest-neighbors and n-body potential fields. *J. ACM*, 42(1):67–90, 1995. doi:10.1145/200836.200853.

12 Marco Cuturi and Arnaud Doucet. Fast computation of Wasserstein barycenters. In *Proc. of the 31th Internat. Conf. on Machine Learning*, pages 685–693, 2014. URL: http://jmlr.org/proceedings/papers/v32/cuturi14.html.

13 Alexandre Gramfort, Gabriel Peyré, and Marco Cuturi. Fast optimal transport averaging of neuroimaging data. In *Proc. of the 24th Internat. Conf. on Infor. Processing in Medical Imaging*, pages 261–272. Springer, 2015.

14 Kristen Grauman and Trevor Darrell. Fast contour matching using approximate earth mover's distance. In *Proc. of the 24th Ann. IEEE Conf. on Comp. Vision and Pattern Recog.*, volume 1, pages I–220. IEEE, 2004.

15 Sariel Har-Peled. *Geometric Approximation Algorithms*, volume 173. American Mathematical Society Providence, 2011.

16 Piotr Indyk. A near linear time constant factor approximation for Euclidean bichromatic matching (cost). In *Proc. of the 18th Ann. ACM-SIAM Symp. on Discrete Algo.*, pages 39–42, 2007. URL: http://dl.acm.org/citation.cfm?id=1283383.1283388.

17 Haim Kaplan, Wolfgang Mulzer, Liam Roditty, Paul Seiferth, and Micha Sharir. Dynamic planar Voronoi diagrams for general distance functions and their algorithmic applications. *CoRR*, abs/1604.03654, 2016. URL: http://arxiv.org/abs/1604.03654.

18 Yin Tat Lee and Aaron Sidford. Path finding methods for linear programming: Solving linear programs in $\tilde{O}$(vrank) iterations and faster algorithms for maximum flow. In *Proc. of the 55th Ann. IEEE Symp. on Found. of Comp. Sci.*, pages 424–433, 2014.

19 James B. Orlin. A faster strongly polynominal minimum cost flow algorithm. In *Proc. of the 20th Annual ACM Symp. on Theory of Comp., May 2-4, 1988, Chicago, Illinois, USA*, pages 377–387, 1988. doi:10.1145/62212.62249.

20 Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas. A metric for distributions with applications to image databases. In *6th Internat. Conf. on Comp. Vision*, pages 59–66, 1998. doi:10.1109/ICCV.1998.710701.

21 R. Sharathkumar and Pankaj K. Agarwal. Algorithms for the transportation problem in geometric settings. In *Proc. of the 23rd Ann. ACM-SIAM Symp. on Discrete Algo.*,

pages 306–317, 2012. URL: http://portal.acm.org/citation.cfm?id=2095145&CFID=63838676&CFTOKEN=79617016.

22 R. Sharathkumar and Pankaj K. Agarwal. A near-linear time ε-approximation algorithm for geometric bipartite matching. In *Proc. of the 44th Ann. ACM Symp. on Theory of Comp.*, pages 385–394, 2012. doi:10.1145/2213977.2214014.

23 Justin Solomon, Raif M. Rustamov, Leonidas J. Guibas, and Adrian Butscher. Earth mover's distances on discrete surfaces. *ACM Transactions on Graphics*, 33(4):67:1–67:12, 2014. doi:10.1145/2601097.2601175.

24 Kunal Talwar. Bypassing the embedding: Algorithms for low dimensional metrics. In *Proc. of the 36th Ann. ACM Symp. on Theory of Comp.*, pages 281–290, 2004. doi:10.1145/1007352.1007399.

25 Pravin M. Vaidya. Geometry helps in matching. *SIAM J. Comput.*, 18(6):1201–1225, 1989.

26 Kasturi R. Varadarajan and Pankaj K. Agarwal. Approximation algorithms for bipartite and non-bipartite matching in the plane. In *Proc. of the 10th Ann. ACM-SIAM Symp. on Discrete Algo.*, pages 805–814, 1999. URL: http://dl.acm.org/citation.cfm?id=314500.314918.

27 Cédric Villani. *Optimal Transport: Old and New*, volume 338. Springer Science & Business Media, 2008.

A Superlinear Lower Bound on the Number of 5-Holes*

Oswin Aichholzer[1], Martin Balko[2], Thomas Hackl[3], Jan Kynčl[4], Irene Parada[5], Manfred Scheucher[6], Pavel Valtr[7], and Birgit Vogtenhuber[8]

1 **Institute for Software Technology, Graz University of Technology, Austria**
oaich@ist.tugraz.at
2 **Dept. of Applied Mathematics and Institute for Theoretical Computer Science, Faculty of Mathematics and Physics, Charles University, Czech Republic; and Alfréd Rényi Institute of Mathematics, Hungarian Academy of Sciences, Budapest, Hungary**
balko@kam.mff.cuni.cz
3 **Institute for Software Technology, Graz University of Technology, Austria**
thackl@ist.tugraz.at
4 **Dept. of Applied Mathematics and Institute for Theoretical Computer Science, Faculty of Mathematics and Physics, Charles University, Czech Republic**
kyncl@kam.mff.cuni.cz
5 **Institute for Software Technology, Graz University of Technology, Austria**
iparada@ist.tugraz.at
6 **Institute for Software Technology, Graz University of Technology, Austria; and Alfréd Rényi Institute of Mathematics, Hungarian Academy of Sciences, Budapest, Hungary**
mscheuch@ist.tugraz.at
7 **Dept. of Applied Mathematics and Institute for Theoretical Computer Science, Faculty of Mathematics and Physics, Charles University, Czech Republic; and Alfréd Rényi Institute of Mathematics, Hungarian Academy of Sciences, Budapest, Hungary**
valtr@kam.mff.cuni.cz
8 **Institute for Software Technology, Graz University of Technology, Austria**
bvogt@ist.tugraz.at

Abstract

Let P be a finite set of points in the plane in *general position*, that is, no three points of P are on a common line. We say that a set H of five points from P is a 5-*hole in* P if H is the vertex set of a convex 5-gon containing no other points of P. For a positive integer n, let $h_5(n)$ be the minimum number of 5-holes among all sets of n points in the plane in general position.

Despite many efforts in the last 30 years, the best known asymptotic lower and upper bounds for $h_5(n)$ have been of order $\Omega(n)$ and $O(n^2)$, respectively. We show that $h_5(n) = \Omega(n \log^{4/5} n)$, obtaining the first superlinear lower bound on $h_5(n)$.

* The research for this article was partially carried out in the course of the bilateral research project "Erdős–Szekeres type questions for point sets" between Graz and Prague, supported by the OEAD project CZ 18/2015 and project no. 7AMB15A T023 of the Ministry of Education of the Czech Republic. Aichholzer, Scheucher, and Vogtenhuber were partially supported by the ESF EUROCORES programme EuroGIGA – CRP ComPoSe, Austrian Science Fund (FWF): I648-N18. Parada was supported by the Austrian Science Fund (FWF): W1230. Balko and Valtr were partially supported by the grant GAUK 690214. Balko, Kynčl, and Valtr were partially supported by the project CE-ITI no. P202/12/G061 of the Czech Science Foundation (GAČR). Hackl and Scheucher were partially supported by the Austrian Science Fund (FWF): P23629-N18. Balko, Scheucher, and Valtr were partially supported by the ERC Advanced Research Grant no. 267165 (DISCONV).

33rd International Symposium on Computational Geometry (SoCG 2017).
Editors: Boris Aronov and Matthew J. Katz; Article No. 8; pp. 8:1–8:16

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

The following structural result, which might be of independent interest, is a crucial step in the proof of this lower bound. If a finite set P of points in the plane in general position is partitioned by a line ℓ into two subsets, each of size at least 5 and not in convex position, then ℓ intersects the convex hull of some 5-hole in P. The proof of this result is computer-assisted.

1998 ACM Subject Classification G.2.1 Combinatorics

Keywords and phrases Erdős–Szekeres type problem, k-hole, empty k-gon, empty pentagon, planar point set

Digital Object Identifier 10.4230/LIPIcs.SoCG.2017.8

1 Introduction

We say that a set of points in the plane is in *general position* if it contains no three points on a common line. A point set is in *convex position* if it is the vertex set of a convex polygon. In 1935, Erdős and Szekeres [16] proved the following theorem, which is a classical result both in combinatorial geometry and Ramsey theory.

▶ **Theorem** ([16], The Erdős–Szekeres Theorem)**.** *For every integer $k \geq 3$, there is a smallest integer $n = n(k)$ such that every set of at least n points in general position in the plane contains k points in convex position.*

The Erdős–Szekeres Theorem motivated a lot of further research, including numerous modifications and extensions of the theorem. Here we mention only results closely related to the main topic of our paper.

Let P be a finite set of points in general position in the plane. We say that a set H of k points from P is a *k-hole in P* if H is the vertex set of a convex k-gon containing no other points of P. In the 1970s, Erdős [15] asked whether, for every positive integer k, there is a k-hole in every sufficiently large finite point set in general position in the plane. Harborth [21] proved that there is a 5-hole in every set of 10 points in general position in the plane and gave a construction of 9 points in general position with no 5-hole. After unsuccessful attempts of researchers to answer Erdős' question affirmatively for any fixed integer $k \geq 6$, Horton [22] constructed, for every positive integer n, a set of n points in general position in the plane with no 7-hole. His construction was later generalized to so-called *Horton sets* and *squared Horton sets* [29] and to higher dimensions [30]. The question whether there is a 6-hole in every sufficiently large finite planar point set remained open until 2007 when Gerken [19] and Nicolás [23] independently gave an affirmative answer.

For positive integers n and k, let $h_k(n)$ be the minimum number of k-holes in a set of n points in general position in the plane. Due to Horton's construction, $h_k(n) = 0$ for every n and every $k \geq 7$. Asymptotically tight estimates for the functions $h_3(n)$ and $h_4(n)$ are known. The best known lower bounds are due to Aichholzer et al. [5] who showed that $h_3(n) \geq n^2 - \frac{32n}{7} + \frac{22}{7}$ and $h_4(n) \geq \frac{n^2}{2} - \frac{9n}{4} - o(n)$. The best known upper bounds $h_3(n) \leq 1.6196n^2 + o(n^2)$ and $h_4(n) \leq 1.9397n^2 + o(n^2)$ are due to Bárány and Valtr [12].

For $h_5(n)$ and $h_6(n)$, no matching bounds are known. So far, the best known asymptotic upper bounds on $h_5(n)$ and $h_6(n)$ were obtained by Bárány and Valtr [12] and give $h_5(n) \leq 1.0207n^2 + o(n^2)$ and $h_6(n) \leq 0.2006n^2 + o(n^2)$. For the lower bound on $h_6(n)$, Valtr [31] showed $h_6(n) \geq n/229 - 4$.

In this paper we give a new lower bound on $h_5(n)$. It is widely conjectured that $h_5(n)$ grows quadratically in n, but to this date only lower bounds on $h_5(n)$ that are linear in

n have been known. As noted by Bárány and Füredi [10], a linear lower bound of $\lfloor n/10 \rfloor$ follows directly from Harborth's result [21]. Bárány and Károlyi [11] improved this bound to $h_5(n) \geq n/6 - O(1)$. In 1987, Dehnhardt [14] showed $h_5(11) = 2$ and $h_5(12) = 3$, obtaining $h_5(n) \geq 3\lfloor n/12 \rfloor$. However, his result remained unknown to the scientific community until recently. García [18] then presented a proof of the lower bound $h_5(n) \geq 3\lfloor \frac{n-4}{8} \rfloor$ and a slightly better estimate $h_5(n) \geq \lceil 3/7(n-11) \rceil$ was shown by Aichholzer, Hackl, and Vogtenhuber [6]. Quite recently, Valtr [31] obtained $h_5(n) \geq n/2 - O(1)$. This was strengthened by Aichholzer et al. [5] to $h_5(n) \geq 3n/4 - o(n)$. All improvements on the multiplicative constant were achieved by utilizing the values of $h_5(10)$, $h_5(11)$, and $h_5(12)$. In the bachelor's thesis of Scheucher [26] the exact values $h_5(13) = 3$, $h_5(14) = 6$, and $h_5(15) = 9$ were determined and $h_5(16) \in \{10, 11\}$ was shown. During the preparation of this paper, we further determined the value $h_5(16) = 11$; see our webpage [25]. The values $h_5(n)$ for $n \leq 16$ can be used to obtain further improvements on the multiplicative constant. By revising the proofs of [5, Lemma 1] and [5, Theorem 3], one can obtain $h_5(n) \geq n - 10$ and $h_5(n) \geq 3n/2 - o(n)$, respectively. We also note that it was shown in [24] that if $h_3(n) \geq (1+\epsilon)n^2 - o(n^2)$, then $h_5(n) = \Omega(n^2)$.

As our main result, we give the first superlinear lower bound on $h_5(n)$. This solves an open problem, which was explicitely stated, for example, in a book by Brass, Moser, and Pach [13, Chapter 8.4, Problem 5] and in the survey [2].

▶ **Theorem 1.** *There is an absolute constant $c > 0$ such that for every integer $n \geq 10$ we have $h_5(n) \geq cn \log^{4/5} n$.*

Let P be a finite set of points in the plane in general position and let ℓ be a line that contains no point of P. We say that P is *ℓ-divided* if there is at least one point of P in each of the two halfplanes determined by ℓ. For an ℓ-divided set P, we use $P = A \cup B$ to denote the fact that ℓ partitions P into the subsets A and B.

The following result, which might be of independent interest, is a crucial step in the proof of Theorem 1.

▶ **Theorem 2.** *Let $P = A \cup B$ be an ℓ-divided set with $|A|, |B| \geq 5$ and with neither A nor B in convex position. Then there is an ℓ-divided 5-hole in P.*

The proof of Theorem 2 is computer-assisted. We reduce the result to several statements about point sets of size at most 11 and then verify each of these statements by an exhaustive computer search. To verify the computer-aided proofs we have implemented two independent programs, which, in addition, are based on different abstractions of point sets; see Subsection 4.2. Some of our tools originate from the bachelor's theses of Scheucher [26, 27].

In the rest of the paper, we assume that every point set P is planar, finite, and in general position. We also assume, without loss of generality, that all points in P have distinct x-coordinates. We use $\mathrm{conv}(P)$ to denote the convex hull of P and $\partial\,\mathrm{conv}(P)$ to denote the boundary of the convex hull of P.

A subset Q of P that satisfies $P \cap \mathrm{conv}(Q) = Q$ is called an *island of P*. Note that every k-hole in an island Q of P is also a k-hole in P. For any subset R of the plane, if R contains no point of P, then we say that R is *empty of points of P*.

In Section 2 we derive quite easily Theorem 1 from Theorem 2. Then, in Section 3, we give some preliminaries for the proof of Theorem 2, which is presented in Section 4.

2 Proof of Theorem 1

We apply Theorem 2 to obtain a superlinear lower bound on the number of 5-holes in a given set of n points. Without loss of generality, we assume that $n = 2^t$ for some integer $t \geq 5^5$.

We prove by induction on $t \geq 5^5$ that the number of 5-holes in an arbitrary set P of $n = 2^t$ points is at least $f(t) := c \cdot 2^t t^{4/5} = c \cdot n \log_2^{4/5} n$ for some absolute constant $c > 0$. For $t = 5^5$, we have $n > 10$ and, by the result of Harborth [21], there is at least one 5-hole in P. If c is sufficiently small, then $f(t) = c \cdot n \log_2^{4/5} n \leq 1$ and we have at least $f(t)$ 5-holes in P, which constitutes our base case.

For the inductive step we assume that $t > 5^5$. We first partition P with a line ℓ into two sets A and B of size $n/2$ each. Then we further partition A and B into smaller sets using the following well-known lemma, which is, for example, implied by a result of Steiger and Zhao [28, Theorem 1].

▶ **Lemma 3** ([28]). *Let $P' = A' \cup B'$ be an ℓ-divided set and let r be a positive integer such that $r \leq |A'|, |B'|$. Then there is a line that is disjoint from P' and that determines an open halfplane h with $|A' \cap h| = r = |B' \cap h|$.*

We set $r := \lfloor \log_2^{1/5} n \rfloor$, $s := \lfloor n/(2r) \rfloor$, and apply Lemma 3 iteratively in the following way to partition P into islands $P_1, \ldots, P_{s+1}$ of P so that the sizes of $P_i \cap A$ and $P_i \cap B$ are exactly r for every $i \in \{1, \ldots, s\}$. Let $P'_0 := P$. For every $i = 1, \ldots, s$, we consider a line that is disjoint from P'_{i-1} and that determines an open halfplane h with $|P'_{i-1} \cap A \cap h| = r = |P'_{i-1} \cap B \cap h|$. Such a line exists by Lemma 3 applied to the ℓ-divided set P'_{i-1}. We then set $P_i := P'_{i-1} \cap h$, $P'_i := P'_{i-1} \setminus P_i$, and continue with $i + 1$. Finally, we set $P_{s+1} := P'_s$.

For every $i \in \{1, \ldots, s\}$, if one of the sets $P_i \cap A$ and $P_i \cap B$ is in convex position, then there are at least $\binom{r}{5}$ 5-holes in P_i and, since P_i is an island of P, we have at least $\binom{r}{5}$ 5-holes in P. If this is the case for at least $s/2$ islands P_i, then, given that $s = \lfloor n/(2r) \rfloor$ and thus $s/2 \geq \lfloor n/(4r) \rfloor$, we obtain at least $\lfloor n/(4r) \rfloor \binom{r}{5} \geq c \cdot n \log_2^{4/5} n$ 5-holes in P for a sufficiently small $c > 0$.

We thus further assume that for more than $s/2$ islands P_i, neither of the sets $P_i \cap A$ nor $P_i \cap B$ is in convex position. Since $r = \lfloor \log_2^{1/5} n \rfloor \geq 5$, Theorem 2 implies that there is an ℓ-divided 5-hole in each such P_i. Thus there is an ℓ-divided 5-hole in P_i for more than $s/2$ islands P_i. Since each P_i is an island of P and since $s = \lfloor n/(2r) \rfloor$, we have more than $s/2 \geq \lfloor n/(4r) \rfloor$ ℓ-divided 5-holes in P. As $|A| = |B| = n/2 = 2^{t-1}$, there are at least $f(t-1)$ 5-holes in A and at least $f(t-1)$ 5-holes in B by the inductive assumption. Since A and B are separated by the line ℓ, we have at least

$$2f(t-1) + n/(4r) = 2c(n/2)\log_2^{4/5}(n/2) + n/(4r) \geq cn(t-1)^{4/5} + n/(4t^{1/5})$$

5-holes in P. The right side of the above expression is at least $f(t) = cnt^{4/5}$, because the inequality $cn(t-1)^{4/5} + n/(4t^{1/5}) \geq cnt^{4/5}$ is equivalent to the inequality $(t-1)^{4/5}t^{1/5} + 1/(4c) \geq t$, which is true if c is sufficiently small, as $(t-1)^{4/5}t^{1/5} \geq t - 1$. This completes the proof of Theorem 1.

3 Preliminaries

Before proceeding with the proof of Theorem 2, we first introduce some notation and definitions, and state some immediate observations.

Let a, b, c be three distinct points in the plane. We denote the line segment spanned by a and b as ab, the ray starting at a and going through b as $\overrightarrow{ab}$, and the line through a and b

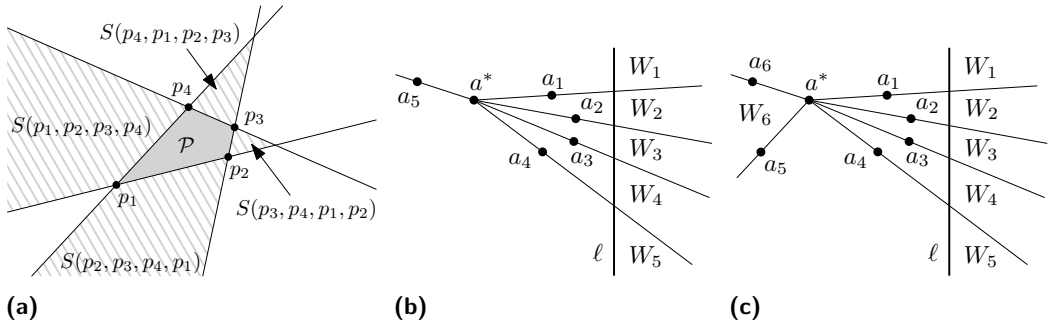


Figure 1 (a) An example of sectors. (b) An example of a^*-wedges with $t = |A| - 1$. (c) An example of a^*-wedges with $t < |A| - 1$.

directed from a to b as $\overline{ab}$. We say c is to the *left* (*right*) of $\overline{ab}$ if the triple (a, b, c) traced in this order is oriented counterclockwise (clockwise). Note that c is to the left of $\overline{ab}$ if and only if c is to the right of $\overline{ba}$, and that the triples (a, b, c), (b, c, a), and (c, a, b) have the same orientation. We say a point set S is to the *left* (*right*) of $\overline{ab}$ if every point of S is to the left (right) of $\overline{ab}$.

Let $P = A \cup B$ be an ℓ-divided set. In the rest of the paper, we assume without loss of generality that ℓ is vertical and directed upwards, A is to the left of ℓ, and B is to the right of ℓ.

Sectors of polygons

For an integer $k \geq 3$, let $\mathcal{P}$ be a convex polygon with vertices $p_1, p_2, \ldots, p_k$ traced counterclockwise in this order. We denote by $S(p_1, p_2, \ldots, p_k)$ the open convex region to the left of each of the three lines $\overline{p_1p_2}$, $\overline{p_1p_k}$, and $\overline{p_{k-1}p_k}$. We call $S(p_1, p_2, \ldots, p_k)$ a *sector* of $\mathcal{P}$. Note that every convex k-gon defines exactly k sectors. Figure 1(a) gives an illustration.

We use $\triangle(p_1, p_2, p_3)$ to denote the closed triangle with vertices p_1, p_2, p_3. We also use $\square(p_1, p_2, p_3, p_4)$ to denote the closed quadrilateral with vertices p_1, p_2, p_3, p_4 traced in the counterclockwise order along the boundary.

The following simple observation summarizes some properties of sectors of polygons.

▶ **Observation 4.** *Let $P = A \cup B$ be an ℓ-divided set with no ℓ-divided 5-hole in P. Then the following conditions are satisfied.*

(i) *Every sector of an ℓ-divided 4-hole in P is empty of points of P.*

(ii) *If S is a sector of a 4-hole in A and S is empty of points of A, then S is empty of points of B.*

ℓ-critical sets and islands

An ℓ-divided set $C = A \cup B$ is called *ℓ-critical* if it fulfills the following two conditions.

(i) Neither A nor B is in convex position.

(ii) For every extremal point x of C, one of the sets $(C \setminus \{x\}) \cap A$ and $(C \setminus \{x\}) \cap B$ is in convex position.

Note that every ℓ-critical set $C = A \cup B$ contains at least four points in each of A and B. If $P = A \cup B$ is an ℓ-divided set with neither A nor B in convex position, then there exists an ℓ-critical island of P. This can be seen by iteratively removing extremal points so that none of the parts is in convex position after the removal.

a-wedges and a^*-wedges

Let $P = A \cup B$ be an ℓ-divided set. For a point a in A, the rays $\overrightarrow{aa'}$ for all $a' \in A \setminus \{a\}$ partition the plane into $|A| - 1$ regions. We call the closures of those regions *a-wedges* and label them as $W_1^{(a)}, \ldots, W_{|A|-1}^{(a)}$ in the clockwise order around a, where $W_1^{(a)}$ is the topmost a-wedge that intersects ℓ. Let $t^{(a)}$ be the number of a-wedges that intersect ℓ. Note that $W_1^{(a)}, \ldots, W_{t^{(a)}}^{(a)}$ are the a-wedges that intersect ℓ sorted in top-to-bottom order on ℓ. Also note that all a-wedges are convex if a is an inner point of A, and that there exists exactly one non-convex a-wedge otherwise. The indices of the a-wedges are considered modulo $|A| - 1$. In particular, $W_0^{(a)} = W_{|A|-1}^{(a)}$ and $W_{|A|}^{(a)} = W_1^{(a)}$.

If A is not in convex position, we denote the rightmost inner point of A as a^* and write $t := t^{(a^*)}$ and $W_k := W_k^{(a^*)}$ for $k = 1, \ldots, |A| - 1$. Recall that a^* is unique, since all points have distinct x-coordinates. Figures 1(b) and 1(c) give an illustration. We set $w_k := |B \cap W_k|$ and label the points of A so that W_k is bounded by the rays $\overrightarrow{a^* a_{k-1}}$ and $\overrightarrow{a^* a_k}$ for $k = 1, \ldots, |A| - 1$. Again, the indices are considered modulo $|A| - 1$. In particular, $a_0 = a_{|A|-1}$ and $a_{|A|} = a_1$.

▶ **Observation 5.** *Let $P = A \cup B$ be an ℓ-divided set with A not in convex position. Then the points $a_1, \ldots, a_{t-1}$ lie to the right of a^* and the points $a_t, \ldots, a_{|A|-1}$ lie to the left of a^*.*

4 Proof of Theorem 2

First, we give a high-level overview of the main ideas of the proof of Theorem 2. We proceed by contradiction and we suppose that there is no ℓ-divided 5-hole in a given ℓ-divided set $P = A \cup B$ with $|A|, |B| \geq 5$ and with neither A nor B in convex position. If $|A|, |B| = 5$, then the statement follows from the result of Harborth [21]. Thus we assume that $|A| \geq 6$ or $|B| \geq 6$. We reduce P to an island Q of P by iteratively removing points from the convex hull until one of the two parts $Q \cap A$ and $Q \cap B$ contains exactly five points or Q is ℓ-critical with $|Q \cap A|, |Q \cap B| \geq 6$. If $|Q \cap A| = 5$ and $|Q \cap B| \geq 6$ or vice versa, then we reduce Q to an island of Q with eleven points and, using a computer-aided result (Lemma 12), we show that there is an ℓ-divided 5-hole in that island and hence in P. If Q is ℓ-critical with $|Q \cap A|, |Q \cap B| \geq 6$, then we show that $|A \cap \partial \operatorname{conv}(Q)|, |B \cap \partial \operatorname{conv}(Q)| \leq 2$ and that, if $|A \cap \partial \operatorname{conv}(Q)| = 2$, then a^* is the single interior point of $Q \cap A$ and similarly for B (Lemma 17). Without loss of generality, we assume that $|A \cap \partial \operatorname{conv}(Q)| = 2$ and thus a^* is the single interior point of $Q \cap A$. Using this assumption, we prove that $|Q \cap B| < |Q \cap A|$ (Proposition 19). By exchanging the roles of $Q \cap A$ and $Q \cap B$, we obtain $|Q \cap A| \leq |Q \cap B|$ (Proposition 22), which gives a contradiction.

To bound $|Q \cap B|$, we use three results about the sizes of the parameters $w_1, \ldots, w_t$ for the ℓ-divided set Q, that is, about the numbers of points of $Q \cap B$ in the a^*-wedges $W_1, \ldots, W_t$ of Q. We show that if we have $w_i = 2 = w_j$ for some $1 \leq i < j \leq t$, then $w_k = 0$ for some k with $i < k < j$ (Lemma 10). Further, for any three or four consecutive a^*-wedges whose union is convex and contains at least four points of $Q \cap B$, each of those a^*-wedges contains at most two such points (Lemma 16). Finally, we show that $w_1, \ldots, w_t \leq 3$ (Lemma 18). The proofs of Lemmas 16 and 18 rely on some results about small ℓ-divided sets with computer-aided proofs (Lemmas 13, 14, and 15). Altogether, this is sufficient to show that $|Q \cap B| < |Q \cap A|$.

We now start the proof of Theorem 2 by showing that if there is an ℓ-divided 5-hole in the intersection of P with a union of consecutive a^*-wedges, then there is an ℓ-divided 5-hole in P.

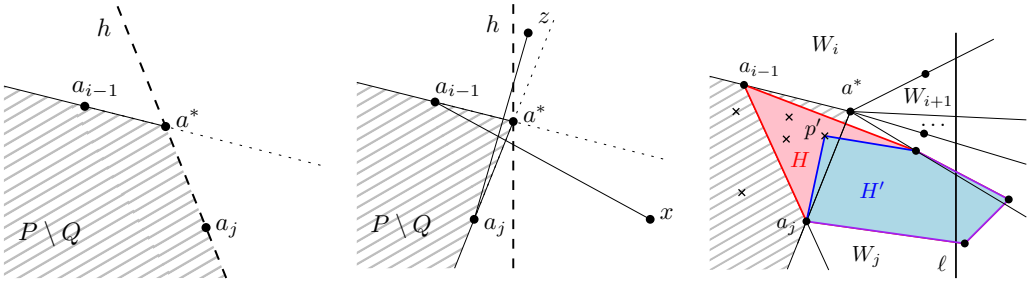


Figure 2 Illustration of the proof of Lemma 6. (a) The point a_j is to the right of a^*. (b) The point a_j is to the left of a^*. (c) The hole H properly intersects the ray $\overrightarrow{a^*a_j}$. The boundary of the convex hull of H is drawn red and the convex hull of H' is drawn blue.

▶ **Lemma 6.** *Let $P = A \cup B$ be an ℓ-divided set with A not in convex position. For integers i, j with $1 \le i \le j \le t$, let $W := \bigcup_{k=i}^{j} W_k$ and $Q := P \cap W$. If there is an ℓ-divided 5-hole in Q, then there is an ℓ-divided 5-hole in P.*

Proof. If W is convex then Q is an island of P and the statement immediately follows. Hence we assume that W is not convex. The region W is bounded by the rays $\overrightarrow{a^*a_{i-1}}$ and $\overrightarrow{a^*a_j}$ and all points of $P \setminus Q$ lie in the convex region $\mathbb{R}^2 \setminus W$; see Figure 2.

Since W is non-convex and every a^*-wedge contained in W intersects ℓ, at least one of the points a_{i-1} and a_j lies to the left of a^*. Moreover, the points $a_i, \ldots, a_{j-1}$ are to the right of a^* by Observation 5. Without loss of generality, we assume that a_{i-1} is to the left of a^*.

Let H be an ℓ-divided 5-hole in Q. If a_j is to the left of a^*, then we let h be the closed halfplane determined by the vertical line through a^* such that a_{i-1} and a_j lie in h. Otherwise, if a_j is to the right of a^*, then we let h be the closed halfplane determined by the line $\overline{a^*a_j}$ such that a_{i-1} lies in h. In either case, $h \cap A \cap Q = \{a^*, a_{i-1}, a_j\}$.

We say that H *properly intersects* a ray r if there are points $p, q \in H$ such that the interior of the segment pq intersects r. Now we show that if H properly intersects the ray $\overrightarrow{a^*a_j}$, then H contains a_{i-1}. Assume there are points $p, q \in H$ such that pq properly intersects $r := \overrightarrow{a^*a_j}$. Since r lies in h and neither of p and q lies in r, at least one of the points p and q lies in $h \setminus r$. Without loss of generality, we assume $p \in h \setminus r$. From $h \cap A \cap Q = \{a^*, a_{i-1}, a_j\}$ we have $p = a_{i-1}$. By symmetry, if H properly intersects the ray $\overrightarrow{a^*a_{i-1}}$, then H contains a_j.

Suppose for contradiction that H properly intersects both rays $\overrightarrow{a^*a_{i-1}}$ and $\overrightarrow{a^*a_j}$. Then H contains the points a_{i-1}, a_j, x, y, z for some points $x, y, z \in Q$, where $a_{i-1}x$ intersects $\overrightarrow{a^*a_j}$, and a_jz intersects $\overrightarrow{a^*a_{i-1}}$. Observe that z is to the left of $\overline{a_{i-1}a^*}$ and that x is to the right of $\overline{a_ja^*}$. If a_j lies to the right of a^*, then z is to the left of a^*, and thus z is in A; see Figure 2(a). However, this is impossible as z also lies in h. Hence, a_j lies to the left of a^*; see Figure 2(b). As x and z are both to the right of a^*, the point a^* is inside the convex quadrilateral $\square(a_{i-1}, a_j, x, z)$. This contradicts the assumption that H is a 5-hole in Q.

So assume that H properly intersects exactly one of the rays $\overrightarrow{a^*a_{i-1}}$ and $\overrightarrow{a^*a_j}$, say $\overrightarrow{a^*a_j}$; see Figure 2(c). In this case, H contains a_{i-1}. The interior of the triangle $\triangle(a^*, a_{i-1}, a_j)$ is empty of points of Q, since the triangle is contained in h. Moreover, $\mathrm{conv}(H)$ cannot intersect the line that determines h both strictly above and strictly below a^*. Thus, all remaining points of $H \setminus \{a_{i-1}\}$ lie to the right of $\overline{a_{i-1}a^*}$ and to the right of $\overline{a_ja^*}$. If H is empty of points of $P \setminus Q$, we are done. Otherwise, we let $H' := (H \setminus \{a_{i-1}\}) \cup \{p'\}$ where $p' \in P \setminus Q$ is a point inside $\triangle(a^*, a_{i-1}, a_j)$ closest to $\overline{a_ja^*}$. Note that the point p' might not

be unique. By construction, H' is an ℓ-divided 5-hole in P. An analogous argument shows that there is an ℓ-divided 5-hole in P if H properly intersects $\overrightarrow{a^*a_{i-1}}$.

Finally, if H does not properly intersect any of the rays $\overrightarrow{a^*a_{i-1}}$ and $\overrightarrow{a^*a_j}$, then $\text{conv}(H)$ contains no point of $P \setminus Q$ in its interior, and hence H is an ℓ-divided 5-hole in P. ◀

4.1 Sequences of a^*-wedges with at most two points of B

In this subsection we consider an ℓ-divided set $P = A \cup B$ with A not in convex position. We consider the union W of consecutive a^*-wedges, each containing at most two points of B, and derive an upper bound on the number of points of B that lie in W if there is no ℓ-divided 5-hole in $P \cap W$; see Corollary 11.

▶ **Observation 7.** *Let $P = A \cup B$ be an ℓ-divided set with A not in convex position. Let W_k be an a^*-wedge with $w_k \geq 1$ and $1 \leq k \leq t$ and let b be the leftmost point in $W_k \cap B$. Then the points a^*, a_{k-1}, b, and a_k form an ℓ-divided 4-hole in P.*

From Observation 4(i) and Observation 7 we obtain the following result.

▶ **Observation 8.** *Let $P = A \cup B$ be an ℓ-divided set with A not in convex position and with no ℓ-divided 5-hole in P. Let W_k be an a^*-wedge with $w_k \geq 2$ and $1 \leq k \leq t$ and let b be the leftmost point in $W_k \cap B$. For every point b' in $(W_k \cap B) \setminus \{b\}$, the line $\overline{bb'}$ intersects the segment $a_{k-1}a_k$. Consequently, b is inside $\triangle(a_{k-1}, a_k, b')$, to the left of $\overline{a_k b'}$, and to the right of $\overline{a_{k-1}b'}$.*

The following lemma states that there is an ℓ-divided 5-hole in P if two consecutive a^*-wedges both contain exactly two points of B. Its proof can be found in the full version of the paper [4].

▶ **Lemma 9.** *Let $P = A \cup B$ be an ℓ-divided set with A not in convex position and with $|A|, |B| \geq 5$. Let W_i and W_{i+1} be consecutive a^*-wedges with $w_i = 2 = w_{i+1}$ and $1 \leq i < t$. Then there is an ℓ-divided 5-hole in P.*

Next we show that if there is a sequence of consecutive a^*-wedges where the first and the last a^*-wedge both contain two points of B and every a^*-wedge in between them contains exactly one point of B, then there is an ℓ-divided 5-hole in P.

▶ **Lemma 10.** *Let $P = A \cup B$ be an ℓ-divided set with A not in convex position and with $|A| \geq 5$ and $|B| \geq 6$. Let $W_i, \ldots, W_j$ be consecutive a^*-wedges with $1 \leq i < j \leq t$, $w_i = 2 = w_j$, and $w_k = 1$ for every k with $i < k < j$. Then there is an ℓ-divided 5-hole in P.*

The proof of Lemma 10 can be found in the full version of the paper [4]. We now use Lemma 10 to show the following upper bound on the total number of points of B in a sequence $W_i, \ldots, W_j$ of consecutive a^*-wedges with $w_i, \ldots, w_j \leq 2$.

▶ **Corollary 11.** *Let $P = A \cup B$ be an ℓ-divided set with no ℓ-divided 5-hole, with A not in convex position, and with $|A| \geq 5$ and $|B| \geq 6$. For $1 \leq i \leq j \leq t$, let $W_i, \ldots, W_j$ be consecutive a^*-wedges with $w_k \leq 2$ for every k with $i \leq k \leq j$. Then $\sum_{k=i}^{j} w_k \leq j - i + 2$.*

Proof. Let n_0, n_1, and n_2 be the number of a^*-wedges from $W_i, \ldots, W_j$ with 0, 1, and 2 points of B, respectively. Due to Lemma 10, we can assume that between any two a^*-wedges from $W_i, \ldots, W_j$ with two points of B each, there is an a^*-wedge with no point of B. Thus $n_2 \leq n_0 + 1$. Since $n_0 + n_1 + n_2 = j - i + 1$, we have $\sum_{k=i}^{j} w_k = 0n_0 + 1n_1 + 2n_2 = (j - i + 1) + (n_2 - n_0) \leq j - i + 2$. ◀

4.2 Computer-assisted results

We now provide lemmas that are key ingredients in the proof of Theorem 2. All these lemmas have computer-aided proofs. Each result was verified by two independent implementations, which are also based on different abstractions of point sets; see below for details.

▶ **Lemma 12.** *Let $P = A \cup B$ be an ℓ-divided set with $|A| = 5$, $|B| = 6$, and with A not in convex position. Then there is an ℓ-divided 5-hole in P.*

▶ **Lemma 13.** *Let $P = A \cup B$ be an ℓ-divided set with no ℓ-divided 5-hole in P, $|A| = 5$, $4 \leq |B| \leq 6$, and with A in convex position. Then for every point a of A, every convex a-wedge contains at most two points of B.*

▶ **Lemma 14.** *Let $P = A \cup B$ be an ℓ-divided set with no ℓ-divided 5-hole in P, $|A| = 6$, and $|B| = 5$. Then for each point a of A, every convex a-wedge contains at most two points of B.*

▶ **Lemma 15.** *Let $P = A \cup B$ be an ℓ-divided set with no ℓ-divided 5-hole in P, $5 \leq |A| \leq 6$, $|B| = 4$, and with A in convex position. Then for every point a of A, if the non-convex a-wedge is empty of points of B, every a-wedge contains at most two points of B.*

We remark that all the assumptions in the statements of Lemmas 12 to 15 are necessary; see the full version of the paper [4]. To prove these lemmas, we employ an exhaustive computer search through all combinatorially different sets of $|P| \leq 11$ points in the plane. Since none of these statements depends on the actual coordinates of the points but only on the relative positions of the points, we distinguish point sets only by orientations of triples of points as proposed by Goodman and Pollack [20]. That is, we check all possible equivalence classes of point sets in the plane with respect to their triple-orientations, which are known as *order types*.

We wrote two independent programs to verify Lemmas 12 to 15. Both programs are available online [25, 8].

The first implementation is based on programs from the two bachelor's theses of Scheucher [26, 27]. For our verification purposes we reduced the framework from there to a very compact implementation [25]. The program uses the order type database [3, 7], which stores all order types realizable as point sets of size up to 11. The order types realizable as sets of ten points are available online [1] and the ones realizable as sets of eleven points need about 96 GB and are available upon request from Aichholzer. The running time of each of the programs in this implementation does not exceed two hours on a standard computer.

The second implementation [8] neither uses the order type database nor the program used to generate the database. Instead it relies on the description of point sets by so-called *signature functions* [9, 17]. In this description, points are sorted according to their x-coordinates and every unordered triple of points is represented by a sign from $\{-, +\}$, where the sign is $-$ if the triple traced in the order by increasing x-coordinates is oriented clockwise and the sign is $+$ otherwise. Every 4-tuple of points is then represented by four signs of its triples, which are ordered lexicographically. There are only eight 4-tuples of signs that we can obtain (out of 16 possible ones); see [9, Theorem 3.2] or [17, Theorem 7] for details. In our algorithm, we generate all possible signature functions using a simple depth-first search algorithm and verify the conditions from our lemmas for every signature. The running time of each of the programs in this implementation may take up to a few hundreds of hours.

4.3 Applications of the computer-assisted results

Here we present some applications of the computer-assisted results from Section 4.2.

▶ **Lemma 16.** *Let $P = A \cup B$ be an ℓ-divided set with no ℓ-divided 5-hole in P, with $|A| \geq 6$, and with A not in convex position. Then the following two conditions are satisfied.*

- **(i)** *Let W_i, W_{i+1}, W_{i+2} be three consecutive a^*-wedges whose union is convex and contains at least four points of B. Then $w_i, w_{i+1}, w_{i+2} \leq 2$.*
- **(ii)** *Let $W_i, W_{i+1}, W_{i+2}, W_{i+3}$ be four consecutive a^*-wedges whose union is convex and contains at least four points of B. Then $w_i, w_{i+1}, w_{i+2}, w_{i+3} \leq 2$.*

Proof. To show part (i), let $W := W_i \cup W_{i+1} \cup W_{i+2}$, $A' := A \cap W$, $B' := B \cap W$, and $P' := A' \cup B'$. Since W is convex, P' is an island of P and thus there is no ℓ-divided 5-hole in P'. Note that $|A'| = 5$ and A' is in convex position. If $|B'| \leq 5$, then every convex a^*-wedge in P' contains at most two points of B' by Lemma 13 applied to P'. So assume that $|B'| \geq 6$. We remove points from P' from the right to obtain $P'' = A' \cup B''$, where B'' contains exactly six points of B'. Note that there is no ℓ-divided 5-hole in P'', since P'' is an island of P'. By Lemma 13, each a^*-wedge in P'' contains exactly two points of B''. Let $\tilde{B}$ be the set of points of B that are to the left of the rightmost point of B'', including this point, and let $\tilde{P} := A \cup \tilde{B}$. Note that $B'' \subseteq \tilde{B}$. Since $|B''| = 6$ and since $W \cap \tilde{B} = B''$, each of the a^*-wedges W_i, W_{i+1}, W_{i+2} contains exactly two points of $\tilde{B}$. The a^*-wedges W_i, W_{i+1}, and W_{i+2} are also a^*-wedges in $\tilde{P}$. Thus, Lemma 9 applied to $\tilde{P}$ and W_i, W_{i+1} then gives us an ℓ-divided 5-hole in $\tilde{P}$. From the choice of $\tilde{P}$, we then have an ℓ-divided 5-hole in P, a contradiction.

To show part (ii), let $W := W_i \cup W_{i+1} \cup W_{i+2} \cup W_{i+3}$, $A' := A \cap W$, $B' := B \cap W$, and $P' := A' \cup B'$. Since W is convex, P' is an island of P and thus there is no ℓ-divided 5-hole in P'. Note that $|A'| = 6$ and A' is in convex position. If $|B'| = 4$, then the statement follows from Lemma 15 applied to P' since a^* is an extremal point of P'. If $|B'| = 5$, then the statement follows from Lemma 14 applied to P' and thus we can assume $|B'| \geq 6$. Suppose for contradiction that $w_j \geq 3$ for some $i \leq j \leq i + 3$. We remove points from P from the right to obtain P'' so that $B'' := P'' \cap B$ contains exactly six points of $W \cap B$. By applying part (i) for P'' and $W_i \cup W_{i+1} \cup W_{i+2}$ and $W_{i+1} \cup W_{i+2} \cup W_{i+3}$, we obtain that $|B'' \cap W_i|, |B'' \cap W_{i+3}| = 3$ and $|B'' \cap W_{i+1}|, |B'' \cap W_{i+2}| = 0$. Let b be the rightmost point from $P'' \cap W$. By Lemma 14 applied to $W \cap (P'' \setminus \{b\})$, there are at most two points of $B'' \setminus \{b\}$ in every a^*-wedge in $W \cap (P'' \setminus \{b\})$. This contradicts the fact that either $|(B'' \cap W_i) \setminus \{b\}| = 3$ or $|(B'' \cap W_{i+3}) \setminus \{b\}| = 3$. ◀

4.4 Extremal points of ℓ-critical sets

Recall the definition of ℓ-critical sets: An ℓ-divided point set $C = A \cup B$ is called ℓ-critical if neither $C \cap A$ nor $C \cap B$ is in convex position and if for every extremal point x of C, one of the sets $(C \setminus \{x\}) \cap A$ and $(C \setminus \{x\}) \cap B$ is in convex position.

In this section, we consider an ℓ-critical set $C = A \cup B$ with $|A|, |B| \geq 5$. We first show that C has at most two extremal points in A and at most two extremal points in B. Later, under the assumption that there is no ℓ-divided 5-hole in C, we show that $|B| \leq |A| - 1$ if A contains two extremal points of C (Section 4.4.1) and that $|B| \leq |A|$ if B contains two extremal points of C (Section 4.4.2).

▶ **Lemma 17.** *Let $C = A \cup B$ be an ℓ-critical set. Then the following statements are true.*

- **(i)** *If $|A| \geq 5$, then $|A \cap \partial \operatorname{conv}(C)| \leq 2$.*
- **(ii)** *If $A \cap \partial \operatorname{conv}(C) = \{a, a'\}$, then a^* is the single interior point in A and every point of $A \setminus \{a, a'\}$ lies in the convex region spanned by the lines $\overline{a^*a}$ and $\overline{a^*a'}$ that does not have any of a and a' on its boundary.*

(iii) *If $A \cap \partial\,\mathrm{conv}(C) = \{a, a'\}$, then the a^*-wedge that contains a and a' contains no point of B.*

By symmetry, analogous statements hold for B.

The proof of Lemma 17 can be found in the full version of the paper [4].

We remark that the assumption $|A| \geq 5$ in part (i) of Lemma 17 is necessary. In fact, arbitrarily large ℓ-critical sets with only four points in A and with three points of A on $\partial\,\mathrm{conv}(C)$ exist, and analogously for B.

▶ **Lemma 18.** *Let $C = A \cup B$ be an ℓ-critical set with no ℓ-divided 5-hole in C and with $|A| \geq 6$. Then $w_i \leq 3$ for every $1 < i < t$. Moreover, if $|A \cap \partial\,\mathrm{conv}(C)| = 2$, then $w_1, w_t \leq 3$.*

Proof. Recall that, since C is ℓ-critical, we have $|B| \geq 4$. Let i be an integer with $1 \leq i \leq t$. We assume that there is a point a in $A \cap \partial\,\mathrm{conv}(C)$, which lies outside of W_i, as otherwise there is nothing to prove for W_i (either $|A \cap \partial\,\mathrm{conv}(C)| = 1$ and $i \in \{1, t\}$ or $|A \cap \partial\,\mathrm{conv}(C)| = 2$ and, by Lemma 17(iii), $W_i \cap B = \emptyset$). We consider $C' := C \setminus \{a\}$. Since C is an ℓ-critical set, $A' := C' \cap A$ is in convex position. Thus, there is a non-convex a^*-wedge W' of C'. Since W' is non-convex, all other a^*-wedges of C' are convex. Moreover, since W' is the union of the two a^*-wedges of C that contain a, all other a^*-wedges of C' are also a^*-wedges of C. Let W be the union of all a^*-wedges of C that are not contained in W'. Note that W is convex and contains at least $|A| - 3 \geq 3$ a^*-wedges of C. Since $|A| \geq 6$, the statement follows from Lemma 16(i). ◀

4.4.1 Two extremal points of C in A

▶ **Proposition 19.** *Let $C = A \cup B$ be an ℓ-critical set with no ℓ-divided 5-hole in C, with $|A|, |B| \geq 6$, and with $|A \cap \partial\,\mathrm{conv}(C)| = 2$. Then $|B| \leq |A| - 1$.*

Proof. Since $|A \cap \partial\,\mathrm{conv}(C)| = 2$, Lemma 18 implies that $w_i \leq 3$ for every $1 \leq i \leq t$. Let a and a' be the two points in $A \cap \partial\,\mathrm{conv}(C)$. By Lemma 17(ii), all points of $A \setminus \{a, a'\}$ lie in the convex region R spanned by the lines $\overline{a^*a}$ and $\overline{a^*a'}$ that does not have any of a and a' on its boundary. That is, without loss of generality, $a = a_{h-1}$ and $a' = a_h$ for some $1 \leq h \leq |A| - 1$ and, by Lemma 17(iii), we have $w_h = 0$. Since all points of $A \setminus \{a, a'\}$ lie in the convex region R, the regions $W := \mathrm{cl}(\mathbb{R}^2 \setminus (W_{h-1} \cup W_h))$ and $W' := \mathrm{cl}(\mathbb{R}^2 \setminus (W_h \cup W_{h+1}))$ are convex. Here $\mathrm{cl}(X)$ denotes the closure of a set $X \subseteq \mathbb{R}^2$. Recall that the indices of the a^*-wedges are considered modulo $|A| - 1$ and that $\mathbb{R}^2$ is the union of all a^*-wedges.

First, suppose for contradiction that $|A| = 6$ and $|B| \geq 6$. There are exactly five a^*-wedges $W_1, \ldots, W_5$, and only four of them can contain points of B, since $w_h = 0$. We apply Lemma 16(i) to W and to W' and obtain that either $w_i \leq 2$ for every $1 \leq i \leq t$ or $w_{h-1}, w_{h+1} = 3$ and $w_i = 0$ for every $i \notin \{h-1, h+1\}$. In the first case, Corollary 11 implies that $|B| \leq 5$ and in the latter case Lemma 14 applied to $P \setminus \{b\}$, where b is the rightmost point of B, gives $|B| \leq 5$, a contradiction. Hence, we assume $|A| \geq 7$.

▶ **Claim 20.** *For $1 \leq k \leq t - 3$, if one of the four consecutive a^*-wedges W_k, W_{k+1}, W_{k+2}, or W_{k+3} contains 3 points of B, then $w_k + w_{k+1} + w_{k+2} + w_{k+3} = 3$.*

There are $|A| - 1 \geq 6$ a^*-wedges and, in particular, W and W' are both unions of at least four a^*-wedges. For every W_i with $w_i = 3$ and $1 \leq i \leq t$, the a^*-wedge W_i is either contained in W or in W'. Thus we can find four consecutive a^*-wedges $W_k, W_{k+1}, W_{k+2}, W_{k+3}$ whose union is convex and contains W_i. Lemma 16(ii) implies that each of $W_k, W_{k+1}, W_{k+2}, W_{k+3}$ except of W_i is empty of points of B. This finishes the proof of Claim 20.

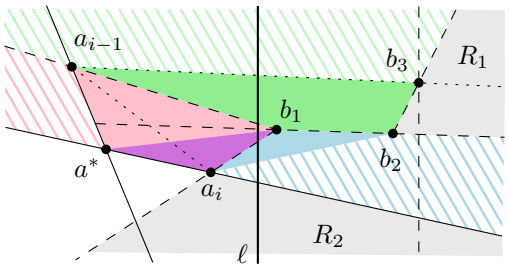


Figure 3 An illustration of the proof of Proposition 22.

▶ **Claim 21.** *For all integers i and j with $1 \leq i < j \leq t$, we have $\sum_{k=i}^{j} w_k \leq j - i + 2$.*

Let $S := (w_i, \ldots, w_j)$ and let S' be the subsequence of S obtained by removing every 1-entry from S. If S contains only 1-entries, the statement clearly follows. Thus we can assume that S' is non-empty. Recall that S' contains only 0-, 2-, and 3-entries, since $w_i \leq 3$ for all $1 \leq i \leq t$. Due to Claim 20, there are at least three consecutive 0-entries between every pair of nonzero entries of S' that contains a 3-entry. Together with Lemma 10, this implies that there is at least one 0-entry between every pair of 2-entries in S'.

By applying the following iterative procedure, we show that $\sum_{s \in S'} s \leq |S'| + 1$. While there are at least two nonzero entries in S', we remove the first nonzero entry s from S'. If $s = 2$, then we also remove the 0-entry from S' that succeeds s in S. If $s = 3$, then we also remove the two consecutive 0-entries from S' that succeed s in S'. The procedure stops when there is at most one nonzero element s' in the remaining subsequence S'' of S'. If $s' = 3$, then S'' contains at least one 0-entry and thus S'' contains at least $s' - 1$ elements. Since the number of removed elements equals the sum of the removed elements in every step of the procedure, we have $\sum_{s \in S'} s \leq |S'| + 1$. This implies

$$\sum_{k=i}^{j} w_k = \sum_{s \in S} s = |S| - |S'| + \sum_{s \in S'} s \leq |S| - |S'| + |S'| + 1 = j - i + 2$$

and finishes the proof of Claim 21.

If W_h does not intersect ℓ, that is, $t < h \leq |A| - 1$, then the statement follows from Claim 21 applied with $i = 1$ and $j = t$. Otherwise, we have $h = 1$ or $h = t$ and we apply Claim 21 with $(i, j) = (2, t)$ or $(i, j) = (1, t - 1)$, respectively. Since $t \leq |A| - 1$ and $w_h = 0$, this gives us $|B| \leq |A| - 1$. ◀

4.4.2 Two extremal points of C in B

▶ **Proposition 22.** *Let $C = A \cup B$ be an ℓ-critical set with no ℓ-divided 5-hole in C, with $|A|, |B| \geq 6$, and with $|B \cap \partial \operatorname{conv}(C)| = 2$. Then $|B| \leq |A|$.*

Proof. If $w_k \leq 2$ for all $1 \leq k \leq t$, then the statement follows from Corollary 11, since $|B| = \sum_{k=1}^{t} w_k \leq t + 1 \leq |A|$. Therefore we assume that there is an a^*-wedge W_i that contains at least three points of B. Let b_1, b_2, and b_3 be the three leftmost points in $W_i \cap B$ from left to right. Without loss of generality, we assume that b_3 is to the left of $\overline{b_1 b_2}$. Otherwise we can consider a vertical reflection of P. Figure 3 gives an illustration.

Let R_1 be the region that lies to the left of $\overline{b_1 b_2}$ and to the right of $\overline{b_2 b_3}$ and let R_2 be the region that lies to the right of $\overline{a_i b_1}$ and to the right of $\overline{a^* a_i}$. Let $B' := B \setminus \{b_1, b_2, b_3\}$.

▶ **Claim 23.** *Every point of B' lies in $R_1 \cup R_2$.*

We first show that every point of B' that lies to the left of $\overline{b_1b_2}$ lies in R_1. Then we show that every point of B' that lies to the right of $\overline{b_1b_2}$ lies in R_2.

By Observation 8, both lines $\overline{b_1b_2}$ and $\overline{b_1b_3}$ intersect the segment $a_{i-1}a_i$. Since the segment $a_{i-1}b_1$ intersects ℓ and since b_1 is the leftmost point of $W_i \cap B$, all points of B' that lie to the left of $\overline{b_1b_2}$ lie to the left of $\overline{a_{i-1}b_1}$. The four points a_{i-1}, b_1, b_2, b_3 form an ℓ-divided 4-hole in P, since a_{i-1} is the leftmost and b_3 is the rightmost point of a_{i-1}, b_1, b_2, b_3 and both a_{i-1} and b_3 lie to the left of $\overline{b_1b_2}$. By Observation 4(i), the sector $S(a_{i-1}, b_1, b_2, b_3)$ is empty of points of P (green shaded area in Figure 3). Altogether, all points of B' that lie to the left of $\overline{b_1b_2}$ are to the right of $\overline{b_2b_3}$ and thus lie in R_1.

Since the segment a_ib_1 intersects ℓ and since b_1 is the leftmost point of $W_i \cap B$, all points of B' that lie to the right of $\overline{b_1b_2}$ lie to the right of $\overline{a_ib_1}$. By Observation 4(i), the sector $S(b_1, b_2, b_3, a_{i-1})$ is empty of points of P. Combining this with the fact that a^* is to the right of $\overline{a_{i-1}b_3}$, we see that a^* lies to the right of $\overline{b_1b_2}$. Since b_1 and b_2 both lie to the left of $\overline{a^*a_i}$ and since a^* and a_i both lie to the right of $\overline{b_1b_2}$, the points b_2, b_1, a^*, a_i form an ℓ-divided 4-hole in P. By Observation 4(i), the sector $S(b_2, b_1, a^*, a_i)$ (blue shaded area in Figure 3) is empty of points of P. Altogether, all points of B' that lie to the right of $\overline{b_1b_2}$ are to the right of $\overline{a^*a_i}$ and to the right of $\overline{a_ib_1}$ and thus lie in R_2. This finishes the proof of Claim 23.

▶ **Claim 24.** *If b_4 is a point from $B' \setminus R_1$, then b_2 lies inside the triangle $\triangle(b_3, b_1, b_4)$.*

By Claim 23, b_4 lies in R_2 and thus to the right of $\overline{a_ib_1}$ and to the right of $\overline{a^*a_i}$. We recall that b_4 lies to the right of $\overline{b_1b_2}$.

We distinguish two cases. First, we assume that the points b_2, b_3, b_1, a_i are in convex position. Then b_2, b_3, b_1, a_i form an ℓ-divided 4-hole in P and, by Observation 4(i), the sector $S(b_2, b_3, b_1, a_i)$ is empty of points from P. Thus b_4 lies to the right of $\overline{b_2b_3}$ and the statement follows.

Second, we assume that the points b_2, b_3, b_1, a_i are not in convex position. Due to Observation 8, b_2 and b_3 both lie to the right of $\overline{a_ib_1}$. Moreover, since b_3 is the rightmost of those four points, b_2 lies inside the triangle $\triangle(b_3, b_1, a_i)$. In particular, a_i lies to the right of $\overline{b_2b_3}$. Therefore, since b_2 and b_3 are to the left of $\overline{a^*a_i}$, the line $\overline{b_2b_3}$ intersects ℓ in a point p above $\ell \cap \overline{a^*a_i}$. Let q be the point $\ell \cap \overline{b_1b_2}$. Note that q is to the left of $\overline{a^*a_i}$. The point b_4 is to the right of $\overline{b_2b_3}$, as otherwise b_4 lies in $\triangle(p, q, b_2)$, which is impossible because the points p, q, b_2 are in W_i while b_4 is not. Altogether, b_2 is inside $\triangle(b_3, b_1, b_4)$ and this finishes the proof of Claim 24.

▶ **Claim 25.** *Either every point of B' is to the right of b_3 or b_3 is the rightmost point of B.*

By Observation 4(i), the sector $S(b_3, a_{i-1}, b_1, b_2)$ is empty of points of P and thus all points of $B' \cap R_1$ lie to the left of $\overline{a_{i-1}b_3}$ and, in particular, to the right of b_3.

Suppose for contradiction that the claim is not true. That is, there is a point $b_4 \in B'$ that is the rightmost point in B and there is a point $b_5 \in B'$ that is to the left of b_3. Note that b_4 is an extremal point of C. By Claim 23 and by the fact that all points of $B' \cap R_1$ lie to the right of b_3, b_5 lies in $R_2 \setminus R_1$. By Claim 24, b_2 lies in the triangle $\triangle(b_1, b_5, b_3)$, and thus $B \setminus \{b_4\}$ is not in convex position. This contradicts the assumption that C is an ℓ-critical island. This finishes the proof of Claim 25.

▶ **Claim 26.** *The point b_3 is the third leftmost point of B. In particular, W_i is the only a^*-wedge with at least three points of B.*

Suppose for contradiction that b_3 is not the third leftmost point of B. Then by Claim 25, b_3 is the rightmost point of B and therefore an extremal point of B. This implies that $B' \subseteq R_2 \setminus R_1$, since all points of $B' \cap R_1$ lie to the right of b_3. By Claim 24, each point of B'

then forms a non-convex quadrilateral together with b_1, b_2, and b_3. Since neither b_1 nor b_2 are extremal points of C and since $|B \cap \partial \operatorname{conv}(C)| = 2$, there is a point $b_4 \in B$ that is an extremal point of C. Since $|B| \geq 5$, the set $C \setminus \{b_4\}$ has none of its parts separated by ℓ in convex position, which contradicts the assumption that C is an ℓ-critical set. Since W_i is an arbitrary a^*-wedge with $w_i \geq 3$, Claim 26 follows.

▶ **Claim 27.** *Let W be a union of four consecutive a^*-wedges that contains W_i. Then $|W \cap B| \leq 4$.*

Suppose for contradiction that $|W \cap B| \geq 5$. Let $C' := C \cap W$. Note that $|C' \cap A| = 6$ and that a^*, a_{i-1}, a_i lie in C'. By Lemma 6, there is no ℓ-divided 5-hole in C'. We obtain C'' by removing points from C' from the right until $|C'' \cap B| = 5$. Since C'' is an island of C', there is no ℓ-divided 5-hole in C''. From Claim 26 we know that b_1, b_2, b_3 are the three leftmost points in C and thus lie in C''. We apply Lemma 14 to C'' and, since b_1, b_2, b_3 lie in a convex a^*-wedge of C'', we obtain a contradiction. This finishes the proof of Claim 27.

We now complete the proof of Proposition 22. First, we assume that $1 \leq i \leq 4$. Let $W := W_1 \cup W_2 \cup W_3 \cup W_4$. By Claim 27, $|W \cap B| \leq 4$. Claim 26 implies that $w_k \leq 2$ for every k with $5 \leq k \leq t$. By Corollary 11, we have

$$|B| = \sum_{k=1}^{4} w_k + \sum_{k=5}^{t} w_k \leq 4 + (t-3) = t + 1 \leq |A|.$$

The case $t - 3 \leq i \leq t$ follows by symmetry.

Second, we assume that $5 \leq i \leq t-4$. Let $W := W_{i-3} \cup W_{i-2} \cup W_{i-1} \cup W_i$. Note that W is convex, since $2 \leq i - 3$ and $i < t$. By Lemma 16(ii), we have $w_{i-3} + w_{i-2} + w_{i-1} + w_i \leq 3$ and $w_i + w_{i+1} + w_{i+2} + w_{i+3} \leq 3$. By Claim 26, $w_k \leq 2$ for all k with $1 \leq k \leq i - 4$. Thus, by Corollary 11, $\sum_{k=1}^{i-4} w_k \leq i - 3$. Similarly, we have $\sum_{k=i+4}^{t} w_k \leq t - i - 2$. Altogether, we obtain that

$$|B| = \sum_{k=1}^{i-4} w_k + \sum_{k=i-3}^{i-1} w_k + w_i + \sum_{k=i+1}^{i+3} w_k + \sum_{k=i+4}^{t} w_k \leq (i-3) + 3 + (t-i-2) = t-2 \leq |A| - 3.$$

◀

4.5 Finalizing the proof of Theorem 2

We are now ready to prove Theorem 2. Namely, we show that for every ℓ-divided set $P = A \cup B$ with $|A|, |B| \geq 5$ and with neither A nor B in convex position there is an ℓ-divided 5-hole in P.

Suppose for the sake of contradiction that there is no ℓ-divided 5-hole in P. By the result of Harborth [21], every set P of ten points contains a 5-hole in P. In the case $|A|, |B| = 5$, the statement then follows from the assumption that neither of A and B is in convex position.

So assume that at least one of the sets A and B has at least six points. We obtain an island Q of P by iteratively removing extremal points so that neither part is in convex position after the removal and until one of the following conditions holds.

(i) One of the parts $Q \cap A$ and $Q \cap B$ has only five points.

(ii) Q is an ℓ-critical island of P with $|Q \cap A|, |Q \cap B| \geq 6$.

In case (i), we have $|Q \cap A| = 5$ or $|Q \cap B| = 5$. If $|Q \cap A| = 5$ and $|Q \cap B| \geq 6$, then we let Q' be the union of $Q \cap A$ with the six leftmost points of $Q \cap B$. Since $Q \cap A$ is not in convex position, Lemma 12 implies that there is an ℓ-divided 5-hole in Q', which is also an ℓ-divided 5-hole in Q, since Q' is an island of Q. However, this is impossible as then there is

an ℓ-divided 5-hole in P because Q is an island of P. If $|Q \cap A| \geq 6$ and $|Q \cap B| = 5$, then we proceed analogously.

In case (ii), we have $|Q \cap A|, |Q \cap B| \geq 6$. There is no ℓ-divided 5-hole in Q, since Q is an island of P. By Lemma 17(i), we can assume without loss of generality that $|A \cap \partial \operatorname{conv}(Q)| = 2$. Then it follows from Proposition 19 that $|Q \cap B| < |Q \cap A|$. By exchanging the roles of $Q \cap A$ and $Q \cap B$ and by applying Proposition 22, we obtain that $|Q \cap A| \leq |Q \cap B|$, a contradiction. This completes the proof of Theorem 2.

References

1 O. Aichholzer. Enumerating order types for small point sets with applications. `http://www.ist.tugraz.at/aichholzer/research/rp/triangulations/ordertypes/`.

2 O. Aichholzer. [Empty] [colored] k-gons. Recent results on some Erdős–Szekeres type problems. In *Proceedings of XIII Encuentros de Geometría Computacional*, pages 43–52, Zaragoza, Spain, 2009.

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

4 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. `http://arXiv.org/abs/1703.05253`, 2017.

5 O. Aichholzer, R. Fabila-Monroy, T. Hackl, C. Huemer, A. Pilz, and B. Vogtenhuber. Lower bounds for the number of small convex k-holes. *Computational Geometry: Theory and Applications*, 47(5):605–613, 2014.

6 O. Aichholzer, T. Hackl, and B. Vogtenhuber. On 5-gons and 5-holes. *Lecture Notes in Computer Science*, 7579:1–13, 2012.

7 O. Aichholzer and H. Krasser. Abstract order type extension and new results on the rectilinear crossing number. *Computational Geometry: Theory and Applications*, 36(1):2–15, 2007.

8 M. Balko. `http://kam.mff.cuni.cz/~balko/superlinear5Holes`.

9 M. Balko, R. Fulek, and J. Kynčl. Crossing numbers and combinatorial characterization of monotone drawings of K_n. *Discrete & Computational Geometry*, 53(1):107–143, 2015.

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

11 I. Bárány and Gy. Károlyi. Problems and results around the Erdős–Szekeres convex polygon theorem. In Akiyama, Kano, and Urabe, editors, *Discrete and Computational Geometry*, volume 2098 of *Lecture Notes in Computer Science*, pages 91–105. Springer, 2001.

12 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.

13 P. Brass, W. Moser, and J. Pach. *Research Problems in Discrete Geometry*. Springer, 2005.

14 K. Dehnhardt. *Leere konvexe Vielecke in ebenen Punktmengen*. PhD thesis, TU Braunschweig, Germany, 1987. In German.

15 P. Erdős. Some more problems on elementary geometry. *Australian Mathematical Society Gazette*, 5(2):52–54, 1978.

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

17 S. Felsner and H. Weil. Sweeps, arrangements and signotopes. *Discrete Applied Mathematics*, 109(1–2):67–94, 2001.

18 A. García. A note on the number of empty triangles. *Lecture Notes in Computer Science*, 7579:249–257, 2012.

19 T. Gerken. Empty convex hexagons in planar point sets. *Discrete & Computational Geometry*, 39(1–3):239–272, 2008.

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

21 H. Harborth. Konvexe Fünfecke in ebenen Punktmengen. *Elemente der Mathematik*, 33:116–118, 1978. In German.

22 J. D. Horton. Sets with no empty convex 7-gons. *Canadian Mathematical Bulletin*, 26(4):482–484, 1983.

23 C. M. Nicolás. The empty hexagon theorem. *Discrete & Computational Geometry*, 38(2):389–397, 2007.

24 R. Pinchasi, R. Radoičić, and M. Sharir. On empty convex polygons in a planar point set. *Journal of Combinatorial Theory, Series A*, 113(3):385–419, 2006.

25 M. Scheucher. `http://www.ist.tugraz.at/scheucher/5holes`.

26 M. Scheucher. Counting convex 5-holes, Bachelor's thesis, 2013. In German.

27 M. Scheucher. On order types, projective classes, and realizations, Bachelor's thesis, 2014.

28 W. Steiger and J. Zhao. Generalized ham-sandwich cuts. *Discrete & Computational Geometry*, 44(3):535–545, 2010.

29 P. Valtr. Convex independent sets and 7-holes in restricted planar point sets. *Discrete & Computational Geometry*, 7(2):135–152, 1992.

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

31 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.

A Universal Slope Set for 1-Bend Planar Drawings

Patrizio Angelini[1], Michael A. Bekos[2], Giuseppe Liotta[3], and Fabrizio Montecchiani[4]

1 **Wilhelm-Schickhard-Institut für Informatik, Universität Tübingen, Tübingen, Germany**
angelini@informatik.uni-tuebingen.de
2 **Wilhelm-Schickhard-Institut für Informatik, Universität Tübingen, Tübingen, Germany**
bekos@informatik.uni-tuebingen.de
3 **Universitá degli Studi di Perugia, Perugia, Italy**
giuseppe.liotta@unipg.it
4 **Universitá degli Studi di Perugia, Perugia, Italy**
fabrizio.montecchiani@unipg.it

Abstract

We describe a set of $\Delta-1$ slopes that are universal for 1-bend planar drawings of planar graphs of maximum degree $\Delta \geq 4$; this establishes a new upper bound of $\Delta-1$ on the 1-bend planar slope number. By universal we mean that every planar graph of degree Δ has a planar drawing with at most one bend per edge and such that the slopes of the segments forming the edges belong to the given set of slopes. This improves over previous results in two ways: Firstly, the best previously known upper bound for the 1-bend planar slope number was $\frac{3}{2}(\Delta-1)$ (the known lower bound being $\frac{3}{4}(\Delta-1)$); secondly, all the known algorithms to construct 1-bend planar drawings with $O(\Delta)$ slopes use a different set of slopes for each graph and can have bad angular resolution, while our algorithm uses a universal set of slopes, which also guarantees that the minimum angle between any two edges incident to a vertex is $\frac{\pi}{(\Delta-1)}$.

1998 ACM Subject Classification G.2.1 Combinatorics, G.2.2 Graph Theory

Keywords and phrases Slope number, 1-bend drawings, planar graphs, angular resolution

Digital Object Identifier 10.4230/LIPIcs.SoCG.2017.9

1 Introduction

This paper is concerned with planar drawings of graphs such that each edge is a poly-line with few bends, each segment has one of a limited set of possible slopes, and the drawing has good angular resolution, i.e. it forms large angles between consecutive edges incident on a common vertex. Besides their theoretical interest, visualizations with these properties find applications in software engineering and information visualization (see, e.g., [12, 26, 40]). For example, planar graphs of maximum degree four (degree-4 planar graphs) are widely used in database design, where they are typically represented by orthogonal drawings, i.e. crossing-free drawings such that every edge segment is a polygonal chain of horizontal and vertical segments. Clearly, orthogonal drawings of degree-4 planar graphs are optimal both in terms of angular resolution and in terms of number of distinct slopes for the edges. Also, a classical result in the graph drawing literature is that every degree-4 planar graph, except the octahedron, admits an orthogonal drawing with at most two bends per edge [5, 34].

It is immediate to see that more than two slopes are needed in a planar drawing of a graph with vertex degree $\Delta \geq 5$. The *k-bend planar slope number* of a graph G with degree Δ is the

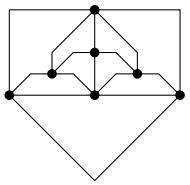


Figure 1 A 1-bend planar drawing with 4 slopes and angular resolution $\frac{\pi}{4}$ of a graph with $\Delta = 5$.

minimum number of distinct slopes that are sufficient to compute a crossing-free drawing of G with at most k bends per edge. Keszegh et al. [30] generalize the technique by Biedl and Kant [5] and prove that for any $\Delta \geq 5$, the 2-bend planar slope number of a degree-Δ planar graph is $\lceil \Delta/2 \rceil$; the construction in their proof has optimal angular resolution, that is $\frac{2\pi}{\Delta}$.

For the case of drawings with one bend per edge, Keszegh et al. [30] also show an upper bound of 2Δ and a lower bound of $\frac{3}{4}(\Delta - 1)$ on the 1-bend planar slope number, while a recent paper by Knauer and Walczak [31] improves the upper bound to $\frac{3}{2}(\Delta - 1)$. Both these papers use a similar technique: First, the graph is realized as a contact representation with T-shapes [10], which is then transformed into a planar drawing where vertices are points and edges are poly-lines with at most one bend. The set of slopes depends on the initial contact representation and may change from graph to graph; also, each slope is either very close to horizontal or very close to vertical, which gives rise to bad angular resolution. Knauer and Walczak [31] also proved that the 1-bend (outer)planar slope number of outerplanar graphs with $\Delta > 2$ is $\lceil \frac{\Delta}{2} \rceil$ and that $\Delta + 1$ slopes suffice for planar bipartite graphs.

We study the trade-off between number of slopes, angular resolution, and number of bends per edge in a planar drawing of a graph with maximum degree Δ. We improve the upper bounds in [31] on the 1-bend planar slope number, and at the same time we achieve $\Omega(\frac{1}{\Delta})$ angular resolution. More precisely, we prove the following.

▶ **Theorem 1.** *For any $\Delta \geq 4$, there exists an equispaced universal set S of $\Delta - 1$ slopes for 1-bend planar drawings of planar graphs with maximum degree Δ. That is, every such graph has a planar drawing with the following properties: (i) each edge has at most one bend; (ii) each edge segment uses one of the slopes in S; and (iii) the minimum angle between any two consecutive edge segments incident on a vertex or a bend is at least $\frac{\pi}{\Delta - 1}$.*

Theorem 1, in conjuction with [27], implies that the 1-bend planar slope number of planar graphs with $n \geq 5$ vertices and maximum degree $\Delta \geq 3$ is at most $\Delta - 1$. We prove the theorem by using an approach that is conceptually different from that of Knauer and Walczak [31]: We do not construct an intermediate representation, but we prove the existence of a *universal* set of slopes and use it to directly compute a 1-bend planar drawing of any graph with degree at most Δ. The universal set of slopes consists of $\Delta - 1$ distinct slopes such that the minimum angle between any two of them is $\frac{\pi}{(\Delta - 1)}$. An immediate consequence of the $\frac{3}{4}(\Delta - 1)$ lower bound argument in [30] is that a 1-bend planar drawing with the minimum number of slopes cannot have angular resolution larger than $\frac{4}{3}\frac{\pi}{(\Delta - 1)}$. Hence, the angular resolution of our drawings is optimal up to a multiplicative factor of at most 0.75; also, note that the angular resolution of a graph of degree Δ is at most $\frac{2\pi}{\Delta}$ even when the number of slopes and the number of bends along the edges are not bounded.

The proof of Theorem 1 is constructive and it gives rise to a linear-time algorithm assuming the real RAM model of computation. A drawing computed with this algorithm is shown in Fig. 1. The construction for triconnected planar graphs uses a variant of the shifting method of de Fraysseix, Pach and Pollack [11]; this is used as a building block for

the drawing algorithm for biconnected planar graphs, which is based on the SPQR-tree decomposition of the graph into its triconnected components (see, e.g., [12]). By using a BC-tree decomposition, our approach can be further extended to general planar graphs, details can be found in the extended version of this paper [1]. If the graph is disconnected, since we use a universal set of slopes, the distinct connected components can be drawn independently.

Related work. The results on the slope number of graphs are mainly classified into two categories based on whether the input graph is planar or not. For a (planar) graph G of maximum degree Δ, the *slope number* (*planar slope number*) is the minimum number of slopes that are sufficient to compute a straight-line (planar) drawing of G. The slope number of non-planar graphs is lower bounded by $\lceil \Delta/2 \rceil$ [41] but it can be arbitrarily large, even when $\Delta = 5$ [2]. For $\Delta = 3$ this number is 4 [35], while it is unknown for $\Delta = 4$. Upper bounds on the slope number are known for complete graphs [41] and outer 1-planar graphs [14]. Deciding whether a graph has slope number 2 is NP-complete [15, 19]. For a planar graph G of maximum degree Δ, the planar slope number of G is lower bounded by $3\Delta - 6$ and upper bounded by $O(2^{\Delta})$ [30]. Improved upper bounds are known for special subclasses of planar graphs, e.g., planar graphs with $\Delta \leq 3$ [15, 13, 28], outerplanar graphs with $\Delta \geq 4$ [32], partial 2-trees [33], planar partial 3-trees [25]. Note that determining the planar slope number of a graph is hard in the existential theory of the reals [24].

Closely related to our problem is the problem of finding *d-linear* drawings of graphs, in which all angles (formed either between consecutive segments of an edge or between edge-segments incident to the same vertex) are multiples of $2\pi/d$. For $d = 4$, an angular resolution of $2\pi/d$ implies d-linearity, while for $d > 4$ this is not always true [8]. Special types of d-linear drawings are the *orthogonal* [5, 7, 21, 39] and the *octilinear* [3, 4, 36] drawings, for which $d = 2$ and $d = 4$ holds, respectively. As already recalled, any planar graph with $\Delta \leq 4$ (except the octahedron) admits a planar orthogonal drawing with at most two bends per edge [5, 34]. Deciding whether a degree-4 planar graph has an orthogonal drawing with no bends is NP-complete [21], while it is solvable in polynomial time if one bend per edge is allowed [6]. Octilinear drawings have been mainly studied in the context of metro map visualization and map schematization [37, 38]. Nöllenburg [36] proved that deciding whether an embedded planar graph with $\Delta \leq 8$ admits a bendless planar octilinear drawing is NP-complete. Bekos et al. [3] showed that any planar graph with $\Delta \leq 5$ admits a planar octilinear drawing with at most one bend per edge and this is not always possible if $\Delta \geq 6$. Note that in our work we generalize their positive result to any Δ. Later, Bekos et al. [4] studied bounds on the total number of bends of planar octilinear drawings. Finally, trade-offs between number of bends, angular resolution, and area requirement of planar drawings of graphs with maximum degree Δ are, for example, studied in [9, 16, 17, 18, 20, 22].

Paper organization. The rest of this paper is organized as follows. Preliminaries are given in Section 2. In Section 3, we describe a drawing algorithm for triconnected planar graphs. The technique is extended to biconnected planar graphs in Sections 4. Finally, in Section 5 we discuss further implications of Theorem 1 and we list open problems.

2 Preliminaries

A graph G containing neither loops nor multiple edges is *simple*. We consider simple graphs, if not otherwise specified. The *degree* of a vertex of G is the number of its neighbors. We say

that G has *maximum degree* Δ if it contains a vertex with degree Δ but no vertex with degree larger than Δ. A graph is *connected*, if for any pair of vertices there is a path connecting them. Graph G is *k-connected*, if the removal of $k-1$ vertices leaves the graph connected. A 2-connected (3-connected) graph is also called *biconnected* (*triconnected*, respectively).

A *drawing* Γ of G maps each vertex of G to a point in the plane and each edge of G to a Jordan arc between its two endpoints. A drawing is *planar*, if no two edges cross (except at common endpoints). A planar drawing divides the plane into connected regions, called *faces*. The unbounded one is called *outer face*. A graph is *planar*, if it admits a planar drawing. A *planar embedding* of a planar graph is an equivalence class of planar drawings that combinatorially define the same set of faces and outer face.

The *slope* s of a line ℓ is the angle that a horizontal line needs to be rotated counterclockwise in order to make it overlap with ℓ. The slope of an edge-segment is the slope of the line containing the segment. Let S be a set of slopes sorted in increasing order; assume (up to a rotation) that S contains the 0 angle, which we call *horizontal slope*. A *1-bend* planar drawing Γ of graph G *on* S is a planar drawing of G in which every edge is composed of at most two straight-line segments, each of which has a slope that belongs to S. We say that S is *equispaced* if and only if the difference between any two consecutive slopes of S is $\frac{\pi}{|S|}$. For a vertex v in G, each slope $s \in S$ defines two different *rays* that emanate from v and have slope s. If s is the horizontal slope, then these rays are called *horizontal*. Otherwise, one of them is the *top* and the other one is the *bottom* ray of v. Consider a 1-bend planar drawing Γ of a graph G and a ray r_v emanating from a vertex v of G. We say that r_v is *free* if there is no edge attached to v through r_v. We also say that r_v is *incident* to face f of Γ if and only if r_v is free and the first face encountered when moving from v along r_v is f.

Let Γ be a 1-bend planar drawing of a graph and let e be an edge incident to the outer face of Γ that has a horizontal segment. A *cut at* e is a y-monotone curve that

(i) starts at any point of the horizontal segment of e,
(ii) ends at any point of a horizontal segment of an edge $e' \neq e$ incident to the outer face of Γ, and
(iii) crosses only horizontal segments of Γ.

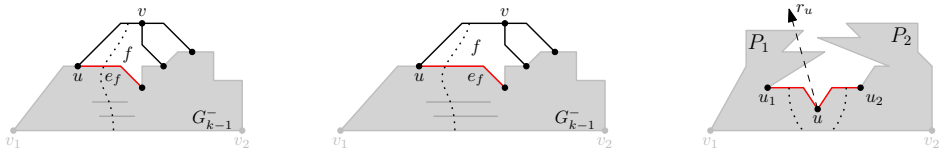
Central in our approach is the canonical order of triconnected planar graphs which can be computed in linear time [11, 29]. Let $G = (V, E)$ be a triconnected planar graph and let $\Pi = (P_0, \ldots, P_m)$ be a partition of V into paths, such that $P_0 = \{v_1, v_2\}$, $P_m = \{v_n\}$, edges (v_1, v_2) and (v_1, v_n) exist and belong to the outer face of G. For $k = 0, \ldots, m$, let G_k be the subgraph induced by $\cup_{i=0}^{k} P_i$ and denote by C_k the outer face of G_k. Π is a *canonical order* of G if for each $k = 1, \ldots, m-1$ the following hold:

(i) G_k is biconnected,
(ii) all neighbors of P_k in G_{k-1} are on C_{k-1},
(iii) $|P_k| = 1$ or the degree of each vertex of P_k is two in G_k, and
(iv) all vertices of P_k with $0 \leq k < m$ have at least one neighbor in P_j for some $j > k$.

An SPQR-tree $\mathcal{T}$ represents the decomposition of a biconnected graph G into its triconnected components and it can be computed in linear time [12, 23]. Every triconnected component of G is associated with a node μ of $\mathcal{T}$. The triconnected component itself is called the *skeleton* of μ, denoted by G_μ^{skel}. A node μ in $\mathcal{T}$ can be of four different types:

(i) μ is an *R-node*, if G_μ^{skel} is a triconnected graph,
(ii) a simple cycle of length at least three classifies μ as an *S-node*,
(iii) a bundle of at least three parallel edges classifies μ as a *P-node*,
(iv) the leaves of $\mathcal{T}$ are *Q-nodes*, whose skeleton consists of two parallel edges.

Neither two S- nor two P-nodes are adjacent in $\mathcal{T}$. A *virtual edge* in G_μ^{skel} corresponds to a



(a) (b) (c)

Figure 2 Illustrations for (a-b) Lemma 2 and (c) Lemma 3.

tree node ν that is adjacent to μ in $\mathcal{T}$, more precisely, to another virtual edge in G_ν^{skel}. If we assume that $\mathcal{T}$ is rooted at a Q-node ρ, then every skeleton (except the one of ρ) contains exactly one virtual edge, called *reference edge* and whose endpoints are the *poles* of μ, that has a counterpart in the skeleton of its parent. Every subtree $\mathcal{T}_\mu$ rooted at a node μ of $\mathcal{T}$ induces a subgraph G_μ of G called *pertinent*, that is described by $\mathcal{T}_\mu$ in the decomposition.

3 Triconnected Planar Graphs

Let G be a triconnected planar graph with $\Delta \geq 4$ and let S be a set of $\Delta - 1$ equispaced slopes containing the horizontal one. We consider the vertices of G according to a canonical order $\Pi = (P_0, \ldots, P_m)$. For $k = 0, \ldots, m$, let G_k^- be the planar graph obtained by removing edge (v_1, v_2) from G_k, and let C_k^- be the path from v_1 to v_2 obtained by removing (v_1, v_2) from C_k. We construct a 1-bend planar drawing of G_k^- on S satisfying the following invariants.

(I.1) No part of the drawing lies below vertices v_1 and v_2, which have the same y-coordinate.
(I.2) Every edge on C_k^- has a horizontal segment.
(I.3) Each vertex v on C_k^- has at least as many free top rays incident to the outer face of G_k^- as the number of its neighbors in $G \setminus G_k$.

Once a 1-bend planar drawing on S of G_m^- satisfying Invariants I.3–I.3 has been constructed, a 1-bend planar drawing on S of $G = G_m^- \cup \{(v_1, v_2)\}$ can be obtained by drawing (v_1, v_2) as a polyline composed of two straight-line segments, one attaching at the first clockwise bottom ray of v_1 and the other one at the first anti-clockwise bottom ray of v_2. Since S has at least three slopes, these two rays cross. Invariant I.3 ensures that edge (v_1, v_2) does not introduce any crossing. Next, we state two useful properties of any 1-bend planar drawing on S satisfying Invariants I.3–I.3, whose proofs can be found in [1]; see also Figs. 2a–2c).

▶ **Lemma 2.** *Let Γ_k be a 1-bend planar drawing on S of G_k^- satisfying Invariants I.3–I.3. Let (u, v) be an edge of C_k^- such that u precedes v along path C_k^- and let $\sigma > 0$. There exists a 1-bend planar drawing Γ_k' on S of G_k^-, satisfying Invariants I.3–I.3, in which the horizontal distance between any two consecutive vertices along C_k^- is the same as in Γ_k, except for u and v, whose horizontal distance is increased by σ.*

▶ **Lemma 3.** *Let Γ_k be a 1-bend planar drawing on S of G_k^- satisfying Invariants I.3–I.3. Let u be any vertex of C_k^-, and let r_u be any free top ray of u that is incident to the outer face of G_k^- in Γ_k. Then, it is possible to construct a 1-bend planar drawing Γ_k' on S of G_k^-, satisfying Invariants I.3–I.3, in which r_u does not cross any edge of Γ_k'.*

We now describe our algorithm. First, we draw $P_0 = \{v_1, v_2\}$ and $P_1 = \{v_3, \ldots, v_j\}$ such that $v_1, v_3, \ldots, v_j, v_2$ lie along a horizontal line, in this order (edge (v_1, v_2) is not considered). Invariants I.3 and I.3 hold. Invariant I.3 follows since S contains $\Delta - 2$ top rays and all vertices drawn so far (including v_1 and v_2) have at most $\Delta - 2$ neighbors later in the canonical

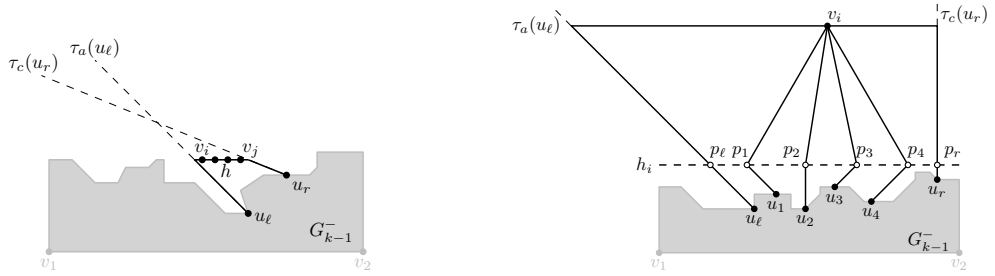


Figure 3 Illustration of the cases of: (a) a chain, (b) a singleton of degree δ_i in G_k.

order. We now show how to add path P_k, for some $k > 1$, to a drawing Γ_{k-1} satisfying Invariants I.3–I.3, so that the resulting drawing Γ_k of G_k^- is a 1-bend planar drawing on S satisfying Invariants I.3–I.3. We distinguish the cases in which P_k is a chain or a singleton.

Suppose first that P_k is a chain, say $\{v_i, v_{i+1}, \ldots, v_j\}$; refer to Fig. 3a. Let u_ℓ and u_r be the neighbors of v_i and v_j in C^-_{k-1}, respectively. By Invariant I.3, each of u_ℓ and u_r has at least one free top ray that is incident to the outer face of Γ_{k-1}; among them, we denote by $\tau_a(u_\ell)$ the first one in anti-clockwise order for u_ℓ, and by $\tau_c(u_r)$ the first one in clockwise order for u_r. By Lemma 3, we can assume that $\tau_a(u_\ell)$ and $\tau_c(u_r)$ do not cross any edge in Γ_{k-1}. This implies that there exists a horizontal line-segment h whose left and right endpoints are on $\tau_a(u_\ell)$ and $\tau_c(u_r)$, respectively, that does not cross any edge of Γ_{k-1}. We place all the vertices $v_i, v_{i+1}, \ldots, v_j$ of P_k on interior points of h, in this left-to-right order. Then, we draw edge (u_ℓ, v_i) with a segment along h and the other one along $\tau_a(u_\ell)$; we draw edge (u_r, v_j) with a segment along h and the other one along $\tau_c(u_r)$, and we draw every edge (v_q, v_{q+1}), with $q = i, \ldots, j-1$, with a unique segment along h.

By construction, Γ_k is a planar drawing on S. All the vertices of P_k lie above u_ℓ and u_r, since $\tau_a(u_\ell)$ and $\tau_c(u_r)$ are top rays of u_ℓ and u_r, respectively. Hence, these vertices and their incident edges lie above v_1 and v_2, and thus Invariant I.3 is satisfied by Γ_k. Invariant I.3 is satisfied since every edge that is drawn at this step has a segment along h, which is horizontal. Invariant I.3 is satisfied since we attached edges (u_ℓ, v_i) and (u_r, v_j) at vertices u_ℓ and u_r using the first anti-clockwise free top ray of u_ℓ and the first clockwise free top ray of u_r among those incident to the outer face, respectively. Thus, we reduced only by one the number of free top rays incident to the outer face for u_ℓ and u_r. For the other vertices of P_k, the invariant is satisfied since their $\Delta - 2$ top rays are free and incident to the outer face. This concludes our description for the case in which P_k is a chain.

Suppose now that P_k is a singleton $\{v_i\}$ of degree $\delta_i \leq \Delta$ in G_k^-. This also includes the case in which $P_k = P_m$ is the last path of Π. If $\delta_i = 2$, then v_i is placed as in the case of a chain. So, we may assume that $\delta_i \geq 3$. Let $u_\ell, u_1, u_2, \ldots, u_{\delta_i - 2}, u_r$ be the neighbors of v_i as they appear along C^-_{k-1}; see Fig. 3b. By Invariant I.3, each neighbor of v_i in C^-_{k-1} has at least one free top ray incident to the outer face of Γ_{k-1}; let $\tau_a(u_\ell)$ be the first of them in anti-clockwise order for u_ℓ and $\tau_c(u_r)$ be the first of them in clockwise order for u_r. Also, for each vertex u_q, with $q = 1, \ldots, \delta_i - 2$, let $\tau(u_q)$ be any of these rays. By Lemma 3, we can assume that these rays do not cross any edge in Γ_{k-1}.

Consider a horizontal line h_i above all vertices of Γ_{k-1}. Rays $\tau_a(u_\ell)$, $\tau(u_1), \ldots, \tau(u_{\delta_i-2})$, $\tau_c(u_r)$ cross h_i; however, the corresponding intersection points p_ℓ, $p_1, \ldots, p_{\delta_i-2}$, p_r may not appear in this left-to-right order along h_i; see Fig. 4a. To guarantee this property, we perform a sequence of stretchings of Γ_{k-1} by repediately applying Lemma 2. First, if p_ℓ

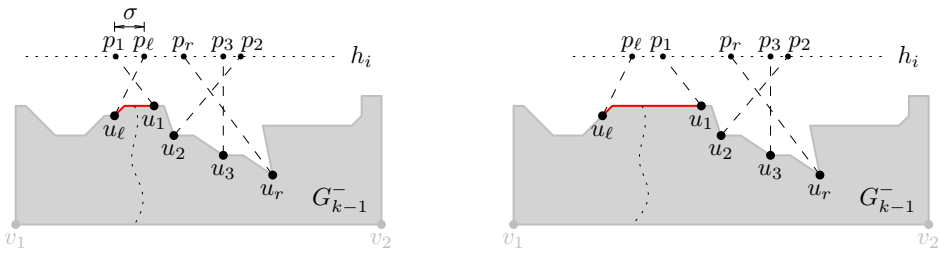


Figure 4 (a) Intersection points $p_\ell, p_1, \ldots, p_{\delta_i-2}, p_r$ appear in a wrong order along h_i. (b) Applying Lemma 2 to make p_ℓ be the leftmost intersection point.

is not the leftmost of these intersection points, let σ be the distance between p_ℓ and the leftmost intersection point. We apply Lemma 2 on any edge between u_ℓ and u_1 along C^-_{k-1} to stretch Γ_{k-1} so that all the vertices in the path of C^-_k from u_1 to v_2 are moved to the right by a quantity σ' slightly larger than σ. This implies that p_ℓ is not moved, while all the other intersection points are moved to the right by a quantity σ', and thus they all lie to the right of p_ℓ in the new drawing; see Fig. 4b. Analogously, we can move p_1 to the left of every other intersection point, except for p_ℓ, by applying Lemma 2 on any edge between u_1 and u_2 along C^-_{k-1}. Repeating this argument allows us to assume that in Γ_{k-1} all the intersection points appear in the correct left-to-right order along h_i.

We now describe the placement of v_i. Let $\beta_1(v_i), \ldots, \beta_{\delta_i-2}(v_i)$ be any set of $\delta_i - 2$ consecutive bottom rays of v_i; since S contains $\Delta - 1$ slopes and $\delta_i \leq \Delta$, v_i has enough bottom rays. If we place v_i above h_i, rays $\beta_1(v_i), \beta_2(v_i), \ldots, \beta_{\delta_i-2}(v_i)$ intersect h_i in this left-to-right order. Let $\rho_1, \ldots, \rho_{\delta_i-2}$ be the corresponding intersection points. The goal is to place v_i so that each ρ_q, with $q = 1, \ldots, \delta_i - 2$, coincides with p_q. To do so, we consider the line λ_1 through p_1 with the same slope as $\beta_1(v_i)$. Note that placing v_i on λ_1 above h_i results in ρ_1 to coincide with p_1. Also note that, while moving v_i upwards along λ_1, the distance $d(\rho_q, \rho_{q+1})$ between any two consecutive points ρ_q and ρ_{q+1}, with $q = 1, \ldots, \delta_i - 3$, increases.

We move v_i upwards along λ_1 so that $d(\rho_q, \rho_{q+1}) > d(p_1, p_{\delta_i-2})$, for each $q = 1, \ldots, \delta_i - 3$. This implies that all points $p_2, \ldots, p_{\delta_i-2}$ lie strictly between ρ_1 and ρ_2. Then, we apply Lemma 2 on any edge between u_1 and u_2 along C^-_{k-1} to stretch Γ_{k-1} so that all the vertices in the path of C^-_k from u_2 to v_2 are moved to the right by a quantity $d(p_2, \rho_2)$. In this way, u_1 is not moved and so p_1 still coincides with ρ_1; also, p_2 is moved to the right to coincide with ρ_2; finally, since $d(\rho_2, \rho_3) > d(p_1, p_{\delta_i-2}) > d(p_2, p_{\delta_i-2})$, all points $p_3, \ldots, p_{\delta_i-2}$ lie strictly between ρ_2 and ρ_3. By repeating this transformation for all points $p_3, \ldots, p_{\delta_i-2}$, if any, we guarantee that each ρ_q, with $q = 1, \ldots, \delta_i - 2$, coincides with p_q. We draw each edge (v_i, u_q), with $q = 1, \ldots, \delta_i - 2$, with a segment along $\tau(u_q)$ and the other one along $\beta_q(v_i)$.

It remains to draw edges (v_i, u_ℓ) and (v_i, u_r), which by Invariant I.3 must have a horizontal segment. After possibly applying Lemma 2 on any edge between u_ℓ and u_1 along C^-_{k-1} to stretch Γ_{k-1}, we can guarantee that $\tau_a(u_\ell)$ crosses the horizontal line through v_i to the left of v_i. Similarly, we can guarantee that $\tau_c(u_r)$ crosses the horizontal line through v_i to the right of v_i by applying Lemma 2 on any edge between u_{δ_i-2} and u_r. We draw (v_i, u_ℓ) with one segment along $\tau_a(u_\ell)$ and one along the horizontal line through v_i, and we draw (v_i, u_r) with one segment along $\tau_c(u_r)$ and one along the horizontal line through v_i. A drawing produced by this algorithm is illustrated in Fig. 3b.

The fact that Γ_k is a 1-bend planar drawing on S satisfying Invariant I.3–I.3 can be shown as for the case in which P_k is a chain. In particular, for Invariants I.3 and I.3, note that vertices $u_1, \ldots, u_{\delta_i-2}$ do not have neighbors in $G \setminus G_k$ and do not belong to C_k^-. Thus, they do not need to have any free top ray incident to the outer face of G_k^- and the edges connecting them to v_i do not need to have a horizontal segment. This concludes our description for the case in which P_k is a singleton, and yields the following theorem (a discussion about the time complexity can be found in [1]).

▶ **Theorem 4.** *For any $\Delta \geq 4$, there exists a equispaced universal set S of $\Delta - 1$ slopes for 1-bend planar drawings of triconnected planar graphs with maximum degree Δ. Also, for any such graph on n vertices, a 1-bend planar drawing on S can be computed in $O(n)$ time.*

4 Biconnected Planar Graphs

In this section we describe how to extend Theorem 4 to biconnected planar graphs, using the SPQR-tree data structure described in Section 2.

The idea is to traverse the SPQR-tree of the input biconnected planar graph G bottom-up and to construct for each visited node a drawing of its pertinent graph (except for its two poles) inside a rectangle, which we call *chip*. Besides being a 1-bend planar drawing on S, this drawing must have an additional property, namely that it is possible to increase its width while changing neither its height nor the slope of any edge-segment. We call this property *horizontal stretchability*. We give a formal definition of this drawing and describe how to compute it for each type of node of the SPQR-tree.

Let $\mathcal{T}$ be the SPQR-tree of G rooted at an arbitrary Q-node ρ. Let μ be a node of $\mathcal{T}$ with poles s_μ and t_μ. Let G_μ be the pertinent graph of μ. Let $\overline{G}_\mu$ be the graph obtained from G_μ as follows. First, remove edge (s_μ, t_μ), if it exists; then, subdivide each edge incident to s_μ (to t_μ) with a dummy vertex, which is a *pin of* s_μ (is a *pin of* t_μ); finally, remove s_μ and t_μ, and their incident edges. Note that, if μ is a Q-node other than the root ρ, then $\overline{G}_\mu$ is the empty graph. We denote by $\delta(s_\mu, \mu)$ and $\delta(t_\mu, \mu)$ the degree of s_μ and t_μ in G_μ, respectively; note that the number of pins of s_μ (of t_μ) is $\delta(s_\mu, \mu) - 1$ (is $\delta(t_\mu, \mu) - 1$), if edge (s_μ, t_μ) exists in G, otherwise it is $\delta(s_\mu, \mu)$ (it is $\delta(t_\mu, \mu)$).

We construct a 1-bend planar drawing of $\overline{G}_\mu$ on S inside an axis-aligned rectangle, called the *chip* of μ and denoted by $C(\mu)$, so that the following properties are satisfied (see Fig. 5a):

(P.1) All the pins of s_μ lie on the left side of $C(\mu)$ and all the pins of t_μ lie on its right side;
(P.2) for each pin, the unique edge incident to it is horizontal; and
(P.3) there exist pins on the bottom-left and on the bottom-right corners of $C(\mu)$.

We call *horizontally-stretchable* (or *stretchable*, for short) a drawing of $\overline{G}_\mu$ satisfying Properties P.4-P.4. Note that a stretchable drawing Γ remains stretchable after any uniform scaling, any translation, and any horizontal or vertical flip, since the horizontal slope is in S and the slopes are equispaced. On the other hand, it is generally not possible to perform any non-uniform scaling of Γ (in particular, a horizontal or a vertical scaling) without altering the slopes of some segments. However, we can simulate a horizontal scaling up of Γ by elongating the horizontal segments incident to all the pins lying on the same vertical side of the chip, thus obtaining a new stretchable drawing inside a new chip with the same height and a larger width. Conversely, a horizontal scaling down cannot always be simulated in this way.

Before giving the details of the algorithm, we describe a subroutine that we will often use to add the poles of a node μ to a stretchable drawing of $\overline{G}_\mu$ and draw the edges incident to them. The proof of the next lemma can be found in [1]; see also Fig. 5b.

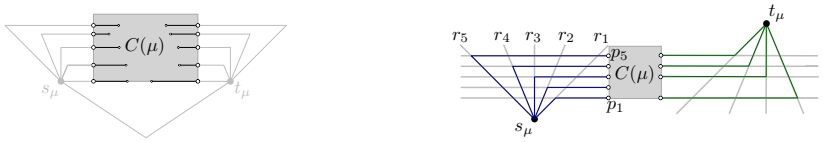


Figure 5 Illustrations (a) of a pin $C(\mu)$ and (b) for Lemma 5.

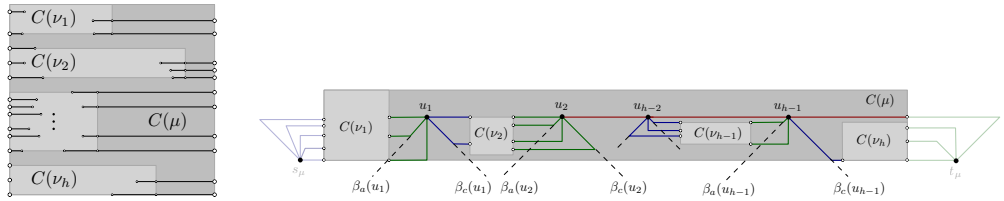


Figure 6 Illustrations for the cases in which μ is: (a) a P-node and (b) an S-node.

▶ **Lemma 5.** *Let $u \in \{s_\mu, t_\mu\}$ be a pole of a node $\mu \in \mathcal{T}$ and let $u_1, \ldots, u_q$ be q neighbors of u in $\overline{G}_\mu$. Consider a stretchable drawing of $\overline{G}_\mu$ inside a chip $C(\mu)$, whose pins $p_1, \ldots, p_q$ correspond to $u_1, \ldots, u_q$. Suppose that there exists a set of q consecutive free rays of u and that the elongation of the edge incident to each pin $p_1, \ldots, p_q$ intersects all these rays. Then, it is possible to draw edges $(u, u_1), \ldots, (u, u_q)$ with two straight-line segments whose slopes are in S, without introducing any crossing between two edges incident to u or between an edge incident to u and an edge of $\overline{G}_\mu$.*

We now describe the algorithm. At each step of the bottom-up traversal of $\mathcal{T}$, we consider a node $\mu \in \mathcal{T}$ with children $\nu_1, \ldots, \nu_h$, and we construct a stretchable drawing of $\overline{G}_\mu$ inside a chip $C(\mu)$ starting from the stretchable drawings of $\overline{G}_{\nu_1}, \ldots, \overline{G}_{\nu_h}$ inside chips $C(\nu_1), \ldots, C(\nu_h)$ that have been already constructed. We distinguish the different cases in which μ is a Q-, a P-, an S-, or an R-node.

Suppose that μ is a Q-node. If μ is not the root ρ of $\mathcal{T}$, we do not do anything, since $\overline{G}_\mu$ is the empty graph; the edge (s_μ, t_μ) of G corresponding to μ will be drawn when visiting either the parent ξ of μ, if ξ is not a P-node, or the parent of ξ.

Otherwise, $\mu = \rho$ and hence it has only one child ν_1. Since $\overline{G}_\mu$ coincides with $\overline{G}_{\nu_1}$, the stretchable drawing of $\overline{G}_{\nu_1}$ is also a stretchable drawing of $\overline{G}_\mu$. Vertices s_μ and t_μ, and their incident edges, will be added at the end of the traversal of $\mathcal{T}$.

Suppose that μ is a P-node; refer to Fig. 6a. We consider a chip $C(\mu)$ for μ whose height is larger than the sum of the heights of chips $C(\nu_1), \ldots, C(\nu_h)$ and whose width is larger than the one of any of $C(\nu_1), \ldots, C(\nu_h)$. Then, we place chips $C(\nu_1), \ldots, C(\nu_h)$ inside $C(\mu)$ so that no two chips overlap, their left sides lie along the left side of $C(\mu)$, and the bottom side of $C(\nu_h)$ lies along the bottom side of $C(\mu)$. Finally, we elongate the edges incident to all the pins on the right side of $C(\nu_1), \ldots, C(\nu_h)$ till reaching the right side of $C(\mu)$. The resulting drawing is stretchable since the drawings of $\overline{G}_{\nu_1}, \ldots, \overline{G}_{\nu_h}$ are. In particular, Property P.4 holds for $C(\mu)$ since it holds for $C(\nu_h)$.

Suppose that μ is an S-node; refer to Fig. 6b. Let $u_1, \ldots, u_{h-1}$ be the internal vertices of the path of virtual edges between s_μ and t_μ obtained by removing the virtual edge (s_μ, t_μ) from the skeleton of μ. We construct a stretchable drawing of $\overline{G}_\mu$ as follows.

First, we place vertices $u_1, \ldots, u_{h-1}$ in this order along a horizontal line l_μ. For $i = 1, \ldots, h-1$, let $\beta_a(u_i)$ and $\beta_c(u_i)$ be the first bottom rays of u_i in anti-clockwise and in clockwise order, respectively. To place each chip $C(\nu_i)$, with $i = 2, \ldots, h-1$, we first flip it vertically, so that it has pins on its top-left and top-right corners, by Property P.4. Then, after possibly scaling it down uniformly, we place it in such a way that its left side is to the right of u_{i-1}, its right side is to the left of u_i, it does not cross $\beta_c(u_{i-1})$ and $\beta_a(u_i)$, and either its top side lies on line l_μ (if edge $(u_{i-1}, u_i) \notin G$; see $C(\nu_2)$ in Fig. 6b), or it lies slightly below it (otherwise; see $C(\nu_{h-1})$ in Fig. 6b).

Then, we place $C(\nu_1)$ and $C(\nu_h)$, after possibly scaling them up uniformly, in such a way that:

- **(i)** Chip $C(\nu_1)$ lies to the left of u_1 and does not cross $\beta_a(u_1)$. Also, if $(s_\mu, u_1) \in G$, then $C(\nu_1)$ lies entirely below l_μ; otherwise, as in Fig. 6b, the topmost pin on its right side has the same y-coordinate as u_1.
- **(ii)** Chip $C(\nu_h)$ lies to the right of u_h and does not cross $\beta_c(u_h)$. Also, if $(u_h, t_\mu) \in G$, as in Fig. 6b, then $C(\nu_h)$ lies entirely below u_h; otherwise, the topmost pin on its left side has the same y-coordinate as u_h.
- **(iii)** The bottom sides of $C(\nu_1)$ and of $C(\nu_h)$ have the same y-coordinate, which is smaller than the one of the bottom side of any other chip $C(\nu_2), \ldots, C(\nu_{h-1})$.

We now draw all the edges incident to each vertex u_i, with $i = 1, \ldots, h-1$. If edge $(u_{i-1}, u_i) \in G$, then it can be drawn as a horizontal segment, by construction. Otherwise, u_i can be connected with a horizontal segment to its neighbor in $\overline{G}_{\nu_i}$ corresponding to the topmost pin on the right side of $C(\nu_i)$. In both cases, one of these edges is attached at a horizontal ray of u_i. Analogously, one of the edges connecting u_i to its neighbors in $\overline{G}_{\nu_{i+1}} \cup \{u_{i+1}\}$ is attached at the other horizontal ray of u_i. Thus, it is possible to draw the remaining $\delta(u_i, \nu_i) + \delta(u_i, \nu_{i+1}) - 2 \leq \Delta - 2$ edges incident to u_i by attaching them at the $\Delta - 2$ bottom rays of u_i, by applying Lemma 5. In fact, since $C(\nu_i)$ and $C(\nu_{i+1})$ lie to the left and to the right of u_i, respectively, and do not cross $\beta_c(u_i)$ and $\beta_a(u_i)$, the elongations of the edges incident to the pins of u_i in $C(\nu_i)$ and in $C(\nu_{i+1})$ corresponding to these edges intersect all the bottom rays of u_i, hence satisfying the preconditions to apply the lemma.

Finally, we construct chip $C(\mu)$ as the smallest rectangle enclosing all the current drawing. Note that the left side of $C(\mu)$ contains the left side of $C(\nu_1)$, while the right side of $C(\mu)$ contains the right side of $C(\nu_h)$. Thus, all the pins of s_μ, possibly except for the one corresponding to edge (s_μ, u_1), lie on the left side of $C(\mu)$. Also, if (s_μ, u_1) exists, we can add the corresponding pin since, by construction, $C(\nu_1)$ lies entirely below u_1. The same discussion applies for the pins of t_μ. This proves that the constructed drawing satisfies properties P.4 and P.4. To see that it also satisfies P.4, note that the bottom side of $C(\mu)$ contains the bottom sides of $C(\nu_1)$ and of $C(\nu_h)$, by construction, which have a pin on both corners, by Property P.4. Thus, the constructed drawing of $\overline{G}_\mu$ is stretchable.

Suppose that μ is an R-node. We compute a stretchable drawing of $\overline{G}_\mu$ as follows. First, we compute a 1-bend planar drawing on S of the whole pertinent G_μ of μ, including its poles s_μ and t_μ; then, we remove the poles of μ and their incident edges, we define chip $C(\mu)$, and we place the pins on its two vertical sides so to satisfy Properties P.4–P.4.

Since the skeleton G_μ^{skel} of μ is triconnected, we use the algorithm described in Section 3 as a main tool for drawing G_μ, with suitable modifications to take into account the fact

that each virtual edge (u, v) of G_μ^{skel} actually corresponds to a whole subgraph, namely the pertinent graph G_ν of the child ν of μ with poles $s_\nu = u$ and $t_\nu = v$. Thus, when the virtual edge (u, v) is considered, we have to add the stretchable drawing of $\overline{G}_\nu$ inside a chip $C(\nu)$; this enforces additional requirements for our drawing algorithm.

The first obvious requirement is that (u, v) will occupy $\delta(u, \nu)$ consecutive rays of u and $\delta(v, \nu)$ consecutive rays of v, and not just a single ray for each of them, as in the triconnected case. However, reserving the correct amount of rays of u and v is not always sufficient to add $C(\nu)$ and to draw the edges between u, v, and vertices in $\overline{G}_\nu$. In fact, we need to ensure that there exists a placement for $C(\nu)$ such that the elongations of the edges incident to its pins intersect all the reserved rays of the poles u and v of ν, hence satisfying the preconditions to apply Lemma 5. In a high-level description, for the virtual edges that would be drawn with a horizontal segment in the triconnected case (all the edges of a chain, and the first and last edges of a singleton), this can be done by using a construction similar to the one of the case in which μ is an S-node. For the edges that do not have any horizontal segment (the internal edges of a singleton), instead, we need a more complicated construction.

The algorithm is again based on considering the vertices of $H = G_\mu$ according to a canonical order $\Pi = (P_0, \ldots, P_m)$ of H, in which $v_1 = s_\mu$ and $v_2 = t_\mu$, and on constructing a 1-bend planar drawing of H_k^- on S satisfying a modified version of Invariants I.3–I.3.

(M.1) No part of the drawing lies below vertices v_1 and v_2, which have the same y-coordinate.

(M.2) For every virtual edge (w, z) on C_k^-, if (w, z) belongs to H then it has a horizontal segment; also, the edge-segments corresponding to edges incident to the pins of the chip of the child of μ corresponding to (w, z) are horizontal.

(M.3) Each vertex v on C_k^- has at least as many free top rays incident to the outer face of H_k^- as the number of its neighbors in H that have not been drawn yet.

We note that Invariant M.4 is identical to Invariant I.3, while Invariant M.4 is the natural extension of Invariant I.3 to take into account our previous observation. Finally, Invariant M.4 corresponds to Invariant I.3, as it ensures that we can still apply Lemma 2 and Lemma 3.

At the first step, we draw $P_0 = \{v_1, v_2\}$ and $P_1 = \{v_3, \ldots, v_j\}$. Consider the path of virtual edges $(v_1, v_3), (v_3, v_4), \ldots, (v_j, v_2)$. Let $\nu_{1,3}, \nu_{3,4}, \ldots, \nu_{j,2}$ be the corresponding children of μ, and let $C(\nu_{1,3}), C(\nu_{3,4}), \ldots, C(\nu_{j,2})$ be their chips. We consider this path as the skeleton of an S-node with poles v_1 and v_2, and we we apply the same algorithm as in the case in which μ is an S-node to draw the subgraph composed of $v_3, \ldots, v_j$ and of chips $C(\nu_{1,3}), C(\nu_{3,4}), \ldots, C(\nu_{j,2})$ inside a larger chip, denoted by $C(v_1, v_2)$. Note that, by construction, $C(v_1, v_2)$ has pins on its bottom-left and on its bottom-right corners. We then place v_1 and v_2 with the same y-coordinate as the bottom side of $C(v_1, v_2)$, with v_1 to the left and v_2 to the right of $C(v_1, v_2)$. We draw one of the edges incident to v_1 horizontal, and the remaining $\delta(v_1, \nu_{1,3}) - 1$ by applying Lemma 5, and the same for v_2. Invariants M.4 and M.4 are satisfied by construction. For Invariant M.4, note that $v_3, \ldots, v_j$ have all their $\Delta - 2$ top rays free, by construction, and at least two of their neighbors have already been drawn. Also, v_1 and v_2 have consumed only $\delta(v_1, \nu_{1,3}) - 1$ and $\delta(v_2, \nu_{j,2}) - 1$ top rays, respectively. Since edge (v_1, v_2) does not belong to H_k^- (but belongs to H), v_1 and v_2 satisfy Invariant M.4.

We now describe how to add path P_k, for some $k > 1$, to the current drawing Γ_{k-1} in the two cases in which P_k is a chain or a singleton.

Suppose that P_k is a chain $\{v_i, v_{i+1}, \ldots, v_j\}$; let u_ℓ and u_r be the neighbors of v_i and v_j in C_k^-. Let $\nu_\ell, \nu_i, \ldots, \nu_{j-1}, \nu_r$ be the children of μ corresponding to virtual edges $(u_\ell, v_i), (v_i, v_{i+1}), \ldots, (v_{j-1}, v_j), (v_j, v_r)$, and $C(\nu_\ell), C(\nu_i), \ldots, C(\nu_{j-1}), C(\nu_r)$ be their chips.

We define rays $\tau_a(u_\ell)$ and $\tau_c(u_r)$, and the horizontal segment h between them, as in the triconnected case. Due to Lemma 3, we can assume that $\tau_a(u_\ell)$ and the $\delta(u_\ell, \nu_\ell) - 1$ top

rays of u_ℓ following it in anti-clockwise order do not cross any edge of Γ_{k-1}, and the same for $\tau_c(u_r)$ and the $\delta(u_r, \nu_r) - 1$ top rays of u_r following it in clockwise order. Note that, by Invariant M.4, all these rays are free. As in the step in which we considered P_0 and P_1 of Π, we use the algorithm for the case in which μ is an S-node to construct a drawing of the subgraph composed of $v_i, \dots, v_j$ and of chips $C(\nu_\ell), C(\nu_i), \dots, C(\nu_{j-1}), C(\nu_r)$ inside a larger chip $C(u_\ell, u_r)$, which has pins on its bottom-left and on its bottom-right corners. We then place $C(u_\ell, u_r)$ so that its bottom side lies on h and it does not cross $\tau_a(u_\ell)$ and $\tau_c(u_r)$, after possibly scaling it down uniformly. Finally, we draw the $\delta(u_\ell, \nu_\ell)$ edges between u_ℓ and its neighbors in $\overline{G}_{\nu_\ell} \cup \{v_i\}$, and the $\delta(u_r, \nu_r)$ edges between u_r and its neighbors in $\overline{G}_{\nu_r} \cup \{v_j\}$, by applying Lemma 5, whose preconditions are satisfied. The fact that the constructed drawing satisfies the three invariants can be proved as in the previous case.

Suppose that P_k is a singleton $\{v_i\}$ of degree $\delta_i \leq \Delta$ in H_k^-. As in the triconnected case, we shall assume that $\delta_i \geq 3$. Let $u_\ell, u_1, \dots, u_{\delta_i-2}, u_r$ be the neighbors of v_i as they appear along C_{k-1}^-, let $\nu_\ell, \nu_1, \dots, \nu_{\delta_i-2}, \nu_r$ be the children of μ corresponding to the virtual edges connecting v_i with these vertices, and let $C(\nu_\ell), C(\nu_1), \dots, C(\nu_{\delta_i-2}), C(\nu_r)$ be their chips.

For each $q = 1, \dots, \delta_i - 2$, we select any set T_q of consecutive $\delta(u_q, \nu_q)$ free top rays of u_q incident to the outer face and a set B_q of consecutive $\delta(v_i, \nu_q)$ bottom rays of v_i; see Fig. 7b. Sets $B_1, \dots, B_{\delta_i-2}$ are selected so that the rays in B_q precede those in B_{q+1} in anti-clockwise order. Since $\delta(v_i, \nu_\ell) + \delta(v_i, \nu_r) \geq 2$, we have that v_i has enough bottom rays for sets $B_1, \dots, B_{\delta_i-2}$. Sets T_ℓ and T_r contain the first $\delta(u_\ell, \nu_\ell)$ free top rays of u_ℓ in anti-clockwise order and of the first $\delta(u_r, \nu_r)$ free top rays of u_r in clockwise order, respectively.

We then select a horizontal line h_i lying above every vertex in Γ_{k-1}. As in the algorithm described in Section 3, after possibly applying $O(\Delta)$ times Lemma 2, we can assume that all the rays in sets $T_\ell, T_1, \dots, T_{\delta_i-2}, T_r$ intersect h_i in the correct order. Namely, when moving along h_i from left to right, we encounter all the intersections with the rays in T_ℓ, then all those with the rays in T_1, and so on. This property is already guaranteed for the rays in $B_1, \dots, B_{\delta_i-2}$. This defines two total left-to-right orders $\mathcal{O}_T$ and $\mathcal{O}_B$ of the intersection points of $T_\ell, T_1, \dots, T_{\delta_i-2}, T_r$ and of $B_1, \dots, B_{\delta_i-2}$ along h_i, respectively. For simplicity, we extend these orders to the rays in $T_\ell, T_1, \dots, T_{\delta_i-2}, T_r$ and in $B_1, \dots, B_{\delta_i-2}$, respectively.

Our goal is to merge the two sets of intersection points, while respecting $\mathcal{O}_T$ and $\mathcal{O}_B$, in such a way that the following condition holds for each $q = 1, \dots, \delta_i - 2$. If edge (v_i, u_q) belongs to H, then the first intersection point of T_q in $\mathcal{O}_T$ coincides with the first intersection point of B_q in $\mathcal{O}_B$, and the second intersection point of T_q in $\mathcal{O}_T$ is to the right of the last intersection point of B_q in $\mathcal{O}_B$; see T_1 and B_1 in Fig. 7b. Otherwise, $(v_i, u_q) \notin H$ and the first intersection point of T_q in $\mathcal{O}_T$ is to the right of the last intersection point of B_q in $\mathcal{O}_B$; see T_3 and B_3 in Fig. 7b. In both cases, all the intersection points of T_q and B_q are to the left of all the intersection points of T_{q+1} and B_{q+1}.

To obtain this goal, we perform a procedure analogous to the one described in Section 3 to make points $p_1, \dots, p_{\delta_i-2}$ coincide with points $\rho_1, \dots, \rho_{\delta_i-2}$. Namely, we consider a line λ_1, whose slope is the one of the first ray in B_1, that starts at the first intersection point of T_1 in $\mathcal{O}_T$, if edge (v_1, u_1) belongs to H, or at any point between the last intersection point of T_1 and the first intersection point of T_2 in $\mathcal{O}_T$, otherwise. Then, we place v_i along λ_1, far enough from h_i so that the distance between any two consecutive intersection points in $\mathcal{O}_B$ is larger than the distance between the first and the last intersection points in $\mathcal{O}_T$; see Fig. 7a. Finally, we apply Lemma 2 at most $\delta_i - 3$ times to move the intersection points of sets $T_2, \dots, T_{\delta_i-2}$, one by one, in the their correct positions; see Fig. 7b.

Once the required ordering of intersection points along h_i has been obtained, we consider another horizontal line h_i' lying above h_i and close enough to it so that its intersections with

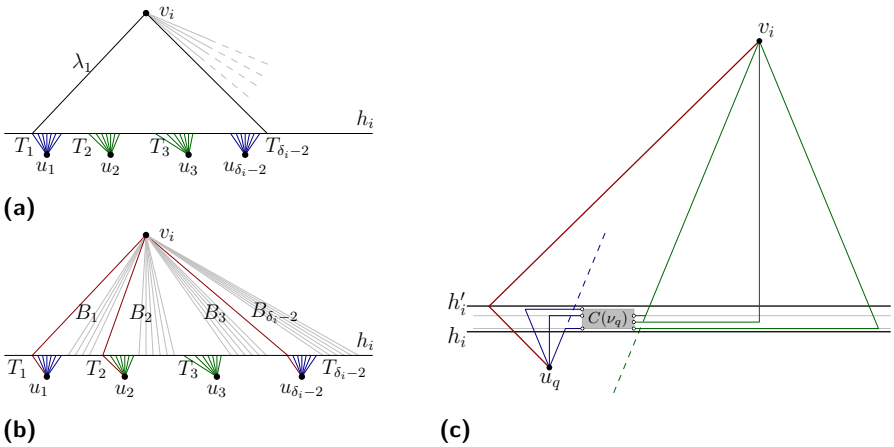


Figure 7 Ilustrations for placing singleton v_i in the case of an R-node.

the rays in $T_\ell, T_1, \ldots, T_{\delta_i-2}, T_r$ and $B_1, \ldots, B_{\delta_i-2}$ appear along it in the same order as along h_i. We place each chip $C(\nu_q)$, with $q = 1, \ldots, \delta_i - 2$, after possibly scaling it down uniformly, in the interior of the region delimited by these two lines, by the last ray in T_q, and by a ray in B_q (either the second or the first, depending on whether $(v_i, u_q) \in H$ or not); see Fig. 7c.

We draw the edges incident to v_i and u_q, for each $q = 1, \ldots, \delta_i - 2$, as follows. If (v_i, u_q) belongs to H, we draw it with one segment along the first ray in T_q and one along the first ray in B_q (red edge in Fig. 7c). For the other edges we apply Lemma 5 twice, whose preconditions are satisfied due to the placement of $C(\nu_q)$ (blue and green edges in Fig. 7c).

We conclude by drawing the edges connecting v_i, u_ℓ, and vertices in $\overline{G}_{\nu_\ell}$; the edges connecting v_i, u_r and vertices in $\overline{G}_{\nu_r}$ are drawn symmetrically. First, after possibly applying Lemma 2, we assume that the last ray of T_ℓ intersects the horizontal line through v_i to the left of v_i, at a point p_i. After possibly scaling $C(\nu_\ell)$ down uniformly, we place it so that its left side is to the right of p_i, its right side is to the left of v_i, it does not cross the first top ray of v_i in clockwise order, and its bottom side is horizontal and lies either above the horizontal line through v_i, if $(u_\ell, v_i) \in H$, or along it, otherwise. Then, we draw (u_ℓ, v_i), if it belongs to H, with one segment along the last ray of T_ℓ and the other one along the horizontal line through v_i. Otherwise, $(u_\ell, v_i) \notin H$ and we can draw one of the edges incident to v_i with a horizontal segment. We finally apply Lemma 5 twice, to draw the edges from u_ℓ to its neighbors in $\overline{G}_{\nu_\ell}$, and from v_i to its other neighbors in $\overline{G}_{\nu_\ell}$. The fact that the constructed drawing satisfies Invariants M.4–M.4 can be proved as in the triconnected case.

Once the last path P_m of Π has been added, we have a drawing Γ_μ of $H = G_\mu$ satisfying Invariants M.4–M.4. We construct chip $C(\mu)$ as the smallest axis-aligned rectangle enclosing Γ_μ. By Invariant M.4, vertices v_1 and v_2 lie on the bottom side of $C(\mu)$. Also, by Invariant M.4, all the edges incident to v_1 or to v_2 have a horizontal segment. Thus, it is possible to obtain a drawing of $\overline{G}_\mu$ inside $C(\mu)$ by removing v_1 and v_2 (and their incident edges) from Γ_μ, by elongating the horizontal segments incident to them till reaching the vertical sides of $C(\mu)$, and by placing pins at their ends. The fact that this drawing satisfies Properties P.4–P.4 follows from the observation that v_1 and v_2 were on the bottom side of $C(\mu)$. This concludes the case in which μ is an R-node.

Once we have visited the root ρ of $\mathcal{T}$, we have a stretchable drawing of $\overline{G}_\rho$ inside a chip $C(\rho)$, which we extend to a drawing of G as follows. Refer to Fig. 5a. We place s_ρ and t_ρ at the same y-coordinate as the bottom side of $C(\rho)$, one to its left and one to its right, so

that $C(\rho)$ does not cross any of the rays of s_ρ and of t_ρ. Then, we draw edge (s_ρ, t_ρ) with one segment along the first bottom ray in clockwise order of s_ρ and the other one along the first bottom ray in anti-clockwise order of t_ρ. Also, we draw the edges connecting s_ρ and t_ρ to the vertices corresponding to the lowest pins on the two vertical sides of $C(\rho)$ as horizontal segments. Finally, we draw all the remaining edges incident to s_ρ and t_ρ by applying Lemma 5 twice. The following theorem summarizes the discussion in this section (a discussion about the time complexity can be found in [1]).

▶ **Theorem 6.** *For any $\Delta \geq 4$, there exists a equispaced universal set S of $\Delta - 1$ slopes for 1-bend planar drawings of biconnected planar graphs with maximum degree Δ. Also, for any such graph on n vertices, a 1-bend planar drawing on S can be computed in $O(n)$ time.*

5 Conclusions and Open Problems

In this paper, we improved the upper bound on the 1-bend planar slope number from $\frac{3}{2}(\Delta-1)$ to $\Delta - 1$, for $\Delta \geq 4$. We mention two side-results of our work. Since the angular resolution of our drawings is at least $\frac{\pi}{\Delta-1}$, at the cost of increased drawing area our main result also improves the best-known upper bound of $\frac{\pi}{4\Delta}$ on the angular resolution of 1-bend poly-line planar drawings by Duncan and Kobourov [17]. For $\Delta = 4$, it also guarantees that planar graphs with maximum degree 4 admit 1-bend planar drawings on a set of slopes $\{0, \frac{\pi}{3}, \frac{2\pi}{3}\}$, while previously it was known that such graphs can be embedded with one bend per edge on a set of slopes $\{0, \frac{\pi}{4}, \frac{\pi}{2}, \frac{3\pi}{4}\}$ [3] and with two bends per edge on a set of slopes $\{0, \pi\}$ [5].

Our work raises several open problems.

- **(i)** Reduce the gap between the $\frac{3}{4}(\Delta - 1)$ lower bound and the $\Delta - 1$ upper bound.
- **(ii)** Our algorithm may produce drawings with super-polynomial area. Is this unavoidable for 1-bend planar drawings with few slopes and good angular resolution?
- **(iii)** Study the straight-line case (e.g., when $\Delta = 4$). Note that stretching might be difficult in this setting.
- **(iv)** We proved that a set of $\Delta - 1$ equispaced slopes is universal for 1-bend planar drawings. Is *every* set of $\Delta - 1$ slopes universal? Note that for $\Delta \leq 4$ a positive answer descends from our work and from a result by Dujmovic et al. [15], who proved that any planar graph that can be drawn on a particular set of three slopes can also be drawn on any set of three slopes. If the answer to this question is negative for $\Delta > 4$, what is the minimum value $s(\Delta)$ such that every set of $s(\Delta)$ slopes is universal?

Acknowledgements. This work started at the 19^{th} *Korean Workshop on Computational Geometry*. We wish to thank the organizers and the participants for creating a pleasant and stimulating atmosphere and in particular Fabian Lipp and Boris Klemz for useful discussions.

References

1 Patrizio Angelini, Michael A. Bekos, Giuseppe Liotta, and Fabrizio Montecchiani. Universal slope sets for 1-bend planar drawings. *CoRR*, 1703.04283, 2017.

2 János Barát, Jirí Matoušek, and David R. Wood. Bounded-degree graphs have arbitrarily large geometric thickness. *Electr. J. Comb.*, 13(1), 2006.

3 Michael A. Bekos, Martin Gronemann, Michael Kaufmann, and Robert Krug. Planar octilinear drawings with one bend per edge. *J. Graph Algorithms Appl.*, 19(2):657–680, 2015. doi:10.7155/jgaa.00369.

4 Michael A. Bekos, Michael Kaufmann, and Robert Krug. On the total number of bends for planar octilinear drawings. In *LATIN*, volume 9644 of *LNCS*, pages 152–163. Springer, 2016. doi:10.1007/978-3-662-49529-2_12.

5 Therese C. Biedl and Goos Kant. A better heuristic for orthogonal graph drawings. *Comput. Geom.*, 9(3):159–180, 1998. doi:10.1016/S0925-7721(97)00026-6.

6 Thomas Bläsius, Marcus Krug, Ignaz Rutter, and Dorothea Wagner. Orthogonal graph drawing with flexibility constraints. *Algorithmica*, 68(4):859–885, 2014. doi:10.1007/s00453-012-9705-8.

7 Thomas Bläsius, Sebastian Lehmann, and Ignaz Rutter. Orthogonal graph drawing with inflexible edges. *Comput. Geom.*, 55:26–40, 2016.

8 Hans L. Bodlaender and Gerard Tel. A note on rectilinearity and angular resolution. *J. Graph Algorithms Appl.*, 8:89–94, 2004.

9 Nicolas Bonichon, Bertrand Le Saëc, and Mohamed Mosbah. Optimal area algorithm for planar polyline drawings. In *WG*, volume 2573 of *LNCS*, pages 35–46. Springer, 2002.

10 Hubert de Fraysseix, Patrice Ossona de Mendez, and Pierre Rosenstiehl. On triangle contact graphs. *Combinatorics, Probability & Computing*, 3:233–246, 1994. doi:10.1017/S0963548300001139.

11 Hubert de Fraysseix, János Pach, and Richard Pollack. How to draw a planar graph on a grid. *Combinatorica*, 10(1):41–51, 1990. doi:10.1007/BF02122694.

12 Giuseppe Di Battista, Peter Eades, Roberto Tamassia, and Ioannis G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice-Hall, 1999.

13 Emilio Di Giacomo, Giuseppe Liotta, and Fabrizio Montecchiani. The planar slope number of subcubic graphs. In *LATIN*, volume 8392 of *LNCS*, pages 132–143. Springer, 2014. doi:10.1007/978-3-642-54423-1_12.

14 Emilio Di Giacomo, Giuseppe Liotta, and Fabrizio Montecchiani. Drawing outer 1-planar graphs with few slopes. *J. Graph Algorithms Appl.*, 19(2):707–741, 2015. doi:10.7155/jgaa.00376.

15 Vida Dujmović, David Eppstein, Matthew Suderman, and David R. Wood. Drawings of planar graphs with few slopes and segments. *Comput. Geom.*, 38(3):194–212, 2007.

16 Christian A. Duncan, David Eppstein, Michael T. Goodrich, Stephen G. Kobourov, and Martin Nöllenburg. Drawing trees with perfect angular resolution and polynomial area. *Discrete & Computational Geometry*, 49(2):157–182, 2013.

17 Christian A. Duncan and Stephen G. Kobourov. Polar coordinate drawing of planar graphs with good angular resolution. *J. Graph Algorithms Appl.*, 7(4):311–333, 2003.

18 Stephane Durocher and Debajyoti Mondal. Trade-offs in planar polyline drawings. In *GD*, volume 8871 of *LNCS*, pages 306–318. Springer, 2014.

19 Michael Formann, Torben Hagerup, James Haralambides, Michael Kaufmann, Frank Thomson Leighton, Antonios Symvonis, Emo Welzl, and Gerhard J. Woeginger. Drawing graphs in the plane with high resolution. *SIAM J. Comput.*, 22(5):1035–1052, 1993. doi:10.1137/0222063.

20 Ashim Garg and Roberto Tamassia. Planar drawings and angular resolution: Algorithms and bounds (extended abstract). In *ESA*, volume 855 of *LNCS*, pages 12–23. Springer, 1994.

21 Ashim Garg and Roberto Tamassia. On the computational complexity of upward and rectilinear planarity testing. *SIAM J. Comput.*, 31(2):601–625, 2001. doi:10.1137/S0097539794277123.

22 Carsten Gutwenger and Petra Mutzel. Planar polyline drawings with good angular resolution. In *GD*, volume 1547 of *LNCS*, pages 167–182. Springer, 1998.

23 Carsten Gutwenger and Petra Mutzel. A linear time implementation of SPQR-trees. In *GD*, volume 1984 of *LNCS*, pages 77–90. Springer, 2000. doi:10.1007/3-540-44541-2_8.

24 Udo Hoffmann. On the complexity of the planar slope number problem. *J. Graph Algorithms Appl.*, 21(2):183–193, 2017.

25 Vít Jelínek, Eva Jelínková, Jan Kratochvíl, Bernard Lidický, Marek Tesar, and Tomás Vyskocil. The planar slope number of planar partial 3-trees of bounded degree. *Graphs and Comb.*, 29(4):981–1005, 2013. doi:10.1007/s00373-012-1157-z.

26 Michael Jünger and Petra Mutzel, editors. *Graph Drawing Software*. Springer, 2004.

27 Goos Kant. Drawing planar graphs using the lmc-ordering (extended abstract). In *FOCS*, pages 101–110. IEEE Computer Society, 1992. doi:10.1109/SFCS.1992.267814.

28 Goos Kant. Hexagonal grid drawings. In *WG*, volume 657 of *LNCS*, pages 263–276. Springer, 1992. doi:10.1007/3-540-56402-0_53.

29 Goos Kant. Drawing planar graphs using the canonical ordering. *Algorithmica*, 16(1):4–32, 1996. doi:10.1007/BF02086606.

30 Balázs Keszegh, János Pach, and Dömötör Pálvölgyi. Drawing planar graphs of bounded degree with few slopes. *SIAM J. Discrete Math.*, 27(2):1171–1183, 2013. doi:10.1137/100815001.

31 Kolja Knauer and Bartosz Walczak. Graph drawings with one bend and few slopes. In *LATIN*, volume 9644 of *LNCS*, pages 549–561. Springer, 2016. doi:10.1007/978-3-662-49529-2_41.

32 Kolja B. Knauer, Piotr Micek, and Bartosz Walczak. Outerplanar graph drawings with few slopes. *Comput. Geom.*, 47(5):614–624, 2014. doi:10.1016/j.comgeo.2014.01.003.

33 William Lenhart, Giuseppe Liotta, Debajyoti Mondal, and Rahnuma Islam Nishat. Planar and plane slope number of partial 2-trees. In *GD*, volume 8242 of *LNCS*, pages 412–423. Springer, 2013. doi:10.1007/978-3-319-03841-4_36.

34 Yanpei Liu, Aurora Morgana, and Bruno Simeone. A linear algorithm for 2-bend embeddings of planar graphs in the two-dimensional grid. *Discrete Applied Mathematics*, 81(1-3):69–91, 1998. doi:10.1016/S0166-218X(97)00076-0.

35 Padmini Mukkamala and Dömötör Pálvölgyi. Drawing cubic graphs with the four basic slopes. In *GD*, volume 7034 of *LNCS*, pages 254–265. Springer, 2011. doi:10.1007/978-3-642-25878-7_25.

36 Martin Nöllenburg. Automated drawings of metro maps. Technical Report 2005-25, Fakultät für Informatik, Universität Karlsruhe, 2005.

37 Martin Nöllenburg and Alexander Wolff. Drawing and labeling high-quality metro maps by mixed-integer programming. *IEEE Trans. Vis. Comput. Graph.*, 17(5):626–641, 2011. doi:10.1109/TVCG.2010.81.

38 Jonathan M. Stott, Peter Rodgers, Juan Carlos Martinez-Ovando, and Stephen G. Walker. Automatic metro map layout using multicriteria optimization. *IEEE Trans. Vis. Comput. Graph.*, 17(1):101–114, 2011. doi:10.1109/TVCG.2010.24.

39 Roberto Tamassia. On embedding a graph in the grid with the minimum number of bends. *SIAM J. Comput.*, 16(3):421–444, 1987. doi:10.1137/0216030.

40 Roberto Tamassia, editor. *Handbook on Graph Drawing and Visualization*. CRC Press, 2013.

41 Greg A. Wade and Jiang-Hsing Chu. Drawability of complete graphs using a minimal slope set. *Comput. J.*, 37(2):139–142, 1994. doi:10.1093/comjnl/37.2.139.

Near-Optimal ε-Kernel Construction and Related Problems*

Sunil Arya[†1], **Guilherme D. da Fonseca**[2], **and David M. Mount**[‡3]

1 **Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong**
arya@cse.ust.hk

2 **Université Clermont Auvergne and LIMOS, Clermont-Ferrand, France**
fonseca@isima.fr

3 **Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, MD, USA**
mount@cs.umd.edu

Abstract

The computation of (i) ε-kernels, (ii) approximate diameter, and (iii) approximate bichromatic closest pair are fundamental problems in geometric approximation. In this paper, we describe new algorithms that offer significant improvements to their running times. In each case the input is a set of n points in $\mathbb{R}^d$ for a constant dimension $d \geq 3$ and an approximation parameter $\varepsilon > 0$. We reduce the respective running times

(i) from $O((n + 1/\varepsilon^{d-2}) \log \frac{1}{\varepsilon})$ to $O(n \log \frac{1}{\varepsilon} + 1/\varepsilon^{(d-1)/2+\alpha})$,
(ii) from $O((n + 1/\varepsilon^{d-2}) \log \frac{1}{\varepsilon})$ to $O(n \log \frac{1}{\varepsilon} + 1/\varepsilon^{(d-1)/2+\alpha})$, and
(iii) from $O(n/\varepsilon^{d/3})$ to $O(n/\varepsilon^{d/4+\alpha})$,

for an arbitrarily small constant $\alpha > 0$. Result (i) is nearly optimal since the size of the output ε-kernel is $\Theta(1/\varepsilon^{(d-1)/2})$ in the worst case.

These results are all based on an efficient decomposition of a convex body using a hierarchy of Macbeath regions and contrast with previous solutions, which decompose space using quadtrees and grids. By further application of these techniques, we also show that it is possible to obtain near-optimal preprocessing times for the most efficient data structures to approximately answer queries for (iv) nearest-neighbor searching, (v) directional width, and (vi) polytope membership.

1998 ACM Subject Classification F.2.2 Geometrical problems and computations

Keywords and phrases Approximation, diameter, kernel, coreset, nearest neighbor, polytope membership, bichromatic closest pair, Macbeath regions

Digital Object Identifier 10.4230/LIPIcs.SoCG.2017.10

1 Introduction

In this paper we present new faster algorithms to several fundamental geometric approximation problems involving point sets in d-dimensional space. In particular, we present approximation algorithms for ε-kernels, diameter, bichromatic closest pair, and the minimum bottleneck spanning tree. Our results arise from a recently developed shape-sensitive approach to approximating convex bodies, which is based on the classical concept of Macbeath

* A full version of the paper is available at http://arxiv.org/abs/1703.10868.
† Research supported by the Research Grants Council of Hong Kong, China under project number 610012.
‡ Research supported by NSF grant CCF–1618866.

regions. This approach has been applied to computing area-sensitive bounds for polytope approximation [5], polytope approximations with low combinatorial complexity [6], answering approximate polytope-membership queries [7], and approximate nearest-neighbor searching [7]. The results of [7] demonstrate the existence of data structures for these query problems but did not discuss preprocessing in detail. We complete the story by presenting efficient algorithms for building data structures for three related queries: approximate polytope membership, approximate directional width, and approximate nearest-neighbors.

Throughout, we assume that the dimension d is a constant. Our running times will often involve expressions of the form $1/\varepsilon^{\alpha}$. In such cases, $\alpha > 0$ is constant that can be made arbitrarily small. The approximation parameter ε is treated as an asymptotic variable that approaches 0. We assume throughout that $\varepsilon < 1$, which guarantees that $\log \frac{1}{\varepsilon} > 0$.

In Section 1.1, we present our results for ε-kernels, diameter, bichromatic closest pair, and minimum bottleneck tree. In Section 1.2, we present our results for the data structure problems. In Section 1.3, we give an overview of the techniques used.

Concurrently and independently, Timothy Chan has reported complexity bounds that are very similar to our results [18]. Remarkably, the computational techniques seem to be very different, based on Chebyshev polynomials.

1.1 Static Results

Kernel. Given a set S of n points in $\mathbb{R}^d$ and an approximation parameter $\varepsilon > 0$, an *ε-coreset* is an (ideally small) subset of S that approximates some measure over S (see [2] for a survey). Given a nonzero vector $v \in \mathbb{R}^d$, the *directional width* of S in direction v, $\text{width}_v(S)$ is the minimum distance between two hyperplanes that enclose S and are orthogonal to v. A *coreset for the directional width* (also known as an *ε-kernel* and as a *coreset for the extent measure*) is a subset $Q \subseteq S$ such that $\text{width}_v(Q) \geq (1-\varepsilon)\,\text{width}_v(S)$, for all $v \in \mathbb{R}^d$. Kernels are among the most fundamental constructions in geometric approximation, playing a role similar to that of convex hulls in exact computations. Kernels have been used to obtain approximation algorithms to several problems such as diameter, minimum width, convex hull volume, minimum enclosing cylinder, minimum enclosing annulus, and minimum-width cylindrical shell [1, 2].

The concept of ε-kernels was introduced by Agarwal et al. [1]. The existence of ε-kernels with $O(1/\varepsilon^{(d-1)/2})$ points is implied in the works of Dudley [19] and Bronshteyn and Ivanov [16], and this is known to be optimal in the worst case. Agarwal et al. [1] demonstrated how to compute such a kernel in $O(n + 1/\varepsilon^{3(d-1)/2})$ time, which reduces to $O(n)$ when $n = \Omega(1/\varepsilon^{3(d-1)/2})$. While less succinct ε-kernels with $O(1/\varepsilon^{d-1})$ points can be constructed in time $O(n)$ time for all n [1, 14], no linear-time algorithm is known to build an ε-kernel of optimal size. Hereafter, we use the term ε-kernel to refer exclusively to an ε-kernel of size $O(1/\varepsilon^{(d-1)/2})$.

Chan [17] showed that an ε-kernel can be constructed in $O((n + 1/\varepsilon^{d-2}) \log \frac{1}{\varepsilon})$ time, which is nearly linear when $n = \Omega(1/\varepsilon^{d-2})$. He posed the open problem of obtaining a faster algorithm. A decade later, Arya and Chan [11] showed how to build an ε-kernel in roughly $O(n + \sqrt{n}/\varepsilon^{d/2})$ time using discrete Voronoi diagrams. In this paper, we attain the following near-optimal construction time.

▶ **Theorem 1.1.** *Given a set S of n points in $\mathbb{R}^d$ and an approximation parameter $\varepsilon > 0$, it is possible to construct an ε-kernel of S with $O(1/\varepsilon^{(d-1)/2})$ points in $O(n \log \frac{1}{\varepsilon} + 1/\varepsilon^{(d-1)/2+\alpha})$ time, where α is an arbitrarily small positive constant.*

Because the worst-case output size is $O(1/\varepsilon^{(d-1)/2})$, we may assume that n is at least this large, for otherwise we can simply take S itself to be the kernel. Since $1/\varepsilon^{\alpha}$ dominates

$\log \frac{1}{\varepsilon}$, the above running time can be expressed as $O(n/\varepsilon^{\alpha})$, which is nearly linear given that α can be made arbitrarily small.

Diameter. An important application of ε-kernels is to approximate the diameter of a point set. Given n data points, the *diameter* is defined to be the maximum distance between any two data points. An *ε-approximation* of the diameter is a pair of points whose distance is at least $(1-\varepsilon)$ times the exact diameter. There are multiple algorithms to approximate the diameter [1, 3, 11, 13, 17]. The fastest running times are $O((n+1/\varepsilon^{d-2})\log\frac{1}{\varepsilon})$ [17] and roughly $O(n+\sqrt{n}/\varepsilon^{d/2})$ [11]. The algorithm from [17] essentially computes an ε-kernel Q and then determines the maximum value of $\text{width}_v(Q)$ among a set of $k = O(1/\varepsilon^{(d-1)/2})$ directions v by brute force [1]. Discrete Voronoi diagrams [11] permit this computation in roughly $O(n+\sqrt{n}/\varepsilon^{d/2})$ time. Therefore, combining the kernel construction of Theorem 1.1 with discrete Voronoi diagrams [11], we reduce n to $O(1/\varepsilon^{(d-1)/2})$ and obtain an algorithm to ε-approximate the diameter in roughly $O(n+1/\varepsilon^{3d/4})$ time. However, we show that it is possible to obtain a much faster algorithm, as presented in the following theorem.

▶ **Theorem 1.2.** *Given a set S of n points in $\mathbb{R}^d$ and an approximation parameter $\varepsilon > 0$, it is possible to compute an ε-approximation to the diameter of S in $O(n\log\frac{1}{\varepsilon}+1/\varepsilon^{(d-1)/2+\alpha})$ time.*

Bichromatic Closest Pair. In the *bichromatic closest pair* (BCP) problem, we are given n points from two sets, designated red and blue, and we want to find the closest red-blue pair. In the *ε-approximate* version, the goal is to find a red-blue pair of points whose distance is at most $(1+\varepsilon)$ times the exact BCP distance. Approximations to the BCP problem were introduced in [23], and the most efficient randomized approximation algorithm runs in roughly $O(n/\varepsilon^{d/3})$ expected time [11]. We present the following result.

▶ **Theorem 1.3.** *Given n red and blue points in $\mathbb{R}^d$ and an approximation parameter $\varepsilon > 0$, there is a randomized algorithm that computes an ε-approximation to the bichromatic closest pair in $O(n/\varepsilon^{d/4+\alpha})$ expected time.*

Euclidean Trees. Given a set S of n points in $\mathbb{R}^d$, a *Euclidean minimum spanning tree* is the spanning tree with vertex set S that minimizes the sum of the edge lengths, while a *Euclidean minimum bottleneck tree* minimizes the maximum edge length. In the approximate version we respectively approximate the sum and the maximum of the edge lengths. A minimum spanning tree is a minimum bottleneck tree (although the converse does not hold). However, an approximation to the minimum spanning tree is not necessarily an approximation to the minimum bottleneck tree. A recent approximation algorithm to the Euclidean minimum spanning tree takes roughly $O(n\log n+n/\varepsilon^2)$ time, regardless of the (constant) dimension [9]. On the other hand, the fastest algorithm to approximate the minimum bottleneck tree takes roughly $O((n\log n)/\varepsilon^{d/3})$ expected time [11]. The algorithm uses BCP to simultaneously attain an approximation to the minimum bottleneck and the minimum spanning trees. We prove the following theorem.

▶ **Theorem 1.4.** *Given n points in $\mathbb{R}^d$ and an approximation parameter $\varepsilon > 0$, there is a randomized algorithm that computes a tree T that is an ε-approximation to both the Euclidean minimum bottleneck and the Euclidean minimum spanning trees in $O((n\log n)/\varepsilon^{d/4+\alpha})$ expected time.*

1.2 Data Structure Results

Polytope membership. Let P denote a convex polytope in $\mathbb{R}^d$, represented as the bounded intersection of n halfspaces. The *polytope membership problem* consists of preprocessing P so that it is possible to determine efficiently whether a given query point $q \in \mathbb{R}^d$ lies within P. In the ε-approximate version, we consider an expanded convex body $K \supset P$. A natural way to define this expansion would be to consider the set of points that lie within distance $\varepsilon \cdot \text{diam}(P)$ of P, thus defining a body whose Hausdorff distance from P is $\varepsilon \cdot \text{diam}(P)$. However, this definition has the shortcoming that it is not sensitive to the directional width of P. Instead, we define K as follows. For any nonzero vector $v \in \mathbb{R}^d$, consider the two supporting hyperplanes for P that are normal to v. Translate each of these hyperplanes outward by a distance of $\varepsilon \cdot \text{width}_v(P)$, and consider the closed slab-like region lying between them. Define K to be the intersection of this (infinite) set of slabs. This is clearly a stronger approximation than the Hausdorff-based definition. An *ε-approximate polytope membership query* (ε-APM query) returns a positive result if the query point q is inside P, a negative result if q is outside K, and may return either result otherwise.[1]

We recently proposed an optimal data structure to answer approximate polytope membership queries, but efficient preprocessing remained an open problem [7]. In this paper, we present a similar data structure that not only attains optimal storage and query time, but can also be preprocessed in near-optimal time.

▶ **Theorem 1.5.** *Given a convex polytope P in $\mathbb{R}^d$ represented as the intersection of n halfspaces and an approximation parameter $\varepsilon > 0$, there is a data structure that can answer ε-approximate polytope membership queries with query time $O(\log \frac{1}{\varepsilon})$, space $O(1/\varepsilon^{(d-1)/2})$, and preprocessing time $O(n \log \frac{1}{\varepsilon} + 1/\varepsilon^{(d-1)/2+\alpha})$.*

Directional width. Applying the previous data structure in the dual space, we obtain a data structure for the following *ε-approximate directional width* problem, which is closely related to ε-kernels. Given a set S of n points in a constant dimension d and an approximation parameter $\varepsilon > 0$, the goal is to preprocess S to efficiently ε-approximate $\text{width}_v(S)$, for a nonzero query vector v. We present the following result.

▶ **Theorem 1.6.** *Given a set S of n points in $\mathbb{R}^d$ and an approximation parameter $\varepsilon > 0$, there is a data structure that can answer ε-approximate directional width queries with query time $O(\log^2 \frac{1}{\varepsilon})$, space $O(1/\varepsilon^{(d-1)/2})$, and preprocessing time $O(n \log \frac{1}{\varepsilon} + 1/\varepsilon^{(d-1)/2+\alpha})$.*

Nearest Neighbor. Let S be a set of n points in $\mathbb{R}^d$. Given any $q \in \mathbb{R}^d$, an *ε-approximate nearest neighbor* (ANN) of q is any point of S whose distance from q is at most $(1 + \varepsilon)$ times the distance to q's closest point in S. The objective is to preprocess S in order to answer such queries efficiently. Data structures for approximate nearest neighbor searching (in fixed dimensions) have been proposed by several authors, offering space-time tradeoffs (see [7] for an overview of the tradeoffs). Applying the reduction from approximate nearest neighbor to approximate polytope membership established in [4] together with Theorem 1.5, we obtain the following result, which matches the best bound [7] up to an $O(\log \frac{1}{\varepsilon})$ factor in the query time, but offers faster preprocessing time.

[1] Our earlier works on ε-APM queries [4, 7] use the weaker Hausdorff form to define the problem, but the solutions presented there actually achieve the stronger direction-sensitive form.

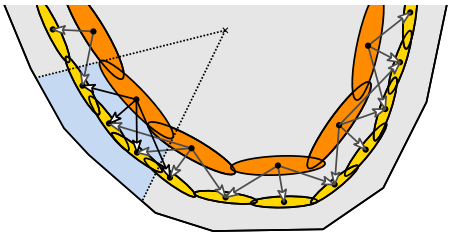


Figure 1 Two levels of the ellipsoid hierarchy.

▶ **Theorem 1.7.** *Given a set S of n points in $\mathbb{R}^d$, an approximation parameter $\varepsilon > 0$, and m such that $\log \frac{1}{\varepsilon} \leq m \leq 1/(\varepsilon^{d/2} \log \frac{1}{\varepsilon})$, there is a data structure that can answer Euclidean ε-approximate nearest neighbor queries with query time $O(\log n + (\log \frac{1}{\varepsilon})/(m \cdot \varepsilon^{d/2}))$ space $O(n\,m)$, and preprocessing time $O(n \log n \log \frac{1}{\varepsilon} + n\,m/\varepsilon^{\alpha})$.*

1.3 Techniques

In contrast to previous kernel constructions, which are based on grids and the execution of Bronshteyn and Ivanov's algorithm, our construction employs a classical structure from the theory of convexity, called *Macbeath regions* [24]. Macbeath regions, which will be defined in Section 2.1, have found numerous uses in the theory of convex sets and the geometry of numbers (see Bárány [12] for an excellent survey). They have also been applied to several problems in the field of computational geometry. However, most previous results were either in the form of lower bounds [8, 10, 15] or focused on existential results [5, 6, 20, 25].

In [7] the authors introduced a data structure employing a hierarchy of ellipsoids based on Macbeath regions to answer approximate polytope membership queries, but the efficient computation of the hierarchy was not considered. In this paper, we show how to efficiently construct the Macbeath regions that form the basis of this hierarchy.

Let P denote a convex polytope in $\mathbb{R}^d$. Each level i in the hierarchy corresponds to a δ_i-approximation of the boundary of P by a set of $O(1/\delta_i^{(d-1)/2})$ ellipsoids, where $\delta_i = \Theta(1/2^i)$. Each ellipsoid is sandwiched between two Macbeath regions and has $O(1)$ children, which correspond to the ellipsoids of the following level that approximate the same portion of the boundary (see Figure 1). The hierarchy starts with $\delta_0 = \Theta(1)$ and stops after $O(\log \frac{1}{\delta})$ levels when $\delta_i = \delta$, for a desired approximation δ. We present a simple algorithm to construct the hierarchy in $O(n + 1/\delta^{3(d-1)/2})$ time. The polytope P can be presented as either the intersection of n halfspaces or the convex hull of n points. We present the relevant background in Section 3.

Our algorithm to compute an ε-kernel in time $O(n \log \frac{1}{\varepsilon} + 1/\varepsilon^{(d-1)/2+\alpha})$ (Theorem 1.1) is based on a bootstrapping process. Since the time to build the ε-approximation hierarchy for the convex hull is prohibitively high, we use an approximation parameter $\delta = \varepsilon^{1/3}$ to build a δ-approximation hierarchy in $O(n + 1/\varepsilon^{(d-1)/2})$ time. By navigating through this hierarchy, we partition the n points among the leaf Macbeath ellipsoids in $O(n \log \frac{1}{\varepsilon})$ time, discarding points that are too far from the boundary. We then compute an (ε/δ)-kernel for the set of points in each leaf ellipsoid and return the union of the kernels computed.

Given an algorithm to compute an ε-kernel in $O(n \log \frac{1}{\varepsilon} + 1/\varepsilon^{t(d-1)})$ time, the previous procedure produces an ε-kernel in $O(n \log \frac{1}{\varepsilon} + 1/\varepsilon^{t'(d-1)})$ time, where $t' = (4t+1)/6$. By bootstrapping the construction a constant number of times, the value of t decreases from 1 to a value that is arbitrarily close to $\frac{1}{2}$. (This accounts for the $O(1/\varepsilon^{\alpha})$ factors in our running times.) The construction and its analysis are presented in Section 4.

In Section 5, we use our kernel construction in the dual space to efficiently build a polytope membership data structure, proving Theorem 1.5. The key idea is to compute multiple kernels in order to avoid examining the whole polytope when building each Macbeath region. Again, we use bootstrapping to obtain a near-optimal preprocessing time. The remaining theorems follow from Theorems 1.1 and 1.5, together with several known reductions.

2 Geometric Preliminaries

Consider a convex body K in d-dimensional space $\mathbb{R}^d$. Let ∂K denote the boundary of K. Let O denote the origin of $\mathbb{R}^d$. Given a parameter $0 < \gamma \leq 1$, we say that K is γ-*fat* if there exist concentric Euclidean balls B and B', such that $B \subseteq K \subseteq B'$, and $\text{radius}(B)/\text{radius}(B') \geq \gamma$. We say that K is *fat* if it is γ-fat for a constant γ (possibly depending on d, but not on ε).

Unless otherwise specified, the notion of ε-*approximation* between convex bodies will be based on the direction-sensitive definition given in Section 1.2. We say that a convex body K' is an *absolute* ε-*approximation* to another convex body K if they are within Hausdorff error ε of each other. Further, we say that K' is an *inner* (resp., *outer*) approximation if $K' \subseteq K$ (resp., $K' \supseteq K$).

Let B_0 denote a ball of radius $r_0 = \frac{1}{2}$ centered at the origin. For $0 < \gamma \leq 1$, let γB_0 denote the concentric ball of radius $\gamma r_0 = \frac{\gamma}{2}$. We say that a convex body K is in γ-*canonical form* if it is nested between γB_0 and B_0. A body in γ-canonical form is γ-fat and has diameter $\Theta(1)$. We will refer to point O as the *center* of P.

For any point $x \in K$, define $\delta(x)$ to be minimum distance from x to any point on ∂K. For the sake of ray-shooting queries, it is useful to define a ray-based notion of distance as well. Given $x \in K$, define the *ray-distance* of x to the boundary, denoted $\text{ray}(x)$, as follows. Consider the intersection point p of ∂K and the ray emanating from O that passes through x. We define $\text{ray}(x) = \|xp\|$. The following utility lemma will be helpful in relating distances to the boundary.

▶ **Lemma 2.1.** *Given a convex body K in γ-canonical form:*

(a) *For any point $x \in P$, $\delta(x) \leq \text{ray}(x) \leq \delta(x)/\gamma$.*

(b) *Let h be a supporting hyperplane of K. Let p be any point inside K at distance at most some distance w from h, where $w \leq \gamma/4$. Let p' denote the intersection of the ray Op and h. Then $\|pp'\| \leq 2w/\gamma$.*

(c) *Let p be any point on the boundary of K, and let h be a supporting hyperplane at p. Let h' denote the hyperplane obtained by translating h in the direction of the outward normal by some distance w. Let p' denote the intersection of the ray Op with h'. Then $\|pp'\| \leq w/\gamma$.*

We omit the straightforward proof. The lower bound on $\text{ray}(x)$ for part (a) is trivial, and the upper bound follows by a straightforward adaption of Lemma 4.2 of [6]. Part (b) is an adaptation of Lemma 2.11 of [7], and part (c) is similar.

2.1 Caps and Macbeath Regions

Much of the material in this section has been presented in [6, 7]. We include it here for the sake of completeness. Given a convex body K, a *cap* C is defined to be the nonempty intersection of K with a halfspace (see Figure 2(a)). Let h denote the hyperplane bounding this halfspace. We define the *base* of C to be $h \cap K$. The *apex* of C is any point in the cap such that the supporting hyperplane of K at this point is parallel to h. The *width* of C, denoted $\text{width}(C)$, is the distance between h and this supporting hyperplane. Given any cap

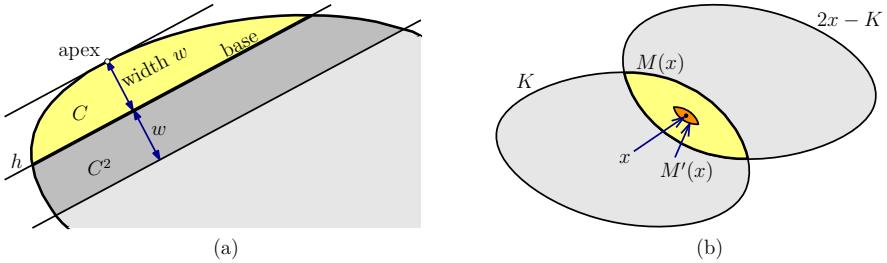


Figure 2 (a) Cap concepts and (b) Macbeath regions.

C of width w and a real $\lambda \geq 0$, we define its λ-*expansion*, denoted C^λ, to be the cap of K cut by a hyperplane parallel to and at distance λw from this supporting hyperplane. (Note that $C^\lambda = K$, if λw exceeds the width of K along the defining direction.)

Given a point $x \in K$ and real parameter $\lambda \geq 0$, the *Macbeath region* $M^\lambda(x)$ (also called an *M-region*) is defined as:

$$M^\lambda(x) \;=\; x + \lambda((K - x) \cap (x - K)).$$

It is easy to see that $M^1(x)$ is the intersection of K and the reflection of K around x (see Figure 2(b)). Clearly, $M^1(x)$ is centrally symmetric about x, and $M^\lambda(x)$ is a scaled copy of $M^1(x)$ by the factor λ about x. We refer to x as the *center* of $M^\lambda(x)$ and to λ as its *scaling factor*. As a convenience, we define $M(x) = M^1(x)$ and $M'(x) = M^{1/5}(x)$. We refer to the latter as the *shrunken* Macbeath region.

We now present a few lemmas that encapsulate key properties of Macbeath regions. The first lemma shows that if two shrunken Macbeath regions have a nonempty intersection, then a constant factor expansion of one contains the other [7, 15, 21].

▶ **Lemma 2.2.** *Let K be a convex body, and let $\lambda \leq 1/5$ be any real. If $x, y \in K$ such that $M^\lambda(x) \cap M^\lambda(y) \neq \emptyset$, then $M^\lambda(y) \subseteq M^{4\lambda}(x)$.*

The following lemma shows that all points in a shrunken Macbeath region have similar distances from the boundary of K. The proof appears in [7].

▶ **Lemma 2.3.** *Let K be a convex body. If $x \in K$ and $x' \in M'(x)$, then $4\delta(x)/5 \leq \delta(x') \leq 4\delta(x)/3$.*

For any $\delta > 0$, define the δ-erosion of a convex body K, denoted $K(\delta)$, to be the closed convex body formed by removing from K all points lying within distance δ of ∂K. The next lemma bounds the number of disjoint Macbeath regions that can be centered on the boundary of $K(\delta)$. The proof appears in [7].

▶ **Lemma 2.4.** *Consider a convex body $K \subset \mathbb{R}^d$ in γ-canonical form for some constant γ. Define $\Delta_0 = \frac{1}{2}(\gamma^2/(4d))^d$. For any fixed constant $0 < \lambda \leq 1/5$ and real parameter $\delta \leq \Delta_0$, let $\mathcal{M}$ be a set of disjoint λ-scaled Macbeath regions whose centers lie on the boundary of $K(\delta)$. Then $|\mathcal{M}| = O(1/\delta^{(d-1)/2})$.*

2.2 Shadows of Macbeath regions

Shrunken Macbeath regions reside within the interior of the convex body, but it is useful to identify the portion of the body's boundary that this Macbeath region will be responsible for approximating. For this purpose, we introduce the shadow of a Macbeath region. Given a

convex body K that contains the origin O and a region $R \subseteq K$, we define the *shadow* of R (with respect to K), denoted shadow(R), to be the set of points $x \in K$ such that the line segment Ox intersects R.

We also define a set of *normal directions* for R, denoted normals(R). Consider the set of all hyperplanes that support K at some point in the shadow of R. Define normals(R) to be the set of outward unit normals to these supporting hyperplanes. Typically, the region R in our constructions will be a (scaled) Macbeath region or an associated John ellipsoid (as defined in Section 3), close to the boundary of K. The following lemma captures a salient feature of these shadows, namely, that the shadow of a Macbeath region $M'(x)$ can be enclosed in an ellipsoid whose width in all normal directions is $O(\delta(x))$. The proof is presented in the full version.)

▶ **Lemma 2.5.** *Let $K \subset \mathbb{R}^d$ be a convex body in γ-canonical form for some constant γ. Let $x \in K$ be a point at distance δ from the boundary of K, where $\delta \leq \Delta_0$. Let $M = M'(x), S = \text{shadow}(M)$, $N = \text{normals}(M)$, and $\widehat{M} = M^{4/\gamma}(x)$. Then:*
(a) *$S \subseteq \widehat{M}$.*
(b) *$\text{width}_v(S) \leq c_1\delta$ for all $v \in N$. Here c_1 is the constant $8/(3\gamma)$.*
(c) *$\text{width}_v(\widehat{M}) \leq c_2\delta$ for all $v \in N$. Here c_2 is the constant $160/(3\gamma^2)$.*

2.3 Representation Conversions

Convex sets are naturally described in two ways, as the convex hull of a discrete set of points and as the intersection of a discrete set of halfspaces. Some computational tasks are more easily performed using one representation or the other, and hence it will be useful to convert between them. Also, when approximate representations suffice, it will be useful to prune a large set down to a smaller size. In this section we will present a few technical utilities to perform these conversions. We refer the reader to the full version for the missing proofs.

Given an n-element point set in $\mathbb{R}^d$, Chan showed that it is possible to construct an ε-kernel of size $O(1/\varepsilon^{(d-1)/2})$ in time $O(n + 1/\varepsilon^{d-1})$ [17]. The following lemma shows that, by applying Chan's construction, is is possible to concisely approximate the convex hull of n points as the intersection of halfspaces.

▶ **Lemma 2.6.** *Let $\gamma < 1$ be a positive constant, and $\varepsilon > 0$ be a real parameter. Let P be a polytope in γ-canonical form represented as the convex hull of n points. In $O(n + 1/\varepsilon^{d-1})$ time it is possible to compute a polytope P' represented as the intersection of $O(1/\varepsilon^{(d-1)/2})$ halfspaces such that P' is an inner absolute ε-approximation of P.*

The following lemma is useful when representing polytopes by the intersection of halfspaces.

▶ **Lemma 2.7.** *Let $\gamma < 1$ be a positive constant, and $\varepsilon > 0$ be a real parameter. Let P be a polytope in γ-canonical form represented as the intersection of n halfspaces. In $O(n+1/\varepsilon^{d-1})$ time it is possible to compute a polytope P' represented as the intersection of $O(1/\varepsilon^{(d-1)/2})$ halfspaces such that P' is an outer absolute ε-approximation of P.*

▶ Remark. Theorem 1.1 shows that an ε-kernel of size $O(1/\varepsilon^{(d-1)/2})$ can be computed in time $O(n \log \frac{1}{\varepsilon} + 1/\varepsilon^{(d-1)/2+\alpha})$. The construction time in Lemma 2.7 (which is derived in the full version) is asymptotically dominated by the time needed to construct an ε-kernel. Therefore, the construction time can be reduced to this quantity.

3 Hierarchy of Macbeath Ellipsoids

The data structure presented in [7] for the approximate polytope membership problem is based on constructing a hierarchy of ellipsoids. In this section, we present a variant of this structure, which will play an important role in our constructions.

For a Macbeath region $M^\lambda(x)$, we denote its circumscribing John ellipsoid by $E^\lambda(x)$, which we call a *Macbeath ellipsoid*. Since Macbeath regions are centrally symmetric and the constant in John's Theorem [22] is $\sqrt{d}$ for centrally symmetric bodies, we have $E^\lambda(x) \subseteq M^{\lambda\sqrt{d}}(x)$. Recall the constant $\Delta_0 = \frac{1}{2}(\gamma^2/4d)^d$ defined in the statement of Lemma 2.4, and define $\lambda_0 = 1/(20d)$. We omit the proof of the following lemma due to space limitations. (We caution the reader that in the lemmas of this section, the value of n used in the application of the lemma may differ from the original input size.)

▶ **Lemma 3.1.** *Let $\gamma < 1$ be a positive constant, and let $0 < \delta \leq \Delta_0$ be a real parameter. Let P be a polytope in γ-canonical form, represented as the intersection of n halfspaces. In $O(n/\delta^{d-1} + 1/\delta^{3(d-1)/2})$ time, we can construct a DAG structure satisfying the following properties:*

- **(a)** *The total number of nodes (including leaves), and the total space used by the DAG are each $O(1/\delta^{(d-1)/2})$.*
- **(b)** *Each leaf is associated with an ellipsoid $E^{4\lambda_0\sqrt{d}}(x)$, where $x \in \partial P(\delta)$. The union of the ellipsoids associated with all the leaves covers $\partial P(\delta)$.*
- **(c)** *Given a query ray Oq, in $O(\log \frac{1}{\delta})$ time, we can find a leaf node such that the associated ellipsoid intersects this ray.*

Given a convex body K and query point q, an *absolute ε-APM* query returns a positive result if q lies within K, a negative result if q is at distance at least ε from K, and otherwise it may return either result. After a small enhancement, this DAG can be used for answering absolute ε-APM queries for a polyope P in γ-canonical form. We assume that P is represented as the intersection of a set H of n halfspaces. We invoke the above lemma for $\delta = \varepsilon\gamma/(2c_1)$, where c_1 is the constant of Lemma 2.5(b). We then associate each leaf of the DAG with a halfspace as follows. Let x denote the center of the leaf ellipsoid and let p denote the intersection of the ray Ox with ∂P. Let $h \in H$ denote any supporting halfspace of P (containing P) at p. We store h with this leaf. By exhaustive search, we can determine h in $O(n)$ time, so the total time for this step is $O(n/\varepsilon^{(d-1)/2})$. Asymptotically, this does not affect the time it takes to construct the data structure. Given a query point q, we answer queries by first determining a leaf whose ellipsoid intersects the ray Oq. By Lemma 3.1(c), this takes $O(\log \frac{1}{\varepsilon})$ time. We return a positive answer if and only if q is contained in the associated halfspace.

The following lemma summarizes the result, whose proof is presented in the full version.

▶ **Lemma 3.2.** *Let $\gamma < 1$ be a positive constant, and let $\varepsilon > 0$ be a real parameter. Let P be a polytope in γ-canonical form, represented as the intersection of n halfspaces. In $O(n/\varepsilon^{d-1} + 1/\varepsilon^{3(d-1)/2})$ time, we can construct a data structure that uses $O(1/\varepsilon^{(d-1)/2})$ space and answers absolute ε-APM queries in $O(\log \frac{1}{\varepsilon})$ time.*

4 Kernel Construction

In this section we establish Theorem 1.1 by showing how to build an ε-kernel efficiently. The input to an ε-kernel construction consists of the approximation parameter ε and a set S of n points. Our algorithm is based on a bootstrapping strategy. We assume that we

have access to an algorithm that can construct an ε-kernel of $O(1/\varepsilon^{(d-1)/2})$ size in time $O(n \log \frac{1}{\varepsilon} + 1/\varepsilon^{(1/2+\beta)(d-1)})$, where $\beta > 0$ is a parameter. Recall that the size of the kernel is asymptotically optimal in the worst case. We will present a method for improving the running time of this algorithm. Recall that Chan [17] gave an algorithm for constructing kernels of optimal size which runs in time $O(n \log \frac{1}{\varepsilon} + 1/\varepsilon^{d-1})$. By setting $\beta = \frac{1}{2}$, this will form the basis of our bootstrapping, which is described below. Throughout, let $\delta = \varepsilon^{1/3}$.

1. Fatten the input point set S by computing an affine transformation that maps S to S', such that $\mathrm{conv}(S')$ is in γ-canonical form for some constant γ. By standard results (see, e.g., the journal version of [4]), this can be done in $O(n)$ time.
2. Using Lemma 2.6, build a polytope P, represented as the intersection of $O(1/\delta^{(d-1)/2})$ halfspaces, such that P is an inner absolute δ-approximation of $\mathrm{conv}(S')$. This step takes $O(n + 1/\delta^{d-1}) = O(n + 1/\varepsilon^{(d-1)/3})$ time.
3. Apply Lemma 3.1 to construct a DAG structure for P using the parameter δ. Replacing n in the statement of the lemma by $O(1/\delta^{(d-1)/2})$, it follows that this step takes $O(1/\delta^{3(d-1)/2}) = O(1/\varepsilon^{(d-1)/2})$ time.
4. By Lemma 3.1(c), for each point $p \in S'$, find a leaf of the DAG such that the associated ellipsoid $E^{4\lambda_0\sqrt{d}}(x)$ intersects the ray Op. Recall that $x \in \partial P(\delta)$. This takes $O(\log \frac{1}{\delta})$ per point. In $O(1)$ additional time, determine whether p lies in the shadow of this ellipsoid (with respect to $\mathrm{conv}(S')$). If so, associate p with this ellipsoid, and otherwise discard it. All the points of S' can be processed in time $O(n \log \frac{1}{\delta}) = O(n \log \frac{1}{\varepsilon})$.
5. For each leaf ellipsoid of the DAG, build a $(c_3\varepsilon/\delta)$-kernel for the points of S' that lie in its shadow, where c_3 is a suitably small constant that will be selected later. This kernel is computed using the aforementioned algorithm that computes the ε-kernel of a point set of size n in time $O(n \log \frac{1}{\varepsilon} + 1/\varepsilon^{(1/2+\beta)(d-1)})$. The size of the $O(\varepsilon/\delta)$-kernel computed for each shadow is $O((\delta/\varepsilon)^{(d-1)/2})$ and the time required is $O(n_i \log \frac{\delta}{\varepsilon} + (\delta/\varepsilon)^{(1/2+\beta)(d-1)})$, where n_i denotes the number of points of S' in the shadow. Summed over all the shadows, it follows that the total time required is

$$O\left(n \log \frac{\delta}{\varepsilon} + \left(\frac{1}{\delta}\right)^{\frac{d-1}{2}} \left(\frac{\delta}{\varepsilon}\right)^{\left(\frac{1}{2}+\beta\right)(d-1)}\right) = O\left(n \log \frac{1}{\varepsilon} + \left(\frac{1}{\varepsilon}\right)^{\left(\frac{1}{2}+\frac{2\beta}{3}\right)(d-1)}\right).$$

   Here we have used the facts that each point of S' is assigned to at most one shadow and the total number of shadows, which is bounded by the number of leaves in the DAG, is $O(1/\delta^{(d-1)/2})$.
6. Let $S'' \subseteq S'$ be the union of the kernels computed in the previous step. Since the number of shadows is $O(1/\delta^{(d-1)/2})$ and the size of the kernel for each shadow is $O((\delta/\varepsilon)^{(d-1)/2})$, it follows that $|S''| = O(1/\varepsilon^{(d-1)/2})$. Apply the inverse of the affine transformation computed in Step 1 to the points of S'', and output the resulting set of points as the desired ε-kernel for S.

We have shown that the size of the output kernel is $O(1/\varepsilon^{(d-1)/2})$, as desired. The running time of Step 5 dominates the time complexity. Our next lemma establishes the correctness of this construction.

▶ **Lemma 4.1.** *The construction yields an ε-kernel.*

Proof. Throughout this proof, for a given convex body K, we use $M_K(x)$, $E_K(x)$, and $\delta_K(x)$ to denote the quantities $M(x)$, $E(x)$, and $\delta(x)$ with respect to K. Let $P' = \mathrm{conv}(S')$. By standard results on fattening, it suffices to show that $\mathrm{conv}(S'')$ is an absolute $O(\varepsilon)$-approximation of P'. Let v be an arbitrary direction. Consider the extreme point p of S'

in direction v. Clearly $p \in \partial P'$. Recall that P is an inner δ-approximation of P', and the ellipsoids associated with the leaves of the DAG cover the boundary of $P(\delta)$. Thus, there must be an ellipsoid $E = E_P^{4\lambda_0\sqrt{d}}(x)$, $x \in \partial P(\delta)$, such that p is assigned to the shadow of E in Step 4. Note that this shadow and all shadows throughout this proof are assumed to be with respect to the polytope P' (and not P). We claim that $\text{width}_v(\text{shadow}(E)) \leq 2c_1\delta$, where c_1 is the constant of Lemma 2.5(b). Assuming this claim for now, let us complete the proof of the lemma. Recall that in Step 5, we built a $(c_3\varepsilon/\delta)$-kernel for all the points of S' that are assigned to the shadow of E, and S'' includes all the points of this kernel. It follows that the distance between the supporting hyperplanes of $\text{conv}(S')$ and $\text{conv}(S'')$ in direction v is at most $(c_3\varepsilon/\delta) \cdot \text{width}_v(\text{shadow}(E)) \leq (c_3\varepsilon/\delta) \cdot (2c_1\delta) = 2c_1c_3\varepsilon$. By choosing c_3 sufficiently small, we can ensure that this quantity is smaller than any desired constant times ε, which proves the lemma.

It remains to show that $\text{width}_v(\text{shadow}(E)) \leq 2c_1\delta$. Recall that

$$E \;=\; E_P^{4\lambda_0\sqrt{d}}(x) \;\subseteq\; M_P^{4\lambda_0 d}(x) \;=\; M'_P(x).$$

Furthermore, since $P \subseteq P'$, a straightforward consequence of the definition of Macbeath regions is that $M'_P(x) \subseteq M'_{P'}(x)$. To simplify the notation, let M denote $M'_{P'}(x)$. Putting it together, we obtain $E \subseteq M$. Thus $\text{shadow}(E) \subseteq \text{shadow}(M)$, which implies that $\text{width}_v(\text{shadow}(E)) \leq \text{width}_v(\text{shadow}(M))$. By Lemma 2.5(b),

$$\text{width}_v(\text{shadow}(M)) \;\leq\; c_1\delta_{P'}(x).$$

Using the triangle inequality and the fact that P is an inner δ-approximation of P', we obtain $\delta_{P'}(x) \leq \delta_P(x) + \delta = 2\delta$. Thus $\text{width}_v(\text{shadow}(E)) \leq \text{width}_v(\text{shadow}(M)) \leq 2c_1\delta$, as desired. ◀

We are now ready to establish the main result of this section.

Proof. (of Theorem 1.1) Our proof is based on a constant number of applications of the algorithm from this section. It suffices to show that there is an algorithm that can construct an ε-kernel of $O(1/\varepsilon^{(d-1)/2})$ size in time $O(n \log \frac{1}{\varepsilon} + 1/\varepsilon^{(1/2+\beta')(d-1)})$, where $\beta' = \alpha/(d-1)$.

We initialize the bootstrapping process by Chan's algorithm [17], which has $\beta = \frac{1}{2}$. Observe that the value of β is initially $\frac{1}{2}$ and falls by a factor of $\frac{2}{3}$ with each application of the algorithm. It follows that after $O(\log \frac{1}{\alpha})$ applications, we will obtain an algorithm with the desired running time. This completes the proof. ◀

5 Approximate Polytope Membership

In this section we show how to obtain a data structure for approximate polytope membership, proving Theorem 1.5. Our best data structure for APM achieves query time $O(\log \frac{1}{\varepsilon})$ with storage $O(1/\varepsilon^{(d-1)/2})$ and preprocessing time $O(n \log \frac{1}{\varepsilon} + 1/\varepsilon^{(d-1)/2+\alpha})$. As with kernels, our construction here is again based on a bootstrapping strategy. To initialize the process, we will use a data structure that achieves the aforementioned query time with the same storage but with preprocessing time $O(n + 1/\varepsilon^{3(d-1)/2})$. The data structure is based on Lemma 3.2. Recall that the input is a polytope represented as the intersection of n halfspaces.

We begin by "fattening" the input polytope. As before, we use an affine transformation to map the input polytope to a polytope P' that is in γ-canonical form. This step takes $O(n)$ time [4]. By standard results, it suffices to build a data structure for answering absolute $O(\varepsilon)$-APM queries with respect to P' (see, e.g., Lemma 7.1 of the journal version of [4]).

Next, we apply Lemma 2.7 to construct an outer absolute $O(\varepsilon)$-approximation P of P', where P is represented as the intersection of $O(1/\varepsilon^{(d-1)/2})$ halfspaces. This step takes $O(n + 1/\varepsilon^{d-1})$ time. Finally, we use Lemma 3.2 to construct a data structure for answering absolute $O(\varepsilon)$-APM queries with respect to P. Replacing n in the statement of the lemma by $O(1/\varepsilon^{(d-1)/2})$, it follows that this step takes $O(1/\varepsilon^{3(d-1)/2}))$ time.

The total construction time is $O(n + 1/\varepsilon^{3(d-1)/2})$. To answer a query, we map the query point using the same transformation used to fatten the polytope, and then use the data structure constructed above to determine whether the resulting point lies in polytope P. Subject to an appropriate choice of constant factors, the correctness of this method follows from the fact that P is an outer absolute $O(\varepsilon)$-approximation of P'.

We summarize this result in the following lemma.

▶ **Lemma 5.1.** *Let $\varepsilon > 0$ be a real parameter and let P be a polytope, represented as the intersection of n halfspaces. In $O(n + 1/\varepsilon^{3(d-1)/2})$ time, we can construct a data structure that uses $O(1/\varepsilon^{(d-1)/2})$ space and answers ε-APM queries in $O(\log \frac{1}{\varepsilon})$ time.*

We now present the details of our bootstrapping approach. We assume that for a parameter $\beta > 0$, in time $O(n \log \frac{1}{\varepsilon} + 1/\varepsilon^{(1/2+\beta)(d-1)})$ we can construct a data structure that can answer ε-APM queries in $O(\log \frac{1}{\varepsilon})$ time with $O(1/\varepsilon^{(d-1)/2})$ storage. We present a method for constructing a new data structure that matches the same storage and query time but has a lower preprocessing time. Throughout, let $\delta = \varepsilon^{\beta/(1+\beta)}$.

1. As in the kernel construction, first fatten the input polytope by applying an affine transformation that maps the input polytope to a polytope P' that is in γ-canonical form. By standard results (see, e.g., [4]), this step takes $O(n)$ time, and it suffices to build a data structure for answering absolute $O(\varepsilon)$-APM queries with respect to P'.
2. Using Lemma 2.7, build an outer absolute $O(\varepsilon)$-approximation of P', denoted P, which is represented as the intersection of $O(1/\varepsilon^{(d-1)/2})$ halfspaces. By the remark following Lemma 2.7, this step takes $O(n \log \frac{1}{\varepsilon} + 1/\varepsilon^{(d-1)/2+\alpha})$ time.
3. Apply Lemma 3.1 to construct a DAG structure for P using the parameter δ. Replacing n in the statement of the lemma by $O(1/\varepsilon^{(d-1)/2})$, it follows that this step takes $O((1/\delta)^{d-1} \cdot (1/\varepsilon)^{(d-1)/2})$ time.
4. For each leaf of the DAG, construct an APM data structure as follows. Let $E = E^{4\lambda_0\sqrt{d}}(x)$ denote the ellipsoid associated with the leaf. Let R denote the minimum enclosing hyperrectangle of the ellipsoid $E^{4/\gamma}(x)$. We will see later that R contains the shadow of E (with respect to P), and its width in any direction in normals(E) is at most $c_2 d\delta = O(\delta)$, where c_2 is the constant in Lemma 2.5(c).
   Using the aforementioned algorithm, construct an APM data structure for this region with approximation parameter $c_3\varepsilon/\delta$, where c_3 is a sufficiently small constant that we will select later. Note that each such region can be expressed as the intersection of $n_i = O(1/\varepsilon^{(d-1)/2})$ halfspaces, namely, all the halfspaces defining P together with the $2d$ halfspaces defined by the facets of R. The construction time of the APM data structure for each leaf is
$$O\left(n_i \log \frac{\delta}{\varepsilon} + \left(\frac{\delta}{\varepsilon}\right)^{\left(\frac{1}{2}+\beta\right)(d-1)}\right) = O\left(\left(\frac{1}{\varepsilon}\right)^{\frac{d-1}{2}} \log \frac{\delta}{\varepsilon} + \left(\frac{\delta}{\varepsilon}\right)^{\left(\frac{1}{2}+\beta\right)(d-1)}\right),$$
   and the space used is $O((\delta/\varepsilon)^{(d-1)/2})$. Since there are $O(1/\delta^{(d-1)/2})$ leaves, it follows that the total space is $O(1/\varepsilon^{(d-1)/2})$, and the total construction time is the product of $O(1/\delta^{(d-1)/2})$ and the above construction time for each leaf.

Summing up the time over all the four steps, we obtain a total construction time on the order of

$$\left(n\log\frac{1}{\varepsilon}+\left(\frac{1}{\varepsilon}\right)^{\frac{d-1}{2}+\alpha}\right)+\left(\frac{1}{\delta}\right)^{d-1}\left(\frac{1}{\varepsilon}\right)^{\frac{d-1}{2}}+\left(\frac{1}{\delta}\right)^{\frac{d-1}{2}}\cdot\left(\left(\frac{1}{\varepsilon}\right)^{\frac{d-1}{2}}\log\frac{\delta}{\varepsilon}+\left(\frac{\delta}{\varepsilon}\right)^{\left(\frac{1}{2}+\beta\right)(d-1)}\right).$$

Recalling that $\delta=\varepsilon^{\beta/(1+\beta)}$ and assuming that the constant α is much smaller than β, it follows that the construction time is

$$O\left(n\log\frac{1}{\varepsilon}+\left(\frac{1}{\varepsilon}\right)^{\left(\frac{1}{2}+\frac{\beta}{1+\beta}\right)(d-1)}\right).$$

We answer queries as follows. We apply the affine transformation of Step 1 to the input query point to obtain a point q. Recall that it suffices to answer absolute $O(\varepsilon)$-APM queries for q with respect to P'. As P is an outer absolute $O(\varepsilon)$-approximation of P', it suffices to answer absolute $O(\varepsilon)$-APM queries for q with respect to P. To answer this query, we identify a leaf of the DAG such that the associated ellipsoid E intersects the ray Oq. This takes time $O(\log\frac{1}{\delta})$. Let y denote an intersection point of this ray with the ellipsoid E. If q lies on the segment Oy, then q is declared as lying inside P. Otherwise we return the answer we get for query q using the APM data structure we built for this leaf. It takes time $O(\log\frac{\delta}{\varepsilon})$ to answer this query. Including the time to locate the leaf, the total query time is $O(\log\frac{1}{\varepsilon})$. Our next lemma shows that queries are answered correctly.

▶ **Lemma 5.2.** *The query procedure returns a valid answer to the ε-APM query.*

Proof. We borrow the terminology from the query procedure given above. As mentioned, it suffices to show that our algorithm correctly answers absolute $O(\varepsilon)$-APM queries for q with respect to the polytope P. Recall that we identify a leaf of the DAG whose associated ellipsoid $E=E^{4\lambda_0\sqrt{d}}(x)$ intersects the ray Oq. Recall that y is a point on the intersection of the ray Oq with E. Clearly, if q lies on segment Oy, then $q\in P$ and q is correctly declared as lying inside P.

It remains to show that queries are answered correctly when $\|Oq\|>\|Oy\|$. In this case, we handle the query using the APM data structure we built for the leaf. Recall that this structure is built for the polytope formed by intersecting P with the smallest enclosing hyperrectangle R of the ellipsoid $E^{4/\gamma}(x)$. It suffices to show: (i) $\text{shadow}(E)\subseteq R$ and (ii) $\text{width}_v(R)\le c_2 d\delta$ for all $v\in\text{normals}(E)$, where c_2 is the constant in Lemma 2.5(c).

To establish (i), recall that $M^{\lambda}(x)\subseteq E^{\lambda}(x)\subseteq M^{\lambda\sqrt{d}}(x)$ for any $\lambda>0$. Using this fact, it follows that $M^{4/\gamma}(x)\subseteq E^{4/\gamma}(x)\subseteq M^{4\sqrt{d}/\gamma}(x)$. By Lemma 2.5(a), $\text{shadow}(E)\subseteq M^{4/\gamma}(x)$. Thus $\text{shadow}(E)\subseteq E^{4/\gamma}(x)\subseteq R$, which proves (i). To prove (ii), note that $R\subseteq E^{4\sqrt{d}/\gamma}(x)$, since R is the smallest enclosing hyperrectangle of $E^{4/\gamma}(x)$. Also $E^{4\sqrt{d}/\gamma}(x)\subseteq M^{4d/\gamma}(x)$. Thus $R\subseteq M^{4d/\gamma}(x)$. By Lemma 2.5(c), $\text{width}_v(M^{4/\gamma}(x))\le c_2\delta$ for all $v\in\text{normals}(M'(x))$. Since $R\subseteq M^{4d/\gamma}(x)$ and $E\subseteq M'(x)$, it follows that $\text{width}_v(R)\le c_2 d\delta$ for all $v\in\text{normals}(E)$.

We return to showing that queries are correctly answered when $\|Oq\|>\|Oy\|$. We consider two possibilities depending on whether q is inside or outside P. If $q\in P$ then $q\in\text{shadow}(E)$. By part (i) of the above claim, $\text{shadow}(E)\subseteq R$, and thus $q\in P\cap R$. It follows that the APM structure built for the leaf will declare this point as lying inside $P\cap R$, and hence the overall algorithm will correctly declare that q lies in P.

Finally, we consider the case when $q\notin P$. To complete the proof, we need to show that if the distance of q from the boundary of P is greater than ε, then q is declared as lying outside P. Let p denote the point of intersection of the ray Oq with ∂P, let h denote a hyperplane

supporting P at p, and let v denote the outward normal to h. Recall by part (i) of the claim that shadow$(E) \subseteq R$. It follows that h is a supporting hyperplane of $P \cap R$ at p. By part (ii) of the claim, $\text{width}_v(R) \leq c_2 d\delta$, and hence $\text{width}_v(P \cap R) \leq c_2 d\delta$. Recall that the APM data structure for the leaf is built using the approximation parameter $c_3\varepsilon/\delta$ for some constant c_3. By definition of APM query (in the standard, direction-sensitive sense), the absolute error allowed in direction v is at most $(c_3\varepsilon/\delta) \cdot \text{width}_v(P \cap R) \leq (c_3\varepsilon/\delta)(c_2 d\delta)$. By choosing c_3 sufficiently small we can ensure that this error is at most $\varepsilon\gamma$. To make this more precise, let h' denote the hyperplane parallel to h (outside P), and at distance $\varepsilon\gamma$ from it. Consider the halfspace bounded by h' and containing P. By the definition of APM query, if q is not contained in this halfspace, then q would be declared as lying outside $P \cap R$, and the overall algorithm would declare q as lying outside P. Let p' denote the point of intersection of the ray Oq with h'. By Lemma 2.1(c), $\|pp'\| \leq (\varepsilon\gamma)/\gamma = \varepsilon$. Thus, if the distance of q from ∂P is greater than ε, then q cannot lie on segment pp' and q is correctly declared as lying outside P. This completes the proof of correctness. ◀

We now establish the main result of this section.

Proof. (of Theorem 1.5) Our proof is based on a constant number of applications of the method presented in this section. It suffices to show that there is a data structure with space and query time as in the theorem and preprocessing time $O(n \log \frac{1}{\varepsilon} + 1/\varepsilon^{(1/2+\beta')(d-1)})$, where $\beta' = \alpha/(d-1)$.

We initialize the bootstrapping process by the data structure described in the beginning of this section, which has $\beta = 1$. Recall that applying the method once changes the value of β to $\beta/(1+\beta)$. It is easy to show that after i applications, the value of β will fall to $1/(i+1)$. Thus, after $O(1/\alpha)$ applications, we will obtain a data structure with the desired preprocessing time. ◀

The remaining theorems follow from previous reductions. Theorem 1.2 follows from performing $O(1/\varepsilon^{(d-1)/2})$ width queries [3, 17] using Theorem 1.6. Theorem 1.3 is a consequence of Theorem 1.5 together with [4, Lemma 9.2 of the journal version] and the construction from [11, Theorem 3.2]. Theorem 1.4 follows from 1.3 using [11, Theorem 4.1]. Theorem 1.6 follows from Theorem 1.5 by using duality and binary search. Theorem 1.7 is a consequence of Theorem 1.5 and the reduction presented in [4, Lemma 9.3 of the journal version].

References

1. P. K. Agarwal, S. Har-Peled, and K. R. Varadarajan. Approximating extent measures of points. *J. Assoc. Comput. Mach.*, 51:606–635, 2004.
2. P. K. Agarwal, S. Har-Peled, and K. R. Varadarajan. Geometric approximation via coresets. In J. E. Goodman, J. Pach, and E. Welzl, editors, *Combinatorial and Computational Geometry*. MSRI Publications, 2005.
3. P. K. Agarwal, J. Matoušek, and S. Suri. Farthest neighbors, maximum spanning trees and related problems in higher dimensions. *Comput. Geom. Theory Appl.*, 1(4):189–201, 1992.
4. S. Arya, G. D. da Fonseca, and D. M. Mount. Approximate polytope membership queries. In *Proc. 43rd Annu. ACM Sympos. Theory Comput.*, pages 579–586, 2011. doi:10.1145/1993636.1993713.
5. S. Arya, G. D. da Fonseca, and D. M. Mount. Optimal area-sensitive bounds for polytope approximation. In *Proc. 28th Annu. Sympos. Comput. Geom.*, pages 363–372, 2012.
6. S. Arya, G. D. da Fonseca, and D. M. Mount. On the combinatorial complexity of approximating polytopes. In *Proc. 32nd Internat. Sympos. Comput. Geom.*, pages 11:1–11:15, 2016. doi:10.4230/LIPIcs.SoCG.2016.11.

7 S. Arya, G. D. da Fonseca, and D. M. Mount. Optimal approximate polytope membership. In *Proc. 28th Annu. ACM-SIAM Sympos. Discrete Algorithms*, pages 270–288, 2017.

8 S. Arya, T. Malamatos, and D. M. Mount. The effect of corners on the complexity of approximate range searching. *Discrete Comput. Geom.*, 41:398–443, 2009.

9 S. Arya and D. M. Mount. A fast and simple algorithm for computing approximate Euclidean minimum spanning trees. In *Proc. 27th Annu. ACM-SIAM Sympos. Discrete Algorithms*, pages 1220–1233, 2016.

10 S. Arya, D. M. Mount, and J. Xia. Tight lower bounds for halfspace range searching. *Discrete Comput. Geom.*, 47:711–730, 2012. doi:10.1007/s00454-012-9412-x.

11 Sunil Arya and Timothy M. Chan. Better ε-dependencies for offline approximate nearest neighbor search, Euclidean minimum spanning trees, and ε-kernels. In *Proc. 30th Annu. Sympos. Comput. Geom.*, pages 416–425, 2014.

12 I. Bárány. The technique of M-regions and cap-coverings: A survey. *Rend. Circ. Mat. Palermo*, 65:21–38, 2000.

13 G. Barequet and S. Har-Peled. Efficiently approximating the minimum-volume bounding box of a point set in three dimensions. *J. Algorithms*, 38(1):91–109, 2001.

14 J. L. Bentley, M. G. Faust, and F. P. Preparata. Approximation algorithms for convex hulls. *Commun. ACM*, 25(1):64–68, 1982. doi:10.1145/358315.358392.

15 H. Brönnimann, B. Chazelle, and J. Pach. How hard is halfspace range searching. *Discrete Comput. Geom.*, 10:143–155, 1993.

16 E. M. Bronshteyn and L. D. Ivanov. The approximation of convex sets by polyhedra. *Siberian Math. J.*, 16:852–853, 1976.

17 T. M. Chan. Faster core-set constructions and data-stream algorithms in fixed dimensions. *Comput. Geom. Theory Appl.*, 35(1):20–35, 2006. doi:10.1016/j.comgeo.2005.10.002.

18 T. M. Chan. Applications of Chebyshev polynomials to low-dimensional computational geometry. In *Proc. 33rd Internat. Sympos. Comput. Geom.*, pages 26:1–15, 2017.

19 R. M. Dudley. Metric entropy of some classes of sets with differentiable boundaries. *J. Approx. Theory*, 10(3):227–236, 1974.

20 K. Dutta, A. Ghosh, B. Jartoux, and N. H. Mustafa. Shallow packings, semialgebraic set systems, Macbeath regions and polynomial partitioning. In *Proc. 33rd Internat. Sympos. Comput. Geom.*, pages 38:1–15, 2017.

21 G. Ewald, D. G. Larman, and C. A. Rogers. The directions of the line segments and of the r-dimensional balls on the boundary of a convex body in Euclidean space. *Mathematika*, 17:1–20, 1970.

22 F. John. Extremum problems with inequalities as subsidiary conditions. In *Studies and Essays Presented to R. Courant on his 60th Birthday*, pages 187–204. Interscience Publishers, Inc., New York, 1948.

23 S. Khuller and Y. Matias. A simple randomized sieve algorithm for the closest-pair problem. *Information and Computation*, 118(1):34–37, 1995.

24 A. M. Macbeath. A theorem on non-homogeneous lattices. *Ann. of Math.*, 56:269–293, 1952.

25 N. H. Mustafa and S. Ray. Near-optimal generalisations of a theorem of Macbeath. In *Proc. 31st Internat. Sympos. on Theoret. Aspects of Comp. Sci.*, pages 578–589, 2014.

Exact Algorithms for Terrain Guarding*

Pradeesha Ashok[1], Fedor V. Fomin[2], Sudeshna Kolay[3], Saket Saurabh[4], and Meirav Zehavi[5]

1 **The Institute of Mathematical Sciences, Chennai, India**
pradeesha@imsc.res.in
2 **University of Bergen, Bergen, Norway**
fomin@ii.uib.no
3 **The Institute of Mathematical Sciences, Chennai, India**
skolay@imsc.res.in
4 **The Institute of Mathematical Sciences, Chennai, India; and University of Bergen, Bergen, Norway**
saket@imsc.res.in
5 **University of Bergen, Bergen, Norway**
meirav.zehavi@ii.uib.no

Abstract

Given a 1.5-dimensional terrain T, also known as an x-monotone polygonal chain, the Terrain Guarding problem seeks a set of points of minimum size on T that *guards* all of the points on T. Here, we say that a point p guards a point q if no point of the line segment $\overline{pq}$ is strictly below T. The Terrain Guarding problem has been extensively studied for over 20 years. In 2005 it was already established that this problem admits a constant-factor approximation algorithm [SODA 2005]. However, only in 2010 King and Krohn [SODA 2010] finally showed that Terrain Guarding is NP-hard. In spite of the remarkable developments in approximation algorithms for Terrain Guarding, next to nothing is known about its parameterized complexity. In particular, the most intriguing open questions in this direction ask whether it admits a subexponential-time algorithm and whether it is fixed-parameter tractable. In this paper, we answer the first question affirmatively by developing an $n^{\mathcal{O}(\sqrt{k})}$-time algorithm for both Discrete Terrain Guarding and Continuous Terrain Guarding. We also make non-trivial progress with respect to the second question: we show that Discrete Orthogonal Terrain Guarding, a well-studied special case of Terrain Guarding, is fixed-parameter tractable.

1998 ACM Subject Classification F.2.2 [Nonnumerical Algorithms and Problems] Geometrical problems and computations

Keywords and phrases Terrain Guarding, Art Gallery, Exponential-Time Algorithms

Digital Object Identifier 10.4230/LIPIcs.SoCG.2017.11

1 Introduction

The study of terrains, also known as x-monotone polygonal chains, has attracted widespread and growing interest over the last few decades in the field of Discrete Computational Geometry. A terrain is a graph where each vertex v_i, $1 \leq i \leq n$, is associated with a point (x_i, y_i) on the two-dimensional Euclidean plane such that $x_1 < x_2 < \ldots < x_n$, and the edge-set is

* The research leading to these results received funding from the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreement no. 306992 (S. Saurabh). Part of this work was done while F. V. Fomin and M. Zehavi were visiting the Simons Institute for the Theory of Computing.

$E = \{\{v_i, v_{i+1}\} : 1 \leq i \leq n\}$. In the Terrain Guarding problem the task is to decide whether one can place guards on at most k points of a given terrain such that each point on the terrain is seen by at least one guard. Here, we say that a point p sees a point q if no point of the line segment $\overline{pq}$ is strictly below T. The Terrain Guarding problem arises in a wide-variety of applications relevant to the design of various communication technologies such as cellular telephony and line-of-sight transmission networks for radio broadcasting. It also arises in applications of coverage of highways, streets and walls with street lights or security cameras [3, 14].

The visibility graphs of terrains exhibit unique properties which render the complexity of the Terrain Guarding problem difficult to elucidate. Some of these properties have already been observed in 1995 by Abello *et al.* [1], and some of them remain unknown despite recent advances to identify them [13]. Indeed, the Terrain Guarding problem has been extensively studied since 1995, when an NP-hardness proof was claimed but never completed by Chen *et al.* [5]. Almost 10 years later King and Krohn [23] finally showed that this problem is NP-hard.

Particular attention has been given to the Terrain Guarding problem from the viewpoint of approximation algorithms. In 2005, Ben-Moshe *et al.* [3] obtained the first constant-factor approximation algorithm for Terrain Guarding. Afterward, the approximation factor was gradually improved in [6, 22, 12], until a PTAS was proposed by Gibson *et al.* [16]. Recently, Friedrichs *et al.* [14] showed that even if the terrain is continuous, the Terrain Guarding problem still admits a PTAS.

The Terrain Guarding problem has also gained interest due to its deceptive resemblance to the Art Gallery problem, where instead of a terrain, it is necessary to guard a polygon. The Art Gallery problem was introduced by Klee in 1976, and it is arguably one of the most well-known problems in Discrete Computational Geometry. For more information on the Art Gallery problem, we refer to the books dedicated to its study [26, 28, 18]. Note that the Art Gallery problem does not admit a subexponential-time algorithm. Indeed, the known NP-hardness reduction for the Art Gallery problem, even when restricted to orthogonal polygons, reduces a 3-SAT instance on n variables and m clauses to an instance of Art Gallery with $\mathcal{O}(n+m)$ vertices [27, 26]. This reduction combined with the Exponential Time Hypothesis (ETH) [19, 7] implies the following result.

▶ **Corollary 1** (Folklore). *Unless ETH fails, there is no algorithm for* Art Gallery*, even when restricted to orthogonal polygons, that achieves running time of* $2^{o(n)}$*. That is, the* Art Gallery *problem does not admit a subexponential-time algorithm.*

In the parameterized setting, where n is the number of vertices in the polygon and k is the number of guards, clearly one can design an algorithm for the Art Gallery problem running in time $n^{\mathcal{O}(k)}$ by enumerating all subsets of vertices of size at most k. Interestingly, by the very recent result of Bonnet and Miltzow [4] this trivial brute-force algorithm is essentially optimal. More precisely, they proved that an algorithm solving Art Gallery in time $f(k) \cdot n^{o(k/\log k)}$ for any function f would imply that the ETH fails. The reduction given in [4] also implies that Art Gallery is W[1]-hard parameterized by k. Thus it is highly unlikely that Art Gallery is fixed-parameter tractable (FPT).

Orthogonal Terrain Guarding is a problem of independent interest that is a special case of Terrain Guarding. In this problem, the terrain is orthogonal: for each vertex v_i, $2 \leq i \leq n-1$, either both $x_{i-1} = x_i$ and $y_i = y_{i+1}$ or both $y_{i-1} = y_i$ and $x_i = x_{i+1}$. In other words, each edge is either a horizontal line segment or a vertical line segment, and each vertex is incident to at most one horizontal edge and at most one vertical edge. The Orthogonal Terrain Guarding problem has already been studied from the perspective

of algorithms theory [20, 24, 25, 11]. Katz and Roisman [20] gave a relatively simple 2-approximation algorithm for the the problem of guarding all vertices of an orthogonal terrain by vertices. Recently, Lyu and Üngör improved upon this result by developing a linear-time 2-approximation algorithm for Orthogonal Terrain Guarding. The papers [25] and [11] studied restrictions under which Orthogonal Terrain Guarding can be solved in polynomial time.

While by now we have quite satisfactory understanding of the approximability of Terrain Guarding, the parameterized hardness of this problem is unknown. Currently, the most fundamental open questions regarding the complexity of the Terrain Guarding problem are the following:

- Does Terrain Guarding admit a subexponential-time algorithm?
- Is Terrain Guarding FPT with respect to k?

Indeed, King and Krohn [23] state that "the biggest remaining question regarding the complexity of Terrain Guarding is whether or not it is FPT". Moreover, interest in the design of efficient, exact exponential-time algorithms for this problem has been expressed at workshops such as the Lorentz Workshop on Fixed-Parameter Computational Geometry [15]. To the best of our knowledge, the only work which is somewhat related to the second question is the one by Khodakarami *et al.* [21], who introduced the parameter "the depth of the onion peeling of a terrain" and showed that Terrain Guarding is FPT with respect to this parameter.

In this paper, we address both of these questions. First, we completely resolve the first question by designing a subexponential-time algorithm for Terrain Guarding in both discrete and continuous domains. For this purpose, we develop an $n^{\mathcal{O}(\sqrt{k})}$-time algorithm for Terrain Guarding in discrete domains. Friedrichs *et al.* [14] proved that given an instance of Terrain Guarding in a continuous domain, one can construct (in polynomial time) an equivalent instance of Terrain Guarding in a discrete domain. More precisely, given an instance $(T = (V, E), k)$ of Terrain Guarding in a continuous domain, Friedrichs *et al.* [14] designed a discretization procedure that outputs an instance $(T' = (V', E'), k)$ of Terrain Guarding in a discrete domain such that $(T = (V, E), k)$ is a yes-instance if and only if $(T' = (V', E'), k)$ is a yes-instance. Unfortunately, this reduction blows up the number of vertices of the terrain to $\mathcal{O}(n^3)$, and therefore the existence of a subexponential-time algorithm for Terrain Guarding in discrete domains does not imply that there exists such an algorithm for Terrain Guarding in continuous domains. However, observe that the reduction *does not change* the value of the parameter k. Thus, since we solve Terrain Guarding in discrete domains in time $n^{\mathcal{O}(\sqrt{k})}$ rather than $n^{\mathcal{O}(\sqrt{n})}$, we are able to deduce that Terrain Guarding in continuous domains is solvable in time $n^{\mathcal{O}(\sqrt{k})}$. Observe that, in both discrete and continuous domains, it can be assumed that $k \leq n$: to guard all of the points that lie on a terrain, it is sufficient to place guards only on the vertices of the terrain. Hence, when we solve Terrain Guarding in continuous domains, we assume that $k \leq n$ where n is the number of vertices of the input continuous terrain and *not* of the discrete terrain outputted by the reduction. The next theorem summarizes our algorithmic contribution.

▶ **Theorem 2.** Terrain Guarding *in both discrete and continuous domains is solvable in time* $n^{\mathcal{O}(\sqrt{k})}$. *Thus, it is also solvable in time* $n^{\mathcal{O}(\sqrt{n})}$.

Observe that our result, Theorem 2, demonstrates an interesting dichotomy in the complexities of Terrain Guarding and the Art Gallery problem: Corollary 1 implies that the Art Gallery problem does not admit an algorithm with running time $2^{o(n)}$, while

Terrain Guarding in both discrete and continuous domains is solvable in time $2^{\mathcal{O}(\sqrt{n}\log n)}$. When we measure the running time in terms of both n and k, the Art Gallery problem does not admit an algorithm with running time $f(k)\cdot n^{o(k/\log k)}$ for any function f [4], while Terrain Guarding in both discrete and continuous domains is solvable in time $n^{\mathcal{O}(\sqrt{k})}$.

Our solution is based on the definition of a planar graph that has a small domination number and which captures *both* the manner in which a hypothetical solution guards the terrain and some information on the layout of the terrain itself. Having this planar graph, we are able to "guess" separators whose exploitation, which involves additional guesses guided by the structure of the graph, essentially results in a divide-and-conquer algorithm. The design of the divide-and-conquer algorithm is also nontrivial since given our guesses, it is not possible to divide the problem into two simpler subproblems in the obvious way – that is, we cannot divide the terrain into two disjoint subterrains that can be handled separately. We overcome this difficulty by dividing not the terrain itself, but a set of points of interest on the terrain.

We also shed light on the second question by showing that Orthogonal Terrain Guarding of vertices of the orthogonal terrain with vertices is FPT with respect to the parameter k. More precisely, we obtain the following result.

▶ **Theorem 3.** Orthogonal Terrain Guarding *of vertices of the terrain with vertices is solvable in time* $k^{\mathcal{O}(k)}\cdot n^{\mathcal{O}(1)}$.

Our algorithm is based on new insights into the structure of orthogonal terrains, particularly into the relations between their left and right reflex and convex vertices. We integrate these insights in the design of an algorithm that is based on the proof that one can ignore "exposed vertices", which are vertices seen by too many vertices of a specific type, greedy localization, and a non-trivial branching strategy that we call "double-branching". We conclude the introduction by posing the following open problems: Are Terrain Guarding and Orthogonal Terrain Guarding in continuous domains FPT?

2 Preliminaries

For a positive integer k, we use $[k]$ as a shorthand for $\{1,2,\ldots,k\}$.

Graphs. We use standard notation and terminology from the book of Diestel [9] for graph-related terms which are not explicitly defined here. We only consider simple undirected graphs. Given a graph H, $V(H)$ and $E(H)$ denote its vertex-set and edge-set, respectively. Given a subset $U\subseteq V(H)$, the subgraph of H induced by U is denoted by $H[U]$. A *dominating set* of H is a subset $S\subseteq V(H)$ such that each vertex in $V(H)$ either belongs to S or has a neighbor in S. The *domination number* of H, denoted by $\gamma(H)$, is the minimum size of a dominating set of H. A *clique cover* of H is a partition $(V_1,V_2,\ldots,V_t)$ of $V(H)$ for some $t\in\mathbb{N}$ such that for any $i\in[t]$, $H[V_i]$ is a clique. The size of the clique cover is t. The *clique cover number* of H, denoted by $\kappa(H)$, is the minimum size of a clique cover of H. An *independent set* of H is a subset $U\subseteq V(H)$ such that there do not exist two vertices in U that are neighbors in H. The *independence number* of H, denoted by $\alpha(H)$, is the maximum size of an independent set of H. A *chordal graph* is a graph that has no induced cycle on more than three vertices. In the context of chordal graphs, we will need to rely on the following well-known results.

▶ **Theorem 4** ([17])**.** *Let H be a chordal graph. Then*

- *A clique cover of H of minimum size can be found in linear time.*
- *An independent set of H of maximum size can be found in linear time.*
- $\kappa(H)=\alpha(H)$.

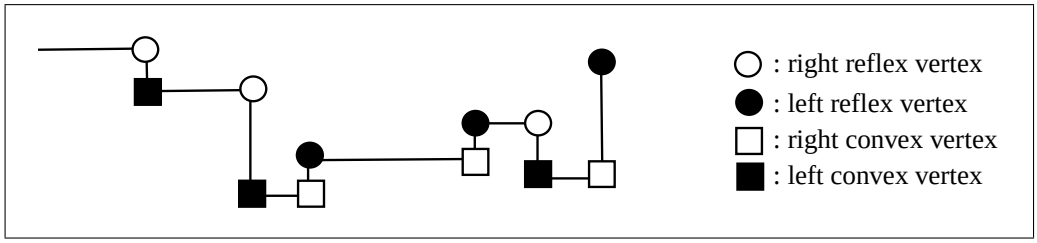


Figure 1 Reflex and convex vertices.

Terrains. A *1.5-dimensional terrain* $T = (V, E)$, or *terrain* for short, is a graph on vertex-set $V = \{v_1, v_2, \ldots, v_n\}$ where each vertex v_i is associated with a point (x_i, y_i) on the two-dimensional Euclidean plane such that $x_1 < x_2 < \ldots < x_n$, and the edge-set is $E = \{\{v_i, v_{i+1}\} : i \in [n-1]\}$. We say that a point p *sees* a point q if every point of the line segment $\overline{pq}$ is either on or above T. Note that if a point p sees a point q, then the point q sees the point p as well. More generally, we say that a set of points P *sees* a set of points Q if each point in Q is seen by at least one point in P.

An *orthogonal terrain*, also known as a *rectilinear terrain*, is a terrain $T = (V, E)$ where for each vertex v_i, $2 \leq i \leq n-1$, either both $x_{i-1} = x_i$ and $y_i = y_{i+1}$ or both $y_{i-1} = y_i$ and $x_i = x_{i+1}$. In other words, an orthogonal terrain is a terrain where each edge is either a horizontal line segment or vertical line segment, and each vertex is incident to at most one horizontal edge and at most one vertical edge. A vertex v_i, $2 \leq i \leq n-1$ belongs to one of the four following categories: if $x_i = x_{i+1}$ and $y_i > y_{i+1}$, it is a *right reflex vertex*; if $x_i = x_{i+1}$ and $y_i < y_{i+1}$, it is a *right convex vertex*; if $x_i = x_{i-1}$ and $y_i > y_{i-1}$, it is a *left reflex vertex*; if $x_i = x_{i-1}$ and $y_i < y_{i-1}$, it is a *left convex vertex*. Moreover, if $x_1 = x_2$ and $y_1 > y_2$, v_1 is a right reflex vertex; if $x_1 = x_2$ and $y_1 < y_2$, it is a right convex vertex; otherwise it is a left convex vertex. Symmetrically, if $x_n = x_{n-1}$ and $y_n > y_{n-1}$, v_n is a left reflex vertex; if $x_n = x_{n-1}$ and $y_n < y_{n-1}$, it is a left convex vertex; otherwise it is a right convex vertex. We also say that a vertex is a *reflex vertex* if it is either a left reflex vertex or a right reflex vertex, and otherwise it is a *convex vertex*. Furthermore, we say that left reflex/convex vertices are *opposite* to right reflex/convex vertices. An illustrative example of these notions is given in Fig. 1

Let $T = (V, E)$ be a terrain and let U be a subset of V. We use $\textsc{vis}(U)$ to denote the set containing every vertex in V that is seen by at least one vertex in U. In case $U = \{u\}$, we abuse notation and write $\textsc{vis}(u)$ to refer to $\textsc{vis}(U)$. We use $\textsc{cut}(U)$ to denote the set of (maximal) subterrains of T that result from the removal of the vertices in U. That is, $\textsc{cut}(U)$ is the set of each subterrain $T' = (V', E')$ for which there exist $i < j$ such that $V' = \{v_i, v_{i+1}, \ldots, v_j\} \subseteq V \setminus U$, either $i = 1$ or $v_{i-1} \in U$, and either $j = n$ or $v_{j+1} \in U$. An illustrative example of this notation is given in Fig. 2. Given a subset $X \subseteq V$ and subterrain $T' = (V', E')$, we define $X[T'] = X \cap V'$. Moreover, given a set of terrains $\mathcal{T}$, we let $X[\mathcal{T}]$ be set of vertices that is the union of the sets in $\{X[T'] : T' \in \mathcal{T}\}$.

Terrain Guarding Problems. The decision version of the (Discrete) Terrain Guarding problem is defined as follows. Its input consists of a terrain $T = (V, E)$ on n vertices and a positive integer $k \leq n$, and the objective is to determine whether there is a subset $S \subseteq V$ of size at most k that sees V. We say that such a subset S is a solution. In the special case where the input terrain is an orthogonal terrain, the problem is known as the Orthogonal Terrain Guarding problem.

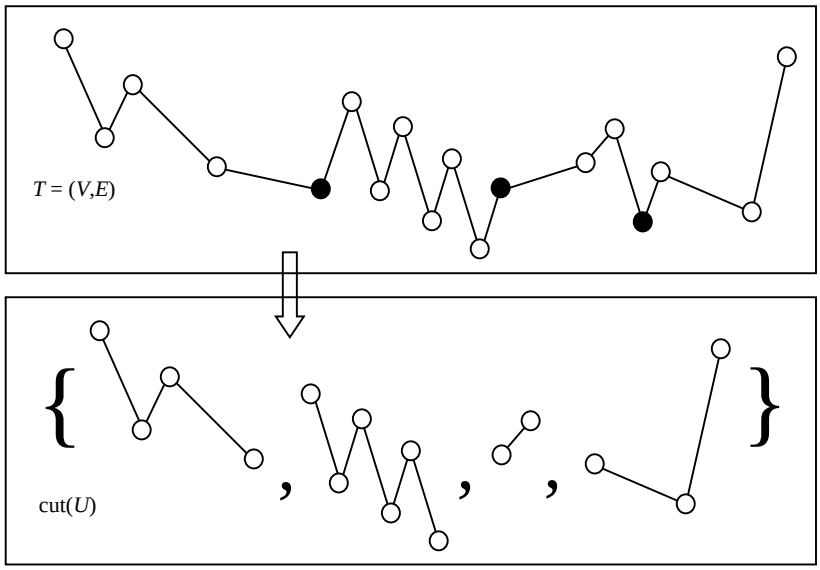


Figure 2 The result of the operation CUT(U) where U is the set of black vertices.

The Terrain Guarding problem is also defined in the context of continuous domains, in which case it is called the Continuous Terrain Guarding problem. The input for the Continuous Terrain Guarding problem is the same as the input for the Discrete Terrain Guarding problem. We say that a point lies on the terrain T if it is either a vertex in V or a point on an edge between two adjacent vertices. The objective is to determine whether there is a subset of points of size at most k that lie on T and which see every point that lies on T.

To develop our algorithms for Discrete Terrain Guarding, it will be more convenient to solve a problem generalizing Discrete Terrain Guarding, that we call Annotated Terrain Guarding. Roughly speaking, Annotated Terrain Guarding is the variant of Discrete Terrain Guarding where one cannot place a "guard" on any vertex, but only on vertices from a given set G, and where it is not necessary to "cover" all of the vertices in V, but only those belonging to a given set C. Formally, the input consists of a terrain $T = (V, E)$ on n vertices, a positive integer $k \leq n$, and subsets $G, C \subseteq V$. The objective is to determine whether there is a subset $S \subseteq G$ of size at most k that sees C. We say that such a subset S is a solution. Clearly, Terrain Guarding is the special case of Annotated Terrain Guarding where $G = C = V$. We will refer to the special case where the input terrain is an orthogonal terrain as the Annotated Orthogonal Terrain Guarding problem.

Treewidth. A *tree decomposition* of a graph H is a pair (D, β), where D is a rooted tree and $\beta : V(D) \to 2^{V(G)}$ is a mapping that satisfies the following conditions.

- For each vertex $v \in V(H)$, the set $\{d \in V(D) : v \in \beta(d)\}$ induces a nonempty and connected subtree of D.
- For each edge $\{v, u\} \in E(H)$, there exists $d \in V(D)$ such that $\{v, u\} \subseteq \beta(d)$.

A vertex d in $V(D)$ is called a node, and the set $\beta(d)$ is called the bag at d. We let DESCENDANTS(d) denote the set of descendants of d in D. The width of (D, β) is the size of the largest bag minus one (i.e., $\max_{d \in V(D)} |\beta(d)| - 1$). The *treewidth* of H, denoted by $tw(H)$, is the minimum width among all possible tree decompositions of H.

Standard arguments on trees, see e.g. [7, Lemma 7.20], imply the correctness of the following observation.

▶ **Observation 5.** *Let (D, β) be a tree decomposition of a graph H where D is a binary tree, and let S be a subset of $V(H)$. Then, there exists a node $d \in V(D)$ such that $|S|/3 \leq |\bigcup_{d' \in \text{DESCENDANTS}(d)} \beta(d') \cap S|$ and $|\bigcup_{d' \in \text{DESCENDANTS}(d) \setminus \{d\}} \beta(d') \cap S| \leq 2|S|/3$.*

Parameterized Complexity. In Parameterized Complexity each problem instance is accompanied by a parameter k. A central notion in this field is the one of *fixed-parameter tractability (FPT)*. This means, for a given instance (I, k), solvability in time $f(k)|I|^{\mathcal{O}(1)}$ where f is some function of k. For more information on Parameterized Complexity we refer the reader to monographs such as [10, 7].

Bit Vectors. A *t-length bit vector* is a vector $\overline{v} = (v_1, v_2, \ldots, v_t)$ such that for any $i \in [t]$, $v_i \in \{0, 1\}$. Given two t-length bit vectors $\overline{v}$ and $\overline{u}$, the *Hamming distance* between them, denoted by $H(\overline{v}, \overline{u})$, is the number of indices $i \in [t]$ such that $v_i \neq u_i$.

3 Subexponential Algorithm

In this section we prove that Annotated Terrain Guarding can be solved in time $n^{\mathcal{O}(\sqrt{n})}$. In fact, we obtain a somewhat stronger result:

▶ **Theorem 6.** Annotated Terrain Guarding *is solvable in time $n^{\mathcal{O}(\sqrt{k})}$.*

Since Discrete Terrain Guarding is a special case of Annotated Terrain Guarding, we derive the following result.

▶ **Corollary 7.** Discrete Terrain Guarding *is solvable in time $n^{\mathcal{O}(\sqrt{k})}$.*

We also derive the following result.

▶ **Corollary 8.** Continuous Terrain Guarding *is solvable in time $n^{\mathcal{O}(\sqrt{k})}$.*

Throughout this section, we let $(T = (V, E), n, k, G, C)$ denote the input instance of Annotated Terrain Guarding. First, in Section 3.1, we carefully define a planar graph P_S that captures relations between a hypothetical solution and the set C. Then, in Section 3.2, we rely on properties of P_S to show that there exists a partition of $G \cup C$ into two sets which rarely "alternate" along the terrain and which are both relatively "small". In Section 3.3 we show how the existence of this partition allows us to design an algorithm for Annotated Terrain Guarding. The algorithm is based on the method of divide-and-conquer, although the subproblems we obtain are not associated with subterrains smaller than the original one.

3.1 The Planar Graph P_S

In this section we assume that the input instance is a yes-instance. Let S be some hypothetical solution, that is, a subset of G of size at most k that sees C. We define three sets of edges:

- The set E_1 contains an edge $\{v_i, v_j\}$ between any two vertices $v_i, v_j \in S \cup C$ such that there is no $v_t \in S \cup C$ with $i < t < j$.
- The set E_2 contains an edge $\{v_i, v_j\}$ between any two vertices $v_i \in S$ and $v_j \in C \cap \text{VIS}(v_i)$ such that $i < j$ and there is no $v_t \in S$ with $t < i$ and $v_j \in \text{VIS}(v_t)$.
- The set E_3 contains an edge $\{v_i, v_j\}$ between any two vertices $v_i \in S$ and $v_j \in C \cap \text{VIS}(v_i)$ such that $j < i$ and there is no $v_t \in S$ with $i < t$ and $v_j \in \text{VIS}(v_t)$.

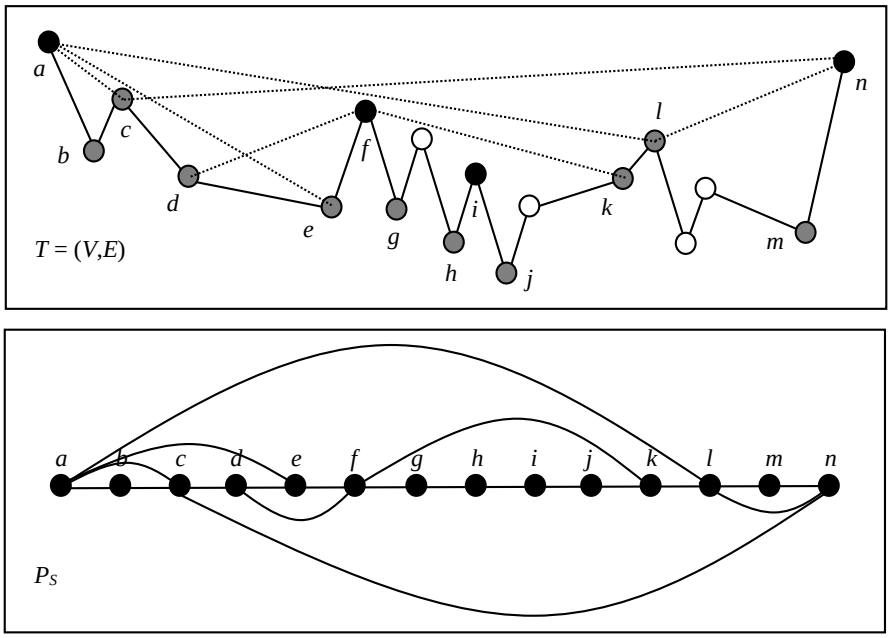


Figure 3 A sketch of the embedding of the planar graph P_S where $S = \{a, f, i, n\}$ (black vertices) and $C = \{b, c, d, e, g, h, j, k, l, m\}$ (grey vertices). Here, $E_1 = \{\{a, b\}, \{b, c\}, \{c, d\}, \{d, e\}, \{e, f\}, \{f, g\}, \{g, h\}, \{h, i\}, \{i, j\}, \{j, k\}, \{k, l\}, \{l, m\}, \{m, n\}\}$, $E_2 = \{\{a, b\}, \{a, c\}, \{a, e\}, \{a, l\}, \{f, g\}, \{f, k\}, \{i, j\}\}$ and $E_3 = \{\{c, n\}, \{d, f\}, \{e, f\}, \{h, i\}, \{l, n\}, \{m, n\}\}$.

We define P_S as the graph on the vertex set $V(P_S) = S \cup C$ and the edge set $E(P_S) = E_1 \cup E_2 \cup E_3$. Denote the vertices in $V(P_S)$ by $u_1, u_2, \ldots, u_{|V(P_S)|}$ with respect to the order (from left to right) in which they appear on the terrain T. An illustrative example is given in Fig. 3. We show that P_S isa planar graph using techniques similar to that in [16]. To show that P_S is a planar graph, we will need the following result, known as the Order Claim, which was proved in [3].

▶ **Lemma 9** ([3]). *Let v_i, v_j, v_t, v_r be four vertices in V such that $i < t < j < r$. If v_i sees v_j and v_t sees v_r, then v_i sees v_r.*

Our proof also relies on the following result.

▶ **Lemma 10.** *There is no pair of edges $\{u_i, u_j\}, \{u_t, u_r\} \in E_2$ such that $i < t < j < r$. Symmetrically, there is no pair of edges $\{u_i, u_j\}, \{u_t, u_r\} \in E_3$ such that $i < t < j < r$.*

▶ **Lemma 11.** *The graph P_S is a planar graph.*

Let us remind that we use $tw(H)$ to denote the treewidth and $\gamma(H)$ the dominating number of a graph H. The proof of the following result is given in [2], (see also [8]).

▶ **Lemma 12** ([2]). *There exists a constant c such that for any planar graph H, $tw(H) \leq c\sqrt{\gamma(H)}$.*

From now on, we let c denote the constant mentioned in this lemma. We are now ready to bound the treewidth of P_S.

▶ **Lemma 13.** *$tw(P_S) \leq c\sqrt{k}$.*

3.2 The Existence of Exploitable Partitions

In this section we continue to assume that the input instance is a yes-instance, and again we let S be some hypothetical solution. Given a subset $U \subseteq G \cup C$ and a mapping $f : \textsc{cut}(U) \to \{0,1\}$, denote $\textsc{cut}(f,0) = \{T' \in \textsc{cut}(U) : f(T') = 0\}$ and $\textsc{cut}(f,1) = \{T' \in \textsc{cut}(U) : f(T') = 1\}$. Thus $\textsc{cut}(f,0)$ and $\textsc{cut}(f,1)$ form a partition of the set of subterrains $\textsc{cut}(U)$. Moreover, given a subset $X \subseteq V$, denote $X[f,0] = X[\textsc{cut}(f,0)]$ and $X[f,1] = X[\textsc{cut}(f,1)]$. Roughly speaking, we will use such a carefully chosen set U and a function f to achieve the following goal. The set U will partition the terrain T into subterrains, but these subterrains do not necessarily correspond to independent subproblems. Yet, the function $f : \textsc{cut}(U) \to \{0,1\}$ will partition $\textsc{cut}(U)$ into two sets of subterrains such that each of them will be independent (in a certain exploitable sense) and relatively small.[1]

To make the above mentioned divide-and-conquer approach work, we need the following definition. Each of its properties will be exploited in the following section.

▶ **Definition 14.** Let $U \subseteq G \cup C$, and let f be a mapping $f : \textsc{cut}(U) \to \{0,1\}$. We say that the pair (U,f) is *good* if the following conditions are satisfied.

1. $|U| \leq 2c\sqrt{k}$.
2. $S \cap U$ sees U.
3. $|S[f,0]|, |S[f,1]| \leq \frac{2}{3}|S|$.
4. $S \cap (U \cup G[f,0])$ sees $C[f,0]$; $S \cap (U \cup G[f,1])$ sees $C[f,1]$.

Roughly speaking, the motivation behind the introduction of each property is the following. The first property implies that the set U is small, and therefore it will be possible to "guess" it. The second property implies that guards places on set S see all of the vertices of U. The third property implies that the two subproblems are small in a narrow yet exploitable sense: each subproblem will include the entire terrain and therefore its size will be roughly the same as the size of the original problem, yet the number of guards one should place to solve it will be much smaller than the number of guards one should place to solve to the original problem. We briefly note that each subproblem will be associated with the entire terrain, including vertices on which we cannot place guards and which are already covered/may not be covered, because such vertices play a role in blocking the lines of sights between other vertices on which we can place guards and vertices that should be covered. The last property implies that the subproblems are independent in the sense that we do not need to cover a vertex of one subproblem using a guard that we place when we solve the other subproblem.

The rest of this section focuses on the proof of the existence of a good pair.

▶ **Lemma 15.** *There exists a good pair* (U,f).

3.3 Divide-and-Conquer

In this section we rely on Lemma 15 to design an algorithm, based on the method of divide-and-conquer, that solves Annotated Terrain Guarding in time $n^{\mathcal{O}(\sqrt{k})}$.

We start by presenting an algorithmic interpretation of Lemma 15. To this end, we need the following definition.

[1] In our algorithm, the terrain itself will *not* change – the partition is only meant to control the annotations associated with it).

▶ **Definition 16.** A tuple (U, U', f, k_0, k_1) is *relevant* if the following conditions are satisfied.
1. $U \subseteq G \cup C$ satisfies $|U| \leq 2c\sqrt{k}$.
2. $U' \subseteq U \cap G$ sees U.
3. $f : \text{CUT}(U) \to \{0, 1\}$.
4. $k_0, k_1 \in \{0\} \cup [\lfloor 2k/3 \rfloor]$; $k_0 + k_1 + |U'| = k$.

▶ **Lemma 17.** *One can compute in time $n^{\mathcal{O}(\sqrt{k})}$ a collection Q of relevant tuples whose size is bounded by $n^{\mathcal{O}(\sqrt{k})}$ such that if the input instance is a yes-instance, then there exists a solution S of size k and at least one tuple in Q having the following properties.*
1. *(U, f) is a good pair (with respect to S).*
2. *$U' = U \cap S$.*
3. *$|S[f, 0]| \leq k_0$; $|S[f, 1]| \leq k_1$.*

Let Q be a collection of tuples given by Lemma 17. With each tuple $(U, U', f, k_0, k_1) \in Q$, we associate a pair of instances of Annotated Terrain Guarding, $(I_0(U, U', f, k_0), I_1(U, U', f, k_0))$, as follows: $I_0(U, U', f, k_0) = (T, k_0, G_0, C_0)$ where $G_0 = G[f, 0]$ and $C_0 = C[f, 0] \setminus \text{VIS}(U')$; $I_1(U, U', f, k_1) = (T, k_1, G_1, C_1)$ where $G_1 = G[f, 1]$ and $C_1 = C[f, 1] \setminus \text{VIS}(U')$. We set $I(Q) = \{(I_0(U, U', f, k_0), I_1(U, U', f, k_1)) : (U, U', f, k_0, k_1) \in Q\}$.

▶ **Lemma 18.** *The input instance is a yes-instance if and only if there exists a pair (I_0, I_1) in $I(Q)$ such that both I_0 and I_1 are yes-instances.*

We are now ready to prove Theorem 6.

Proof of Theorem 6. We present a recursive algorithm that solves Annotated Terrain Guarding in the desired time. At each stage, if $k \leq 10c$, it uses brute-force to solve the instance in polynomial time. Otherwise, it computes the set $I(Q)$ where Q is given by Lemma 17. For each pair $(I_0, I_1) \in I(Q)$, it calls itself recursively twice: once with the input I_0 and once with the input I_1. If the answers to both inputs I_0 and I_1 are positive, it returns a positive answer. At the end, if no pair resulted in two positive answers, it returns a negative answer. By Lemma 18, the algorithm returns the correct answer.

By Lemma 17, we have that $|I(Q)| = |Q| = n^{\mathcal{O}(\sqrt{k})}$. Consider some pair $(I_0(U, U', f, k_0), I_1(U, U', f, k_1))$ in $I(Q)$. By the fourth property in Definition 16, $k_0, k_1 \leq 2k/3$. Let $t(k, n)$ denote the running time of our algorithm. Then, there exists a constant d such that $t(k, n) \leq n^{d\sqrt{k}} \cdot t(2k/3)$. Let p be the largest number smaller than $10c$ such that there exists q for which $\sqrt{(2/3)^q k} = p$. Thus, $t(n, k) = n^{d(\sqrt{k} + \sqrt{(2/3)k} + \sqrt{(2/3)^2 k} + \cdots + p)} = n^{\mathcal{O}(\sqrt{k})}$. ◀

4 Parameterized Algorithm for Orthogonal Terrain Guarding

In this section we prove that Discrete Orthogonal Terrain Guarding is FPT:

▶ **Theorem 19.** Discrete Orthogonal Terrain Guarding *is solvable in time $k^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$.*

Throughout this section, we let $\mathcal{R}_\ell$, $\mathcal{R}_r$, $\mathcal{C}_\ell$ and $\mathcal{C}_r$ denote the sets of left reflex vertices, right reflex vertices, left convex vertices and right convex vertices, respectively. We further let $\mathcal{R} = \mathcal{R}_\ell \cup \mathcal{R}_r$ and $\mathcal{C} = \mathcal{C}_\ell \cup \mathcal{C}_r$ denote the sets of reflex vertices and convex vertices, respectively. Katz and Roisman [20] showed that an instance $(T = (V, E), n, k)$ of Discrete Orthogonal Terrain Guarding is a yes-instance if and only if the instance $(T = (V, E), n, k, \mathcal{R}, \mathcal{C})$ of Annotated Orthogonal Terrain Guarding is a yes-instance. In other words, it is sufficient to place guards only on reflex vertices and to guard only convex vertices. Therefore,

we say that an instance $(T = (V, E), n, k, G, C)$ of Annotated Orthogonal Terrain Guarding is *relevant* if $\mathcal{R} = G$ and $\mathcal{C} = C$, and in the rest of this section, we focus on the proof of the following result.

▶ **Lemma 20.** *Relevant instances of* Annotated Orthogonal Terrain Guarding *are solvable in time* $k^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$.

First, in Section 4.1, we show that vertices seen by too many vertices of the opposite type can actually be ignored as they will be guarded even if we do not explicitly demand it. In Section 4.2, we describe solutions via clique covers in chordal graphs. This description will allow us to find a set of size at most k', for any $k' \leq k$, that guards a subset of left convex vertices of interest via left reflex vertices, or provide a witness for the non-existence of such a set. Next, in Section 4.3, we examine the Hamming distance between vectors that describe the way in which convex vertices can be guarded, and show that this distance cannot be too large. Finally, in Section 4.4, we integrate the results obtained in the three previous sections into the design of our double-branching parameterized algorithm for relevant instances of Annotated Orthogonal Terrain Guarding.

4.1 Ignoring Exposed Vertices

In this section, we handle seemingly problematic vertices, which comply with the following definition.

▶ **Definition 21.** A vertex $v \in V(T)$ is *exposed* if it is a convex vertex seen by more than $k + 2$ opposite reflex vertices.

We let E denote the set of exposed vertices, $E_\ell = \mathcal{C}_\ell \cap E$ and $E_r = \mathcal{C}_r \cap E$. The efficiency of the second phase of our double-branching procedure, presented in Section 4.4, relies on the assumption that C does not contain exposed vertices. However, $C = \mathcal{C}$, and the set $\mathcal{C}$ may very well contain exposed vertices. We circumvent this difficulty by showing that vertices in E can actually be ignored. To prove this claim, we need the following notation. Given a vertex $v \in E_\ell$, we let $u_1^v, u_2^v, \ldots, u_{k+3}^v$ denote the $k + 3$ leftmost right reflex vertices that see v, sorted from left-to-right by the order in which they lie on T (see Fig. 4(A)). Symmetrically, given a vertex $v \in E_r$, we let $u_1^v, u_2^v, \ldots, u_{k+3}^v$ denote the $k + 3$ rightmost left reflex vertices that see v, sorted from right-to-left by the order in which they lie on T (see Fig. 4(B)). By the definition of an orthogonal terrain, we have the following observation.

▶ **Observation 22.** *For each vertex $v \in E$, the x-coordinate of u_1^v is the same as the one of v and the y-coordinate of u_1^v is larger than the one of v. For any $2 \leq i \leq k + 3$, the y-coordinate of u_i^v is the same as the one of v, and if $v \in E_\ell$ ($v \in E_r$), the x-coordinate of u_i^v is larger (resp. smaller) than the one of v.*

In the two following lemmata, we continue to examine vertices in E.

▶ **Lemma 23.** *For each vertex $v \in E_\ell$ and index $2 \leq i \leq k + 2$, there exists a vertex in $\mathcal{C}_\ell \setminus E_\ell$ that lies between u_i^v and u_{i+1}^v. Symmetrically, for each vertex $v \in E_r$ and index $2 \leq i \leq k + 2$, there exists a vertex in $\mathcal{C}_r \setminus E_r$ that lies between u_i^v and u_{i+1}^v.*

▶ **Lemma 24.** *Let S be a solution to $(T, n, k, \mathcal{R}, \mathcal{C} \setminus E)$. For each vertex $v \in E_\ell$, there exists an index $2 \leq i \leq k + 2$ and a vertex that lies strictly between u_i^v and u_{i+1}^v which is seen by a vertex in S to the right of u_{i+1}^v. Symmetrically, for each vertex $v \in E_r$, there exists an index $2 \leq i \leq k + 2$ and a vertex that lies strictly between u_i^v and u_{i+1}^v which is seen by a vertex in S to the left of u_{i+1}^v.*

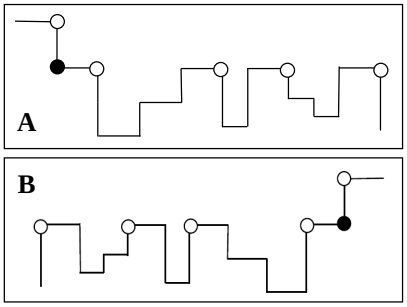


Figure 4 Exposed vertices are black, and reflex vertices of the opposite type that see them are white. The parameter k is 2.

We are now ready to show that vertices in E can be ignored.

▶ **Lemma 25.** *$(T, n, k, \mathcal{R}, \mathcal{C})$ is a yes-instance if and only if $(T, n, k, \mathcal{R}, \mathcal{C} \setminus E)$ is a yes-instance.*

4.2 Describing Solutions via Clique Covers in Chordal Graphs

Katz and Roisman [20] defined two graphs that aim to capture relations between convex vertices. The first graph, G_L, is defined as follows: $V(G_L) = \mathcal{C}_L$ and $E(G_L) = \{\{v, u\} : \text{there exists a vertex in } \mathcal{R}_L \text{ that sees both } v \text{ and } u\}$. The second one, G_R, is defined symmetrically: $V(G_R) = \mathcal{C}_R$ and $E(G_R) = \{\{v, u\} : \text{there exists a vertex in } \mathcal{R}_R \text{ that sees both } v \text{ and } u\}$. For these graphs, Katz and Roisman [20] proved the following useful result.

▶ **Lemma 26** ([20])**.** *The graph G_L satisfies the following properties.*

- *The graph G_L is a chordal graph.*
- *For any subset $U \subseteq V(G_L)$, $G_L[U]$ is a clique if and only if there exists a left reflex vertex that sees all of the vertices in U.*

The symmetric claim holds for the graph G_R.

By relying on Lemma 26, Katz and Roisman [20] showed that one can decide in polynomial time whether there exists a subset $S \subseteq \mathcal{R}_L$ of size k' that sees $\mathcal{C}_L$. To design our double-branching procedure (in Section 4.4), we will need the following stronger claim.

▶ **Lemma 27.** *Let $U \subseteq \mathcal{C}_L$ and $k' \in \mathbb{N}$. Then, one can decide in polynomial time whether there exists a subset $S \subseteq \mathcal{R}_L$ of size k' that sees U. In case such a subset does not exist, one can find in polynomial time a subset $U' \subseteq U$ of size $k' + 1$ such that there does not exist a subset $S \subseteq \mathcal{R}_L$ of size k' that sees U'.*

Symmetrically, we obtain the following claim.

▶ **Lemma 28.** *Let $U \subseteq \mathcal{C}_R$ and $k' \in \mathbb{N}$. Then, one can decide in polynomial time whether there exists a subset $S \subseteq \mathcal{R}_R$ of size k' that sees U. In case such a subset does not exist, one can find in polynomial time a subset $U' \subseteq U$ of size $k' + 1$ such that there does not exist a subset $S \subseteq \mathcal{R}_R$ of size k' that sees U'.*

4.3 Hamming Distance

In this section, we associate vectors with subsets of $\mathcal{R}$, and then examine the Hamming distance between these vectors and a special vector. We start with the definition of the association. Here, we set $m = |\mathcal{C} \setminus E|$, and let $u_1, u_2, \ldots, u_m$ denote the vertices in $\mathcal{C} \setminus E$ sorted from left-to-right by the order in which they lie on T.

▶ **Definition 29.** Let $S \subseteq \mathcal{R}$ be a set that sees $\mathcal{C}$. Then, the vector *associated* with S is the m-length bit vector $(b_1, b_2, \ldots, b_m)$ such that $b_i = 0$ if and only if u_i is seen by a vertex in S that is a left reflex vertex.

In other words, the vector associated with a subset $S \subseteq \mathcal{R}$ that sees $\mathcal{C}$ indicates, for each vertex that we would like to guard, whether it is guarded by at least one left reflex vertex or only by right reflex vertices. Observe that by the definition of an orthogonal terrain, a reflex vertex can see at most two vertices of the opposite type:

▶ **Observation 30.** *Any reflex vertex v sees at most two convex vertices of the opposite type: one has the same x-coordinate as v and the other has the same y-coordinate as v.*

Next, we examine the Hamming distance between a vector associated with a solution and a special vector.

▶ **Lemma 31.** *Let S^* be a solution to $(T, n, k, \mathcal{R}, \mathcal{C} \setminus E)$, and let $\overline{b}^*$ be the m-length bit vector associated with S^*. Let $\overline{b}$ be the m-length bit vector $(b_1, b_2, \ldots, b_m)$ such that $b_i = 0$ if and only if u_i is a left convex vertex. Then, $H(\overline{b}^*, \overline{b}) \leq 2k$.*

4.4 Double-Branching

We are now ready to present $\mathsf{ALG}(T = (V, E), n, \mathcal{R}, C, \delta, k_\ell, k_r)$, our algorithm for relevant instances of Annotated Orthogonal Terrain Guarding. Initially, it is called with the arguments $C = \mathcal{C} \setminus E$, $\delta = 2k$, and every choice of $k_\ell, k_r \in [k]$ such that $k_\ell + k_r = k$. As the execution of the algorithm progresses, vertices are removed from C, and the values of k_ℓ, k_r and δ decrease. Note that there are only k choices of k_ℓ and k_r, and there exists a choice of k_ℓ and k_r such that if there exists a solution S, it holds that $|S \cap \mathcal{R}_\ell| = k_\ell$ and $|S \cap \mathcal{R}_r| = k_r$. Accordingly, and in light of Lemma 31, we say that the input instance (in the context of a pair (k_ℓ, k_r)) is *identifiable* if there exists a solution S such that $|S \cap \mathcal{R}_\ell| = k_\ell$, $|S \cap \mathcal{R}_r| = k_r$ and the Hamming distance between $\overline{b}$ and the vector associated with S is at most δ. Thus, to prove Lemma 20, it is sufficient to prove the following result.

▶ **Lemma 32.** $\mathsf{ALG}(T = (V, E), n, \mathcal{R}, C, \delta, k_\ell, k_r)$ *runs in time $k^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$, and returns YES if and only if the input instance is identifiable.*

The pseudocode of our algorithm is given in Algorithm 1. First, we argue that if the input instance is identifiable, then the algorithm returns YES. In this argument, we follow the pseudocode line-by-line, and also highlight the phases of our double-branching. In case $\delta < 0$, we return NO, since the Hamming distance between any two vectors is nonnegative. Next, suppose that $\delta \geq 0$. By Lemma 27, we may proceed by deciding in polynomial time whether $C \cap \mathcal{C}_L$ cannot be seen by any set of at most k_ℓ vertices from $\mathcal{R}_L$. If the answer is positive, by Lemma 27, we can compute in polynomial time a set $U \subseteq C \cap \mathcal{C}_L$ of size $k_\ell + 1$ that cannot be seen by any set of at most k_ℓ vertices from $\mathcal{R}_L$. In case the input instance is identifiable, there exists a vertex $v \in U$ that should be seen by a right reflex vertex. We try every option to identify the vertex v; this is the first phase of our double-branching. Then, we try every option to identify a vertex $u \in \mathcal{R}_R \cap \textsc{vis}(v)$ that should both see v and belong to a solution; this is the second phase of our double-branching. Since v is not exposed, there are at most $k + 2$ such options to consider. For each such option, we place a guard on u. Therefore, we decrement k_r by 1, remove the vertices in $\textsc{vis}(u)$ from C, and since at least one bit is flipped in $\overline{b}$, we also decrement δ by 1. For an identifiable input instance, we will have made correct choices in at least one of the paths in the branch-tree. Now, if the answer is negative, by performing the symmetric test with respect to the set $C \cap \mathcal{C}_R$, we can safely conclude that an identifiable instance is detected correctly.

Algorithm 1 $\mathsf{ALG}(T=(V,E), n, \mathcal{R}, C, \delta, k_\ell, k_r)$.

if $\delta < 0$ **then**
  Return NO.
else if $C \cap \mathcal{C}_L$ cannot be seen by any set of at most k_ℓ vertices from $\mathcal{R}_L$ **then**
  Compute a set $U \subseteq C \cap \mathcal{C}_L$ of size $k_\ell + 1$ that cannot be seen by any set of at most k_ℓ vertices from $\mathcal{R}_L$.
  for all $v \in U$ **do**
    for all $u \in \mathcal{R}_R \cap \textsc{vis}(v)$ **do** Return $\mathsf{ALG}(T=(V,E), n, \mathcal{R}, C \setminus \textsc{vis}(u), \delta - 1, k_\ell, k_r - 1)$.
  end for
else if $C \cap \mathcal{C}_R$ cannot be seen by any set of at most k_r vertices from $\mathcal{R}_R$ **then**
  Compute a set $U \subseteq C \cap \mathcal{C}_R$ of size $k_r + 1$ that cannot be seen by any set of at most k_r vertices from $\mathcal{R}_R$.
  for all $v \in U$ **do**
    for all $u \in \mathcal{R}_L \cap \textsc{vis}(v)$ **do** Return $\mathsf{ALG}(T=(V,E), n, \mathcal{R}, C \setminus \textsc{vis}(u), \delta - 1, k_\ell - 1, k_r)$.
  end for
else
  Return YES.
end if

5 Conclusion

We studied the well-known Terrain Guarding problem, addressing two fundamental questions relating to its complexity:

- Does Terrain Guarding admit a subexponential-time algorithm?
- Is Terrain Guarding FPT with respect to k?

We have resolved the first question: both Discrete Terrain Guarding and Continuous Terrain Guarding admit subexponential-time algorithms. For discrete orthogonal domains we have also resolved the second question: Discrete Orthogonal Terrain Guarding is FPT.

We would like to conclude our paper by suggesting several directions for further research. First and foremost, it remains to establish the fixed-parameter (in)tractability of Terrain Guarding in general (discrete and continuous) domains, as well as to determine whether Discrete Orthogonal Terrain Guarding is NP-hard or not. In case Terrain Guarding is FPT, one can further ask whether it admits a polynomial kernel. An affirmative answer to this question, combined with our subexponential-time algorithm, would imply that Terrain Guarding admits a subexponential-time parameterized algorithm. Finally, it would also be interesting to investigate whether the running time of our subexponential-time algorithm can be substantially improved, or whether it is essentially tight under reasonable complexity assumptions. We remark that the proof given by King and Krohn [23] to show that Terrain Guarding is NP-hard only implies that unless ETH fails, Terrain Guarding cannot be solved in time $2^{o(n^{\frac{1}{4}})}$.

References

1 J. Abello, O. Egecioglu, and K. Kumar. Visibility graphs of staircase polygons and the weak bruhat order I: From visibility graphs to maximal chains. *Discrete and Computational Geometry*, 14(3):331–358, 1995.

2 Jochen Alber, Hans L. Bodlaender, Henning Fernau, Ton Kloks, and Rolf Niedermeier. Fixed parameter algorithms for dominating set and related problems on planar graphs. *Algorithmica*, 33(4):461–493, 2002.

3 B. Ben-Moshe, M. J. Katz, and J. S. B. Mitchell. A constant-factor approximation algorithm for optimal 1.5d terrain guarding. *SICOMP*, 36(6):1631–1647, 2007.

4 E. Bonnet and T. Miltzow. Parameterized hardness of art gallery problems. In *ESA*, 2016.

5 D. Z. Chen, V. Estivill-Castro, and J. Urrutia. Optimal guarding of polygons and monotone chains. In *CCCG*, pages 133–138, 1995.

6 K. L. Clarkson and K. R. Varadarajan. Improved approximation algorithms for geometric set cover. *Discrete and Computational Geometry*, 37(1):43–58, 2007.

7 M. Cygan, F. V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized algorithms*. Springer, 2015.

8 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. *J. ACM*, 52(6):866–893, 2005.

9 R. Diestel. *Graph Theory, 4th Edition*. Springer, 2012.

10 R. Downey and M. R. Fellows. *Fundamentals of parameterized complexity*. Springer, 2013.

11 S. Durocher, P. C. Li, and S. Mehrabi. Guarding orthogonal terrains. In *CCCG*, 2015.

12 M K Elbassioni, E Krohn, D Matijevic, J Mestre, and D Severdija. Improved approximations for guarding 1.5-dimensional terrains. *Algorithmica*, 60(2):451–463, 2011.

13 W. Evans and N. Saeedi. On characterizing terrain visibility graphs. *JoCG*, 6(1):108–141, 2015.

14 S. Friedrichs, M. Hemmer, J. King, and C. Schmidt. The continuous 1.5D terrain guarding problem: Discretization, optimal solutions, and PTAS. *JoCG*, 7(1):256–284, 2016.

15 P. Giannopoulos. Open problems: Guarding problems. *Lorentz Workshop on Fixed-Parameter Computational Geometry, Leiden, the Netherlands*, page 12, 2016.

16 M. Gibson, G. Kanade, E. Krohn, and K. Varadarajan. Guarding terrains via local search. *JoCG*, 5(1):168–178, 2014.

17 Martin Charles Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. Academic Press, New York, 1980.

18 S. K. Gosh. *Visibility algorithms in the plane*. Cambridge University Press, 2007.

19 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity. *Journal of Computer and System Sciences (JCSS0*, 63(4):512–530, 2001.

20 M. J. Katz and G. S. Roisman. On guarding the vertices of rectilinear domains. *Computational Geometry*, 39(3):219–228, 2008.

21 F. Khodakarami, F. Didehvar, and A. Mohades. A fixed-parameter algorithm for guarding 1.5D terrains. *Theoretical Computer Science*, 595:130–142, 2015.

22 J. King. A 4-approximation algorithm for guarding 1.5-dimensional terrains. In *LATIN*, pages 629–640, 2006.

23 J. King and E. Krohn. Terrain guarding is NP-hard. *SICOMP*, 40(5):1316–1339, 2011.

24 Y. Lyu and A. Üngör. A fast 2-approximation algorithm for guarding orthogonal terrains. In *CCCG*, 2016.

25 S. Mehrabi. Guarding the vertices of an orthogonal terrain using vertex guards. *arXiv:1512.08292*, 2015.

26 J. O'rourke. *Art gallery theorems and algorithms*. Oxford University Press, 1987.

27 D. Schuchardt and H. D. Hecker. Two NP-hard art-gallery problems for ortho-polygons. *Mathematical Logic Quarterly*, 41:261–267, 1995.

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

Covering Lattice Points by Subspaces and Counting Point-Hyperplane Incidences*

Martin Balko[1], Josef Cibulka[2], and Pavel Valtr[3]

1 **Department of Applied Mathematics, Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic; and Alfréd Rényi Institute of Mathematics, Hungarian Academy of Sciences, Budapest, Hungary**
balko@kam.mff.cuni.cz

2 **Department of Applied Mathematics, Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic**
cibulka@kam.mff.cuni.cz

3 **Department of Applied Mathematics, Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic; and Alfréd Rényi Institute of Mathematics, Hungarian Academy of Sciences, Budapest, Hungary**

Abstract

Let d and k be integers with $1 \leq k \leq d-1$. Let Λ be a d-dimensional lattice and let K be a d-dimensional compact convex body symmetric about the origin. We provide estimates for the minimum number of k-dimensional linear subspaces needed to cover all points in $\Lambda \cap K$. In particular, our results imply that the minimum number of k-dimensional linear subspaces needed to cover the d-dimensional $n \times \cdots \times n$ grid is at least $\Omega(n^{d(d-k)/(d-1)-\varepsilon})$ and at most $O(n^{d(d-k)/(d-1)})$, where $\varepsilon > 0$ is an arbitrarily small constant. This nearly settles a problem mentioned in the book of Brass, Moser, and Pach [7]. We also find tight bounds for the minimum number of k-dimensional affine subspaces needed to cover $\Lambda \cap K$.

We use these new results to improve the best known lower bound for the maximum number of point-hyperplane incidences by Brass and Knauer [6]. For $d \geq 3$ and $\varepsilon \in (0,1)$, we show that there is an integer $r = r(d,\varepsilon)$ such that for all positive integers n, m the following statement is true. There is a set of n points in $\mathbb{R}^d$ and an arrangement of m hyperplanes in $\mathbb{R}^d$ with no $K_{r,r}$ in their incidence graph and with at least $\Omega\left((mn)^{1-(2d+3)/((d+2)(d+3))-\varepsilon}\right)$ incidences if d is odd and $\Omega\left((mn)^{1-(2d^2+d-2)/((d+2)(d^2+2d-2))-\varepsilon}\right)$ incidences if d is even.

1998 ACM Subject Classification G.2.1 Combinatorics

Keywords and phrases lattice point, covering, linear subspace, point-hyperplane incidence

Digital Object Identifier 10.4230/LIPIcs.SoCG.2017.12

1 Introduction

In this paper, we study the minimum number of linear or affine subspaces needed to cover points that are contained in the intersection of a given lattice with a given 0-symmetric convex body. We also present an application of our results to the problem of estimating the

* The first and the third author acknowledge the support of the grants GAČR 14-14179S of Czech Science Foundation, ERC Advanced Research Grant no 267165 (DISCONV), and GAUK 690214 of the Grant Agency of the Charles University. The first author is also supported by the grant SVV–2016–260332.

maximum number of incidences between a set of points and an arrangement of hyperplanes. Consequently, this establishes a new lower bound for the time complexity of so-called partitioning algorithms for Hopcroft's problem. Before describing our results in more detail, we first give some preliminaries and introduce necessary definitions.

1.1 Preliminaries

For linearly independent vectors $b_1, \dots, b_d \in \mathbb{R}^d$, the d-dimensional *lattice* $\Lambda = \Lambda(b_1, \dots, b_d)$ with *basis* $\{b_1, \dots, b_d\}$ is the set of all linear combinations of the vectors $b_1, \dots, b_d$ with integer coefficients. We define the *determinant of* Λ as $\det(\Lambda) := |\det(B)|$, where B is the $d \times d$ matrix with the vectors $b_1, \dots, b_d$ as columns. For a positive integer d, we use $\mathcal{L}^d$ to denote the set of d-dimensional lattices Λ, that is, lattices with $\det(\Lambda) \neq 0$.

A convex body K is *symmetric about the origin* 0 if $K = -K$. We let $\mathcal{K}^d$ be the set of d-dimensional compact convex bodies in $\mathbb{R}^d$ that are symmetric about the origin.

For a positive integer n, we use the abbreviation $[n]$ to denote the set $\{1, 2, \dots, n\}$. A point x of a lattice is called *primitive* if whenever its multiple $\lambda \cdot x$ is a lattice point, then λ is an integer. For $K \in \mathcal{K}_d$, let $\mathrm{vol}(K)$ be the d-dimensional Lebesgue measure of K. We say that $\mathrm{vol}(K)$ is the *volume of* K. The closed d-dimensional ball with the radius $r \in \mathbb{R}$, $r \geq 0$, centered in the origin is denoted by $B^d(r)$. If $r = 1$, we simply write B^d instead of $B^d(1)$. For $x \in \mathbb{R}^d$, we use $\|x\|$ to denote the Euclidean norm of x.

Let X be a subset of $\mathbb{R}^d$. We use $\mathrm{aff}(X)$ and $\mathrm{lin}(X)$ to denote the *affine hull of* X and the *linear hull of* X, respectively. The dimension of the affine hull of X is denoted by $\dim(X)$.

For functions $f, g \colon \mathbb{N} \to \mathbb{N}$, we write $f(n) \leq O(g(n))$ if there is a fixed constant c_1 such that $f(n) \leq c_1 \cdot g(n)$ for all $n \in \mathbb{N}$. We write $f(n) \geq \Omega(g(n))$ if there is a fixed constant $c_2 > 0$ such that $f(n) \geq c_2 \cdot g(n)$ for all $n \in \mathbb{N}$. If the constants c_1 and c_2 depend on some parameters $a_1, \dots, a_t$, then we emphasize this by writing $f(n) \leq O_{a_1,\dots,a_t}(g(n))$ and $f(n) \geq \Omega_{a_1,\dots,a_t}(g(n))$, respectively. If $f(n) \leq O_{a_1,\dots,a_t}(n)$ and $f(n) \geq \Omega_{a_1,\dots,a_t}(n)$, then we write $f(n) = \Theta_{a_1,\dots,a_t}(n)$.

1.2 Covering lattice points by subspaces

We say that a collection $\mathcal{S}$ of subsets in $\mathbb{R}^d$ *covers* a set of points P from $\mathbb{R}^d$ if every point from P lies in some set from $\mathcal{S}$.

Let d, k, n, and r be positive integers that satisfy $1 \leq k \leq d-1$. We let $a(d, k, n, r)$ be the maximum size of a set $S \subseteq \mathbb{Z}^d \cap B^d(n)$ such that every k-dimensional *affine* subspace of $\mathbb{R}^d$ contains at most $r-1$ points of S. Similarly, we let $l(d, k, n, r)$ be the maximum size of a set $S \subseteq \mathbb{Z}^d \cap B^d(n)$ such that every k-dimensional *linear* subspace of $\mathbb{R}^d$ contains at most $r-1$ points of S. We also let $g(d, k, n)$ be the minimum number of k-dimensional linear subspaces of $\mathbb{R}^d$ necessary to cover $\mathbb{Z}^d \cap B^d(n)$.

In this paper, we study the functions $a(d, k, n, r)$, $l(d, k, n, r)$, and $g(d, k, n)$ and their generalizations to arbitrary lattices from $\mathcal{L}^d$ and bodies from $\mathcal{K}^d$. We mostly deal with the last two functions, that is, with covering lattice points by linear subspaces. In particular, we obtain new upper bounds on $g(d, k, n)$ (Theorem 4), lower bounds on $l(d, k, n, r)$ (Theorem 5), and we use the estimates for $a(d, k, n, r)$ and $l(d, k, n, r)$ to obtain improved lower bounds for the maximum number of point-hyperplane incidences (Theorem 9). Before doing so, we first give a summary of known results, since many of them are used later in the paper.

The problem of determining $a(d, k, n, r)$ is essentially solved. In general, the set $\mathbb{Z}^d \cap B^d(n)$ can be covered by $(2n+1)^{d-k}$ affine k-dimensional subspaces and thus we have an upper bound $a(d, k, n, r) \leq (r-1)(2n+1)^{d-k}$. This trivial upper bound is asymptotically almost

tight for all fixed d, k, and some r, as Brass and Knauer [6] showed with a probabilistic argument that for every $\varepsilon > 0$ there is an $r = r(d, \varepsilon, k) \in \mathbb{N}$ such that for each positive integer n we have

$$a(d, k, n, r) \geq \Omega_{d,\varepsilon,k}\left(n^{d-k-\varepsilon}\right). \tag{1}$$

For fixed d and r, the upper bound is known to be asymptotically tight in the cases $k = 1$ and $k = d - 1$. This is showed by considering points on the modular moment surface for $k = 1$ and the modular moment curve for $k = d - 1$; see [6].

Covering lattice points by linear subspaces seems to be more difficult than covering by affine subspaces. From the definitions we immediately get $l(d, k, n, r) \leq (r - 1)g(d, k, n)$. In the case $k = d - 1$ and d fixed, Bárány, Harcos, Pach, and Tardos [5] obtained the following asymptotically tight estimates for the functions $l(d, d - 1, n, d)$ and $g(d, d - 1, n)$:

$$l(d, d-1, n, d) = \Theta_d(n^{d/(d-1)}) \qquad \text{and} \qquad g(d, d-1, n) = \Theta_d(n^{d/(d-1)}).$$

In fact, Bárány et al. [5] proved stronger results that estimate the minimum number of $(d - 1)$-dimensional linear subspaces necessary to cover the set $\Lambda \cap K$ in terms of so-called successive minima of a given lattice $\Lambda \in \mathcal{L}^d$ and a body $K \in \mathcal{K}^d$.

For a lattice $\Lambda \in \mathcal{L}^d$, a body $K \in \mathcal{K}^d$, and $i \in [d]$, we let $\lambda_i(\Lambda, K)$ be the *ith successive minimum of* Λ *and* K. That is, $\lambda_i(\Lambda, K) := \inf\{\lambda \in \mathbb{R} \colon \dim(\Lambda \cap (\lambda \cdot K)) \geq i\}$. Since K is compact, it is easy to see that the successive minima are achieved. That is, there are linearly independent vectors $v_1, \ldots, v_d$ from Λ such that $v_i \in \lambda_i(\Lambda, K) \cdot K$ for every $i \in [d]$. Also note that we have $\lambda_1(\Lambda, K) \leq \cdots \leq \lambda_d(\Lambda, K)$ and $\lambda_1(\mathbb{Z}^d, B^d(n)) = \cdots = \lambda_d(\mathbb{Z}^d, B^d(n)) = 1/n$.

▶ **Theorem 1** ([5]). *For an integer $d \geq 2$, a lattice $\Lambda \in \mathcal{L}^d$, and a body $K \in \mathcal{K}^d$, we let $\lambda_i := \lambda_i(\Lambda, K)$ for every $i \in [d]$. If $\lambda_d \leq 1$, then the set $\Lambda \cap K$ can be covered with at most*

$$c2^d d^2 \log_2 d \min_{1 \leq j \leq d-1} (\lambda_j \cdots \lambda_d)^{-1/(d-j)}$$

$(d - 1)$-dimensional linear subspaces of $\mathbb{R}^d$, where c is some absolute constant.

On the other hand, if $\lambda_d \leq 1$, then there is a subset S of $\Lambda \cap K$ of size

$$\frac{1 - \lambda_d}{16d^2} \min_{1 \leq j \leq d-1} (\lambda_j \cdots \lambda_d)^{-1/(d-j)}$$

such that no $(d - 1)$-dimensional linear subspace of $\mathbb{R}^d$ contains d points from S.

We note that the assumption $\lambda_d \leq 1$ is necessary; see the discussion in [5]. Not much is known for linear subspaces of lower dimension. We trivially have $l(d, k, n, r) \geq a(d, k, n, r)$ for all d, k, n, r with $1 \leq k \leq d - 1$. Thus $l(d, k, n, r) \geq \Omega_{d,\varepsilon,k}(n^{d-k-\varepsilon})$ for some $r = r(d, \varepsilon, k)$ by (1). Brass and Knauer [6] conjectured that $l(d, k, n, k + 1) = \Theta_{d,k}(n^{d(d-k)/(d-1)})$ for d fixed. This conjecture was refuted by Lefmann [15] who showed that, for all d and k with $1 \leq k \leq d - 1$, there is an absolute constant c such that we have $l(d, k, n, k + 1) \leq c \cdot n^{d/\lceil k/2 \rceil}$ for every positive integer n. This bound is asymptotically smaller in n than the growth rate conjectured by Brass and Knauer for sufficiently large d and almost all values of k with $1 \leq k \leq d - 1$.

Covering lattice points by linear subspaces is also mentioned in the book by Brass, Moser, and Pach [7], where the authors pose the following problem.

▶ **Problem 2** ([7, Problem 6 in Chapter 10.2]). *What is the minimum number of k-dimensional linear subspaces necessary to cover the d-dimensional $n \times \cdots \times n$ lattice cube?*

1.3 Point-hyperplane incidences

As we will see later, the problem of determining $a(d,k,n,r)$ and $l(d,n,k,r)$ is related to a problem of bounding the maximum number of point-hyperplane incidences. For an integer $d \geq 2$, let P be a set of n points in $\mathbb{R}^d$ and let $\mathcal{H}$ be an arrangement of m hyperplanes in $\mathbb{R}^d$. An *incidence between P and $\mathcal{H}$* is a pair (p,H) such that $p \in P$, $H \in \mathcal{H}$, and $p \in H$. The number of incidences between P and $\mathcal{H}$ is denoted by $\mathrm{I}(P,\mathcal{H})$.

We are interested in the maximum number of incidences between P and $\mathcal{H}$. In the plane, the famous *Szemerédi–Trotter theorem* [23] says that the maximum number of incidences between a set of n points in $\mathbb{R}^2$ and an arrangement of m lines in $\mathbb{R}^2$ is at most $O((mn)^{2/3} + m + n)$. This is known to be asymptotically tight, as a matching lower bound was found earlier by Erdős [9]. The current best known bounds are $\approx 1.27(mn)^{2/3} + m + n$ [19][1] and $\approx 2.44(mn)^{2/3} + m + n$ [1].

For $d \geq 3$, it is easy to see that there is a set P of n points in $\mathbb{R}^d$ and an arrangement $\mathcal{H}$ of m hyperplanes in $\mathbb{R}^d$ for which the number of incidences is maximum possible, that is $\mathrm{I}(P,\mathcal{H}) = mn$. It suffices to consider the case where all points from P lie in an affine subspace that is contained in every hyperplane from $\mathcal{H}$. In order to avoid this degenerate case, we forbid large complete bipartite graphs in the *incidence graph of P and $\mathcal{H}$*, which is denoted by $G(P,\mathcal{H})$. This is the bipartite graph on the vertex set $P \cup \mathcal{H}$ and with edges $\{p,H\}$ where (p,H) is an incidence between P and $\mathcal{H}$.

With this restriction, bounding $\mathrm{I}(P,\mathcal{H})$ becomes more difficult and no tight bounds are known for $d \geq 3$. It follows from the works of Chazelle [8], Brass and Knauer [6], and Apfelbaum and Sharir [2] that the number of incidences between any set P of n points in $\mathbb{R}^d$ and any arrangement $\mathcal{H}$ of m hyperplanes in $\mathbb{R}^d$ with $K_{r,r} \not\subseteq G(P,\mathcal{H})$ satisfies

$$\mathrm{I}(P,\mathcal{H}) \leq O_{d,r}\left((mn)^{1-1/(d+1)} + m + n\right). \tag{2}$$

We note that an upper bound similar to (2) holds in a much more general setting; see the remark in the proof of Theorem 9. The best general lower bound for $\mathrm{I}(P,\mathcal{H})$ is due to a construction of Brass and Knauer [6], which gives the following estimate.

▶ **Theorem 3** ([6]). *Let $d \geq 3$ be an integer. Then for every $\varepsilon > 0$ there is a positive integer $r = r(d,\varepsilon)$ such that for all positive integers n and m there is a set P of n points in $\mathbb{R}^d$ and an arrangement $\mathcal{H}$ of m hyperplanes in $\mathbb{R}^d$ such that $K_{r,r} \not\subseteq G(P,\mathcal{H})$ and*

$$\mathrm{I}(P,\mathcal{H}) \geq \begin{cases} \Omega_{d,\varepsilon}\left((mn)^{1-2/(d+3)-\varepsilon}\right) & \text{if } d \text{ is odd and } d > 3, \\ \Omega_{d,\varepsilon}\left((mn)^{1-2(d+1)/(d+2)^2-\varepsilon}\right) & \text{if } d \text{ is even,} \\ \Omega_{d,\varepsilon}\left((mn)^{7/10}\right) & \text{if } d = 3. \end{cases}$$

For $d \geq 4$, this lower bound has been recently improved by Sheffer [21] in a certain non-diagonal case. Sheffer constructed a set P of n points in $\mathbb{R}^d$, $d \geq 4$, and an arrangement $\mathcal{H}$ of $m = \Theta(n^{(3-3\varepsilon)/(d+1)})$ hyperplanes in $\mathbb{R}^d$ such that $K_{(d-1)/\varepsilon,2} \not\subseteq G(P,\mathcal{H})$ and $\mathrm{I}(P,\mathcal{H}) \geq \Omega\left((mn)^{1-2/(d+4)-\varepsilon}\right)$.

[1] The lower bound claimed by Pach and Tóth [19, Remark 4.2] contains the multiplicative constant ≈ 0.42. This is due to a miscalculation in the last equation in the calculation of the number of incidences. The correct calculation is $I \approx \cdots = 4n\sum_{r=1}^{1/\varepsilon}\phi(r) - 2n\varepsilon^2\sum_{r=1}^{1/\varepsilon} r^2\phi(r) \approx 4n \cdot 3(1/\varepsilon)^2/\pi^2 - 2n\varepsilon^2(3/2)(1/\varepsilon)^4/\pi^2 = 9n/(\varepsilon^2\pi^2)$. This leads to $c \approx 3\sqrt[3]{3/(4\pi^2)} \approx 1.27$.

2 Our results

In this paper, we nearly settle Problem 2 by proving almost tight bounds for the function $g(d,k,n)$ for a fixed d and an arbitrary k from $[d-1]$. For a fixed d, an arbitrary $k \in [d-1]$, and some fixed r, we also provide bounds on the function $l(d,k,n,r)$ that are very close to the bound conjectured by Brass and Knauer [6]. Thus it seems that the conjectured growth rate of $l(d,k,n,r)$ is true if we allow r to be (significantly) larger than $k+1$.

We study these problems in a more general setting where we are given an arbitrary lattice Λ from $\mathcal{L}^d$ and a body K from $\mathcal{K}^d$. Similarly to Theorem 1 by Bárány et al. [5], our bounds are expressed in terms of the successive minima $\lambda_i(\Lambda,K)$, $i \in [d]$.

2.1 Covering lattice points by linear subspaces

First, we prove a new upper bound on the minimum number of k-dimensional linear subspaces that are necessary to cover points in the intersection of a given lattice with a body from $\mathcal{K}^d$.

▶ **Theorem 4.** *For integers d and k with $1 \le k \le d-1$, a lattice $\Lambda \in \mathcal{L}^d$, and a body $K \in \mathcal{K}^d$, we let $\lambda_i := \lambda_i(\Lambda,K)$ for $i = 1,\dots,d$. If $\lambda_d \le 1$, then we can cover $\Lambda \cap K$ with $O_{d,k}(\alpha^{d-k})$ k-dimensional linear subspaces of $\mathbb{R}^d$, where*

$$\alpha := \min_{1\le j\le k} (\lambda_j \cdots \lambda_d)^{-1/(d-j)}.$$

We also prove the following lower bound.

▶ **Theorem 5.** *For integers d and k with $1 \le k \le d-1$, a lattice $\Lambda \in \mathcal{L}^d$, and a body $K \in \mathcal{K}^d$, we let $\lambda_i := \lambda_i(\Lambda,K)$ for $i = 1,\dots,d$. If $\lambda_d \le 1$, then, for every $\varepsilon \in (0,1)$, there is a positive integer $r = r(d,\varepsilon,k)$ and a set $S \subseteq \Lambda \cap K$ of size at least $\Omega_{d,\varepsilon,k}(((1-\lambda_d)\beta)^{d-k-\varepsilon})$, where*

$$\beta := \min_{1\le j\le d-1} (\lambda_j \cdots \lambda_d)^{-1/(d-j)},$$

such that every k-dimensional linear subspace of $\mathbb{R}^d$ contains at most $r-1$ points from S.

We remark that we can get rid of the ε in the exponent if $k=1$ or $k=d-1$; for details, see Theorem 1 for the case $k=d-1$ and the proof in Section 4 for the case $k=1$. Also note that in the definition of α in Theorem 4 the minimum is taken over the set $\{1,\dots,k\}$, while in the definition of β in Theorem 5 the minimum is taken over $\{1,\dots,d-1\}$. There are examples, which show that α cannot be replaced by β in Theorem 4. It suffices to consider $d=3$, $k=1$, and let Λ be the lattice $\{(x_1/n, x_2/2, x_3/2) \in \mathbb{R}^3 \colon x_1,x_2,x_3 \in \mathbb{Z}\}$ for some large positive integer n. Then $\lambda_1(\Lambda,B^3) = 1/n$, $\lambda_2(\Lambda,B^3) = 1/2$, $\lambda_3(\Lambda,B^3) = 1/2$, and thus $\beta = (\lambda_2\lambda_3)^{-1} = 4$. However, it is not difficult to see that we need at least $\Omega(n)$ 1-dimensional linear subspaces to cover $\Lambda \cap B^3$, which is asymptotically larger than $\beta^2 = O(1)$. On the other hand, $\alpha = (\lambda_1\lambda_2\lambda_3)^{-1/2}$ and $O(\alpha^2) = O(n)$ 1-dimensional linear subspaces suffice to cover $\Lambda \cap B^3$. We thus suspect that the lower bound can be improved.

Since $\lambda_i(\mathbb{Z}^d, B^d(n)) = 1/n$ for every $i \in [d]$, we can apply Theorem 5 with $\Lambda = \mathbb{Z}^d$ and $K = B^d(n)$ and obtain the following lower bound on $l(d,k,n,r)$.

▶ **Corollary 6.** *Let d and k be integers with $1 \le k \le d-1$. Then, for every $\varepsilon \in (0,1)$, there is an $r = r(d,\varepsilon,k) \in \mathbb{N}$ such that for every $n \in \mathbb{N}$ we have*

$$l(d,k,n,r) \ge \Omega_{d,\varepsilon,k}(n^{d(d-k)/(d-1)-\varepsilon}).$$

The existence of the set S from Theorem 5 is showed by a probabilistic argument. It would be interesting to find, at least for some value $1 < k < d-1$, some fixed $r \in \mathbb{N}$, and arbitrarily large $n \in \mathbb{N}$, a construction of a subset R of $\mathbb{Z}^d \cap B^d(n)$ of size $\Omega_{d,k}(n^{d(d-k)/(d-1)})$ such that every k-dimensional linear subspace contains at most $r-1$ points from R. Such constructions are known for $k = 1$ and $k = d-1$; see [6, 20].

Since we have $l(d,k,n,r) \le (r-1)g(d,k,n)$ for every $r \in \mathbb{N}$, Theorem 4 and Corollary 6 give the following almost tight estimates on $g(d,k,n)$. This nearly settles Problem 2.

▶ **Corollary 7.** *Let d, k, and n be integers with $1 \le k \le d-1$. Then, for every $\varepsilon \in (0,1)$, we have*

$$\Omega_{d,\varepsilon,k}(n^{d(d-k)/(d-1)-\varepsilon}) \le g(d,k,n) \le O_{d,k}(n^{d(d-k)/(d-1)}).$$

2.2 Covering lattice points by affine subspaces

For *affine* subspaces, Brass and Knauer [6] considered only the case of covering the d-dimensional $n \times \cdots \times n$ lattice cube by k-dimensional affine subspaces. To our knowledge, the case for general $\Lambda \in \mathcal{L}^d$ and $K \in \mathcal{K}^d$ was not considered in the literature. We extend the results of Brass and Knauer to covering $\Lambda \cap K$.

▶ **Theorem 8.** *For integers d and k with $1 \le k \le d-1$, a lattice $\Lambda \in \mathcal{L}^d$, and a body $K \in \mathcal{K}^d$, we let $\lambda_i := \lambda_i(\Lambda, K)$ for $i = 1, \dots, d$. If $\lambda_d \le 1$, then the set $\Lambda \cap K$ can be covered with $O_{d,k}((\lambda_{k+1} \cdots \lambda_d)^{-1})$ k-dimensional affine subspaces of $\mathbb{R}^d$.*

On the other hand, at least $\Omega_{d,k}((\lambda_{k+1} \cdots \lambda_d)^{-1})$ k-dimensional affine subspaces of $\mathbb{R}^d$ are necessary to cover $\Lambda \cap K$.

2.3 Point-hyperplane incidences

As an application of Corollary 6, we improve the best known lower bounds on the maximum number of point-hyperplane incidences in $\mathbb{R}^d$ for $d \ge 4$. That is, we improve the bounds from Theorem 3. To our knowledge, this is the first improvement on the estimates for $\mathrm{I}(P, \mathcal{H})$ in the general case during the last 13 years.

▶ **Theorem 9.** *For every integer $d \ge 2$ and $\varepsilon \in (0,1)$, there is an $r = r(d,\varepsilon) \in \mathbb{N}$ such that for all positive integers n and m the following statement is true. There is a set P of n points in $\mathbb{R}^d$ and an arrangement $\mathcal{H}$ of m hyperplanes in $\mathbb{R}^d$ such that $K_{r,r} \not\subseteq G(P, \mathcal{H})$ and*

$$\mathrm{I}(P,\mathcal{H}) \ge \begin{cases} \Omega_{d,\varepsilon}\left((mn)^{1-(2d+3)/((d+2)(d+3))-\varepsilon}\right) & \text{if } d \text{ is odd,} \\ \Omega_{d,\varepsilon}\left((mn)^{1-(2d^2+d-2)/((d+2)(d^2+2d-2))-\varepsilon}\right) & \text{if } d \text{ is even.} \end{cases}$$

We can get rid of the ε in the exponent for $d \le 3$. That is, we have the bounds $\Omega((mn)^{2/3})$ for $d = 2$ and $\Omega((mn)^{7/10})$ for $d = 3$. For $d = 3$, our bound is the same as the bound from Theorem 3. For larger d, our bounds become stronger. In particular, the exponents in the lower bounds from Theorem 9 exceed the exponents from Theorem 3 by $1/((d+2)(d+3))$ for $d > 3$ odd and by $d^2/((d+2)^2(d^2+2d-2))$ for d even. However, the bounds are not tight.

In the non-diagonal case, when one of n and m is significantly larger that the other, the proof of Theorem 9 yields the following stronger bound.

▶ **Theorem 10.** *For all integers d and k with $0 \le k \le d-2$ and for $\varepsilon \in (0,1)$, there is an $r = r(d,\varepsilon,k) \in \mathbb{N}$ such that for all positive integers n and m the following statement is true.*

There is a set P of n points in $\mathbb{R}^d$ and an arrangement $\mathcal{H}$ of m hyperplanes in $\mathbb{R}^d$ such that $K_{r,r} \not\subseteq G(P,\mathcal{H})$ and

$$\mathrm{I}(P,\mathcal{H}) \geq \Omega_{d,\varepsilon,k}\left(n^{1-(k+1)/((k+2-1/d)(d-k))-\varepsilon} m^{1-(d-1)/(dk+2d-1)-\varepsilon}\right).$$

For example, in the case $m = \Theta(n^{(3-3\varepsilon)/(d+1)})$ considered by Sheffer [21], Theorem 10 gives a slightly better bound than $I(P,\mathcal{H}) \geq \Omega((mn)^{1-2/(d+4)-\varepsilon}))$ if we set, for example, $k = \lfloor (d-1)/4 \rfloor$. However, the forbidden complete bipartite subgraph in the incidence graph is larger than $K_{(d-1)/\varepsilon,2}$.

The following problem is known as the counting version of *Hopcroft's problem* [6, 10]: given n points in $\mathbb{R}^d$ and m hyperplanes in $\mathbb{R}^d$, how fast can we count the incidences between them? We note that the lower bounds from Theorem 9 also establish the best known lower bounds for the time complexity of so-called *partitioning algorithms* [10] for the counting version of Hopcroft's problem; see [6] for more details.

In the proofs of our results, we make no serious effort to optimize the constants. We also omit floor and ceiling signs whenever they are not crucial.

3 Proof of Theorem 4

Here we sketch the proof of the upper bound on the minimum number of k-dimensional linear subspaces needed to cover points from a given d-dimensional lattice that are contained in a body K from $\mathcal{K}^d$. We first prove Theorem 4 in the special case $K = B^d$ (Theorem 14) and then we extend the result to arbitrary $K \in \mathcal{K}^d$. Since the proof is rather long and complicated, we only prove a weaker bound (Corollary 16) and then we give a high-level overview of the main ideas of the full proof, which can be found in the full version of the paper [3].

3.1 Sketch of the proof for balls

We first introduce some auxiliary results that are used later. The following classical result is due to Minkowski [18] and shows a relation between $\mathrm{vol}(K)$, $\det(\Lambda)$, and the successive minima of $\Lambda \in \mathcal{L}^d$ and $K \in \mathcal{K}^d$.

▶ **Theorem 11** (Minkowski's second theorem [18]). *Let d be a positive integer. For every $\Lambda \in \mathcal{L}^d$ and every $K \in \mathcal{K}^d$, we have*

$$\frac{1}{2^d} \cdot \frac{\mathrm{vol}(K)}{\det(\Lambda)} \leq \frac{1}{\lambda_1(\Lambda,K)\cdots\lambda_d(\Lambda,K)} \leq \frac{d!}{2^d} \cdot \frac{\mathrm{vol}(K)}{\det(\Lambda)}.$$

A result similar to the first bound from Theorem 11 can be obtained if the volume is replaced by the point enumerator; see Henk [13].

▶ **Theorem 12** ([13, Theorem 1.5]). *Let d be a positive integer. For every $\Lambda \in \mathcal{L}^d$ and every $K \in \mathcal{K}^d$, we have*

$$|\Lambda \cap K| \leq 2^{d-1} \prod_{i=1}^{d} \left\lfloor \frac{2}{\lambda_i(\Lambda,K)} + 1 \right\rfloor.$$

For $\Lambda \in \mathcal{L}^d$ and $K \in \mathcal{K}^d$, let $v_1,\ldots,v_d$ be linearly independent vectors such that $v_i \in \Lambda \cap (\lambda_i(\Lambda,K)\cdot K)$ for every $i \in [d]$. For $d > 2$, the vectors $v_1,\ldots,v_d$ do not necessarily form a basis of Λ [22, see Section X.5]. However, the following theorem shows that there exists a basis with vectors of lengths not much larger than the lengths of $v_1,\ldots,v_d$.

▶ **Theorem 13** (First finiteness theorem [22, see Lemma 2 in Section X.6])**.** *Let d be a positive integer. For every $\Lambda \in \mathcal{L}^d$ and every $K \in \mathcal{K}^d$, there is a basis $\{b_1, \dots, b_d\}$ of Λ with $b_i \in (3/2)^{i-1}\lambda_i(\Lambda, K) \cdot K$ for every $i \in [d]$.*

Now, let Λ be a d-dimensional lattice with $\lambda_d(\Lambda, B^d) \leq 1$. Throughout this section, we use λ_i to denote the ith successive minimum $\lambda_i(\Lambda, B^d)$ for $i = 1, \dots, d$. Let k be an integer with $1 \leq k \leq d-1$. We show the following result.

▶ **Theorem 14.** *There is a constant $C = C(d,k)$ such that the set $\Lambda \cap B^d$ can be covered with $C \cdot \alpha^{d-k}$ k-dimensional linear subspaces of $\mathbb{R}^d$, where*

$$\alpha := \min_{1 \leq j \leq k} (\lambda_j \cdots \lambda_d)^{-1/(d-j)}.$$

As the first step towards the proof of Theorem 14, we show a weaker bound on the number of k-dimensional linear subspaces needed to cover $\Lambda \cap B^d$; see Corollary 16. To do so, we prove the following lemma that is also used later in the proof of Theorem 8.

▶ **Lemma 15.** *Let d and s be integers with $0 \leq s \leq d-1$. There is a positive integer $r = r(d,s)$ and a projection p of $\mathbb{R}^d$ along s vectors of Λ onto a $(d-s)$-dimensional linear subspace N of $\mathbb{R}^d$ such that $\Lambda \cap B^d$ is mapped to $\Lambda \cap N \cap B^d(r)$ and such that $\lambda_i(\Lambda \cap N, B^d(r) \cap N) = \Theta_{d,s}(\lambda_{i+s})$ for every $i \in [d-s]$.*

Proof. If $s = 0$, then we set p to be the identity on $\mathbb{R}^d$ and $r := 1$. Thus we assume $s \geq 1$.

For $j = 0, \dots, d-1$, we set $r_j := (2^{d^2}+1)^j$. For $j = 0, \dots, d-1$ and a lattice $\Lambda_j \in \mathcal{L}^{d-j}$, we show that there is a projection p_j of $\mathbb{R}^{d-j}$ along a vector $v_j \in \Lambda_j$ onto a $(d-j-1)$-dimensional linear subspace N of $\mathbb{R}^{d-j}$ such that $\Lambda_j \cap B^{d-j}(r_j)$ is mapped to $\Lambda_j \cap N \cap B^{d-j}(r_{j+1})$ by p_j and such that

$$\lambda_{i+1}(\Lambda_j, B^{d-j}(r_j))/(2^{d^2}+1) \leq \lambda_i(\Lambda_j \cap N, B^{d-j}(r_{j+1}) \cap N) \leq \lambda_{i+1}(\Lambda_j, B^{d-j}(r_j))$$

for every $i \in [d-j-1]$. We let p_j be the projection for $\Lambda_j := p_{j-1}(\Lambda_{j-1})$ for every $j = 1, \dots, s-1$, where $\Lambda_0 := \Lambda$ and p_0 is the projection for Λ_0. The statement of the lemma is then obtained by setting $p := p_{s-1} \circ \cdots \circ p_0$.

Let $B = \{b_1, \dots, b_{d-j}\}$ be a basis of Λ_j such that $b_i \in (3/2)^{i-1}\lambda_i(\Lambda_j, B^{d-j}(r_j)) \cdot B^{d-j}(r_j)$ for every $i \in [d-j]$. Such basis exists by the First finiteness theorem (Theorem 13). In particular, b_1 is the shortest vector from $\Lambda_j \cap B^{d-j}(r_j)$. Let $v_j := b_1$ and let N be the linear subspace generated by $b_2, \dots, b_{d-j}$. Let Λ_N be the set $\Lambda_j \cap N$. Note that Λ_N is a $(d-j-1)$-dimensional lattice with the basis $\{b_2, \dots, b_{d-j}\}$.

We consider the projection p_j onto N along v_j. That is, every $x \in \mathbb{R}^{d-j}$ is mapped to $p_j(x) = \sum_{i=2}^{d-j} t_i b_i$, where $x = \sum_{i=1}^{d-j} t_i b_i$, $t_i \in \mathbb{R}$, is the expression of x with respect to the basis B.

We show that $p_j(z) \in \Lambda_N \cap B^{d-j}(r_{j+1})$ for every $z \in \Lambda_j \cap B^{d-j}(r_j)$. We have $p_j(z) \in \Lambda_N$, since B is a basis of Λ_j and $B \setminus \{b_1\}$ is a basis of Λ_N. Let $z = \sum_{i=1}^{d-j} t_i b_i$, $t_i \in \mathbb{Z}$, be the expression of z with respect to B and let v be the Euclidean distance between b_1 and N.

From the definitions of Λ_N and B, we have

$$\lambda_{i+1}(\Lambda_j, B^{d-j}(r_j)) \leq \lambda_i(\Lambda_N, B^{d-j}(r_j) \cap N) \leq (3/2)^i \lambda_{i+1}(\Lambda_j, B^{d-j}(r_j)) \tag{3}$$

for every $i \in [d-j-1]$. Using Minkowski's second theorem (Theorem 11) twice, the upper

bound in (3), and the choice of b_1, we obtain

$$\begin{aligned}\frac{\mathrm{vol}(B^{d-j}(r_j))}{2^{d-j}\det(\Lambda_j)} &\leq \frac{1}{\lambda_1(\Lambda_j, B^{d-j}(r_j))\cdots\lambda_{d-j}(\Lambda_j, B^{d-j}(r_j))}\\ &\leq \frac{r_j}{\|b_1\|}\cdot\frac{(3/2)^{(d-j)(d-j-1)/2}}{\lambda_1(\Lambda_N, B^{d-j}(r_j)\cap N)\cdots\lambda_{d-j-1}(\Lambda_N, B^{d-j}(r_j)\cap N)}\\ &\leq \frac{r_j}{\|b_1\|}\cdot\frac{(3/2)^{(d-j)(d-j-1)/2}\cdot(d-j-1)!\cdot\mathrm{vol}(B^{d-j}(r_j)\cap N)}{2^{d-j-1}\cdot\det(\Lambda_N)}.\end{aligned}$$

Since $\det(\Lambda_j) = v\cdot\det(\Lambda_N)$, we can rewrite this expression as

$$\|b_1\| \leq \frac{r_j\cdot(3/2)^{(d-j)(d-j-1)/2}\cdot(d-j-1)!\cdot 2^{d-j}\cdot\mathrm{vol}(B^{d-j}(r_j)\cap N)\cdot\det(\Lambda_j)}{2^{d-j-1}\cdot\mathrm{vol}(B^{d-j}(r_j))\cdot\det(\Lambda_N)} \leq 2^{d^2}\cdot v.$$

To derive the last inequality, we use the well-known formula

$$\mathrm{vol}(B^m(r)) = \begin{cases}\frac{2((m-1)/2)!(4\pi)^{(m-1)/2}}{m!}\cdot r^m & \text{if } m \text{ is odd,}\\ \frac{\pi^{m/2}}{(m/2)!}\cdot r^m & \text{if } m \text{ is even}\end{cases}$$

for the volume of $B^m(r)$, $m, r\in\mathbb{N}$. Since $\mathrm{vol}(B^{d-j}(r_j)\cap N) = \mathrm{vol}(B^{d-j-1}(r_j))$, we have $\mathrm{vol}(B^{d-j}(r_j)\cap N)/\mathrm{vol}(B^{d-j}(r_j)) \leq 2^{d-j}/r_j$. The Euclidean distance between z and N equals $|t_1|\cdot v$, which is at most r_j, as $z\in B^{d-j}(r_j)$. Thus, since $|t_1|\leq r_j/v$ and $1/v\leq 2^{d^2}/\|b_1\|$, we obtain $|t_1|\leq 2^{d^2}\cdot r_j/\|b_1\|$. This implies

$$\|p_j(z)\| = \|z - t_1b_1\| \leq \|z\| + |t_1|\cdot\|b_1\| \leq r_j + 2^{d^2}r_j = r_{j+1}$$

and we see that $p_j(z)$ lies in $\Lambda_N\cap B^{d-j}(r_{j+1})$.

Note that $\lambda_i(\Lambda_N, B^{d-j}(r_{j+1})\cap N) = (2^{d^2}+1)^{-1}\cdot\lambda_i(\Lambda_N, B^{d-j}(r_j)\cap N)$ for every $i\in[d-j-1]$. Using this fact together with the bounds in (3), we obtain

$$\frac{\lambda_{i+1}(\Lambda_j, B^{d-j}(r_j))}{2^{d^2}+1} \leq \lambda_i(\Lambda_N, B^{d-j}(r_{j+1})\cap N) \leq \frac{(3/2)^{d-j}\lambda_{i+1}(\Lambda_j, B^{d-j}(r_j))}{2^{d^2}+1}$$

for every $i\in[d-j-1]$. ◀

▶ **Corollary 16.** *The set $\Lambda\cap B^d$ can be covered with $O_{d,k}((\lambda_k\cdots\lambda_d)^{-1})$ k-dimensional linear subspaces of $\mathbb{R}^d$.*

Proof. By Lemma 15, there is a positive integer $r = r(d, k-1)$ and a projection p of $\mathbb{R}^d$ along $k-1$ vectors $b_1,\ldots,b_{k-1}\in\Lambda$ onto a $(d-k+1)$-dimensional linear subspace N of $\mathbb{R}^d$ such that $\Lambda\cap B^d$ is mapped to $\Lambda\cap N\cap B^d(r)$ and such that $\lambda_i' := \lambda_i(\Lambda\cap N, B^d(r)\cap N) = \Theta_{d,k}(\lambda_{i+k-1})$ for every $i\in[d-k+1]$. We use Λ_N to denote the $(d-k+1)$-dimensional sublattice $\Lambda\cap N$ of Λ.

We consider the set $\mathcal{S} := \{\mathrm{lin}(\{y, b_1,\ldots,b_{k-1}\})\colon y\in(\Lambda_N\setminus\{0\})\cap B^d(r)\}$. Then $\mathcal{S}$ consists of k-dimensional linear subspaces and its projection $p(\mathcal{S})$ covers $\Lambda_N\cap B^d(r)$. By Theorem 12, the size of $\mathcal{S}$ is at most

$$|\Lambda_N\cap B^d(r)| \leq 2^{d-k}\prod_{i=1}^{d-k+1}\left\lfloor\frac{2}{\lambda_i'}+1\right\rfloor \leq O_{d,k}\left(\prod_{i=1}^{d-k+1}\frac{1}{\lambda_i'}\right) \leq O_{d,k}((\lambda_k\cdots\lambda_d)^{-1}),$$

where the second inequality follows from the assumption $\lambda_d\leq 1$, as then $\lambda_{d-k+1}'\leq O_{d,k}(\lambda_d)$ implies $\lambda_1'\leq\cdots\leq\lambda_{d-k+1}'\leq O_{d,k}(1)$. The last inequality is obtained from $\lambda_i'\geq\Omega_{d,k}(\lambda_{i+k-1})$ for every $i\in[d-k+1]$. Moreover, $\mathcal{S}$ covers $\Lambda\cap K$, since for every $y\in\Lambda_N\cap B^d(r)$ there is $S\in\mathcal{S}$ with $y\in p(S)$ and $p(z)\in\Lambda_N\cap B^d(r)$ for every $z\in\Lambda\cap B^d$. ◀

Let q be an integer from $\{d-k+1,\dots,d\}$ such that $\alpha = (\lambda_{d-q+1}\cdots\lambda_d)^{-1/(q-1)}$. The bound from Corollary 16 matches the bound from Theorem 14 in the case $k=1$. The case $k=d-1$ was shown by Bárány et al. [5]; see Theorem 1. Thus we may assume $d \geq 4$. Corollary 16 also provides the same bound as Theorem 14 if $q = d-k+1$, so we assume $q \geq d-k+2$.

We now sketch the proof of the upper bound $O_{d,k}(\alpha^{d-k})$ if $q \geq d-k+2$. Let Λ^* be the dual lattice of Λ. That is, Λ^* is the set of vectors y from $\mathbb{R}^d$ that satisfy $\langle x,y\rangle \in \mathbb{Z}$ for every $x \in \Lambda$. In the rest of the section, we use μ_i to denote $\lambda_i(\Lambda^*, B^d)$ for every $i \in [d]$. It follows from the results of Mahler [16] and Banaszczyk [4] that $1 \leq \lambda_i \cdot \mu_{d-i+1} \leq d$ holds for every $i \in [d]$. This together with the assumption $\lambda_d \leq 1$ implies $\mu_1 \geq 1$ and $\alpha = \Theta_{d,k}((\mu_1\cdots\mu_q)^{1/(q-1)})$.

We now proceed by induction on $d-k$. The case $d-k=1$ is treated similarly as in the proof of Theorem 1 by Bárány et al. [5]. Using the pigeonhole principle, we can construct a set D' of primitive points from $\Lambda^* \setminus \{0\}$ such that $|D'| \leq O_d(\alpha)$ and such that for every $x \in \Lambda \cap B^d$ there is $z \in D'$ with $\langle x,z\rangle = 0$. We let $\mathcal{S}$ to be the set of hyperplanes that contain the origin and have normal vectors from D'. Observe that $\mathcal{S}$ is a set of $O_d(\alpha) = O_d(\alpha^{d-k})$ $(d-1)$-dimensional linear subspaces that cover $\Lambda \cap B^d$.

For the inductive step, assume that $d-k \geq 2$. We consider the set $\mathcal{S}$ of hyperplanes in $\mathbb{R}^d$ that has been constructed in the base of the induction. For every hyperplane $H(z) \in \mathcal{S}$ with the normal vector $z \in D'$, we let $\Lambda_{H(z)}$ be the set $\Lambda \cap H(z)$. Note that $\Lambda_{H(z)}$ is a lattice of dimension at most $d-1$. We now proceed inductively and cover each set $\Lambda_{H(z)} \cap B^d$ using the inductive hypothesis for $\Lambda_{H(z)}$ and k. To do so, we employ the fact that, for every $z \in D'$, the larger $\|z\|$ is, the fewer k-dimensional linear subspaces we need to cover $\Lambda_{H(z)} \cap B^d$. In particular, we prove that if z is a point from D' and $q \geq d-k+2$, then $\Lambda_{H(z)} \cap B^d$ can be covered with $O_{d,k}\left(((\mu_1\cdots\mu_q)/\|z\|)^{(d-k-1)/(q-2)}\right)$ k-dimensional linear subspaces.

Then we partition D' into subsets $S_1,\dots,S_q$ such that all vectors from S_i have approximately the same Euclidean norm. Then, for every $i \in [q]$, we sum the number c_i of k-dimensional linear subspaces needed to cover $\Lambda_{H(z)} \cap B^d$ for $z \in S_i$ and show that $c_1 + \cdots + c_q \leq O_{d,k}(\alpha^{d-k})$.

3.2 The general case

Here, we finish the proof of Theorem 4 by extending Theorem 14 to arbitrary convex bodies from $\mathcal{K}^d$. This is done by approximating a given body K from $\mathcal{K}^d$ with ellipsoids. A *d-dimensional ellipsoid* in $\mathbb{R}^d$ is an image of B^d under a nonsingular affine map. Such approximation exists by the following classical result, called *John's lemma* [14].

▶ **Lemma 17** (John's lemma [17, see Theorem 13.4.1]). *For every positive integer d and every $K \in \mathcal{K}^d$, there is a d-dimensional ellipsoid E with the center in the origin that satisfies*

$$E/\sqrt{d} \subseteq K \subseteq E.$$

Let $\Lambda \in \mathcal{L}^d$ be a given lattice and let $\lambda_i := \lambda_i(\Lambda, K)$ for every $i \in [d]$. From our assumptions, we know that $\lambda_d \leq 1$. Let E be the ellipsoid from Lemma 17. Since E is an ellipsoid, there is a nonsingular affine map $h\colon \mathbb{R}^d \to \mathbb{R}^d$ such that $E = h(B^d)$. Since E is centered in the origin, we see that h is in fact a linear map. Thus $\Lambda' := h^{-1}(\Lambda) \in \mathcal{L}^d$. Observe that we have $\lambda_i = \lambda_i(\Lambda', h^{-1}(K))$ for every $i \in [d]$.

For every $i \in [d]$, we use λ'_i to denote the ith successive minimum $\lambda_i(\Lambda', B^d) = \lambda_i(\Lambda, E)$. From the choice of E, we have $\lambda_i/\sqrt{d} \leq \lambda'_i \leq \lambda_i$. In particular, $\lambda'_d \leq 1$. Thus, by Theorem 14,

the set $\Lambda' \cap B^d$ can be covered with $O_{d,k}((\alpha')^{d-k})$ k-dimensional linear subspaces, where $\alpha' := \min_{1\le j\le k}(\lambda'_j \cdots \lambda'_d)^{-1/(d-j)}$.

Since $\lambda_i = \Theta_d(\lambda'_i)$ for every $i \in [d]$, we see that the set $\Lambda' \cap h^{-1}(K)$ can be covered with $O_{d,k}(\alpha^{d-k})$ k-dimensional linear subspaces, where $\alpha := \min_{1\le j\le k}(\lambda_j \cdots \lambda_d)^{-1/(d-j)}$. Since every nonsingular linear transformation preserves incidences and successive minima and maps a k-dimensional linear subspace to a k-dimensional linear subspace, the set $\Lambda \cap K$ can be covered with $O_{d,k}(\alpha^{d-k})$ k-dimensional linear subspaces.

4 Proof of Theorem 5

Let d and k be positive integers satisfying $1 \le k \le d-1$ and let K be a body from $\mathcal{K}^d$ with $\lambda_d(\mathbb{Z}^d, K) \le 1$. For every $i \in [d]$, we let λ_i be the ith successive minimum $\lambda_i(\mathbb{Z}^d, K)$. Let ε be a number from $(0,1)$. We use a probabilistic approach to show that there is a set $S \subseteq \mathbb{Z}^d \cap K$ of size at least $\Omega_{d,\varepsilon,k}(((1-\lambda_d)\beta)^{d-k-\varepsilon})$, where $\beta := \min_{1\le j\le d-1}(\lambda_j \cdots \lambda_d)^{-1/(d-j)}$, such that every k-dimensional linear subspace contains at most $r-1$ points from S.

Note that it is sufficient to prove the statement only for the lattice $\mathbb{Z}^d$. For a general lattice $\Lambda \in \mathcal{L}^d$ we can apply a linear transformation h such that $h(\Lambda) = \mathbb{Z}^d$ and then use the result for $\mathbb{Z}^d$ and $h(K)$, since $\lambda_i(\Lambda, K) = \lambda_i(\mathbb{Z}^d, h(K))$ for every $i \in [d]$. We also remark that in the case $k = d-1$ the stronger lower bound $\Omega_d((1-\lambda_d)\beta)$ from Theorem 1 by Bárány et al. [5] applies.

The proof is based on the following two results, first of which is by Bárány et al. [5].

▶ **Lemma 18** ([5]). *For an integer $d \ge 2$ and $K \in \mathcal{K}^d$, if $\lambda_d < 1$ and p is an integer satisfying $1 < p < (1-\lambda_d)\beta/(8d^2)$, then, for every $v \in \mathbb{R}^d$, there exist an integer $1 \le j < p$ and a point $w \in \mathbb{Z}^d$ with $jv + pw \in K$.*

For a prime number p, let $\mathbb{F}_p$ be the finite field of size p. The second main ingredient in the proof of Theorem 5 is the following lemma.

▶ **Lemma 19.** *Let d and k be integers satisfying $2 \le k \le d-2$ and let $\varepsilon \in (0,1)$. Then there is a positive integer $p_0 = p_0(d,\varepsilon,k)$ such that for every prime number $p \ge p_0$ there exists a subset R of $\mathbb{F}_p^{d-1}$ of size at least $p^{d-k-\varepsilon}/2$ such that every $(k-1)$-dimensional affine subspace of $\mathbb{F}_p^{d-1}$ contains at most $r-1$ points from R for $r := \lceil k(d-k+1)/\varepsilon \rceil$.*

Proof. We assume that p is large enough with respect to d, ε, and k so that $p^{k-1} > r$. We set $P := p^{1-k-\varepsilon}$ and we let X be a subset of $\mathbb{F}_p^{d-1}$ obtained by choosing every point from $\mathbb{F}_p^{d-1}$ independently at random with the probability P.

Let A be a $(k-1)$-dimensional affine subspace of $\mathbb{F}_p^{d-1}$. Then $|A| = p^{k-1}$. It is well-known that the number of $(k-1)$-dimensional linear subspaces of $\mathbb{F}_p^{d-1}$ is exactly the *Gaussian binomial coefficient*

$$\begin{aligned}\begin{bmatrix} d-1 \\ k-1 \end{bmatrix}_p &:= \frac{(p^{d-1}-1)(p^{d-1}-p)\cdots(p^{d-1}-p^{k-2})}{(p^{k-1}-1)(p^{k-1}-p)\cdots(p^{k-1}-p^{k-2})} \\ &\le \frac{p^{d-1}\cdot p^{d-2}\cdots p^{d-k+1}}{(p^{k-1}-1)(p^{k-2}-1)\cdots(p-1)} \le p^{(k-1)d-(k-1)k/2-(k-1)(k-2)/2} = p^{(k-1)(d-k+1)}. \quad (4)\end{aligned}$$

We used the fact $p^{k-i} - 1 \ge p^{k-i-1}$ for $k > i$ in the last inequality.

Since every $(k-1)$-dimensional affine subspace A of $\mathbb{F}_p^{d-1}$ is of the form $A = x + L$ for some $x \in \mathbb{F}_p^{d-1}$ and a $(k-1)$-dimensional linear subspace L of $\mathbb{F}_p^{d-1}$ and $x + L = y + L$ if and only if $x - y \in L$, the total number of $(k-1)$-dimensional affine subspaces of $\mathbb{F}_p^{d-1}$

is $p^{d-k}\begin{bmatrix} d-1 \\ k-1 \end{bmatrix}_p$. This is because by considering pairs (x, L), where $x \in \mathbb{F}_p^{d-1}$ and L is a $(k-1)$-dimensional linear subspace of $\mathbb{F}_p^{d-1}$, every $(k-1)$-dimensional affine subspace A is counted p^{k-1} times.

We use the following Chernoff-type bound (see the last bound of [12]) to estimate the probability that A contains at least r points of X. Let $q \in [0,1]$ and let $Y_1, \dots, Y_m$ be independent 0-1 random variables with $\Pr[Y_i = 1] = q$ for every $i \in [m]$. Then, for $mq \le s < m$, we have

$$\Pr[Y_1 + \cdots + Y_m \ge s] \le \left(\frac{mq}{s}\right)^s e^{s-mq}. \tag{5}$$

Choosing Y_x as the indicator variable for the event $x \in A \cap X$ for each $x \in A$, we have $m = |A| = p^{k-1}$ and $q = P$. Since $p, r \ge 1$ and $p^{k-1} > r$, we have $p^{-\varepsilon} = mq \le r < m = p^{k-1}$ and thus the bound (5) implies

$$\Pr[|A \cap X| \ge r] \le \left(\frac{p^{k-1}P}{r}\right)^r e^{r-p^{k-1}P} = \left(\frac{p^{-\varepsilon}}{r}\right)^r e^{r-p^{-\varepsilon}} = p^{-\varepsilon r} e^{r(1-\ln r)-p^{-\varepsilon}} < p^{-\varepsilon r},$$

where the last inequality follows from $r \ge e$, as then $1 - \ln r \le 0$.

By the Union bound, the probability that there is a $(k-1)$-dimensional affine subspace A of $\mathbb{F}_p^{d-1}$ with $|A \cap X| \ge r$ is less than

$$p^{d-k}\begin{bmatrix} d-1 \\ k-1 \end{bmatrix}_p \cdot p^{-\varepsilon r} \le p^{(d-k)+(k-1)(d-k+1)-\varepsilon r} \le p^{k(d-k+1)-1-k(d-k+1)} = p^{-1},$$

where the first inequality follows from (4) and the second inequality is due to the choice of r. From $p \ge 2$, we see that this probability is less than $1/2$.

The expected size of X is $\mathbb{E}[|X|] = |\mathbb{F}_p^{d-1}| \cdot P = p^{d-1}p^{1-k-\varepsilon} = p^{d-k-\varepsilon}$. Since $|X| \sim \mathrm{Bi}(p^{d-1}, P)$, the variance of $|X|$ is $p^{d-1}P(1-P) < p^{d-k-\varepsilon}$ and Chebyshev's inequality implies $\Pr[||X| - \mathbb{E}[|X|]| \ge \sqrt{2p^{d-k-\varepsilon}}] < p^{d-k-\varepsilon}/(2p^{d-k-\varepsilon}) = 1/2$.

Thus there is a set R of size at least $p^{d-k-\varepsilon} - \sqrt{2p^{d-k-\varepsilon}} \ge p^{d-k-\varepsilon}/2$ such that every $(k-1)$-dimensional affine subspace of $\mathbb{F}_p^{d-1}$ contains at most $r-1$ points from R. ◀

Let $\varepsilon \in (0,1)$ be given. To derive Theorem 5, we combine Lemma 18 with Lemma 19. This is a similar approach as in [5], where the authors derive a lower bound for the case $k = d-1$ by combining Lemma 18 with a construction found by Erdős in connection with Heilbronn's triangle problem [20].

Let p be the largest prime number that satisfies the assumptions of Lemma 18. If such p does not exist, then the statement of the theorem is trivial. By Bertrand's postulate, we have $p > (1-\lambda_d)\beta/(16d^2)$. We may assume that $p \ge p_0$, where $p_0 = p_0(d, \varepsilon, k)$ is the constant from Lemma 19, since otherwise the statement of Theorem 5 is trivial.

For $k \ge 2$ and $t := \lceil p^{d-k-\varepsilon}/2 \rceil$, let $R = \{v_1, \dots, v_t\} \subseteq \mathbb{F}_p^{d-1}$ be the set of points from Lemma 19. That is, every $(k-1)$-dimensional affine subspace of $\mathbb{F}_p^{d-1}$ contains at most $r-1$ points from R for $r := \lceil k(d-k+1)/\varepsilon \rceil$. In particular, every r-tuple of points from R contains $k+1$ affinely independent points over the field $\mathbb{F}_p$. For $k = 1$, we can set $r := 2$ and let R be the whole set $\mathbb{F}_p^{d-1}$ of size $t := p^{d-k} = p^{d-1}$. Then every r-tuple of points from R contains two affinely independent points over the field $\mathbb{F}_p$.

For $i = 1, \dots, t$, let $u_i \in \mathbb{Z}^d$ be the vector obtained from v_i by adding 1 as the last coordinate. From the choice of R, every r-tuple of points from $\{u_1, \dots, u_t\}$ contains $k+1$ points that are linearly independent over the field $\mathbb{F}_p$.

By Lemma 18, there exist an integer $1 \le j_i < p$ and a point $w_i \in \mathbb{Z}^d$ for every $i \in [t]$ such that $u_i' := j_i u_i + p w_i$ lies in K. We have $u_i' \equiv j_i u_i \pmod p$ for every $i \in [t]$ and thus every

r-tuple of vectors from $S := \{u'_1, \dots, u'_t\} \subseteq \mathbb{Z}^d$ contains $k+1$ linearly independent vectors over the field $\mathbb{F}_p$, and hence over $\mathbb{R}$. In other words, every k-dimensional linear subspace of $\mathbb{R}^d$ contains at most $r-1$ points from S. Since $|S| = t = \lceil p^{d-k-\varepsilon}/2 \rceil$ and $p > (1-\lambda_d)\beta/(16d^2)$, we have $l(d,k,n,r) \geq \Omega_{d,k}(((1-\lambda_d)\beta)^{d-k-\varepsilon})$. This completes the proof of Theorem 5.

5 Proof of Theorem 8

Let d and k be integers with $1 \leq k \leq d-1$ and let $\Lambda \in \mathcal{L}^d$ and $K \in \mathcal{K}^d$. We let $\lambda_i := \lambda_i(\Lambda, K)$ for every $i \in [d]$ and assume that $\lambda_d \leq 1$. First, we observe that it is sufficient to prove the statement only for $K = B^d$, as we can then strengthen the statement to an arbitrary $K \in \mathcal{K}^d$ using John's lemma (Lemma 17) analogously as in the proof of Theorem 4.

First, we prove the upper bound. That is, we show that $\Lambda \cap B^d$ can be covered with $O_{d,k}((\lambda_{k+1} \cdots \lambda_d)^{-1})$ k-dimensional affine subspaces of $\mathbb{R}^d$. By Lemma 15, there is a positive integer $r = r(d,k)$ and a projection p of $\mathbb{R}^d$ along k vectors $b_1, \dots, b_k$ from Λ onto a $(d-k)$-dimensional linear subspace N of $\mathbb{R}^d$ such that $\Lambda \cap B^d$ is mapped to $\Lambda \cap N \cap B^d(r)$ and such that $\lambda'_i := \lambda_i(\Lambda \cap N, B^d(r) \cap N) = \Theta_{d,k}(\lambda_{i+k})$ for every $i \in [d-k]$.

For each point z of $\Lambda \cap N \cap B^d(r)$, we define $A(z)$ to be the affine hull of the set $\{z, b_1 + z, \dots, b_k + z\}$. Every $A(z)$ is then a k-dimensional affine subspace of $\mathbb{R}^d$ and the set $\mathcal{A} := \{A(z) \colon z \in \Lambda \cap N \cap B^d(r)\}$ covers $\Lambda \cap B^d$, since $p(z) \in \Lambda \cap N \cap B^d(r)$ for every $z \in \Lambda \cap B^d$. We have $|\mathcal{A}| = |\Lambda \cap N \cap B^d(r)|$ and, since $\lambda_d \leq 1$ and $\lambda'_1 \leq \cdots \leq \lambda'_{d-k} \leq O_{d,k}(\lambda_d)$, Theorem 12 implies $|\Lambda \cap N \cap B^d(r)| \leq O_{d,k}((\lambda'_1 \cdots \lambda'_{d-k})^{-1})$. The bound $\lambda'_i \geq \Omega_{d,k}(\lambda_{i+k})$ for every $i \in [d-k]$ then gives $|\mathcal{A}| \leq O_{d,k}((\lambda_{k+1} \cdots \lambda_d)^{-1})$.

To show the lower bound, we prove that we need at least $\Omega_{d,k}((\lambda_{k+1} \cdots \lambda_d)^{-1})$ k-dimensional affine subspaces of $\mathbb{R}^d$ to cover $\Lambda \cap B^d$.

Let A be a k-dimensional affine subspace of $\mathbb{R}^d$. We show that A contains at most $O_{d,k}((\lambda_1 \cdots \lambda_k)^{-1})$ points from $\Lambda \cap B^d$. Let y be an arbitrary point from $\Lambda \cap A \cap B^d$. Then $A = L + y$, where L is a k-dimensional linear subspace of $\mathbb{R}^d$, and $(\Lambda \cap A) - y = \Lambda \cap L$. For every $i \in [k]$, we let $\lambda'_i := \lambda_i(\Lambda \cap L, B^d(2))$ and we observe that $\lambda'_i \geq \lambda_i/2$. By Theorem 12, we have $|\Lambda \cap L \cap B^d(2)| \leq O_{d,k}((\lambda'_1 \cdots \lambda'_s)^{-1})$, where s is the maximum integer j from $[k]$ with $\lambda'_j \leq 1$. Since $\lambda'_i \geq \lambda_i/2$ for every $i \in [k]$, we have $|\Lambda \cap L \cap B^d(2)| \leq O_{d,k}((\lambda_1 \cdots \lambda_k)^{-1})$. For every $x \in A \cap B^d$, we have $\|x - y\| \leq \|x\| + \|y\| \leq 2$ and thus $x - y \in L \cap B^d(2)$. It follows that $(\Lambda \cap A \cap B^d) - y \subseteq \Lambda \cap L \cap B^d(2)$ and thus $|\Lambda \cap A \cap B^d| \leq O_{d,k}((\lambda_1 \cdots \lambda_k)^{-1})$.

Let $\mathcal{A}$ be a collection of k-dimensional affine subspaces of $\mathbb{R}^d$ that covers $\Lambda \cap B^d$. We have $|\mathcal{A}| \geq |\Lambda \cap B^d|/m$, where m is the maximum of $|\Lambda \cap A \cap B^d|$ taken over all subspaces A from $\mathcal{A}$. We know that $m \leq O_{d,k}((\lambda_1 \cdots \lambda_k)^{-1})$. It is a well-known fact that follows from Minkowski's second theorem (Theorem 11) that $|\Lambda \cap B^d| \geq \Omega_{d,k}((\lambda_1 \cdots \lambda_d)^{-1})$. Thus we obtain

$$|\mathcal{A}| \geq \frac{|\Lambda \cap B^d|}{m} \geq \frac{\Omega_{d,k}((\lambda_1 \cdots \lambda_d)^{-1})}{O_{d,k}((\lambda_1 \cdots \lambda_k)^{-1})} \geq \Omega_{d,k}((\lambda_{k+1} \cdots \lambda_d)^{-1}),$$

which finishes the proof of Theorem 8.

6 Proofs of Theorems 9 and 10

We now improve the lower bounds from Theorem 3 on the number of point-hyperplane incidences. We use essentially the same construction as Brass and Knauer [6].

Assume that we are given integers d and k with $0 \leq k \leq d-2$ and let ε be a real number in $(0,1)$. Let $\delta = \delta(d, \varepsilon, k) \in (0,1)$ be a sufficiently small constant. By (1), there is a positive

integer $r_1 = r_1(d, \delta, k)$ and a constant $c_1 = c_1(d, \delta, k)$ such that for every $s \in \mathbb{N}$ there is a subset P of $\mathbb{Z}^d \cap B^d(s)$ of size $c_1 \cdot s^{d-k-\delta}$ such that every k-dimensional affine subspace of $\mathbb{R}^d$ contains at most $r_1 - 1$ points from P. In the case $k = 0$, we can clearly obtain the stronger bound $c_1 \cdot s^d$.

By Corollary 6, there is a positive integer $r_2 = r_2(d, \delta, k)$ and a constant $c_2 = c_2(d, \delta, k)$ such that for every $t \in \mathbb{N}$ there is a subset N' of $\mathbb{Z}^d \cap B^d(t)$ of size $c_2 \cdot t^{d(k+1-\delta)/(d-1)}$ such that every $(d-k-1)$-dimensional linear subspace contains at most $r_2 - 1$ points from N'. In particular, every 1-dimensional linear subspace contains at most $r_2 - 1$ points from N' and thus there is a set $N \subseteq N'$ of size $|N| = |N'|/(r_2 - 1) = c_2 \cdot t^{d(k+1-\delta)/(d-1)}/(r_2 - 1)$ containing only primitive vectors. We note that for $k = 0$ we can apply Theorem 1 instead of Corollary 6 and obtain the stronger bound $|N| = c_2 \cdot t^{d/(d-1)}/(r_2 - 1)$. We let $\mathcal{H}$ be the set of hyperplanes in $\mathbb{R}^d$ with normal vectors from N such that every hyperplane from $\mathcal{H}$ contains at least one point of P.

We show that the graph $G(P, \mathcal{H})$ does not contain K_{r_1,r_2}. If there is an r_2-tuple of hyperplanes from $\mathcal{H}$ with a nonempty intersection, then these hyperplanes have distinct normal vectors that span a linear subspace of dimension at least $d - k$ by the choice of N. The intersection of these hyperplanes is thus an affine subspace of dimension at most k. From the definition of P, it contains at most $r_1 - 1$ points from P.

We set $n := c_1 \cdot s^{d-k-\delta}$ and $m := \frac{3c_2}{r_2-1} \cdot s \cdot t^{d(k+2-1/d-\delta)/(d-1)}$. Then we have $|P| = n$. For every $p \in P$ and $z \in N$, we have $\langle p, z\rangle \in \mathbb{Z}$ and $|\langle p, z\rangle| \le \|p\|\|z\| \le st$ by the Cauchy–Schwarz inequality. Thus every point z from N is the normal vector of at most $2st + 1 \le 3st$ hyperplanes from $\mathcal{H}$. It follows that

$$|\mathcal{H}| \le 3st|N| = 3st\frac{c_2 \cdot t^{d(k+1-\delta)/(d-1)}}{r_2 - 1} = \frac{3c_2}{r_2 - 1} \cdot s \cdot t^{d(k+2-1/d-\delta)/(d-1)} = m.$$

From the definition of $\mathcal{H}$, the number of incidences between P and $\mathcal{H}$ is at least

$$\begin{aligned} |P||N| &= n \cdot \frac{c_2 \cdot t^{d(k+1-\delta)/(d-1)}}{r_2 - 1} = \Omega_{d,\varepsilon,k}\left(n \cdot (m/s)^{(k+1-\delta)/(k+2-1/d-\delta)}\right) \\ &= \Omega_{d,\varepsilon,k}\left(n^{1-(k+1-\delta)/((k+2-1/d-\delta)(d-k-\delta))} m^{(k+1-\delta)/(k+2-1/d-\delta)}\right) \\ &\ge \Omega_{d,\varepsilon,k}\left(n^{1-(k+1)/((k+2-1/d)(d-k))-\varepsilon} m^{(k+1)/(k+2-1/d)-\varepsilon}\right), \end{aligned} \tag{6}$$

where the last inequality holds for δ sufficiently small with respect to d, ε, and k. This finishes the proof of Theorem 10.

To maximize the number of incidences in the diagonal case, we choose $k := \lfloor \frac{d-2}{2} \rfloor$. For d odd, we then have at least

$$\Omega_{d,\varepsilon}\left(n^{1-2(d-1)/((d+1-2/d)(d+3))-\varepsilon} m^{(d-1)/(d+1-2/d)-\varepsilon}\right)$$

incidences by (6). By duality, we may obtain a symmetrical expression by averaging the exponents. Then we obtain

$$\mathrm{I}(P, \mathcal{H}) \ge \Omega_{d,\varepsilon}\left((mn)^{(d^2+3d+3)/(d^2+5d+6)-\varepsilon}\right) = \Omega_{d,\varepsilon}\left((mn)^{1-(2d+3)/((d+2)(d+3))-\varepsilon}\right).$$

For d even, the choice of k implies that the number of incidences is at least

$$\Omega_{d,\varepsilon}\left(n^{1-2d/((d+2-2/d)(d+2))-\varepsilon} m^{d/(d+2-2/d)-\varepsilon}\right)$$

by (6). Using the averaging argument, we obtain

$$\begin{aligned}\mathrm{I}(P,\mathcal{H}) &\geq \Omega_{d,\varepsilon}\left((mn)^{(d^3+2d^2+d-2)/((d+2)(d^2+2d-2))-\varepsilon}\right)\\ &= \Omega_{d,\varepsilon}\left((mn)^{1-(2d^2+d-2)/((d+2)(d^2+2d-2))-\varepsilon}\right).\end{aligned}$$

This completes the proof of Theorem 9. For $d \leq 3$, we have $k = 0$ and thus we can get rid of the ε in the exponent by applying the stronger bounds on m and n.

▶ Remark. An upper bound similar to (2) holds in a much more general setting, where we bound the maximum number of edges in $K_{r,r}$-free *semi-algebraic bipartite graphs* $G = (P \cup Q, E)$ *in* $(\mathbb{R}^d, \mathbb{R}^d)$ *with bounded description complexity* t (see [11] for definitions). Fox et al. [11] showed that the maximum number of edges in such graphs with $|P| = n$ and $|Q| = m$ is at most $O_{d,\varepsilon,r,t}((mn)^{1-1/(d+1)+\varepsilon} + m + n)$ for any $\varepsilon > 0$. Theorem 9 provides the best known lower bound for this problem, as every incidence graph $G(P, \mathcal{H})$ of P and $\mathcal{H}$ in $\mathbb{R}^d$ is a semi-algebraic graph in $(\mathbb{R}^d, \mathbb{R}^d)$ with bounded description complexity.

References

1. E. Ackerman. On topological graphs with at most four crossings per edge. `http://arxiv.org/abs/1509.01932`, 2015.
2. R. Apfelbaum and M. Sharir. Large complete bipartite subgraphs in incidence graphs of points and hyperplanes. *SIAM J. Discrete Math.*, 21(3):707–725, 2007.
3. M. Balko, J. Cibulka, and P. Valtr. Covering lattice points by subspaces and counting point-hyperplane incidences. `http://arxiv.org/abs/1703.04767`, 2017.
4. W. Banaszczyk. New bounds in some transference theorems in the geometry of numbers. *Math. Ann.*, 296(4):625–635, 1993.
5. I. Bárány, G. Harcos, J. Pach, and G. Tardos. Covering lattice points by subspaces. *Period. Math. Hungar.*, 43(1–2):93–103, 2001.
6. P. Brass and C. Knauer. On counting point-hyperplane incidences. *Comput. Geom.*, 25(1–2):13–20, 2003.
7. P. Brass, W. Moser, and J. Pach. *Research problems in discrete geometry.* Springer, New York, 2005.
8. B. Chazelle. Cutting hyperplanes for Divide-and-Conquer. *Discrete Comput. Geom.*, 9(2):145–158, 1993.
9. P. Erdős. On sets of distances of n points. *Amer. Math. Monthly*, 53:248–250, 1946.
10. J. Erickson. New lower bounds for hopcroft's problem. *Discrete Comput. Geom.*, 16(4):389–418, 1996.
11. J. Fox, J. Pach, A. Sheffer, and A. Suk. A semi-algebraic version of Zarankiewicz's problem. `http://arxiv.org/abs/1407.5705`, 2014.
12. T. Hagerup and C. Rüb. A guided tour of Chernoff bounds. *Inform. Process. Lett.*, 33(6):305–308, 1990.
13. M. Henk. Successive minima and lattice points. *Rend. Circ. Mat. Palermo (2) Suppl.*, 70(I):377–384, 2002.
14. F. John. Extremum problems with inequalities as subsidiary conditions. In *Studies and Essays, presented to R. Courant on his 60th birthday, January 8, 1948*, pages 187–204. Interscience Publ., New York, 1948.
15. H. Lefmann. Extensions of the No-Three-In-Line Problem. `www.tu-chemnitz.de/informatik/ThIS/downloads/publications/lefmann_no_three_submitted.pdf`, 2012.
16. K. Mahler. Ein Übertragungsprinzip für konvexe Körper. *Časopis Pěst. Mat. Fys.*, 68:93–102, 1939.

17 J. Matoušek. *Lectures on Discrete Geometry*, volume 212 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 2002.

18 H. Minkowski. *Geometrie der Zahlen*. Leipzig, Teubner, 1910.

19 János Pach and Géza Tóth. Graphs drawn with few crossings per edge. *Combinatorica*, 17:427–439, 1997.

20 K. F. Roth. On a problem of Heilbronn. *J. London Math. Soc.*, 26:198–204, 1951.

21 Adam Sheffer. Lower bounds for incidences with hypersurfaces. *Discrete Anal.*, 2016. Paper No. 16, 14.

22 C. L. Siegel and K. Chandrasekharan. *Lectures on the geometry of numbers*. Springer-Verlag, Berlin, 1989.

23 E. Szemerédi and W. T. Trotter Jr. Extremal problems in discrete geometry. *Combinatorica*, 3(3–4):381–392, 1983.

Subquadratic Algorithms for Algebraic Generalizations of 3SUM

Luis Barba[1], Jean Cardinal[*2], John Iacono[†3], Stefan Langerman[‡4], Aurélien Ooms[§5], and Noam Solomon[¶6]

1 **Department of Computer Science, ETH Zürich, Zürich, Switzerland**
luis.barba@inf.ethz.ch
2 **Département d'Informatique, Université libre de Bruxelles (ULB), Brussels, Belgium**
jcardin@ulb.ac.be
3 **Department of Computer Science and Engineering, New York University (NYU), New York, NY, USA**
socg2017@johniacono.com
4 **Département d'Informatique, Université libre de Bruxelles (ULB), Brussels, Belgium**
slanger@ulb.ac.be
5 **Département d'Informatique, Université libre de Bruxelles (ULB), Brussels, Belgium**
aureooms@ulb.ac.be
6 **School of Computer Science, Tel Aviv University (TAU), Tel Aviv, Israel**
noam.solom@gmail.com

Abstract

The 3SUM problem asks if an input n-set of real numbers contains a triple whose sum is zero. We consider the 3POL problem, a natural generalization of 3SUM where we replace the sum function by a constant-degree polynomial in three variables. The motivations are threefold. Raz, Sharir, and de Zeeuw gave an $O(n^{11/6})$ upper bound on the number of solutions of trivariate polynomial equations when the solutions are taken from the cartesian product of three n-sets of real numbers. We give algorithms for the corresponding problem of counting such solutions. Grønlund and Pettie recently designed subquadratic algorithms for 3SUM. We generalize their results to 3POL. Finally, we shed light on the General Position Testing (GPT) problem: "Given n points in the plane, do three of them lie on a line?", a key problem in computational geometry.

We prove that there exist bounded-degree algebraic decision trees of depth $O(n^{\frac{12}{7}+\varepsilon})$ that solve 3POL, and that 3POL can be solved in $O(n^2(\log\log n)^{\frac{3}{2}}/(\log n)^{\frac{1}{2}})$ time in the real-RAM model. Among the possible applications of those results, we show how to solve GPT in subquadratic time when the input points lie on $o((\log n)^{\frac{1}{6}}/(\log\log n)^{\frac{1}{2}})$ constant-degree polynomial curves. This constitutes the first step towards closing the major open question of whether GPT can be solved in subquadratic time. To obtain these results, we generalize important tools – such as batch range searching and dominance reporting – to a polynomial setting. We expect these new tools to be useful in other applications.

* Supported by the "Action de Recherche Concertée" (ARC) COPHYMA, convention number 4.110.H.000023.
† Research partially completed while on sabbatical at the Algorithms Research Group of the Département d'Informatique at the Université libre de Bruxelles with support from a Fulbright Research Fellowship, the Fonds de la Recherche Scientifique – FNRS, and NSF grants CNS-1229185, CCF-1319648, and CCF-1533564.
‡ Directeur de recherches du F.R.S.-FNRS.
§ Supported by the Fund for Research Training in Industry and Agriculture (FRIA).
¶ Supported by Grant 892/13 from the Israel Science Foundation.

33rd International Symposium on Computational Geometry (SoCG 2017).
Editors: Boris Aronov and Matthew J. Katz; Article No. 13; pp. 13:1–13:15

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

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases 3SUM, subquadratic algorithms, general position testing, range searching, dominance reporting, polynomial curves

Digital Object Identifier 10.4230/LIPIcs.SoCG.2017.13

1 Introduction

The 3SUM problem is defined as follows: given n distinct real numbers, decide whether any three of them sum to zero. A popular conjecture is that no $O(n^{2-\delta})$-time algorithm for 3SUM exists. This conjecture has been used to show conditional lower bounds for problems in P, notably in computational geometry with problems such as GeomBase, general position [21] and Polygonal Containment [7], and more recently for string problems such as Local Alignment [2] and Jumbled Indexing [5], as well as dynamic versions of graph problems [32, 1], triangle enumeration and Set Disjointness [27]. For this reason, 3SUM is considered one of the key subjects of an emerging theory of complexity-within-P, along with other problems such as all-pairs shortest paths, orthogonal vectors, boolean matrix multiplication, and conjectures such as the Strong Exponential Time Hypothesis [3, 26, 11].

Because fixing two of the numbers a and b in a triple only allows for one solution to the equation $a+b+x=0$, an instance of 3SUM has at most n^2 solution triples. An instance with a matching lower bound is for example the set $\{\frac{1-n}{2},\ldots,\frac{n-1}{2}\}$ (for odd n) with $\frac{3}{4}n^2+\frac{1}{4}$ solution triples. One might be tempted to think that the number of solutions to the problem would lower bound the complexity of algorithms for the decision version of the problem, as it is the case for restricted models of computation [18]. This is a common misconception. Indeed, Grønlund and Pettie [23] recently proved that there exist $\tilde{O}(n^{3/2})$-depth linear decision trees and $o(n^2)$-time real-RAM algorithms for 3SUM.

A natural generalization of the 3SUM problem is to replace the sum function by a constant-degree polynomial in three variables $F\in\mathbb{R}[x,y,z]$ and ask to determine whether there exists any triple (a,b,c) of input numbers such that $F(a,b,c)=0$. We call this new problem the *3POL problem*.

For the particular case $F(x,y,z)=f(x,y)-z$ where $f\in\mathbb{R}[x,y]$ is a constant-degree bivariate polynomial, Elekes and Rónyai [16] show that the number of solutions to the 3POL problem is $o(n^2)$ unless f is *special*. Special for f means that f has one of the two special forms $f(u,v)=h(\varphi(u)+\psi(v))$ or $f(u,v)=h(\varphi(u)\cdot\psi(v))$, where h,φ,ψ are univariate polynomials of constant degree. Elekes and Szabó [17] later generalized this result to a broader range of functions F using a wider definition of specialness. Raz, Sharir and Solymosi [37] and Raz, Sharir and de Zeeuw [35] recently improved both bounds on the number of solutions to $O(n^{11/6})$. They translated the problem into an incidence problem between points and constant-degree algebraic curves. Then, they showed that unless f (or F) is special, these curves have low multiplicities. Finally, they applied a theorem due to Pach and Sharir [31] bounding the number of incidences between the points and the curves. Some of these ideas appear in our approach.

In computational geometry, it is customary to assume the real-RAM model can be extended to allow the computation of roots of constant degree polynomials. We distance ourselves from this practice and take particular care of using the real-RAM model and the bounded-degree algebraic decision tree model with only the four arithmetic operators.

1.1 Our results

We focus on the computational complexity of 3POL. Since 3POL contains 3SUM, an interesting question is whether a generalization of Grønlund and Pettie's 3SUM algorithm exists for 3POL. If this is true, then we might wonder whether we can beat the $O(n^{11/6}) = O(n^{1.833\ldots})$ combinatorial bound of Raz, Sharir and de Zeeuw [35] with nonuniform algorithms. We give a positive answer to both questions: we show there exist $O(n^2(\log\log n)^{\frac{3}{2}}/\log n)^{\frac{1}{2}})$-time real-RAM algorithms and $O(n^{12/7+\varepsilon}) = O(n^{1.7143})$-depth bounded-degree algebraic decision trees for 3POL.[1] To prove our main result, we present a fast algorithm for the Polynomial Dominance Reporting (PDR) problem, a far reaching generalization of the Dominance Reporting problem. As the algorithm for Dominance Reporting and its analysis by Chan [13] is used in fast algorithms for all-pairs shortest paths, (min,+)-convolutions, and 3SUM, we expect this new algorithm will have more applications.

Our results can be applied to many degeneracy testing problems, such as the General Position Testing (GPT) problem: "Given n points in the plane, do three of them lie on a line?" It is well known that GPT is 3SUM-hard, and it is open whether GPT admits a subquadratic algorithm. Raz, Sharir and de Zeeuw [35] give a combinatorial bound of $O(n^{11/6})$ on the number of collinear triples when the input points are known to be lying on a constant number of polynomial curves, provided those curves are neither lines nor cubic curves. A corollary of our first result is that GPT where the input points are constrained to lie on $o((\log n)^{\frac{1}{6}}/(\log\log n)^{\frac{1}{2}})$ constant-degree polynomial curves (including lines and cubic curves) admits a subquadratic real-RAM algorithm and a strongly subquadratic bounded-degree algebraic decision tree. Interestingly, both reductions from 3SUM to GPT on 3 lines (map a to $(a,0)$, b to $(b,2)$, and c to $(\frac{c}{2},1)$) and from 3SUM to GPT on a cubic curve (map a to (a^3,a), b to (b^3,b), and c to (c^3,c)) construct such special instances of GPT. This constitutes the first step towards closing the major open question of whether GPT can be solved in subquadratic time. This result is described in the arXiv e-print where we also explain how to apply our algorithms to the problems of counting triples of points spanning unit circles or triangles.

1.2 Definitions

3POL. We look at two different generalizations of 3SUM. In the first one, which we call 3POL, we replace the sum function by a trivariate polynomial of constant degree.

▶ **Problem** (3POL). *Let $F \in \mathbb{R}[x,y,z]$ be a trivariate polynomial of constant degree, given three sets A, B, and C, each containing n real numbers, decide whether there exist $a \in A$, $b \in B$, and $c \in C$ such that $F(a,b,c) = 0$.*

The second one is a special case of 3POL where we restrict the trivariate polynomial F to have the form $F(a,b,c) = f(a,b) - c$.

▶ **Problem** (explicit 3POL). *Let $f \in \mathbb{R}[x,y]$ be a bivariate polynomial of constant degree, given three sets A, B, and C, each containing n real numbers, decide whether there exist $a \in A$, $b \in B$, and $c \in C$ such that $c = f(a,b)$.*

We look at both uniform and nonuniform algorithms for explicit 3POL and 3POL. We begin with an $O(n^{\frac{12}{7}+\varepsilon})$-depth bounded-degree algebraic decision tree for explicit 3POL in §2. In

[1] Throughout this document, ε denotes a positive real number that can be made as small as desired.

§3, we continue by giving a similar real-RAM algorithm for explicit 3POL that achieves subquadratic running time. We show how to generalize those results to the implicit version of 3POL in the arXiv e-print.

Models of Computation Similarly to Grønlund and Pettie [23], we consider both nonuniform and uniform models of computation. For the nonuniform model, Grønlund and Pettie consider linear decision trees, where one is only allowed to manipulate the input numbers through linear queries to an oracle. Each linear query has constant cost and all other operations are free but cannot inspect the input. In this paper, we consider *bounded-degree algebraic decision trees (ADT)* [34, 41, 39], a natural generalization of linear decision trees, as the nonuniform model. In a bounded-degree algebraic decision tree, one performs constant cost branching operations that amount to test the sign of a constant-degree polynomial for a constant number of input numbers. Again, operations not involving the input are free. For the uniform model we consider the real-RAM model with only the four arithmetic operators.

The problems we consider require our algorithms to manipulate polynomial expressions and, potentially, their real roots. For that purpose, we will rely on Collins cylindrical algebraic decomposition (CAD) [14]. To understand the power of this method, and why it is useful for us, we give some background on the related concept of first-order theory of the reals.

▶ **Definition 1.** A Tarski formula $\phi \in \mathbb{T}$ is a grammatically correct formula consisting of real variables ($x \in \mathbb{X}$), universal and existential quantifiers on those real variables ($\forall, \exists\colon \mathbb{X} \times \mathbb{T} \to \mathbb{T}$), the boolean operators of conjunction and disjunction ($\wedge, \vee\colon \mathbb{T}^2 \to \mathbb{T}$), the six comparison operators ($<, \leq, =, \geq, >, \neq\colon \mathbb{R}^2 \to \mathbb{T}$), the four arithmetic operators ($+, -, *, /\colon \mathbb{R}^2 \to \mathbb{R}$), the usual parentheses that modify the priority of operators, and constant real numbers. A Tarski sentence is a fully quantified Tarski formula. The first-order theory of the reals ($\forall\exists\mathbb{R}$) is the set of true Tarski sentences.

Tarski [40] and Seidenberg [38] proved that $\forall\exists\mathbb{R}$ is decidable. However, the algorithm resulting from their proof has nonelementary complexity. This proof, as well as other known algorithms, are based on quantifier elimination, that is, the translation of the input formula to a much longer quantifier-free formula, whose validity can be checked. There exists a family of formulas for which any method of quantifier elimination produces a doubly exponential size quantifier-free formula [15]. Collins CAD matches this doubly exponential complexity.

▶ **Theorem 2** (Collins [14]). *$\forall\exists\mathbb{R}$ can be solved in $2^{2^{O(n)}}$ time.*

See Basu, Pollack, and Roy [9] for additional details, Basu, Pollack, and Roy [8] for a singly exponential algorithm when all quantifiers are existential (existential theory of the reals, $\exists\mathbb{R}$), Caviness and Johnson [12] for an anthology of key papers on the subject, and Mishra [29] for a review of techniques to compute with roots of polynomials.

Collins CAD solves any *geometric* decision problem that does not involve quantification over the integers in time doubly exponential in the problem size. This does not harm our results as we exclusively use this algorithm to solve constant size subproblems. Geometric is to be understood in the sense of Descartes and Fermat, that is, the geometry of objects that can be expressed with polynomial equations. In particular, it allows us to make the following computations in the real-RAM and bounded-degree ADT models:

1. Given a constant-degree univariate polynomial, count its real roots in $O(1)$ operations,
2. Given a constant number of univariate polynomials of constant degree, compute the interleaving of their real roots in $O(1)$ operations,
3. Given a point in the plane and an arrangement of a constant number of constant-degree polynomial planar curves, locate the point in the arrangement in $O(1)$ operations.

Instead of bounded-degree algebraic decision trees as the nonuniform model we could consider decision trees in which each decision involves a constant-size instance of the decision problem in the first-order theory of the reals. The depth of a bounded-degree algebraic decision tree simulating such a tree would only be blown up by a constant factor.

1.3 Previous Results

3SUM. For the sake of simplicity, we consider the following definition of 3SUM

▶ **Problem** (3SUM). *Given 3 sets A, B, and C, each containing n real numbers, decide whether there exist $a \in A$, $b \in B$, and $c \in C$ such that $c = a + b$.*

A quadratic lower bound for solving 3SUM holds in a restricted model of computation: the 3-linear decision tree model. Erickson [18] and Ailon and Chazelle [4] showed that in this model, where one is only allowed to test the sign of a linear expression of up to three elements of the input, there are a quadratic number of critical tuples to test.

▶ **Theorem 3** (Erickson [18]). *The depth of a 3-linear decision tree for 3SUM is $\Omega(n^2)$.*

While no evidence suggested that this lower bound could be extended to other models of computation, it was eventually conjectured that 3SUM requires $\Omega(n^2)$ time.

Baran et al. [6] were the first to give concrete evidence for doubting the conjecture. They gave subquadratic Las Vegas algorithms for 3SUM, where input numbers are restricted to be integer or rational, in the circuit RAM, word RAM, external memory, and cache-oblivious models of computation. Their idea is to exploit the parallelism of the models, using linear and universal hashing.

Grønlund and Pettie [23], using a trick due to Fredman [19], recently showed that there exist subquadratic decision trees for 3SUM when the queries are allowed to be 4-linear.

▶ **Theorem 4** (Grønlund and Pettie [23]). *There is a 4-linear decision tree of depth $O(n^{\frac{3}{2}}\sqrt{\log n})$ for 3SUM.*

They also gave deterministic and randomized subquadratic real-RAM algorithms for 3SUM, refuting the conjecture. Similarly to the subquadratic 4-linear decision trees, these new results use the power of 4-linear queries. These algorithms were later improved by Freund [20] and Gold and Sharir [22].

▶ **Theorem 5** (Grønlund and Pettie [23]). *There is a deterministic $O(n^2(\log\log n)^{2/3}/(\log n)^{2/3})$-time and a randomized $O(n^2(\log\log n)^2/\log n)$-time real-RAM algorithm for 3SUM.*

Since then, the conjecture was eventually updated. This new conjecture is considered an essential part of the theory of complexity-within-P.

▶ **Conjecture 1** (3SUM Conjecture). *There is no $O(n^{2-\delta})$-time algorithm for 3SUM.*

Elekes-Rónyai, Elekes-Szabó. In a series of results spanning fifteen years, Elekes and Rónyai [16], Elekes and Szabó [17], Raz, Sharir and Solymosi [37], and Raz, Sharir and de Zeeuw [35] give upper bounds on the number of solution triples to the 3POL problem. The last and strongest result is the following

▶ **Theorem 6** (Raz, Sharir and de Zeeuw [35]). *Let A, B, C be n-sets of real numbers and $F \in \mathbb{R}[x,y,z]$ be a polynomial of constant degree, then the number of triples $(a,b,c) \in A \times B \times C$ such that $F(a,b,c) = 0$ is $O(n^{11/6})$ unless F has some group related form.*[2]

Raz, Sharir and de Zeeuw [35] also look at the number of solution triples for the General Position Testing problem when the input is restricted to points lying on a constant number of constant-degree algebraic curves.

▶ **Theorem 7** (Raz, Sharir and de Zeeuw [35]). *Let C_1, C_2, C_3 be three (not necessarily distinct) irreducible algebraic curves of degree at most d in $\mathbb{C}^2$, and let $S_1 \subset C_1, S_2 \subset C_2, S_3 \subset C_3$ be finite subsets. Then the number of proper collinear triples in $S_1 \times S_2 \times S_3$ is*

$$O_d(|S_1|^{1/2}|S_2|^{2/3}|S_3|^{2/3} + |S_1|^{1/2}(|S_1|^{1/2} + |S_2| + |S_3|)),$$

unless $C_1 \cup C_2 \cup C_3$ is a line or a cubic curve.

Recently, Nassajian Mojarrad, Pham, Valculescu and de Zeeuw [30] and Raz, Sharir and de Zeeuw [36] proved bounds for versions of the problem where F is a 4-variate polynomial.

2 Nonuniform algorithm for explicit 3POL

We begin with the description of a nonuniform algorithm for explicit 3POL which we use later as a basis for other algorithms. We prove the following:

▶ **Theorem 8.** *There is a bounded-degree ADT of depth $O(n^{\frac{12}{7}+\varepsilon})$ for explicit 3POL.*

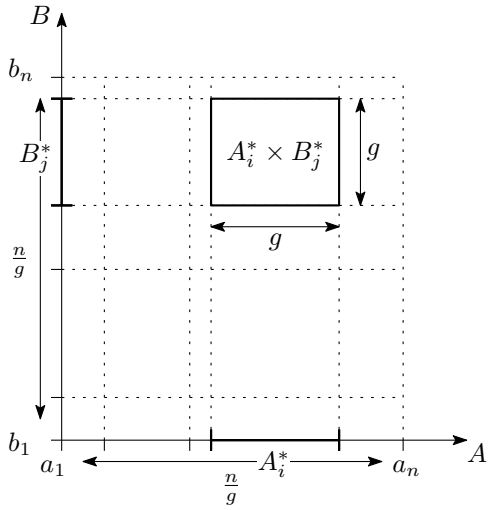
Idea. The idea is to partition the sets A and B into small groups of consecutive elements. That way, we can divide the $A \times B$ grid into cells with the guarantee that each curve $c = f(x,y)$ in this grid intersects a small number of cells. For each such curve and each cell it intersects, we search c among the values $f(a,b)$ for all (a,b) in a given intersected cell. We generalize Fredman's trick [19] – and how it is used in Grønlund and Pettie's paper [23] – to quickly obtain a sorted order on those values, which provides us a logarithmic search time for each cell. Below is a sketch of the algorithm.

Algorithm 1 (Nonuniform algorithm for explicit 3POL).

input $A = \{a_1 < \cdots < a_n\}, B = \{b_1 < \cdots < b_n\}, C = \{c_1 < \cdots < c_n\} \subset \mathbb{R}$.
output *accept* if $\exists\,(a,b,c) \in A \times B \times C$ such that $c = f(a,b)$, *reject* otherwise.
1. Partition the intervals $[a_1, a_n]$ and $[b_1, b_n]$ into blocks A_i^* and B_j^* such that $A_i = A \cap A_i^*$ and $B_j = B \cap B_j^*$ have size g.
2. Sort the sets $f(A_i \times B_j) = \{f(a,b) \colon (a,b) \in A_i \times B_j\}$ for all A_i, B_j. This is the only step that is nonuniform.
3. For each $c \in C$,
3.1. For each cell $A_i^* \times B_j^*$ intersected by the curve $c = f(x,y)$,
3.1.1. Binary search for c in the sorted set $f(A_i \times B_j)$. If c is found, *accept* and halt.
4. *reject* and halt.

Note that it is easy to modify the algorithm to count or report the solutions. In the latter case, the algorithm becomes output sensitive. Like in Grønlund and Pettie's $\tilde{O}(n^{\frac{3}{2}})$ decision tree for 3SUM [23], the tricky part is to give an efficient implementation of step **2**.

[2] Because our results do not depend on the meaning of *group related form*, we do not bother defining it here. We refer the reader to Raz, Sharir and de Zeeuw [35] for the exact definition.



(a) Partitioning A and B.

(b) An xy-monotone arc of $c = f(x, y)$ intersects a staircase of at most $2\frac{n}{g} - 1$ cells in the grid.

Figure 1 Properties of the $A \times B$ grid.

***A* × *B* grid partitioning.** Let $A = \{ a_1 < a_2 < \cdots < a_n \}$ and $B = \{ b_1 < b_2 < \cdots < b_n \}$. For some positive integer g to be determined later, partition the interval $[a_1, a_n]$ into n/g blocks $A_1^*, A_2^*, \ldots, A_{n/g}^*$ such that each block contains g numbers in A. Do the same for the interval $[b_1, b_n]$ with the numbers in B and name the blocks of this partition $B_1^*, B_2^*, \ldots, B_{n/g}^*$. For the sake of simplicity, and without loss of generality, we assume here that g divides n. We continue to make this assumption in the following sections. To each of the $(n/g)^2$ pairs of blocks A_i^* and B_j^* corresponds a cell $A_i^* \times B_j^*$. By definition, each cell contains g^2 pairs in $A \times B$. For the sake of notation, we define $A_i = A \cap A_i^* = \{ a_{i,1} < a_{i,2} < \cdots < a_{i,g} \}$ and $B_j = B \cap B_j^* = \{ b_{j,1} < b_{j,2} < \cdots < b_{j,g} \}$. Figure 1a depicts this construction.

The following two lemmas result from this construction:

▶ **Lemma 9.** *For a fixed value $c \in C$, the curve $c = f(x, y)$ intersects $O(\frac{n}{g})$ cells. Moreover, those cells can be found in $O(\frac{n}{g})$ time.*

Proof. Since f has constant degree, the curve $c = f(x, y)$ can be decomposed into a constant number of xy-monotone arcs. Split the curve into x-monotone pieces, then each x-monotone piece into y-monotone arcs. The endpoints of the xy-monotone arcs are the intersections of $f(x, y) = c$ with its derivatives $f'_x(x, y) = 0$ and $f'_y(x, y) = 0$. By Bézout's theorem, there are $O(\deg(f)^2)$ such intersections and so $O(\deg(f)^2)$ xy-monotone arcs. Figure 1b shows that each such arc intersects at most $2\frac{n}{g} - 1$ cells since the cells intersected by a xy-monotone arc form a staircase in the grid. This proves the first part of the lemma. To prove the second part, notice that for each connected component of $c = f(x, y)$ intersecting at least one cell of the grid either: (1) it intersects a boundary cell of the grid, or (2) it is a (singular) point or contains vertical and horizontal tangency points. The cells intersected by $c = f(x, y)$ are computed by exploring the grid from $O(\frac{n}{g})$ starting cells. Start with an empty set. Find and add all boundary cells containing a point of the curve. Finding those cells is achieved by solving the Tarski sentence $\exists (x, y) c = f(x, y) \wedge x \in A_i^* \wedge y \in B_j^*$, for each cell $A_i^* \times B_j^*$ on the boundary. This takes $O(\frac{n}{g})$ time. Find and add the cells containing endpoints of xy-monotone arcs of $c = f(x, y)$. Finding those cells is achieved by first finding the constant

number of vertical and horizontal slabs $A_i^* \times \mathbb{R}$ and $\mathbb{R} \times B_j^*$ containing such points:

$$\exists(x,y) c = f(x,y) \wedge (f'_x(x,y) = 0 \vee f'_y(x,y) = 0) \wedge x \in A_i^*,$$
$$\exists(x,y) c = f(x,y) \wedge (f'_x(x,y) = 0 \vee f'_y(x,y) = 0) \wedge y \in B_j^*.$$

This takes $O(\frac{n}{g})$ time. Then for each pair of vertical and horizontal slab containing such a point, check that the cell at the intersection of the slab also contains such a point:

$$\exists(x,y) c = f(x,y) \wedge (f'_x(x,y) = 0 \vee f'_y(x,y) = 0) \wedge x \in A_i^* \wedge y \in B_j^*.$$

This takes $O(1)$ time. Note that we can always assume the constant-degree polynomials we manipulate are square-free, as making them square-free is trivial [42]: since $\mathbb{R}[x]$ and $\mathbb{R}[y]$ are unique factorization domains, let $Q = P/\gcd(P, P'_x; x)$ and $\mathrm{sf}(P) = Q/\gcd(P, P'_y; y)$, where $\gcd(P, Q; z)$ is the greatest common divisor of P and Q when viewed as polynomials in $R[z]$ where R is a unique factorization domain and $\mathrm{sf}(P)$ is the square-free part of P. The set now contains, for each component of each type, at least one cell intersected by it. Initialize a list with the elements of the set. While the list is not empty, remove any cell from the list, add each of the eight neighbouring cells to the set and the list, if it contains a point of $c = f(x,y)$ – this can be checked with the same sentences as in the boundary case – and if it is not already in the set. This costs $O(1)$ per cell intersected. The set now contains all cells of the grid intersected by $c = f(x,y)$. ◀

▶ **Lemma 10.** *If the sets A, B, C can be preprocessed in $S_g(n)$ time so that, for any given cell $A_i^* \times B_j^*$ and any given $c \in C$, testing whether $c \in f(A_i \times B_j) = \{ f(a,b) \colon (a,b) \in A_i \times B_j \}$ can be done in $O(\log g)$ time, then, explicit 3POL can be solved in $S_g(n) + O(\frac{n^2}{g} \log g)$ time.*

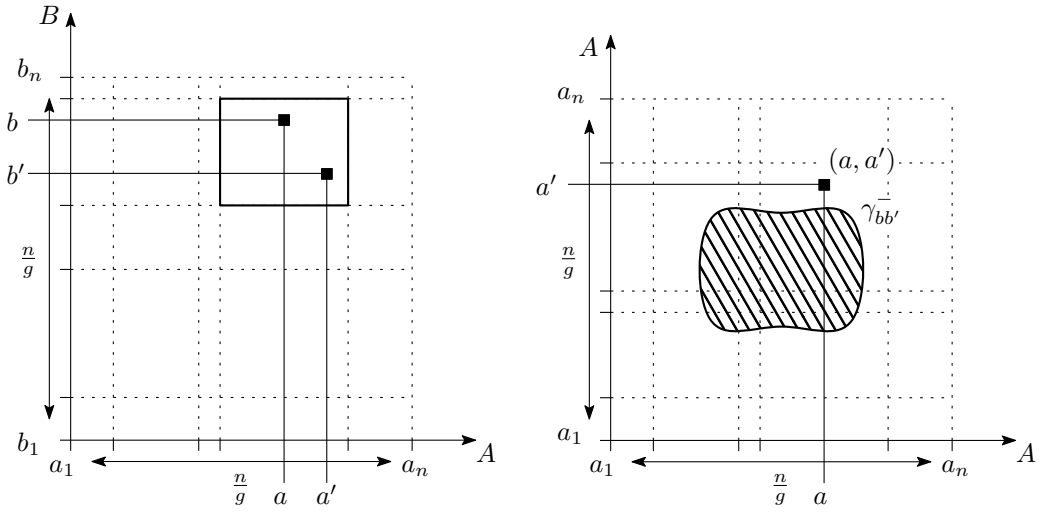
Proof. We need $S_g(n)$ preprocessing time plus the time required to search each of the n numbers $c \in C$ in each of the $O(\frac{n}{g})$ cells intersected by $c = f(x,y)$. Each search costs $O(\log g)$ time. We can compute the cells intersected by $c = f(x,y)$ in $O(\frac{n}{g})$ time by Lemma 9. ◀

▶ Remark. We do not give a $S_g(n)$-time real-RAM algorithm for preprocessing the input, but only a $S_g(n)$-depth bounded-degree ADT. In fact, this preprocessing step is the only nonuniform part of Algorithm 1. A real-RAM implementation of this step is given in §3.

Preprocessing. All that is left to prove is that $S_g(n)$ is subquadratic for some choice of g. To achieve this we sort the points inside each cell using Fredman's trick [19]. Grønlund and Pettie [23] use this trick to sort the sets $A_i + B_j = \{ a + b \colon (a,b) \in A_i \times B_j \}$ with few comparisons: sort the set $D = (\cup_i [A_i - A_i]) \cup (\cup_j [B_j - B_j])$, where $A_i - A_i = \{ a - a' \colon (a,a') \in A_i \times A_i \}$ and $B_j - B_j = \{ b - b' \colon (b,b') \in B_j \times B_j \}$, using $O(n \log n + |D|)$ comparisons, then testing whether $a + b \le a' + b'$ can be done using the free (already computed) comparison $a - a' \le b' - b$. We use a generalization of this trick to sort the sets $f(A_i \times B_j)$. For each B_j, for each pair $(b,b') \in B_j \times B_j$, define the curve $\gamma_{b,b'} = \{ (x,y) \colon f(x,b) = f(y,b') \}$. Define the sets $\gamma^0_{b,b'} = \gamma_{b,b'}, \gamma^-_{b,b'} = \{ (x,y) \colon f(x,b) < f(y,b') \}, \gamma^+_{b,b'} = \{ (x,y) \colon f(x,b) > f(y,b') \}$. The following lemma – illustrated by Figure 2 – follows by definition:

▶ **Lemma 11.** *Given a cell $A_i^* \times B_j^*$ and two pairs $(a,b), (a',b') \in A_i \times B_j$, deciding whether $f(a,b) < f(a',b')$ (respectively $f(a,b) = f(a',b')$ and $f(a,b) > f(a',b')$) amounts to deciding whether the point (a,a') is contained in $\gamma^-_{b,b'}$ (respectively $\gamma^0_{b,b'}$ and $\gamma^+_{b,b'}$).*

There are $N \coloneqq \frac{n}{g} \cdot g^2 = ng$ pairs $(a,a') \in \cup_i [A_i \times A_i]$ and there are N pairs $(b,b') \in \cup_j [B_j \times B_j]$. Sorting the $f(A_i \times B_j)$ for all (A_i, B_j) amounts to solving the following problem:



(a) The pairs $(a, b), (a', b')$.

(b) $(a, a') \in \gamma^+_{b,b'}$ implies $f(a, b) > f(a', b')$.

Figure 2 Generalization of Fredman's trick (Lemma 11).

▶ **Problem** (Polynomial Batch Range Searching). *Given N points and N polynomial curves in $\mathbb{R}^2$, locate each point with respect to each curve.*

We can now refine the description of step **2** in Algorithm 1

Algorithm 2 (Sorting the $f(A_i \times B_j)$ with a nonuniform algorithm).

input $A = \{ a_1 < a_2 < \cdots < a_n \}, B = \{ b_1 < b_2 < \cdots < b_n \} \subset \mathbb{R}$
output The sets $f(A_i \times B_j)$, sorted.
2.1. Locate each point $(a, a') \in \cup_i [A_i \times A_i]$ w.r.t. each curve $\gamma_{b,b'}, (b, b') \in \cup_j [B_j \times B_j]$.
2.2. Sort the sets $f(A_i \times B_j)$ using the information retrieved in step **2.1**.

Note that this algorithm is nonuniform: step **2.2** costs at least quadratic time in the real-RAM model, however, this step does not need to query the input at all, as all the information needed to sort is retrieved during step **2.1**. Step **2.2** incurs no cost in our nonuniform model.

To implement step **2.1**, we use a modified version of the $N^{\frac{4}{3}} 2^{O(\log^* N)}$ algorithm of Matoušek [28] for Hopcroft's problem. We prove the following upper bound in the arXiv e-print:

▶ **Lemma 12.** *Polynomial Batch Range Searching can be solved in $O(N^{\frac{4}{3}+\varepsilon})$ time in the real-RAM model when the input curves are the $\gamma_{b,b'}$.*

Analysis. Combining Lemma 10 and Lemma 12 yields a $O((ng)^{4/3+\varepsilon} + n^2 g^{-1} \log g)$-depth bounded-degree ADT for 3POL. By optimizing over g, we get $g = \Theta(n^{2/7-\varepsilon})$, and the previous expression simplifies to $O(n^{12/7+\varepsilon})$, proving Theorem 8.

3 Uniform algorithm for explicit 3POL

We now build on the first algorithm and prove the following:

▶ **Theorem 13.** *Explicit 3POL can be solved in $O(n^2 (\log \log n)^{\frac{3}{2}} / (\log n)^{\frac{1}{2}})$ time.*

We generalize again Grønlund and Pettie [23]. The algorithm we present is derived from the first subquadratic algorithm in their paper.

Idea. We want the implementation of step **2** in Algorithm 1 to be uniform, because then, the whole algorithm is. We use the same partitioning scheme as before except we choose g to be much smaller. This allows to store all permutations on g^2 items in a lookup table, where g is chosen small enough to make the size of the lookup table $\Theta(n^\varepsilon)$. The preprocessing part of the previous algorithm is replaced by $g^2!$ calls to an algorithm that determines for which cells a given permutation gives the correct sorted order. This preprocessing step stores a constant-size[3] pointer from each cell to the corresponding permutation in the lookup table. Search can now be done efficiently: when searching a value c in $f(A_i \times B_j)$, retrieve the corresponding permutation on g^2 items from the lookup table, then perform binary search on the sorted order defined by that permutation. The sketch of the algorithm is exactly Algorithm 1. The only differences with respect to §2 are the choice of g and the implementation of step **2**.

***A* × *B* grid partitioning.** We use the same partitioning scheme as before, hence Lemma 9 and Lemma 10 hold. We just need to find a replacement for Lemma 12.

Preprocessing. For their simple subquadratic 3SUM algorithm, Grønlund and Pettie [23] explain that for a permutation to give the correct sorted order for a cell, that permutation must define a *certificate* – a set of inequalities – that the cell must verify. They cleverly note – using Fredman's Trick [19] as in Chan [13] and Bremner et al. [10] – that the verification of a single certificate by all cells amounts to solving a red/blue point dominance reporting problem. We generalize their method. For each permutation $\pi\colon [g^2] \to [g]^2$, where $\pi = (\pi_r, \pi_c)$ is decomposed into row and column functions $\pi_r, \pi_c\colon [g^2] \to [g]$, we enumerate all cells $A_i^* \times B_j^*$ for which the following *certificate* holds:

$$f(a_{i,\pi_r(1)}, b_{j,\pi_c(1)}) \leq f(a_{i,\pi_r(2)}, b_{j,\pi_c(2)}) \leq \cdots \leq f(a_{i,\pi_r(g^2)}, b_{j,\pi_c(g^2)}).$$

▶ Remark. Since some entries may be equal, to make sure each cell corresponds to exactly one certificate, we replace $\leq$ symbols by choices of $g^2 - 1$ symbols in $\{=, <\}$. Each permutation π gets a certificate for each of those choices. This adds a 2^{g^2-1} factor to the number of certificates to test, which will eventually be negligible. Note that some of those 2^{g^2-1} certificates are equivalent. We need to skip some of them, as otherwise we might output some cells more than once, and then there will be no guarantee with respect to the output size. For example, the certificate $f(a_{i,9}, b_{j,5}) = f(a_{i,6}, b_{j,7}) < \cdots < f(a_{i,4}, b_{j,4})$ is equivalent to the certificate $f(a_{i,6}, b_{j,7}) = f(a_{i,9}, b_{j,5}) < \cdots < f(a_{i,4}, b_{j,4})$. Among equivalent certificates, we only consider the certificate whose permutation π precedes the others lexicographically. In the previous example, $((6,7),(9,5),\ldots,(4,4)) \prec ((9,5),(6,7),\ldots,(4,4))$ hence we would only process the second certificate. For the sake of simplicity, we will write inequality when we mean strict inequality or equation, and "$\leq$" when we mean "$<$" or "$=$".

Fredman's Trick. This is where Fredman's Trick comes into play. By Lemma 11, each inequality $f(a_{i,\pi_r(t)}, b_{j,\pi_c(t)}) \leq f(a_{i,\pi_r(t+1)}, b_{j,\pi_c(t+1)})$ of a certificate can be checked by computing the relative position of $(a_{i,\pi_r(t)}, a_{i,\pi_r(t+1)})$ with respect to $\gamma_{b_{j,\pi_c(t)}, b_{j,\pi_c(t+1)}}$. For a given certificate, for each A_i and each B_j, define

$$p_i = ((a_{i,\pi_r(1)}, a_{i,\pi_r(2)}), (a_{i,\pi_r(2)}, a_{i,\pi_r(3)}), \ldots, (a_{i,\pi_r(g^2-1)}, a_{i,\pi_r(g^2)})),$$
$$q_j = \big(f(x, b_{j,\pi_c(1)}) \leq f(y, b_{j,\pi_c(2)}), \ldots, f(x, b_{j,\pi_c(g^2-1)}) \leq f(y, b_{j,\pi_c(g^2)})\big).$$

[3] In the real-RAM and word-RAM models.

A certificate is verified by a cell $A_i \times B_j$ if and only if, for all $t \in [g^2-1]$, the point $p_{i,t}$ verifies the inequality $q_{j,t}$. Enumerating all cells $A_i \times B_j$ for which the certificate holds therefore amounts to solving the following problem:

▶ **Problem** (Polynomial Dominance Reporting (PDR)). *Given N k-tuples p_i of points in $\mathbb{R}^2$ and N k-tuples q_j of bivariate polynomial inequalities of degree at most $\deg(f)$, enumerate all pairs (p_i, q_j) where, for all $t \in [k]$, the point $p_{i,t}$ verifies the inequality $q_{j,t}$.*

In the next section, we explain how to solve PDR efficiently and prove the following lemma:

▶ **Lemma 14.** *We can enumerate all ℓ such pairs in time $2^{O(k)} N^{2-\frac{4}{\deg(f)^2+3\deg(f)+2}+\varepsilon} + O(\ell)$.*

We can now give a uniform implementation of step **2** in Algorithm 1:

Algorithm 3 (Sorting the $f(A_i \times B_j)$ with a uniform algorithm).

input $A = \{ a_1 < a_2 < \cdots < a_n \}, B = \{ b_1 < b_2 < \cdots < b_n \} \subset \mathbb{R}$
output The sets $f(A_i \times B_j)$, sorted.
2.1. Initialize a lookup table that will contain all $O(2^{g^2-1}(g^2!))$ certificates on g^2 elements.
2.2. For each permutation $\pi\colon [g^2] \to [g]^2$,
2.2.1. For each choice of g^2-1 symbols in $\{=, <\}$,
2.2.1.1. If there is any "=" symbol that corresponds to a lexicographically decreasing pair of tuples of indices in π, skip this choice of symbols.
2.2.1.2. Append the certificate associated to Π and the choice of symbols to the table.
2.2.1.3. Solve the PDR instance associated to A, B, Π and the choice of symbols.
2.2.1.4. For each output pair (i, j), store a pointer from (i, j) to the last entry in the table.

Analysis. Plugging in $k = g^2 - 1$, $N = \frac{n}{g}$, iterating over all permutations ($\sum_\pi \ell = (n/g)^2$), and adding the binary search step we get that explicit 3POL can be solved in time

$$(g^2!)2^{g^2}2^{O(g^2)}(n/g)^{2-\frac{4}{\deg(f)^2+3\deg(f)+2}+\varepsilon} + O((n/g)^2) + O(n^2 \log g/g).$$

The first two terms correspond to the complexity of step **2** in Algorithm 1, and the last term corresponds to the complexity of step **3** in Algorithm 1. To get subquadratic time we can set $g = c_{\deg(f)}\sqrt{\log n / \log\log n}$, because then for some appropriate choice of the constant factor $c_{\deg(f)}$, $(g^2)!2^{g^2}2^{O(g^2)} = n^\delta$ where $\delta < 4/(\deg(f)^2 + 3\deg(f) + 2) - \varepsilon$, making the first term negligible. The complexity of the algorithm is dominated by $O(n^2 \log g/g) = O(n^2(\log\log n)^{\frac{3}{2}}/(\log n)^{\frac{1}{2}})$. This proves Theorem 13.

4 Polynomial Dominance Reporting

In this section, we combine a standard dominance reporting algorithm [33] with Matoušek's algorithm [28] to prove Lemma 14. We say a pair of blue and red points in $\mathbb{R}^k$ is dominating if for all indices $i \in [k]$ the ith coordinate of the blue point is greater or equal to the ith coordinate of the red point. The standard algorithm [33] solves the following problem:

▶ **Problem.** *Given N blue and M red points in $\mathbb{R}^k$, report all bichromatic dominating pairs.*

Our problem is significantly more complicated and general. Instead of blue points we have blue k-tuples p_i of 2-dimensional points, instead of red points we have red k-tuples q_j of bivariate polynomial inequalities, and we want to report all bichromatic pairs (p_i, q_j) such that, for all $t \in [k]$, the point $p_{i,t}$ verifies the inequality $q_{j,t}$. The standard algorithm essentially works by

a combination of divide and conquer and prune and search, using a one-dimensional cutting (median selection) to split a problem into subproblems. We generalize the standard algorithm by using higher dimensional cuttings, in a way similar to Matoušek's algorithm [28]. For the analysis, we generalize Chan's analysis of the standard algorithm when k is not constant [13].

Proof of Lemma 14. We use the Veronese embedding [25, 24]. Since the polynomials have constant degree, we can trade polynomial inequalities for linear inequalities by lifting everything to a space of higher – but constant – dimension. The degree of each polynomial is at most $\deg(f)$. There are exactly $d = \binom{\deg(f)+2}{2} - 1$ different bivariate monomials of degree at most $\deg(f)$[4]. To each monomial we associate a variable in $\mathbb{R}^d$. By this association, points in the plane are mapped to points in $\mathbb{R}^d$ and bivariate polynomial inequalities are mapped to d-variate linear inequalities.

By abuse of notation, let p_i denote the tuple p_i where each 2-dimensional point has been replaced by its d-dimensional counterpart, and let q_i denote the tuple q_i where each bivariate polynomial inequality has been replaced by its d-variate linear counterpart. We have N k-tuples p_i and M k-tuples q_j. The algorithm checks each of the k components of the tuples in turn and can be described recursively as follows for some positive integer $r > 1$:

Algorithm 4 (Polynomial Dominance Reporting)**.**

input N k-tuples p_i of d-dimensional points, M k-tuples q_j of d-variate linear inequalities.
output All (p_i, q_j) pairs such that, for all $t \in [k]$, the point $p_{i,t}$ verifies the inequality $q_{j,t}$.

1. If $k = 0$, then output all pairs (p_i, q_j) and halt.
2. If $N < r^d$ or $M < r$, solve the problem by brute force in $O((N + M)k)$ time.
3. We now only consider the kth component of each input k-tuple and call these *active* components. To each active d-variate linear inequality corresponds a defining hyperplane in $\mathbb{R}^d$. Construct, as in [28], a hierarchical cutting of $\mathbb{R}^d$ using $O(r^d)$ simplicial cells such that each simplicial cell is intersected by at most $\frac{M}{r}$ of the defining hyperplanes. This construction also gives us for each simplical cell of the cutting the list of defining hyperplanes intersecting it. This takes $O(Mr^{d-1})$ time. Locate each active point inside the hierarchical cutting in time $O(N \log r)$. Let S be a simplicial cell of the hierarchical cutting. Denote by Π_S the set of active points in S. Partition each Π_S into $\left\lceil \frac{|\Pi_S|}{Nr^{-2}} \right\rceil$ disjoint subsets of size at most $\frac{N}{r^d}$. For each simplicial cell, find the active inequalities whose corresponding geometric object (hyperplane, closed or open half-space) contains the cell. This takes $O(Mr^d)$ time. The whole step takes $O(N \log r + Mr^d)$ time.
4. For each of the $O(r^d)$ simplicial cells, recurse on the at most $\frac{N}{r^d}$ k-tuples p_i whose active point is inside the simplicial cell and the at most $\frac{M}{r}$ k-tuples q_j whose active inequality's defining hyperplane intersects the simplicial cell.
5. For each of the $O(r^d)$ simplicial cells, recurse on the at most $\frac{N}{r^d}$ $(k-1)$-prefixes of k-tuples p_i whose active point is inside the simplicial cell and the $(k-1)$-prefixes of k-tuples q_j whose active inequality's corresponding geometric object contains the simplicial cell.

Correctness. In each recursive call, either k is decremented or M and N are divided by some constant, hence, one of the conditions in steps **1** and **2** is met in each of the paths of the recursion tree and the algorithm always terminates. Step **5** is correct because it only recurses on (p_i, q_j) pairs whose suffix pairs are dominating. The base case in step **1** is correct

[4] Not including the independent monomial, namely, 1.

because the only way for a pair (p_i, q_j) to reach this point is to have had all k components checked in step **5**. The base case in step **2** is correct by definition. Each dominating pair is output exactly once because the recursive calls of step **4** and **5** partition the set of pairs (p_i, q_j) that can still claim to be candidate dominating pairs.

Analysis. For $k, N, M \geq 0$, the total complexity $T_k(N, M)$ of computing the inclusions for the first k components, excluding the output cost (steps **1** and **2**), is bounded by

$$T_k(N,M) \leq \underbrace{O(r^d)\, T_{k-1}(N,M)}_{\text{Step } \mathbf{5}} + \underbrace{O(r^d)\, T_k\left(\frac{N}{r^d}, \frac{M}{r}\right)}_{\text{Step } \mathbf{4}} + \underbrace{O(N+M)}_{\text{Step } \mathbf{3}},$$

$$T_0(N,M) = 0, \quad T_k(N,M) = O(Nk) \text{ if } M < r, \quad T_k(N,M) = O(Mk) \text{ if } N < r^d.$$

By point-hyperplane duality, $T_k(N, M) = T_k(M, N)$, hence, we can execute step **4** on dual linear inequalities and dual points to balance the recurrence. For some constant $c_1 \geq 1$,

$$T_k(N,M) \leq c_1 r^{2d} T_{k-1}(N,M) + c_1 r^{2d} T_k\left(\frac{N}{r^{d+1}}, \frac{M}{r^{d+1}}\right) + c_1(N+M).$$

For simplicity, we ignore some problem-size reductions occuring in this balancing step.

Let $T_k(N) = T_k(N, N)$ denote the complexity of solving the problem when $M = N$, excluding the output cost. Hence,

$$T_k(N) \leq c_1 r^{2d} T_{k-1}(N) + c_1 r^{2d} T_k\left(\frac{N}{r^{d+1}}\right) + c_1 N, \tag{1}$$

$$T_0(N) = 0, \quad T_k(N) = O(k) \text{ if } N < r^{d+1}.$$

Solving the recurrence gives $T_k(N) = 2^{O(k)} N^{\frac{2d}{d+1}+\varepsilon_r}$, and since $d = \binom{\deg(f)+2}{2} - 1$, we have

$$T_k(N) = 2^{O(k)} N^{2-\frac{4}{\deg(f)^2+3\deg(f)+2}+\varepsilon_r}.$$

To that complexity we add a constant time unit for each output pair in steps **1** and **2**. ◀

5 3POL

Extending the previous techniques to work for the (implicit) 3POL problem is nontrivial:

1. Instead of sorting the sets $f(A_i \times B_j)$ we need to sort the real roots of the $F(A_i \times B_j, z)$,
2. The $\gamma_{b,b'}$ curves must be redefined. The redefined curve $\gamma_{b,b'}$ is still the zero-set of some constant-degree bivariate polynomial $P(x, y)$. However, retrieving the information we need for sorting becomes more challenging than just computing the sign of the $P(A_i \times A_i)$,
3. The implementation of the certificates for the uniform algorithm gets much more convoluted: each certificate checks the validity of a conjunction of Tarski sentences.

Those extensions are explained in detail in the arXiv e-print. There we show

▶ **Theorem 15.** *There is a bounded-degree ADT of depth $O(n^{\frac{12}{7}+\varepsilon})$ for 3POL.*

▶ **Theorem 16.** *3POL can be solved in $O(n^2(\log\log n)^{\frac{3}{2}}/(\log n)^{\frac{1}{2}})$ time.*

6 Applications

To illustrate the expressive power of 3POL, we give some geometric applications in the arXiv e-print. We show the following:

1. GPT can be solved in subquadratic time provided the input points lie on few parameterized constant-degree polynomial curves.
2. In the plane, given three sets C_i of n unit circles and three points p_i such that a circle $c \in C_i$ contains p_i, deciding whether there exists $(a, b, c) \in C_1 \times C_2 \times C_3$ such that $a \cap b \cap c \neq \emptyset$ can be done in subquadratic time.
3. Given n input points in the plane, deciding whether any triple spawns a unit triangle can be done in subquadratic time, provided the input points lie on few parameterized constant-degree polynomial curves.

References

1 Amir Abboud and Virginia Vassilevska Williams. Popular conjectures imply strong lower bounds for dynamic problems. In *FOCS*, pages 434–443. IEEE Computer Society, 2014.

2 Amir Abboud, Virginia Vassilevska Williams, and Oren Weimann. Consequences of faster alignment of sequences. In *ICALP (1)*, volume 8572 of *LNCS*, pages 39–51, 2014.

3 Amir Abboud, Virginia Vassilevska Williams, and Huacheng Yu. Matching triangles and basing hardness on an extremely popular conjecture. In *STOC*, pages 41–50. ACM, 2015.

4 Nir Ailon and Bernard Chazelle. Lower bounds for linear degeneracy testing. *J. ACM*, 52(2):157–171, 2005.

5 Amihood Amir, Timothy M. Chan, Moshe Lewenstein, and Noa Lewenstein. On hardness of jumbled indexing. In *ICALP (1)*, volume 8572 of *LNCS*, pages 114–125, 2014.

6 Ilya Baran, Erik D. Demaine, and Mihai Pătraşcu. Subquadratic algorithms for 3SUM. *Algorithmica*, 50(4):584–596, 2008.

7 Gill Barequet and Sariel Har-Peled. Polygon containment and translational min Hausdorff distance between segment sets are 3SUM-hard. *Int. J. Comput. Geometry Appl.*, 11(4):465–474, 2001.

8 Saugata Basu, Richard Pollack, and Marie-Françoise Roy. Computing roadmaps of semi-algebraic sets (extended abstract). In *STOC*, pages 168–173. ACM, 1996.

9 Saugata Basu, Richard Pollack, and Marie-Françoise Roy. *Algorithms in real algebraic geometry*, volume 10 of *Algorithms and Computation in Mathematics*. Springer, 2006.

10 David Bremner, Timothy M. Chan, Erik D. Demaine, Jeff Erickson, Ferran Hurtado, John Iacono, Stefan Langerman, Mihai Patrascu, and Perouz Taslakian. Necklaces, Convolutions, and X+Y. *Algorithmica*, 69(2):294–314, 2014.

11 Marco L. Carmosino, Jiawei Gao, Russell Impagliazzo, Ivan Mihajlin, Ramamohan Paturi, and Stefan Schneider. Nondeterministic extensions of the strong exponential time hypothesis and consequences for non-reducibility. In *ITCS*, pages 261–270. ACM, 2016.

12 Bob F. Caviness and Jeremy R. Johnson. *Quantifier elimination and cylindrical algebraic decomposition*. Springer, 2012.

13 Timothy M. Chan. All-pairs shortest paths with real weights in $O(n^3/\log n)$ time. *Algorithmica*, 50(2):236–243, 2008.

14 George E. Collins. Hauptvortrag: Quantifier elimination for real closed fields by cylindrical algebraic decomposition. In *Automata Theory and Formal Languages*, volume 33 of *LNCS*, pages 134–183. Springer, 1975.

15 James H. Davenport and Joos Heintz. Real quantifier elimination is doubly exponential. *J. Symb. Comput.*, 5(1/2):29–35, 1988.

16 György Elekes and Lajos Rónyai. A combinatorial problem on polynomials and rational functions. *J. Comb. Theory, Ser. A*, 89(1):1–20, 2000.

17 György Elekes and Endre Szabó. How to find groups? (and how to use them in Erdős geometry?). *Combinatorica*, 32(5):537–571, 2012.
18 Jeff Erickson. Lower bounds for linear satisfiability problems. *Chicago J. Theor. Comput. Sci.*, 1999.
19 Michael L. Fredman. How good is the information theory bound in sorting? *Theor. Comput. Sci.*, 1(4):355–361, 1976.
20 Ari Freund. Improved subquadratic 3SUM. *Algorithmica*, pages 1–19, 2015.
21 Anka Gajentaan and Mark H. Overmars. On a class of $O(n^2)$ problems in computational geometry. *Comput. Geom.*, 5:165–185, 1995.
22 Omer Gold and Micha Sharir. Improved bounds for 3SUM, k-SUM, and linear degeneracy. *ArXiv e-prints*, 2015. arXiv:1512.05279 [cs.DS].
23 Allan Grønlund and Seth Pettie. Threesomes, degenerates, and love triangles. In *Foundations of Computer Science (FOCS 2014)*, pages 621–630. IEEE, 2014.
24 Joe Harris. *Algebraic geometry: a first course*, volume 133. Springer, 2013.
25 Robin Hartshorne. *Algebraic geometry*, volume 52. Springer, 1977.
26 Monika Henzinger, Sebastian Krinninger, Danupon Nanongkai, and Thatchaphol Saranurak. Unifying and strengthening hardness for dynamic problems via the online matrix-vector multiplication conjecture. In *STOC*, pages 21–30. ACM, 2015.
27 Tsvi Kopelowitz, Seth Pettie, and Ely Porat. Higher lower bounds from the 3SUM conjecture. In *SODA*, pages 1272–1287. SIAM, 2016.
28 Jirí Matoušek. Range searching with efficient hierarchical cutting. *Discrete & Computational Geometry*, 10:157–182, 1993.
29 Bhubaneswar Mishra. Computational real algebraic geometry. In *Handbook of Discrete and Computational Geometry, 2nd Ed.*, pages 743–764. Chapman and Hall/CRC, 2004.
30 H. Nassajian Mojarrad, T. Pham, C. Valculescu, and F. de Zeeuw. Schwartz-Zippel bounds for two-dimensional products. *ArXiv e-prints*, 2016. arXiv:1507.08181 [math.CO].
31 János Pach and Micha Sharir. On the number of incidences between points and curves. *Combinatorics, Probability & Computing*, 7(1):121–127, 1998.
32 Mihai Pătraşcu. Towards polynomial lower bounds for dynamic problems. In *STOC*, pages 603–610. ACM, 2010.
33 Franco P. Preparata and Michael Ian Shamos. *Computational Geometry – An Introduction*. Texts and Monographs in Computer Science. Springer, 1985.
34 Michael O. Rabin. Proving simultaneous positivity of linear forms. *J. Comput. Syst. Sci.*, 6(6):639–650, 1972.
35 Orit E. Raz, Micha Sharir, and Frank de Zeeuw. Polynomials vanishing on cartesian products: The Elekes-Szabó theorem revisited. In *SoCG*, volume 34 of *LIPIcs*, pages 522–536, 2015.
36 Orit E. Raz, Micha Sharir, and Frank de Zeeuw. The elekes-szabó theorem in four dimensions. *ArXiv e-prints*, 2016. arXiv:1607.03600 [math.CO].
37 Orit E. Raz, Micha Sharir, and József Solymosi. Polynomials vanishing on grids: The Elekes-Rónyai problem revisited. In *SoCG*, page 251. ACM, 2014.
38 Abraham Seidenberg. Constructions in algebra. *Transactions of the AMS*, 197:273–313, 1974.
39 J. Michael Steele and Andrew Yao. Lower bounds for algebraic decision trees. *J. Algorithms*, 3(1):1–8, 1982.
40 Alfred Tarski. A decision method for elementary algebra and geometry, 1951. Rand Corporation.
41 Andrew Yao. A lower bound to finding convex hulls. *J. ACM*, 28(4):780–787, 1981.
42 David Yun. On square-free decomposition algorithms. In *SYMSACC*, pages 26–35, 1976.

Towards a Topology-Shape-Metrics Framework for Ortho-Radial Drawings*

Lukas Barth†[1], Benjamin Niedermann[2], Ignaz Rutter[3], and Matthias Wolf[4]

1 **Karlsruhe Institute of Technology, Karlsruhe, Germany**
`lukas.barth@kit.edu`
2 **University of Bonn, Bonn, Germany**
`niedermann@uni-bonn.de`
3 **TU Eindhoven, Eindhoven, The Netherlands**
`i.rutter@tue.nl`
4 **Karlsruhe Institute of Technology, Karlsruhe, Germany**
`matthias.wolf@kit.edu`

Abstract

Ortho-Radial drawings are a generalization of orthogonal drawings to grids that are formed by concentric circles and straight-line spokes emanating from the circles' center. Such drawings have applications in schematic graph layouts, e.g., for metro maps and destination maps.

A plane graph is a planar graph with a fixed planar embedding. We give a combinatorial characterization of the plane graphs that admit a planar ortho-radial drawing without bends. Previously, such a characterization was only known for paths, cycles, and theta graphs [12], and in the special case of rectangular drawings for cubic graphs [11], where the contour of each face is required to be a rectangle.

The characterization is expressed in terms of an *ortho-radial representation* that, similar to Tamassia's *orthogonal representations* for orthogonal drawings describes such a drawing combinatorially in terms of angles around vertices and bends on the edges. In this sense our characterization can be seen as a first step towards generalizing the Topology-Shape-Metrics framework of Tamassia to ortho-radial drawings.

1998 ACM Subject Classification G.2.2 Graph Theory

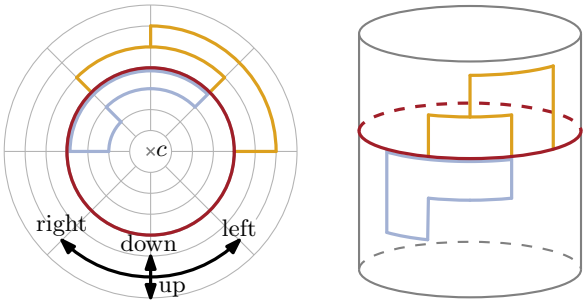
Keywords and phrases Graph Drawing, Ortho-Radial Drawings, Combinatorial Characterization, Bend Minimization, Topology-Shape-Metrics

Digital Object Identifier 10.4230/LIPIcs.SoCG.2017.14

1 Introduction

Grid drawings of graphs map vertices to grid points, and edges to internally disjoint curves on the grid lines connecting their endpoints. The appropriate choice of the underlying grid is decisive for the quality and properties of the drawing. *Orthogonal grids*, where the grid lines are horizontal and vertical lines, are popular and widely used in graph drawing. Their strength lies in their simple structure, their high angular resolution, and the limited number of directions. Graphs admitting orthogonal grid drawings must be *4-planar*, i.e., they must be planar and have maximum degree 4. On such grids a single edge consists of a sequence

* A full version of the paper is available at `https://arxiv.org/abs/1703.06040`.
† Lukas Barth's research was partially supported by DFG Research Training Group 2153



(a) Ortho-Radial Grid. **(b)** Cylinder.

Figure 1 An ortho-radial drawing of a graph on a grid **(a)** and its equivalent interpretation as an orthogonal drawing on a cylinder **(b)**.

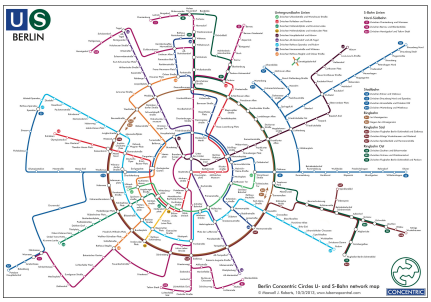


Figure 2 Metro map of Berlin using an ortho-radial layout. Image copyright by Maxwell J. Roberts.

of horizontal and vertical grid segments. A transition between a horizontal and a vertical segment on an edge is called a *bend*. A typical optimization goal is to minimize the number of bends in the drawing, either in total or by the maximum number of bends per edge.

The popularity and usefulness of orthogonal drawing have their foundations in the seminal work of Tamassia [15] who showed that for *plane graphs*, i.e., planar graphs with a fixed embedding, a bend-minimal planar orthogonal drawing can be computed efficiently. More generally, Tamassia [15] established the so-called Topology-Shape-Metrics framework, abbreviated as TSM in the following, for computing orthogonal drawings of 4-planar graphs.

The goal of this work is to provide a similar framework for *ortho-radial drawings*, which are based on ortho-radial grids rather than orthogonal grids. Such drawings have applications in schematic graph layouts, e.g., for metro maps and destination maps; see Fig. 2 for an example. An *ortho-radial grid* is formed by M concentric circles and N spokes, where $M, N \in \mathbb{N}$. More precisely, the radii of the concentric circles are integers, and the N spokes emanate from the center $c = (0, 0)$ of the circles such that they have uniform angular resolution; see Fig. 1. We note that c does not belong to the grid. Again vertices are positioned at grid points and edges are drawn as internally disjoint chains of (1) segments of spokes and (2) arcs of concentric circles. As before, a *bend* is a transition between a spoke segment and an arc segment. We observe that ortho-radial drawings can also be thought of as orthogonal drawings on a cylinder; see Fig. 1. Edge segments that are originally drawn on spokes are parallel to the axis of the cylinder, whereas segments that are originally drawn as arcs are drawn on circles orthogonal to the axis of the cylinder.

It is not hard to see that every orthogonal drawing can be transformed into an ortho-radial drawing with the same number of bends. The converse is, however, not true. It is readily seen that for example a triangle, which requires at least one bend in an orthogonal drawing, admits an ortho-radial drawing without bends, namely in the form of a circle centered at the origin. In fact, by suitably nesting triangles, one can construct graphs that have an ortho-radial drawing without bends but require a linear number of bends in any orthogonal drawing.

Related Work. The case of planar orthogonal drawings is well-researched and there is a plethora of results known; see [9] for a survey. Here, we mention only the most recent results and those which are most strongly related to the problem we consider. A k-embedding is an orthogonal planar drawing such that each edge has at most k bends. It is known that, with

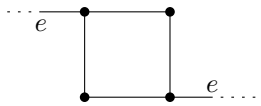


Figure 3 In this drawing, the angles around vertices sum up to 360°, and also the sum of angles for each face is as expected for an ortho-radial drawing. However, the graph does not have an ortho-radial drawing without bends.

the single exception of the octahedron graph, every planar graph has a 2-embedding [2] and that deciding the existence of a 0-embedding is $\mathcal{NP}$-complete [10]. The latter in particular implies that the problem of computing a planar orthogonal drawing with the minimum number of bends is $\mathcal{NP}$-hard. In contrast, the existence of a 1-embedding can be tested efficiently [3]; this has subsequently been generalized to a version that forces up to k edges to have no bends in FPT time w.r.t. k [4], and to an optimization version that optimizes the number of bends beyond the first one on each edge [5]. Moreover, for series-parallel graphs and graphs with maximum degree 3 an orthogonal planar drawing with the minimum number of bends can be computed efficiently [8].

The $\mathcal{NP}$-hardness of the bend minimization problem has also inspired the study of bend minimization for planar graphs with a fixed embedding. In his fundamental work Tamassia [15] showed that for plane graphs, i.e., planar graphs with a fixed planar embedding, bend-minimal planar orthogonal drawings can be computed in polynomial time. The running time has subsequently been improved to $O(n^{1.5})$ [6]. Key to these results is the existence of a combinatorial description of planar orthogonal drawings of a plane graph in terms of the angles surrounding each vertex and the order and directions of bends on the edges but neglecting any kind of geometric information such as coordinates and edge lengths. In particular, this allows to efficiently compute bend-minimal orthogonal drawings of plane graphs by mapping the purely combinatorial problem of determining an orthogonal representation with the minimum number of bends to a flow problem.

In addition, there is a number of works that seek to characterize the plane graphs that can be drawn without bends. In this case an orthogonal representation essentially only describes the angles around the vertices. Rahman, Nishizeki and Naznin [14] characterize the plane graphs with maximum degree 3 that admit such a drawing and Rahman, Nakano and Nishizeki [13] characterize the plane graphs that admit a rectangular drawing where in addition the contour of each face is a rectangle.

In the case of ortho-radial drawings much less is known. A natural generalization of the properties of an orthogonal representation to the ortho-radial case, where the angles around the vertices and inside the faces are constrained, has turned out not be sufficient for drawability; see Fig. 3. So far characterizations of bend-free ortho-radial drawing have only been achieved for paths, cycles, and theta graphs [12]. For the special case of rectangular ortho-radial drawings, i.e., every internal face is bounded by a rectangle, a characterization is known for cubic graphs [11].

Contribution and Outline. Since deciding whether a 4-planar graph can be orthogonally drawn in the plane without any bends is $\mathcal{NP}$-complete [10], it is not surprising that also ortho-radial bend minimization is $\mathcal{NP}$-hard; the proof is in the full version [1] of this paper.

▶ **Theorem 1.1.** *Deciding whether a 4-planar graph has a planar ortho-radial drawing without any bends is $\mathcal{NP}$-complete.*

As our main result we introduce a generalization of an orthogonal representation, which we call *ortho-radial representation*, that characterizes the *4-plane graphs*, i.e., 4-planar graphs with a fixed combinatorial embedding, having bend-free ortho-radial drawings; see Theorem 3.5.

This significantly generalizes the corresponding results for paths, cycles, theta graphs [12], and cubic graphs [11]. Further, this characterization can be seen as a step towards an extension of the TSM framework for computing ortho-radial drawings that may have bends. Namely, once the angles around vertices and the order and directions of bends along each edge of a graph G have been fixed, we can replace each bend by a vertex to obtain a graph G'. The directions of bends and the angles at the vertices of G define a unique ortho-radial representation of G', which is valid if and only if G admits a drawing with the specified angles and bends. Thus, ortho-radial drawings can indeed be described by angles around vertices and orders and directions of bends on edges. Our main result therefore implies that ortho-radial drawings can be computed by a TSM framework, i.e., by fixing a combinatorial embedding in a first "Topology" step, determining a description of the drawing in terms of angles and bends in a second "Shape" step, and computing edge lengths and vertex coordinates in a final "Metrics" step.

In the following, we disregard the "Topology" step and assume that our input consists of a 4-planar graph with a fixed *combinatorial embedding*, i.e., the order of the incident edges around each vertex is fixed and additionally, one outer face and one central face are specified; the latter shall contain the center of the drawing. We present our definition of an ortho-radial representation in Section 3. After introducing some basic tools based on this representation in Section 4, we first present in Section 5 a characterization for rectangular graphs, whose ortho-radial representation is such that internal faces have exactly four 90° angles, while all other incident angles are 180°; i.e., they have to be drawn as rectangles.

The algorithm we use as a proof of drawability corresponds to the "Metrics" step of an ortho-radial TSM framework. The characterization corresponds to the output of the "Shape" step. Based on the special case of rectangular 4-planar graphs, we then present the characterization and the "Metrics" step for general 4-planar graphs in Section 6. Due to space constraints, some proofs are omitted or only sketched; full proofs can be found in the full version of this paper [1].

2 Ortho-Radial Drawings

In this section we introduce basic definitions and conventions on ortho-radial drawings that we use throughout this paper. To that end, we first introduce some basic definitions.

We assume that paths cannot contain vertices multiple times but cycles can, i.e., all paths are simple. We always consider non-selfcrossing cycles as directed clockwise, so that their interior is locally to the right of the cycle. A cycle is part of its interior and its exterior.

For a path P and vertices u and v on P, we denote the subpath of P from u to v (including these vertices) by $P[u, v]$. We may also specify the first and last edge instead and write, e.g., $P[e, e']$ for edges e and e' on P. We denote the concatenation of two paths P and Q by $P + Q$. For a path $P = v_1 \dots v_k$, we define its reverse $\overline{P} = v_k \dots v_1$. For a cycle C that contains any edge at most once (e.g. if C is simple), we extend the notion of subpaths as follows. For two edges e and e' on C, the subpath $C[e, e']$ is the unique path on C that starts with e and ends with e'. If the start vertex v of e identifies e uniquely, i.e., C contains v exactly once, we may write $C[v, e']$ to describe the path on C from v to e'. Analogously, we may identify e' with its endpoint if this is unambiguous.

We are now ready to introduce concepts of ortho-radial drawings. Consider a 4-planar graph $G = (V, E)$ with fixed embedding. We refer to the directed edge from u to v by uv. Let Δ be an ortho-radial drawing of G, and recall that we do not allow bends on edges. In Δ a directed edge e is either drawn clockwise, counter-clockwise, towards the center or away from the center; see Fig. 1. Using the metaphor of drawing G on a cylinder, we say that e points *right*, *left*, *down* or *up*, respectively. Edges pointing left or right are *horizontal* edges and edges pointing up or down are *vertical* edges.

There are two fundamentally different ways of drawing a simple cycle C. The center of the grid may lie in the interior or the exterior of C. In the former case C is *essential* and in the latter case it is *non-essential*. In Fig. 1 the red cycle is essential and the blue cycle is non-essential.

We further observe that Δ has two special faces: One unbounded face, called the *outer face*, and the *central face* containing the center of the drawing. These two faces are equal if and only if Δ contains no essential cycles. All other faces of G are called *regular*. Ortho-radial drawings without essential cycles are equivalent to orthogonal drawings [12]. That is, any such ortho-radial drawing can be transformed to an orthogonal drawing of the same graph with the same outer face and vice versa.

We represent a face as a cycle f in which the interior of the face lies locally to the right of f. Note that f may not be simple since cut vertices may appear multiple times on f. But no directed edge is used twice by f. Therefore, the notation of subpaths of cycles applies to faces. Note furthermore that the cycle bounding the outer face of a graph is directed counter-clockwise, whereas all other faces are bounded by cycles directed clockwise.

A face f in Δ is a *rectangle* if and only if its boundary does not make any left turns. That is, if f is a regular face, there are exactly 4 right turns, and if f is the central or the outer face, there are no turns at all. Note that by this definition f cannot be a rectangle if it is both the outer and the central face.

3 Ortho-Radial Representations

In this section, we define ortho-radial representations. These are a tool to describe the ortho-radial drawing of a graph without fixing any edge lengths. As mentioned in the introduction, they are an analogon to orthogonal representations in the TSM framework.

We observe all directions of all edges being fixed once the direction of one edge and all angles around vertices are fixed. For two edges uv and vw that enclose the angle $\alpha \in \{90°, 180°, 270°, 360°\}$ at v (such that the angle measured lies locally to the right of uvw), we define the rotation $\mathrm{rot}(uvw) = 2 - \alpha/90°$. We note that the rotation is 1 if there is a right turn at v, 0 if uvw is straight, and -1 if a left turn occurs at v. If $u = w$, it is $\mathrm{rot}(uvw) = -2$.

We define the rotation of a path $P = v_1 \dots v_k$ as the sum of the rotations at its internal vertices, i.e., $\mathrm{rot}(P) = \sum_{i=2}^{k-1} \mathrm{rot}(v_{i-1}v_iv_{i+1})$. Similarly, for a cycle $C = v_1 \dots v_k v_1$, its rotation is the sum of the rotations at all its vertices, i.e., $\mathrm{rot}(C) = \sum_{i=1}^{k} \mathrm{rot}(v_{i-1}v_iv_{i+1})$, where we define $v_0 = v_k$ and $v_{k+1} = v_1$.

When splitting a path at an edge, the sum of the rotations of the two parts is equal to the rotation of the whole path.

▶ **Observation 3.1.** *Let P be a simple path with start vertex s and end vertex t. For all edges e on P it holds that* $\mathrm{rot}(P) = \mathrm{rot}(P[s, e]) + \mathrm{rot}(P[e, t])$.

Furthermore, reversing a path changes all left turns to right turns and vice versa. Hence, the sign of the rotation is flipped.

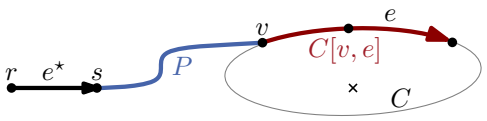


Figure 4 The labeling of e induced by P is $\ell_C^P(e) = \mathrm{rot}(e^\star + P + C[v,e])$.

▶ **Observation 3.2.** *For any path P it is* $\mathrm{rot}(\overline{P}) = -\mathrm{rot}(P)$.

The next observation analyzes *detours*; an illustration is found in [1].

▶ **Observation 3.3.** *Let P be a path from v to w and xy an edge not on P such that x is an internal vertex of P. It is* $\mathrm{rot}\,(P[v,x] + xy) + \mathrm{rot}\,(yx + P[x,w]) = \mathrm{rot}\,(P) + c$, *where $c = -2$ if xy lies locally to the right of P and $c = +2$ if xy lies locally to the left of P.*

An *ortho-radial representation* of a 4-planar graph G consists of a list $H(f)$ of pairs (e, a) for each face f, where e is an edge on f, and $a \in \{90°, 180°, 270°, 360°\}$. Further, the outer face and the central face are fixed and one *reference edge* $e^\star$ in the outer face is given, with $e^\star$ oriented such that the outer face is locally to its left. By convention the reference edge is always drawn such that it points right. We interpret the fields of a pair in $H(f)$ as follows. e denotes the edge on f directed such that f lies to the right of e. The field a represents the angle inside f from e to the following edge in degrees. Using this information we define the *rotation* of such a pair $t = (e, a)$ as $\mathrm{rot}(t) = (180° - a)/90°$.

Not every ortho-radial representation also yields an ortho-radial drawing of a graph. In order to characterize valid ortho-radial representations, we introduce *labelings* of essential cycles. These labelings prove to be a valuable tool to ensure that all the essential cycles of the graph can be drawn in such a way that they are compatible with each other.

Let G be an embedded 4-planar graph and let $e^\star = rs$ be the reference edge of G. Further, let C be a simple, essential cycle in G, and let P be a path from s to a vertex v on C. The *labeling* ℓ_C^P of C induced by P is defined for each edge e on C by $\ell_C^P(e) = \mathrm{rot}(e^\star + P + C[v,e])$; see Fig. 4 for an illustration. We are mostly interested in labelings that are induced by paths starting at s and intersecting C only at their endpoints. We call such paths *elementary paths*.

We now introduce properties characterizing bend-free ortho-radial drawings, as we prove in Theorem 3.5.

▶ **Definition 3.4.** An ortho-radial representation is *valid* if the following conditions hold:

1. The angle sum of all edges around each vertex given by the a-fields is 360.
2. For each face f, it is

$$\mathrm{rot}(f) = \begin{cases} 4, & f \text{ is a regular face} \\ 0, & f \text{ is the outer or the central face but not both} \\ -4, & f \text{ is both the outer and the central face} \end{cases}$$

3. For each simple, essential cycle C in G, there is a labeling ℓ_C^P of C induced by an elementary path P such that either $\ell_C^P(e) = 0$ for all edges e of C, or there are edges e_+ and e_- on C such that $\ell_C^P(e_+) > 0$ and $\ell_C^P(e_-) < 0$.

The first two conditions ensure that the angle-sums at vertices and inside the faces are correct. Since the labels of neighboring edges differ by at most 1, the last condition ensures that on each essential cycle with not only horizontal edges there are edges with labels 1 and -1, i.e., edges pointing up and down. This reflects the characterization for cycles [12].

Intuitively, basing all labels on the reference edge guarantees that all cycles in the graph can be drawn together consistently.

For an essential cycle C not satisfying the last condition there are two possibilities. Either all labels of edges on C are non-negative and at least one label is positive, or all are non-positive and at least one is negative. In the former case C is called *decreasing* and in the latter case it is *increasing*. We call both *monotone* cycles. Cycles with only the label 0 are not monotone.

We show that a graph with a given ortho-radial representation can be drawn if and only if the representation is valid, which yields our main result.

▶ **Theorem 3.5.** *A 4-plane graph admits a bend-free orthogonal drawing if and only if it admits a valid ortho-radial representation.*

While we defer the proof that the conditions for valid ortho-radial representations are sufficient for the existence of a drawing to Section 6.2, the necessity of the conditions is easier to see.

▶ **Theorem 3.6.** *For any ortho-radial drawing Δ of a 4-planar graph G there is a valid ortho-radial representation of G.*

Proof Sketch. We note that any drawing Δ fixes an ortho-radial representation up to the choice of the reference edge. Let Γ be such an ortho-radial representation where we pick an edge $e^\star$ on the outer face as the reference edge such that $e^\star$ points to the right and lies on the outermost circle that is used by Δ. By [12] the representation Γ satisfies Conditions 1 and 2 of Definition 3.4. To prove that Γ also satisfies Condition 3, i.e., Γ does not contain any monotone cycles, we reduce the general case to the more restricted one where all faces are rectangles. To that end we geometrically augment Δ to Δ' such that all faces are rectangles. Given an essential cycle C in Δ' we iteratively construct a path P from the topmost point of C in Δ' to $e^\star$ such that each edge on P goes either up or left. For this Δ' needs to be rectangular. We show that the reversed path $\overline{P}$ induces a labeling of C that is 0 on each edge or has positive and negative labels. ◀

In the next section, we introduce further tools based on labelings. Subsequently, in Section 5 we prove Theorem 3.5 for rectangular graphs. In Section 6, we generalize the result to 4-planar graphs.

4 Properties of Labelings

In this section we study the properties of labelings in more detail to derive useful tools for proving Theorem 3.5. Throughout this section, G is a 4-planar graph with ortho-radial representation Γ that satisfies Conditions 1 and 2 of Definition 3.4. Further let $e^\star$ be a reference edge. From Condition 2 we obtain that the rotation of all essential cycles is 0.

▶ **Observation 4.1.** *For any simple essential cycle C of G it is* $\mathrm{rot}(C) = 0$.

Together with the reference edge, an ortho-radial representation fixes the direction of all edges. For the direction of an edge e, we consider a path P from the reference edge to e including both edges. Different such paths may have different rotations but we observe that these rotations differ by a multiple of 4. An edge e is directed *right*, *down*, *left*, or *up*, if $\mathrm{rot}(P)$ is congruent to 0, 1, 2, or 3 modulo 4, respectively. Note that by this definition the reference edge always points to the right. Because the rotation of essential cycles is 0 by Observation 4.1, for two edges on an essential cycle we observe the following.

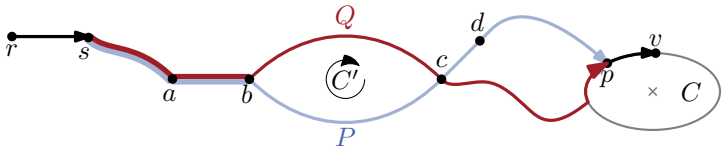


Figure 5 Two paths P and Q from s to p. The cycle C' is formed by $Q[b,c]$ and $\overline{P}[c,b]$.

▶ **Observation 4.2.** *For any path P and for any two edges e and e' on a simple, essential cycle C, it holds that* $\mathrm{rot}(C[e,e']) = \ell_C^P(e') - \ell_C^P(e)$.

We now show that two elementary paths to the same essential cycle C induce identical labelings of C. Two paths P and Q from $e^\star$ to vertices on C are *equivalent* ($P \equiv_C Q$) if the corresponding labelings agree on all edges of C, i.e., $\ell_C^P(e) = \ell_C^Q(e)$ for every edge e of C.

▶ **Lemma 4.3.** *Let C be an essential cycle in G and let $e^\star = rs$. If P and Q are paths from s to vertices on C such that P and Q lie in the exterior of C, they are equivalent.*

Proof Sketch. From the definition of labelings it is not hard to see that two paths P and Q are equivalent if there exists an edge e on C with $\ell_C^P(e) = \ell_C^Q(e)$.

We prove the equivalence of P and Q by showing that such an edge e exists. To that end assume that one of the paths, say Q, is elementary. Let p and q be the endpoints of P and Q, respectively. Let v be the vertex following p on C when C is directed such that the central face lies in its interior. We define $Q' = Q + C[q,p]$. It is easy to verify that both P and Q' are paths that lie in the exterior of C. We show that

$$\ell_C^P(pv) = \mathrm{rot}(e^\star + P + pv) = \mathrm{rot}(e^\star + Q' + pv) = \ell_C^Q(pv). \tag{1}$$

Hence, the edge pv is the desired edge e. So far we have assumed that Q is elementary. If neither P nor Q are elementary, choose any elementary path R to a vertex on C. The argument above shows that $P \equiv_C R$ and $R \equiv_C Q$, and thus $P \equiv_C Q$.

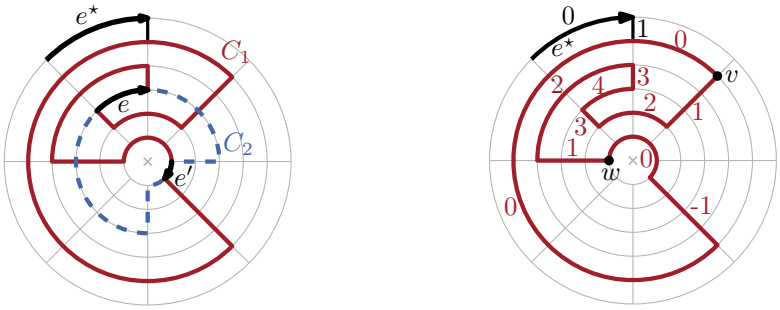
To show Equation 1 we do an induction over the number k of directed edges on Q that do not lie on P. Without loss of generality we assume that Q also ends at p; otherwise we extend Q to $Q + C[q,p]$. If $k = 0$, P contains Q completely and the claim follows immediately.

If $k > 0$, there is a first edge ab on Q such that the following edge does not lie on P. Let c be the first vertex on Q after b that lies on P and let d be the vertex on $P + pv$ that follows c immediately. Fig. 5 illustrates the situation. Consider the cycle $C' = Q[b,c] + \overline{P}[c,b]$. For both cases, namely that C' is essential and C' is non-essential, we argue that $\mathrm{rot}(ab + P[b,c] + cd) = \mathrm{rot}(ab + Q[b,c] + cd)$.

In both cases it then follows from $P[s,b] = Q[s,b]$ that $\mathrm{rot}(rs + P[s,c] + cd) = \mathrm{rot}(rs + Q[s,c] + cd)$. For $P' = Q[s,c] + P[c,p]$ it therefore holds that $\mathrm{rot}(rs + P + pv) = \mathrm{rot}(rs + P' + pv)$. As P' includes the part of Q between b and c it misses fewer edges from Q than P does. Hence, the inductive hypothesis implies $\mathrm{rot}(rs + Q + pv) = \mathrm{rot}(rs + P' + pv)$, and thus $\mathrm{rot}(rs + Q + pv) = \mathrm{rot}(rs + P + pv)$. ◀

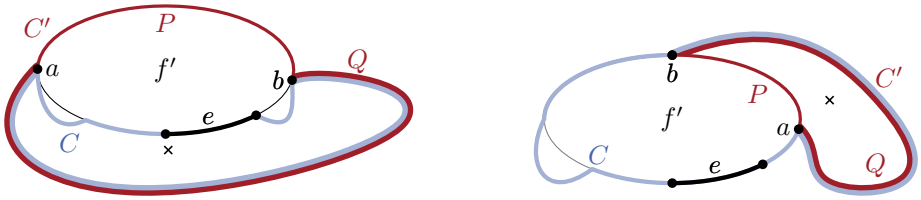
In the remainder of this section all labelings are induced by elementary paths. By Lemma 4.3, the labelings are independent of the choice of the elementary path. Hence, we drop the superscript P and write $\ell_C(e)$ for the labeling of an edge e on an essential cycle C.

If an edge e lies on two simple, essential cycles C_1 and C_2, the labels $\ell_{C_1}(e)$ and $\ell_{C_2}(e)$ may not be equal in general. In Fig. 6a the labels of the edge e are different for C_1 and C_2 respectively, but e' has label 0 on both cycles. Note that e' is incident to the central face of



(a) (b)

Figure 6 (a) Two cycles C_1 and C_2 may have both common edges with different labels ($\ell_{C_1}(e) = 4 \neq 0 = \ell_{C_2}(e)$) and ones with equal labels ($\ell_{C_1}(e') = \ell_{C_2}(e') = 0$). (b) All labels of $C_1[v, w]$ are positive, implying that C_1 goes down. Note that not all edges of $C_1[v, w]$ point downwards.



(a) The cycle bounding the outer face is C'. **(b)** The cycle bounding the central face is C'.

Figure 7 The edge e cannot lie on both the outer and the central face, which is marked by a cross. (a) e does not lie on the outer face, and hence the cycle bounding this face is defined as C'. (b) C' is the cycle bounding the central face. In both cases C' can be subdivided in two paths P and Q on C and f', respectively. Here, these paths are separated by the vertices a and b.

$C_1 + C_2$. One can show that this is a sufficient condition for the equality of the labels; see the full version [1] for a proof.

▶ **Lemma 4.4.** *Let C_1 and C_2 be two essential cycles and let $H = C_1 + C_2$ be the subgraph of G formed by these two cycles. Let v be a common vertex of C_1 and C_2 on the central face of H and consider the edge vw on C_1. Denote the vertices before v on C_1 and C_2 by u_1 and u_2, respectively. Then $\ell_{C_1}(u_1v) + \mathrm{rot}(u_1vw) = \ell_{C_2}(u_2v) + \mathrm{rot}(u_2vw)$.*

Further, if vw belongs to both C_1 and C_2, the labels of e are equal, i.e., $\ell_{C_1}(vw) = \ell_{C_2}(vw)$.

For the correctness proof in Section 6, a crucial insight is that for cycles using an edge which is part of a face, we can find an alternative cycle without this edge in a way that preserves labels on the common subpath, which we show in the next lemma; see Fig. 7 for an illustration.

▶ **Lemma 4.5.** *If an edge e belongs to both a simple essential cycle C and a regular face f' with $C \neq f'$, then there is a simple essential cycle C' not containing e such that C' can be decomposed into two paths P and Q, where P or $\overline{P}$ lies on f' and $Q = C \cap C'$.*

Applying this lemma on an edge e, we construct an essential cycle C' without e from an essential cycle C including e. Also, C and C' have a common path P lying on the central face of $C + f'$. Thus, Lemma 4.4 implies labelings of C and C' being equal on P.

▶ **Corollary 4.6.** *For essential cycles C, C' and the path $P = C \cap C'$ from Lemma 4.5, it is $\ell_C(e) = \ell_{C'}(e)$ for all edges e on P.*

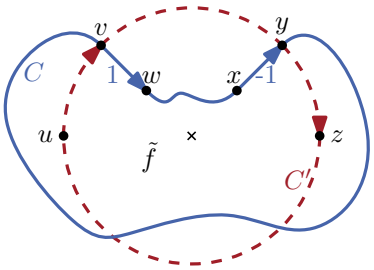
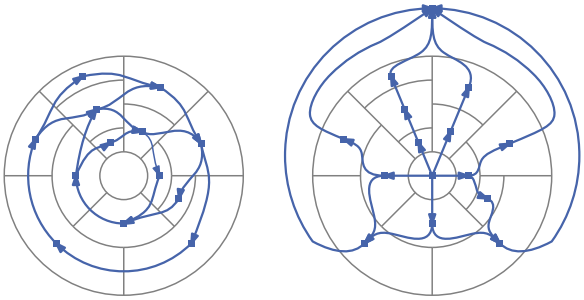


Figure 8 All edges of C' are labeled with 0. In this situation there are edges on C with labels -1 and 1. Hence, C is neither increasing nor decreasing.



(a) N_{ver} **(b)** N_{rad}

Figure 9 Flow networks N_{ver} and N_{rad} for an example graph G. For simplicity, the edge from the outer to the central face in N_{rad} is omitted.

A monotone essential cycle cannot intersect an essential cycle whose edges are all labeled 0.

▶ **Lemma 4.7.** *Let C and C' be two essential cycles that have at least one common vertex. If all edges on C' are labeled with 0, C is neither increasing nor decreasing.*

Proof Sketch. Consider Fig. 8. Let v be a vertex after which C and C' diverge. Assume C leaves this vertex into the interior (the exterior) of C'. Let y be the vertex where both cycles converge again. Then, C enters y from the interior (the exterior) of C. Thus, one of the edges uw and xy must be labeled positively and the other one negatively. ◀

5 Characterization of Rectangular Graphs

In this section, we prove Theorem 3.5 for rectangular graphs. A *rectangular graph* is a graph together with an ortho-radial representation in which every face is a rectangle, i.e., all incident angles are 90° or 180°. We define two flow networks that assign consistent lengths to the graph's edges, one for the vertical and one for the horizontal edges. These networks are straightforward adaptions of the networks used for drawing rectangular graphs in the plane [7]. In the following, *vertex* and *edge* refer to the vertices and edges of the graph G, whereas *node* and *arc* are used for the flow networks.

The network $N_{\text{ver}} = (F_{\text{ver}}, A_{\text{ver}})$ with nodes F_{ver} and arcs A_{ver} for the vertical edges contains one node for each face of G except for the central and the outer face. All nodes have a demand of 0. For each vertical edge e in G, which we assume to be directed upwards, there is an arc $a_e = fg$ in N_{ver}, where f and g denote the face left and right of e, respectively. The flow on a_e has the lower bound $l(a_e) = 1$ and upper bound $u(a_e) = \infty$. An example of this flow network is shown in Fig. 9a.

To obtain a drawing from a flow in N_{ver}, we set the length of a vertical edge e to the flow on a_e. The conservation of the flow at each node f ensures that the two vertical sides of the face f have the same length.

Similarly, the network N_{rad} assigns lengths to the radial edges. There is a node for each face of G, and an arc $a_e = fg$ for every horizontal edge e in G, which we assume to be oriented clockwise. Additionally, N_{rad} includes one arc from the outer to the central face. Again, all edges require a minimum flow of 1 and have infinite capacity. The demand of all nodes is 0. Fig. 9b shows an example of such a flow network.

▶ **Observation 5.1.** *A pair of valid flows in N_{rad} and N_{ver} bijectively corresponds to a valid ortho-radial drawing of G respecting Γ.*

This immediately follows from the construction of the flow networks.

▶ **Theorem 5.2.** *Let (G, Γ) be a rectangular graph and its ortho-radial representation. If Γ is valid, there exists a bend-free ortho-radial drawing of G respecting Γ.*

Proof Sketch. The fact that such a flow exists in N_{rad} is analogous to the orthogonal case [15]. The key is to show that a valid circulation F_{ver} in N_{ver} exists if Γ is valid. The main idea is to determine for each arc a of N_{ver} a cycle C_a in N_{ver} that contains a. If F_a denotes the circulation that pushes one unit of flow along the arcs of C_a and is 0 elsewhere, then $F_{\text{ver}} = \sum_{a \in A} F_a$, where A denotes the arc set of N_{ver}, is the desired flow. The only reason why such a cycle C_a might not exist is if there is a set S of vertices in N_{ver} such that there exists an arc entering S but no arc exiting S. Without loss of generality, we assume the subnetwork of N_{ver} induced by S to be weakly connected, which implies that S corresponds to a connected set $\mathcal{S}$ of faces in G. Note that S contains a directed cycle of N_{ver}, which is an essential cycle. Let C and C' denote the smallest and largest essential cycle of G, respectively, such that all faces in $\mathcal{S}$ lie in the interior of C and in the exterior of C'. We show that C is increasing or C' is decreasing.

Assume there is an arc a entering S that crosses C (an incoming arc crossing C' is similar). Since all faces are rectangles, there is an elementary path P from $e^\star$ to a vertex v on C using only right and down edges of G. Thus, if w is the first vertex of C after v, it is $\ell_C(vw) = 0$ if vw is horizontal and $\ell_C(vw) = -1$ if vw points up. Since no edge on C is pointing downward, i.e., its label is congruent to $1 \mod 4$, and the labels between adjacent edges differ by $-1, 0$, or 1, it follows that $\ell_C(e') \in \{-2, -1, 0\}$ for all edges e' of C, i.e., $\ell_C(e') \leq 0$. However, the edge e corresponding to the incoming arc a of S is pointing upwards, and therefore $\ell_C(e) = -1$. Hence C is increasing. ◀

Combining this result with Theorem 3.6 results in the characterization of ortho-radial drawings for rectangular graphs.

▶ **Corollary 5.3** (Theorem 3.5 for Rectangular Graphs)**.** *A rectangular 4-plane graph admits a bend-free ortho-radial drawing if and only if its ortho-radial representation is valid.*

6 Characterization of 4-Planar Graphs

In the previous section we proved for rectangular graphs that there is an ortho-radial drawing if and only if the ortho-radial representation is valid. We extend this result to 4-planar graphs by reduction to the rectangular case. In this section we provide a high-level overview of the reduction; a detailed description is found in [1].

In Section 6.1 we present an algorithm that augments G to a rectangular graph. In Section 6.2 we then apply Corollary 5.3 to show the remaining implication of Theorem 3.5.

6.1 Rectangulation Algorithm

Given a graph G and its ortho-radial representation Γ as input, we present a rectangulation algorithm that augments Γ producing a graph G' with ortho-radial representation Γ' such that all faces except the central and outer face of Γ' are rectangles. Moreover, we ensure that the outer and the central face of Γ' make no turns.

Let u be a vertex that has a left turn on a face f and let vw be an edge on f. Let further t be the vertex before u on f. We obtain an *augmentation* Γ^u_{vw} from Γ by splitting the edge vw into two edges vz and zw introducing a new vertex z. Further, we insert the edge

uz in the interior of f such that uz points in the same direction as tu. For an illustration of an augmentation see Fig. 10.

Without loss of generality, we assume the central and outer faces to be rectangular; this can be achieved by inserting 3-cycles into both faces and connecting them to the graph. Every regular face that is not a rectangle has a left turn. We augment that face with an additional edge such that the left turn is resolved and no further left turns are introduced, which guarantees that the procedure of augmenting the graph terminates.

We further argue that we augment the given representation in such a way that it remains valid. Conditions 1 and 2 of Definition 3.4 are easy to preserve if we choose the targets of the augmentation correctly; the proof is analogous to the proof by Tamassia [15]. In particular, the next observation helps us to check for these two conditions.

▶ **Observation 6.1.** *The representation Γ^u_{vw} satisfies Conditions 1 and 2 of Definition 3.4 if and only if* $\mathrm{rot}(f[u, vw]) = 2$.

We further have to check Condition 3, i.e., we need to ensure that we do not introduce monotone cycles when augmenting the graph.

We now describe the approach in more detail. Tamassia [15] shows that if there is a left turn, there also is a left turn on a face f such that the next two turns on f are right turns. We resolve that left turn by augmenting G, which we sketch in the following. Let u be the vertex at which face f takes that left turn. Let t be the vertex before u on f. We differentiate two major cases, namely that tu is either vertical or horizontal.

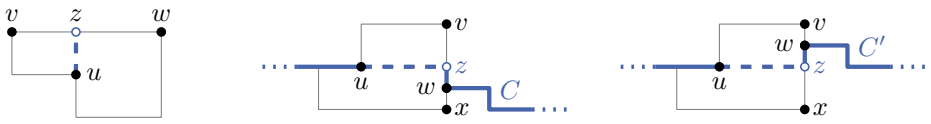
Case 1, tu is vertical. Let vw be the edge that appears on f after the two right turns following the left turn at u. We show that creating a new vertex z on vw and adding the edge uz (cf. to Fig. 10a for an example) always upholds validity; see Section 6.2 for details.

Case 2, tu is horizontal. This case is more intricate, because we may introduce decreasing or increasing cycles by augmenting the graph. Fig. 10b shows an example, where the graph does not include a decreasing cycle without the edge uz, but inserting uz introduces one. We therefore do not just consider Γ^u_{vw}, but Γ^u_e for a set of *candidate edges* e, which include all edges $v'w'$ on f such that $\mathrm{rot}(f[u, v'w']) = 2$. Hence, Condition 1 and Condition 2 are satisfied by Observation 6.1. If there is a candidate e such that Γ^u_e does not contain a monotone cycle, we are done.

So assume that there is no such candidate and, furthermore, assume without loss of generality that tu points to the right. Let the set of candidates be ordered as they appear on f. We show that introducing an edge from u to the first candidate never introduces an increasing cycle, while introducing an edge from u to the last candidate never introduces a decreasing cycle. This also implies that there must be a consecutive pair of candidates vw and $v'w'$ such that introducing an edge from u to vw creates a decreasing cycle and introducing an edge from u to $v'w'$ creates an increasing cycle.

In Section 6.2 we show that in that case, one of the edges uw or uv' can be inserted without introducing a monotone cycle. Together with Observation 6.1, this proves that we can always remove left turns from the representation while maintaining validity.

Altogether, the algorithm consists of first finding a suitable left turn in the representation, then determining which of the two cases applies and finally performing the augmentation. In particular, when augmenting the graph, we need to ensure that we do not introduce monotone cycles. Checking for monotone cycles can trivially be done by testing all essential cycles. However, this may require an exponential number of tests and it is unknown whether testing the existence of a monotone cycle can be done in polynomial time.



(a) Γ^u_{vw} **(b)** Γ^u_{vw} **(c)** Γ^u_{wx}

Figure 10 Examples of augmentations. (a) Inserting uz is valid, if uz points upwards. (b) The representation Γ^u_{vw} is not valid since inserting the new edge introduces a decreasing cycle C. (c) The candidate wx instead gives the valid representation Γ^u_{wx}. The cycle C', which uses the same edges outside of f as C before, is neither in- nor decreasing.

6.2 Correctness of the Rectangulation Algorithm

Following the proof structure outlined in the previous section we argue more detailedly that a valid augmentation always exists. We start with Case 1, in which the inserted edges is vertical and show that a valid augmentation always exists.

▶ **Lemma 6.2.** *Let vw be the first candidate edge after u. If the edge of f entering u points up or down, Γ^u_{vw} is a valid ortho-radial representation of the augmented graph $G + uz$.*

Proof. Assume that Γ^u_{vw} contains a simple monotone cycle C. As Γ is valid, C must contain the new edge uz in either direction (i.e., uz or zu). Let f' be the new rectangular face of $G + uz$ containing u, v and z, and consider the subgraph $H = C + f'$ of $G + uz$. According to Lemma 4.5 there exists a simple essential cycle C' that does not contain uz. Moreover, C' can be decomposed into paths P and Q such that P lies on f' and Q is a part of C.

The goal is to show that also C' is monotone. We present a proof only for the case of increasing C. The proof for decreasing cycles can be obtained by flipping all inequalities.

For each edge e on Q the labels $\ell_C(e)$ and $\ell_{C'}(e)$ are equal by Lemma 4.4, hence $\ell_{C'}(e) \leq 0$. For an edge $e \in P$, there are two possible cases. The edge e either lies on the side of f' parallel to uz or on one of the two other sides. In the first case, the label of e is equal to the label $\ell_C(uz)$ ($\ell_C(zu)$ if C contains zu instead of uz). In particular the label is negative.

In the second case, we first note that $\ell_{C'}(e)$ is even, since e points left or right. Assume that $\ell_{C'}(e)$ is positive and therefore at least 2. Then, let e' be the first edge on C' after e that points to a different direction. Such an edge exists, since otherwise all edges on C' would have the label 2 and therefore C' would be decreasing contradicting the assumption that Γ is valid. This edge e' lies on Q or is parallel to uz. Hence, the argument above implies that $\ell_{C'}(e') \leq 0$. However, $\ell_{C'}(e')$ differs from $\ell_{C'}(e)$ by at most 1, which requires $\ell_{C'}(e') \geq 1$. Therefore, $\ell_{C'}(e)$ cannot be positive.

We conclude that all edges of C' have a non-positive label. If all labels were 0, C would not be an increasing cycle by Lemma 4.7. Thus, there exists an edge on C' with a negative label and C' is an increasing cycle in Γ. But as Γ is valid, such a cycle does not exist, and therefore C does not exist either. Hence, Γ^u_{vw} is valid. ◀

In Case 2 the inserted edge is horizontal. If there is a candidate e such that Γ^u_e is valid, we choose this augmentation and the left turn at u is removed successfully. Otherwise, we make use of the following structural properties. As before we assume that uz points to the right; the other case can be treated analogously. Using a similar, though technically somewhat more difficult approach as in Lemma 6.2 one can show that augmenting with the first and last candidate does not create increasing and decreasing cycles, respectively.

▶ **Lemma 6.3.** *Let vw be the first candidate after u. No increasing cycle exists in Γ^u_{vw}.*

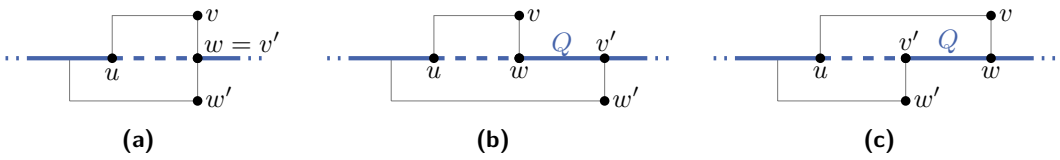


Figure 11 Three possibilities how the path between w and v' can look like: (a) $w = v'$, (b) all edges point right, and (c) all edges point left. In the first two cases the edge uw is inserted and in (c) uv' is added.

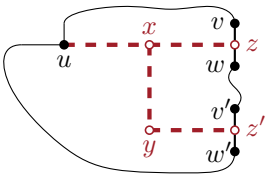


Figure 12 The structure that is used to simulate the insertion of both uz and uz' at the same time. The edge uz is replaced by the path uxz and uz' by $uxyz'$.

▶ **Lemma 6.4.** *Let vw be the last candidate before u. No decreasing cycle exists in Γ^u_{vw}.*

By assumption Γ^u_e contains a monotone cycle for each candidate edge e. From Lemma 6.3 and Lemma 6.4 it follows that there are consecutive candidates vw and $v'w'$ such that Γ^u_{vw} has a decreasing cycle and $\Gamma^u_{v'w'}$ has an increasing cycle.

▶ **Lemma 6.5.** *Let Q be the path on f between the candidates vw and $v'w'$. Then, Q can be completed to a path P in G containing w, v' and u whose edges all point to the right. More precisely, P starts at w or v' and ends at u, and either Q or $\overline{Q}$ forms the first part of P. Moreover, the start vertex of P has no incident edge to its left.*

Proof Sketch. We introduce a construction inside f that simulates the augmentations Γ^u_{vw} and $\Gamma^u_{v'w'}$ at the same time; see Fig. 12. With this construction the resulting representation essentially contains both the increasing cycle of $\Gamma^u_{v'w'}$ and the decreasing cycle of Γ^u_{vw}. Using this, we show that both cycles use Q and thus Q must only point to the right. Similarly, we show that both cycles are equal outside of f, which implies that all corresponding edges also point to the right. Hereby, these edges together with Q form the desired path P. ◀

There are three possible ways how the vertices w and v' can be arranged on P; see Fig. 11. Either $w = v'$, w comes before v', or v' comes before w. In any case we denote the start vertex of P by z. According to Lemma 6.5, no edge is incident to the left of z. Hence, the insertion of the edge uz such that it points right gives a new ortho-radial representation Γ', which is valid by the following lemma.

▶ **Lemma 6.6.** *Let Γ' be the ortho-radial representation that is obtained from Γ by adding the edge uz pointing to the right as in Lemma 6.5. Then, Γ' is valid.*

Proof. By construction, Γ' satisfies Conditions 1 and 2 of Definition 3.4. Let $C' = P + uz$ be the new cycle whose edges point right. It is $\ell_{C'}(e) = 0$ for each edge e of C'. Essential cycles without uz or zu are not monotone, since they are already present in the valid representation Γ. If an essential cycle C contains uz or zu and thus the vertex u, Lemma 4.7 states that C is not monotone. Thus, Γ' satisfies Condition 3 and is therefore valid. ◀

Putting all results together we see that the rectangulation algorithm presented in Section 6.1 works correctly. That is, given a valid ortho-radial representation Γ, the algorithm produces another valid ortho-radial representation Γ' such that all faces of Γ' are rectangles and Γ is contained in Γ'. Combining this result with Theorem 5.2 we obtain the following theorem.

▶ **Theorem 6.7.** *Let Γ be a valid ortho-radial representation of a graph G. Then there is a drawing of G representing Γ.*

This shows the remaining implication of Theorem 3.5 thus completing the characterization.

7 Conclusion

In this paper we considered orthogonal drawings of graphs on cylinders. Our main result is a characterization of the 4-plane graphs with a fixed embedding that can be drawn bend-free on a cylinder in terms of a combinatorial description of such drawings. These ortho-radial representations determine all angles in the drawing without fixing any lengths, and thus are a natural extension of Tamassia's orthogonal representations. However, compared to those, the proof that every valid ortho-radial representation has a corresponding drawing is significantly more involved. The reason for this is the more global nature of the additional property required to deal with the cyclic dimension of the cylinder.

Our ortho-radial representations establish the existence of an ortho-radial TSM framework in the sense that they are a combinatiorial description of the graph serving as interface between the "Shape" and "Metrics" step.

For rectangular 4-plane graphs, we gave an algorithm producing a drawing from a valid ortho-radial representation. Our proof reducing the drawing of general 4-plane graphs with a valid ortho-radial representation to the case of rectangular 4-plane graphs is constructive; however, it requires checking for the violation of our additional consistency criterion. It is an open question whether this condition can be checked in polynomial time. These algorithms correspond to the "Metrics" step in a TSM framework for ortho-radial drawings.

Since the additional property of the characterization is non-local (it is based on paths through the whole graph), the original flow network by Tamassia cannot be easily adapted to compute a bend-minimal valid ortho-radial representation, which would correspond to the "Shape" step of an ortho-radial TSM framework. We leave this as an open question.

References

1 Lukas Barth, Benjamin Niedermann, Ignaz Rutter, and Matthias Wolf. Towards a topology-shape-metrics framework for ortho-radial drawings. *CoRR*, arXiv:1703.06040, 2017.

2 Therese Biedl and Goos Kant. A better heuristic for orthogonal graph drawings. *Computational Geometry: Theory and Applications*, 9:159–180, 1998.

3 Thomas Bläsius, Marcus Krug, Ignaz Rutter, and Dorothea Wagner. Orthogonal graph drawing with flexibility constraints. *Algorithmica*, 68(4):859–885, 2014.

4 Thomas Bläsius, Sebastian Lehmann, and Ignaz Rutter. Orthogonal graph drawing with inflexible edges. *Computational Geometry: Theory and Applications*, 55:26–40, 2016.

5 Thomas Bläsius, Ignaz Rutter, and Dorothea Wagner. Optimal orthogonal graph drawing with convex bend costs. *ACM Transactions on Algorithms*, 12:33:1–33:32, 2016.

6 Sabine Cornelsen and Andreas Karrenbauer. Accelerated bend minimization. *Journal of Graph Algorithms and Applications*, 16(3):635–650, 2012.

7 Giuseppe Di Battista, Peter Eades, Roberto Tamassia, and Ioannis G. Tollis. *Graph Drawing – Algorithms for the Visualization of Graphs*. Prentice Hall, 1999.

8 Giuseppe Di Battista, Giuseppe Liotta, and Francesco Vargiu. Spirality and optimal orthogonal drawings. *SIAM Journal on Computing*, 27(6):1764–1811, 1998. doi:10.1137/S0097539794262847.

9 Christian A. Duncan and Michael T. Goodrich. *Handbook of Graph Drawing and Visualization*, chapter Planar Orthogonal and Polyline Drawing Algorithms, pages 223–246. CRC Press, 2013.

10 Ashim Garg and Roberto Tamassia. On the computational complexity of upward and rectilinear planarity testing. *SIAM Journal on Computing*, 31(2):601–625, 2001.

11 Madieh Hasheminezhad, S. Mehdi Hashemi, Brendan D. McKay, and Maryam Tahmasbi. Rectangular-radial drawings of cubic plane graphs. *Computational Geometry: Theory and Applications*, 43:767–780, 2010.

12 Mahdie Hasheminezhad, S. Mehdi Hashemi, and Maryam Tahmabasi. Ortho-radial drawings of graphs. *Australasian Journal of Combinatorics*, 44:171–182, 2009.

13 Md. Saidur Rahman, Shin-ichi Nakano, and Takao Nishizeki. Rectangular grid drawings of plane graphs. *Computational Geometry: Theory and Applications*, 10:203–220, 1998.

14 Md. Saidur Rahman, Takao Nishizeki, and Mahmuda Naznin. Orthogonal drawings of plane graphs without bends. *Journal of Graph Algorithms and Applications*, 7(3):335–362, 2003.

15 Roberto Tamassia. On embedding a graph in the grid with the minimum number of bends. *SIAM Journal on Computing*, 16(3):421–444, 1987.

On the Number of Ordinary Lines Determined by Sets in Complex Space*†

Abdul Basit[1], Zeev Dvir[2], Shubhangi Saraf[3], and Charles Wolf[4]

1 **Department of Computer Science, Rutgers University, Piscataway, NJ, USA**
basit.abdul@gmail.com
2 **Department of Mathematics and Department of Computer Science, Princeton University, Princeton, NJ, USA**
zeev.dvir@gmail.com
3 **Department of Computer Science and Department of Mathematics, Rutgers University, Piscataway, NJ, USA**
shubhangi.saraf@gmail.com
4 **Department of Mathematics, Rutgers University, Piscataway, NJ, USA**
ciw13@math.rutgers.edu

Abstract

Kelly's theorem states that a set of n points affinely spanning $\mathbb{C}^3$ must determine at least one ordinary complex line (a line passing through exactly two of the points). Our main theorem shows that such sets determine at least $3n/2$ ordinary lines, unless the configuration has $n-1$ points in a plane and one point outside the plane (in which case there are at least $n-1$ ordinary lines). In addition, when at most $n/2$ points are contained in any plane, we prove a theorem giving stronger bounds that take advantage of the existence of lines with four and more points (in the spirit of Melchior's and Hirzebruch's inequalities). Furthermore, when the points span four or more dimensions, with at most $n/2$ points contained in any three dimensional affine subspace, we show that there must be a quadratic number of ordinary lines.

1998 ACM Subject Classification G.2.1 Combinatorics, Counting problems

Keywords and phrases Incidences, Combinatorial Geometry, Designs, Polynomial Method

Digital Object Identifier 10.4230/LIPIcs.SoCG.2017.15

1 Introduction

Let $\mathcal{V} = \{v_1, v_2, \ldots, v_n\}$ be a set of n points in $\mathbb{C}^d$. We denote by $\mathcal{L}(\mathcal{V})$ the set of lines determined by points in $\mathcal{V}$, and by $\mathcal{L}_r(\mathcal{V})$ (resp. $\mathcal{L}_{\geq r}(\mathcal{V})$) the set of lines in $\mathcal{L}(\mathcal{V})$ that contain exactly (resp. at least) r points. Let $t_r(\mathcal{V})$ denote the size of $\mathcal{L}_r(\mathcal{V})$. Throughout the write-up we omit the argument $\mathcal{V}$ when the context makes it clear. We refer to $\mathcal{L}_2$ as the set of *ordinary lines*, and $\mathcal{L}_{\geq 3}$ as the set of *special lines*.

A well known result in combinatorial geometry is the Sylvester-Gallai theorem.

▶ **Theorem 1** (Sylvester-Gallai theorem). *Let $\mathcal{V}$ be a set of n points in $\mathbb{R}^2$ not all on a line. Then there exists an ordinary line determined by points of $\mathcal{V}$.*

* A full version of the paper is available at http://arxiv.org/abs/1611.08740.
† Work of the second author was supported by NSF CAREER award DMS-1451191 and NSF grant CCF-1523816. Work by the third and fourth authors was supported in part by NSF grant CCF-1350572.

The statement was conjectured by Sylvester in 1893 [18], and the first published proof is by Melchior [14]. Later proofs were given by Gallai in 1944 [8] and others; there are now several different proofs of the theorem. Of particular interest is the following result by Melchior [14].

▶ **Theorem 2** (Melchior's inequality [14]). *Let $\mathcal{V}$ be a set of n points in $\mathbb{R}^2$ that are not collinear. Then*

$$t_2(\mathcal{V}) \geq 3 + \sum_{r \geq 4} (r-3) t_r(\mathcal{V}).$$

Theorem 2 in fact proves something stronger than the Sylvester-Gallai theorem, i.e. there are at least three ordinary lines. A natural question to ask is how many ordinary lines must a set of n points, not all on a line, determine. This led to what is known as the *Dirac-Motzkin conjecture*.

▶ **Conjecture 3** (Dirac-Motzkin conjecture). *For every $n \neq 7, 13$, the number of ordinary lines determined by n noncollinear points in the plane is at least $\lceil \frac{n}{2} \rceil$.*

There were several results on this question (see [15, 13, 5]), before Green and Tao [9] resolved it for large enough point sets.

▶ **Theorem 4** (Green and Tao [9]). *Let $\mathcal{V}$ be a set of n points in $\mathbb{R}^2$, not all on a line. Suppose that $n \geq n_0$ for a sufficiently large absolute constant n_0. Then $t_2(\mathcal{V}) \geq \frac{n}{2}$ for even n and $t_2(\mathcal{V}) \geq \lfloor \frac{3n}{4} \rfloor$ for odd n.*

[9] provides a nice history of the problem, and there are several survey articles on the topic, see for example [3].

The Sylvester-Gallai theorem is not true when the field $\mathbb{R}$ is replaced by $\mathbb{C}$. The well known Hesse configuration, realized by the nine inflection points of a non-degenerate cubic, provides a counter example. A more general example is the following:

▶ **Example 5** (Fermat configuration). For any positive integer $k \geq 3$, let $\mathcal{V}$ be inflection points of the Fermat Curve $X^k + Y^k + Z^k = 0$ in $\mathbb{PC}^2$. Then $\mathcal{V}$ has $n = 3k$ points, in particular

$$\mathcal{V} = \bigcup_{i=1}^{k} \{[1 : \omega^i : 0]\} \cup \{[\omega^i : 0 : 1]\} \cup \{[0 : 1 : \omega^i]\},$$

where ω is a k^{th} root of -1.

It is easy to check that $\mathcal{V}$ determines three lines containing k points each, while every other line contains exactly three points. In particular, $\mathcal{V}$ determines no ordinary lines.[1]

In response to a question of Serre [17], Kelly [12] showed that when the points span more than two dimensions, the point set must determine at least one ordinary line.

▶ **Theorem 6** (Kelly's theorem [12]). *Let $\mathcal{V}$ be a set of n points in $\mathbb{C}^3$ that are not contained in a plane. Then there exists an ordinary line determined by points of $\mathcal{V}$.*

Kelly's proof of Theorem 6 used a deep result of Hirzebruch [11] from algebraic geometry. More specifically, it used the following result, known as Hirzebruch's inequality.

[1] We note that the while Fermat configuration as stated lives in the projective plane, it can be made affine by any projective transformation that moves a line with no points to the line at infinity.

▶ **Theorem 7** (Hirzebruch's inequality [11]). *Let $\mathcal{V}$ be a set of n points in $\mathbb{C}^2$, such that $t_n(\mathcal{V}) = t_{n-1}(\mathcal{V}) = t_{n-2}(\mathcal{V}) = 0$. Then*

$$t_2(\mathcal{V}) + \frac{3}{4}t_3(\mathcal{V}) \geq n + \sum_{r\geq 5}(2r-9)t_r(\mathcal{V}).$$

More elementary proofs of Theorem 6 were given in [7] and [6]. To the best of our knowledge, no lower bound greater than one is known for the number of ordinary lines determined by point sets spanning $\mathbb{C}^3$. Improving on the techniques of [6], we make the first progress in this direction.

▶ **Theorem 8.** *Let $\mathcal{V}$ be a set of $n \geq 24$ points in $\mathbb{C}^3$ not contained in a plane. Then $\mathcal{V}$ determines at least $\frac{3}{2}n$ ordinary lines, unless $n-1$ points are on a plane in which case there are at least $n-1$ ordinary lines.*

Clearly if $n-1$ points are coplanar, it is possible to have only $n-1$ ordinary lines. In particular, let $\mathcal{V}$ consist of the Fermat Configuration, for some $k \geq 3$, on a plane and one point v not on the plane. Then $\mathcal{V}$ has $3k+1$ points, and the only ordinary lines determined by $\mathcal{V}$ are lines that contain v, so there are exactly $3k$ ordinary lines. We are not aware of any examples that achieve the $\frac{3}{2}n$ bound when at most $n-2$ points are contained in any plane. Using a similar argument, for point sets in $\mathbb{R}^3$, Theorems 4 and 8 give us the following easy corollary.

▶ **Corollary 9.** *Let $\mathcal{V}$ be a set of n points in $\mathbb{R}^3$ not contained in a plane. Suppose that $n \geq n_0$ for a sufficiently large absolute constant n_0. Then $\mathcal{V}$ determines at least $\frac{3}{2}n - 1$ ordinary lines.*

When $\mathcal{V}$ is sufficiently non-degenerate, i.e. no plane contains too many points, we are able to give a more refined bound in the spirit of Melchior's and Hirzebruch's inequalities, taking into account the existence of lines with more than three points. In particular, we show the following (the constant $1/2$ in Theorem 10 is arbitrary and can be replaced by any positive constant smaller than 1):

▶ **Theorem 10.** *There exists an absolute constant $c > 0$ and a positive integer n_0 such that the following holds. Let $\mathcal{V}$ be a set of $n \geq n_0$ points in $\mathbb{C}^3$ with at most $\frac{1}{2}n$ points contained in any plane. Then*

$$t_2(\mathcal{V}) \geq \frac{3}{2}n + c\sum_{r\geq 4} r^2 t_r(\mathcal{V}).$$

Suppose that $\mathcal{V}$ consists of $n-k$ points on a plane, and k points not on the plane. There are at least $n-k$ lines through each point not on the plane, at most $k-1$ of which could contain three or more points. So we get that there are at least $k(n-2k)$ ordinary lines determined by $\mathcal{V}$. Then if $k = \epsilon n$, for $0 < \epsilon < 1/2$, we get that $\mathcal{V}$ has $\Omega_\epsilon(n^2)$ ordinary lines, where the hidden constant depends on ϵ. Therefore, the bound in Theorem 10 is only interesting when no plane contains too many points.

We note that having at most a constant fraction of the points on any plane is necessary to obtain a bound as in Theorem 10. Indeed, let $\mathcal{V}$ consist of the Fermat Configuration for some $k \geq 3$ on a plane and $o(k)$ points not on the plane. Then $\mathcal{V}$ has $O(k)$ points and determines $o(k^2)$ ordinary lines. On the other hand, $\sum_{r\geq 4} r^2 t_r(\mathcal{V}) = \Omega(k^2)$.

Finally, when a point set $\mathcal{V}$ spans four or more dimensions in a sufficiently non-degenerate manner, i.e. no three dimensional affine subspace contains too many points, we prove that there must be a quadratic number of ordinary lines.

▶ **Theorem 11.** *There exists an absolute constant $c' > 0$ and a positive integer n_0 such that the following holds. Let $\mathcal{V}$ be a set of $n \geq n_0$ points in $\mathbb{C}^4$ with at most $\frac{1}{2}n$ points contained in any three dimensional affine subspace. Then*

$$t_2(\mathcal{V}) \geq c'n^2.$$

Here, again, the constant $1/2$ is arbitrary and can be replaced by any positive constant less than 1. However, increasing this constant will shrink the constant c' in front of n^2. A quadratic lower bound may also be possible if at most $\frac{1}{2}n$ points are contained in any two dimensional space, but we have no proof or counterexample.

Note that while we state Theorems 8 and 10 over $\mathbb{C}^3$ and Theorem 11 over $\mathbb{C}^4$, the same bounds hold in higher dimensions as well since we may project a point set in $\mathbb{C}^d$ onto a generic lower dimensional subspace, preserving the incidence structures. In addition, while these theorems are proved over $\mathbb{C}$, these results are also new and interesting over $\mathbb{R}$.

Organization. In Section 2 we give a short overview of the new ideas in our proof (which builds upon [6]). In Section 3 we develop the necessary machinery on matrix scaling and Latin squares. In Section 4, we prove some key lemmas that will be used in the proofs of our main results. Section 5 gives the proof of Theorem 8, which is considerably simpler than Theorems 10 and 11. In Section 6, we develop additional machinery needed for the proof of Theorem 10 and describe the basic proof idea. The complete proofs of Theorems 10 and 11 can be found in the full version of the paper.

2 Proof overview

The starting point for the proofs of Theorems 8, 10 and 11 is the method developed in [2, 6] which uses rank bounds for *design matrices* – matrices in which the supports of different columns do not intersect in too many positions. We augment the techniques in these papers in several ways which give us more flexibility in analyzing the number of ordinary lines. We devote this short section to an overview of the general framework (starting with [6]) outlining the places where new ideas come into play.

Let $\mathcal{V} = \{v_1, \ldots, v_n\}$ be points in $\mathbb{C}^d$ and denote by V the $n \times (d+1)$ matrix whose i^{th} row is the vector $(v_i, 1) \in \mathbb{C}^{d+1}$, i.e. the vector obtained by appending a 1 to the vector v_i. The dimension of the (affine) space spanned by the point set can be seen to be equal to $\text{rank}(V) - 1$. We would now like to argue that too many collinearities in $\mathcal{V}$ (or too few ordinary lines) imply that all (or almost all) points of $\mathcal{V}$ must be contained in a low dimensional affine subspace, i.e. $\text{rank}(V)$ is small. To do this, we construct a matrix A, encoding the dependencies in $\mathcal{V}$, such that $AV = 0$. Then we must have $\text{rank}(V) \leq n - \text{rank}(A)$, and so it suffices to lower bound the rank of A.

We construct the matrix A in the following manner so that each row of A corresponds to a collinear triple in $\mathcal{V}$. For any collinear triple $\{v_i, v_j, v_k\}$, there exist coefficients a_i, a_j, a_k such that $a_i v_i + a_j v_j + a_k v_k = 0$. We can thus form a row of A by taking these coefficients as the nonzero entries in the appropriate columns. By carefully selecting the triples using constructions of Latin squares (see Lemma 22), we can ensure that A is a design matrix.

The proof in [6] now proceeds to prove a general rank lower bound on any such design-matrix. To understand the new ideas in our proof, we need to 'open the box' and see how the rank bound from [6] is actually proved. The proof in [6] relies on *matrix scaling* techniques to gain control of the matrix. We are allowed to multiply each row and each column of A by a nonzero scalar and would like to reduce to the case where the entries of A are 'mostly

balanced' (see Theorem 14 and Corollary 15). Once scaled, we can consider $M = A^*A$ (note that $rank(M) = rank(A)$). The design properties of A are then used to show that the diagonal entries of M are *large* and the off-diagonal entries are *small*. Such matrices are referred to as *diagonal dominant* matrices, and it is easy to lower bound their rank using trace inequalities (see Lemma 16).

Our proof introduces two new main ideas into this picture. The first idea has to do with the conditions needed to scale A. It is known (see Corollary 15) that a matrix A has a good scaling if it does not contain a 'too large' zero submatrix. This is referred to as having Property-S (see Definition 13). The proof of [6] uses A to construct a new matrix B, whose rows are the same as those of A but with some rows repeating more than once. Then one shows that B has Property-S and continues to scale B (which has rank at most that of A) instead of A. This loses the control on the exact number of rows in A which is crucial for bounding the number of ordinary lines. We instead perform a more careful case analysis: If A has Property-S then we scale A directly and gain more information about the number of ordinary lines. If A does not have Property-S, then we carefully examine the large zero submatrix that violates Property-S. Such a zero submatrix corresponds to a set of points and a set of lines such that no line passes through any of the points. We argue in Lemma 26 that such a submatrix implies the existence of many ordinary lines.

The second new ingredient in our proof comes into play only in the proof of Theorem 10. Here, our goal is to improve on the rank bound of [6] using the existence of lines with four or more points. Recall that our goal is to give a good upper bound on the off-diagonal entries of $M = A^*A$. Consider the (i, j)'th entry of M, obtained by taking the inner product of columns i and j in A. The i'th column of A contains the coefficients of v_i in a set of collinear triples containing v_i (we might not use all collinear triples). In [6] this inner product is bounded using the Cauchy-Schwartz inequality, and uses the fact that we picked our triple family carefully so that v_i and v_j appear together in a small number of collinear triples. One of the key insights of our proof is to notice that since the entries come from linear dependencies, having more than three points on a line gives rise to cancellations in the inner products (which increase the more points we have on a single line).

3 Preliminaries

3.1 Matrix Scaling and Rank Bounds

One of the main ingredients in our proof is rank bounds for design matrices. These techniques were first used for incidence type problems in [2] and improved upon in [6]. We first set up some notation. For a complex matrix A, let A^* denote the matrix conjugated and transposed. Let A_{ij} denote the entry in the i^{th} row and j^{th} column of A. For two complex vectors $u, v \in \mathbb{C}^d$, we denote their inner product by $\langle u, v\rangle = \sum_{i=1}^{d} u_i \cdot \overline{v_i}$.

Central to the obtaining rank bounds for matrices is the notion of matrix scaling. We now introduce this notion and provide some definitions and lemmas.

▶ **Definition 12** (Matrix Scaling)**.** Let A be an $m \times n$ matrix over some field $\mathbb{F}$. For every $\rho \in \mathbb{F}^m, \gamma \in \mathbb{F}^n$ with all entries nonzero, the matrix A' with $A'_{ij} = A_{ij} \cdot \rho_i \cdot \gamma_j$ is referred to as a scaling of A. Note that two matrices that are scalings of each other have the same rank.

We will be interested in scalings of matrices that control the row and column sums. The following property provides a sufficient condition under which such scalings exist.

▶ **Definition 13** (Property-S)**.** Let A be an $m \times n$ matrix over some field. We say that A satisfies Property-S if for every zero submatrix of size $a \times b$, we have

$$\frac{a}{m} + \frac{b}{n} \leq 1.$$

The following theorem is given in [16].

▶ **Theorem 14** (Matrix Scaling theorem)**.** *Let A be an $m \times n$ real matrix with non-negative entries satisfying Property-S. Then, for every $\epsilon > 0$, there exists a scaling A' of A such that the sum of every row of A' is at most $1 + \epsilon$, and the sum of every column of A' is at least $m/n - \epsilon$. Moreover, the scaling coefficients are all positive real numbers.*

We may assume that the sum of every row of the scaling A' is exactly $1 + \epsilon$. Otherwise, we may scale the rows to make the sum $1 + \epsilon$, and note that the column sums can only increase.

The following Corollary to Theorem 14 appeared in [2].

▶ **Corollary 15** (ℓ_2 scaling)**.** *Let A be an $m \times n$ complex matrix satisfying Property-S. Then, for every $\epsilon > 0$, there exists a scaling A' of A such that for every $i \in [m]$*

$$\sum_{j \in [n]} \left|A'_{ij}\right|^2 \leq 1 + \epsilon,$$

and for every $j \in [n]$

$$\sum_{i \in [m]} \left|A'_{ij}\right|^2 \geq \frac{m}{n} - \epsilon\,.$$

Moreover, the scaling coefficients are all positive real numbers.

Corollary 15 is obtained by applying Theorem 14 to the matrix obtained by squaring the absolute values of the entries of the matrix A. Once again, we may assume that $\sum_{j \in [n]} \left|A'_{ij}\right|^2 = 1 + \epsilon$.

To bound the rank of a matrix A, we will bound the rank of the matrix $M = A'^* A'$, where A' is some scaling of A. Then we have that $\mathrm{rank}(A) = \mathrm{rank}(A') = \mathrm{rank}(M)$. We use Corollary 15, along with rank bounds for diagonal dominant matrices. The following lemma is a variant of a folklore lemma on the rank of diagonal dominant matrices (see [1]) and appeared in this form in [6].

▶ **Lemma 16.** *Let A be an $n \times n$ complex hermitian matrix, such that $|A_{ii}| \geq L$ for all $i \in n$. Then*

$$\mathit{rank}(A) \geq \frac{n^2 L^2}{nL^2 + \sum_{i \neq j} |A_{ij}|^2}.$$

The matrix scaling theorem allows us to control the ℓ_2 norms of the columns and rows of A, which in turn allow us to bound the sums of squares of entries of M. To this end, we use the following lemma which appeared in [6].

▶ **Lemma 17.** *Let A be an $m \times n$ matrix over $\mathbb{C}$. Suppose that each row of A has ℓ_2 norm α, the supports of every two columns of A intersect in at most t locations, and the size of the support of every row is q. Let $M = A^* A$. Then*

$$\sum_{i \neq j} |M_{ij}|^2 \leq \left(1 - \frac{1}{q}\right) tm\alpha^4.$$

Lemma 17 is sufficient to prove Theorems 8 and 11. To prove Theorem 10, we need better bounds. A more careful analysis in the proof of Lemma 17 gives us the following lemma. The proof follows the same basic approach and can be found in the full version of the paper. We first need the following definition.

▶ **Definition 18.** Let A be an $m \times n$ matrix over $\mathbb{C}$. Then we define:

$$D(A) := \sum_{i \neq j} \sum_{k<k'} \left| A_{ki}\overline{A_{kj}} - A_{k'i}\overline{A_{k'j}} \right|^2, \text{ and } E(A) := \sum_{k=1}^{m} \sum_{i<j} \left(|A_{ki}|^2 - |A_{kj}|^2 \right)^2.$$

Note that both $D(A)$ and $E(A)$ are non-negative real numbers.

▶ **Lemma 19.** *Let A be an $m \times n$ matrix over $\mathbb{C}$. Suppose that each row of A has ℓ_2 norm α, the supports of every two columns of A intersect in exactly t locations, and the size of the support of every row is q. Let $M = A^*A$. Then*

$$\sum_{i \neq j} |M_{ij}|^2 = \left(1 - \frac{1}{q}\right) tm\alpha^4 - \left(D(A) + \frac{t}{q}E(A)\right).$$

3.2 Latin squares

Latin squares play a central role in our proof. While Latin squares play a role in both [6] and [2], our proof exploits their design properties more strongly.

▶ **Definition 20** (Latin square)**.** An $r \times r$ Latin square is an $r \times r$ matrix L such that $L_{ij} \in [r]$ for all i, j and every number in $[r]$ appears exactly once in each row and exactly once in each column.

If L is a Latin square and $L_{ii} = i$ for all $i \in [r]$, we call it a *diagonal* Latin square.

▶ **Lemma 21.** *For every $r \geq 3$, there exists an $r \times r$ diagonal Latin square. For $r \geq 4$, there exist diagonal Latin squares with the additional property that, for every $i \neq j$, $L_{ij} \neq L_{ji}$.*

Proof. For $r \geq 3$, the existence of $r \times r$ diagonal Latin squares was proved by Hilton [10]. Therefore, we need only show the second part of the theorem. For this we rely on *self-orthogonal Latin squares*.

Two Latin squares L and L' are called *orthogonal* if every ordered pair $(k, l) \in [r]^2$ occurs uniquely as (L_{ij}, L'_{ij}) for some $i, j \in [r]$. A Latin square is called *self-orthogonal* if it is orthogonal to its transpose, denoted by L^T. A theorem of Brayton, Coppersmith, and Hoffman [4] proves the existence of $r \times r$ self-orthogonal Latin squares for $r \in \mathbb{N}$, $r \neq 2, 3, 6$. Let L be a self-orthogonal Latin square. Since $L_{ii} = L^T_{ii}$, the diagonal entries give all pairs of the form (i, i) for every $i \in [r]$, i.e. the diagonal entries must be a permutation of $[r]$. Without loss of generality, we may assume that $L_{ii} = i$ and so L is also a diagonal Latin square. Clearly a self-orthogonal Latin square satisfies the property that $L_{ij} \neq L_{ji}$ if $i \neq j$.

This leaves us only with the case $r = 6$, which requires separate treatment. It is known that 6×6 self-orthogonal Latin squares do not exist. Fortunately, the property we require is weaker and we are able to give an explicit construction of a matrix that is sufficient for our needs. Let L be the following matrix

$$\begin{bmatrix} 1 & 4 & 5 & 3 & 6 & 2 \\ 3 & 2 & 6 & 5 & 1 & 4 \\ 2 & 5 & 3 & 6 & 4 & 1 \\ 6 & 1 & 2 & 4 & 3 & 5 \\ 4 & 6 & 1 & 2 & 5 & 3 \\ 5 & 3 & 4 & 1 & 2 & 6 \end{bmatrix}.$$

It is straightforward to verify that L has the required properties. ◀

The following lemma is a strengthening of a lemma from [2].

▶ **Lemma 22.** *Let $r \geq 3$. Then there exists a set $T \subseteq [r]^3$ of $r^2 - r$ triples, referred to as a* triple system*, that satisfies the following properties:*

1. *Each triple consists of three distinct elements.*
2. *For every pair $i, j \in [r]$, $i \neq j$, there are exactly six triples containing both i and j.*
3. *If $r \geq 4$, for every $i, j \in [r]$, $i \neq j$, there are at least two triples containing i and j such that the remaining elements are distinct.*

Proof. Let L be a Latin square as in Lemma 21. Let T be the set of triples $(i, j, k) \subseteq [r]^3$ with $i \neq j$ and $k = L_{ij}$. Clearly the number of such triples is $r^2 - r$. We verify that the properties mentioned hold.

Recall that we have $L_{ii} = i$ for all $i \in [r]$, and every value appears once in each row and column. So for $i \neq j \in [r]$, it can not happen that $L_{ij} = i$ or $L_{ij} = j$ and we get Property 1, i.e. all elements of a triple must be distinct.

For Property 2, note that a pair i, j appears once as (i, j, L_{ij}) and once as (j, i, L_{ji}). And since every element appears exactly once in every row and column, we have that i must appear once in the j^{th} row, j must appear once in the i^{th} row and the same for the columns. It follows that each of $(*, j, i), (j, *, i), (*, i, j)$ and $(i, *, j)$ appears exactly once, where $*$ is some other element of $[r]$. This gives us that every pair appears in exactly six triples.

For $r \geq 4$ and $i \neq j$, since $L_{ij} \neq L_{ji}$, the triples (i, j, L_{ij}) and (j, i, L_{ji}) are sufficient to satisfy Property 3. ◀

4 The dependency matrix

Let $\mathcal{V} = \{v_1, \ldots, v_n\}$ be a set of n points in $\mathbb{C}^d$. We will use $\dim(\mathcal{V})$ to denote the dimension of the linear span of $\mathcal{V}$ and by affine-dim$(\mathcal{V})$ the dimension of the affine span of $\mathcal{V}$ (i.e., the minimum r such that points of $\mathcal{V}$ are contained in a shift of a linear subspace of dimension r). We projectivize $\mathbb{C}^d$ and consider the set of vectors $\mathcal{V}' = \{v'_1, \ldots, v'_n\}$, where $v'_i = (v_i, 1)$ is the vector in $\mathbb{C}^{d+1}$ obtained by appending a 1 to the vector v_i. Let V be the $n \times (d+1)$ matrix whose i^{th} row is the vector v'_i. Now note that

$$\text{affine-dim}(\mathcal{V}) = \dim(\mathcal{V}') - 1 = \text{rank}(V) - 1.$$

We now construct a matrix A, which we refer to as the dependency matrix of $\mathcal{V}$. Note here that the construction we give here is preliminary, but suffices to prove Theorems 8 and 11. A refined construction is given in Section 6, where we select the triples more carefully. The rows of the matrix will consist of linear dependency coefficients, which we define below.

▶ **Definition 23** (Linear dependency coefficients)**.** Let v_1, v_2 and v_3 be three distinct collinear points in $\mathbb{C}^d$, and let $v'_i = (v_i, 1)$, $i \in \{1, 2, 3\}$, be vectors in $\mathbb{C}^{d+1}$. Recall that v_1, v_2, v_3 are collinear if and only if there exist nonzero coefficients $a_1, a_2, a_3 \in \mathbb{C}$ such that

$$a_1 v'_1 + a_2 v'_2 + a_3 v'_3 = 0.$$

We refer to the a_1, a_2 and a_3 as the linear dependency coefficients between v_1, v_2, v_3. Note that the coefficients are determined up to scaling by a complex number. Throughout our proof, the specific choice of coefficients does not matter, so we fix a canonical choice by setting $a_3 = 1$.

▶ **Definition 24** (Dependency Matrix)**.** For every line $l \in \mathcal{L}_{\geq 3}(\mathcal{V})$, let $\mathcal{V}_l$ denote the points lying on l. Then $|\mathcal{V}_l| \geq 3$ and we assign each line a triple system $T_l \subseteq \mathcal{V}_l^3$, the existence

of which is guaranteed by Lemma 22. Let A be the $m \times n$ matrix obtained by going over every line $l \in \mathcal{L}_{\geq 3}$ and for each triple $(i,j,k) \in T_l$, adding as a row of A a vector with three nonzero coefficients in positions i,j,k corresponding to the linear dependency coefficients among the points v_i, v_j, v_k. We refer to A as the dependency matrix for $\mathcal{V}$.

Note that we have $AV = 0$. Every row of A has exactly three nonzero entries. By Property 2 of Lemma 22, the supports of any distinct two columns intersect in exactly six entries when the two corresponding points lie on a special line[2], and 0 otherwise. That is, the supports of any two distinct columns intersect in at most six entries.

We say a pair of points v_i, v_j, $i \neq j$, *appears* in the dependency matrix A if there exists a row with nonzero entries in columns i and j. The number of times a pair appears is the number of rows with nonzero entries in both columns i and j.

Every pair of points that lies on a special line appears exactly six times. The only pairs not appearing in the matrix are pairs of points that determine ordinary lines. There are $\binom{n}{2}$ pairs of points, $t_2(\mathcal{V})$ of which determine ordinary lines. So the number of pairs appearing in A is $\binom{n}{2} - t_2$. The total number of times these pairs appear is then $6\left(\binom{n}{2} - t_2\right)$. Every row gives three distinct pairs of points, so it follows that the number of rows of A is

$$m = 6\left(\binom{n}{2} - t_2\right)/3 = n^2 - n - 2t_2(\mathcal{V}). \tag{1}$$

Note that $m > 0$, unless $t_2 = \binom{n}{2}$, i.e. all lines are ordinary.

As mentioned in the proof overview, we will consider two cases: when A satisfies Property-S and when it does not. We now prove lemmas dealing with the two cases. The following lemma deals with the former case.

▶ **Lemma 25.** *Let $\mathcal{V}$ be a set of n points affinely spanning $\mathbb{C}^d$, $d \geq 3$, and let A be the dependency matrix for $\mathcal{V}$. Suppose that A satisfies Property-S. Then*

$$t_2(\mathcal{V}) \geq \frac{(d-3)}{2(d+1)}n^2 + \frac{3}{2}n\,.$$

Proof. Fix $\epsilon > 0$. Since A satisfies Property-S, by Lemma 15 there is a scaling A' such that the ℓ_2 norm of each row is at most $\sqrt{1+\epsilon}$ and the ℓ_2 norm of each column is at least $\sqrt{\frac{m}{n} - \epsilon}$. Let $M := A'^* A'$. Then $M_{ii} \geq \frac{m}{n} - \epsilon$ for all i. Since every row in A has support of size three, and the supports of any two columns intersect in at most six locations, Lemma 17 gives us that $\sum_{i \neq j} |M_{ij}|^2 \leq 4m(1+\epsilon)^2$. By applying Lemma 16 to M we get,

$$\operatorname{rank}(M) \geq \frac{n^2(\frac{m}{n} - \epsilon)^2}{n(\frac{m}{n} - \epsilon)^2 + 4m(1+\epsilon)^2}.$$

Taking ϵ to 0, and combining with (1), we get

$$\begin{aligned}\operatorname{rank}(A) = \operatorname{rank}(A') = \operatorname{rank}(M) &\geq \frac{n^2\frac{m^2}{n^2}}{n\frac{m^2}{n^2} + 4m} = \frac{mn}{m+4n} \\ &= n - \frac{4n^2}{m+4n} = n - \frac{4n^2}{n^2 - n - 2t_2(\mathcal{V}) + 4n} \\ &= n - \frac{4n^2}{n^2 + 3n - 2t_2(\mathcal{V})}.\end{aligned}$$

[2] Note that while the triple system T_l consists of ordered triples, the supports of the rows of A are unordered.

Recall that affine-dim$(\mathcal{V}) = d = \text{rank}(V) - 1$. Since $AV = 0$, we have that $\text{rank}(V) \leq n - \text{rank}(A)$. It follows that

$$d + 1 \leq \frac{4n^2}{n^2 + 3n - 2t_2(\mathcal{V})}.$$

Rearranging gives us that

$$t_2(\mathcal{V}) \geq \frac{(d-3)}{2(d+1)}n^2 + \frac{3}{2}n. \qquad \blacktriangleleft$$

We now consider the case when Property-S is not satisfied.

▶ **Lemma 26.** *Let $\mathcal{V}$ be a set of n points in $\mathbb{C}^d$, and let A be the dependency matrix for $\mathcal{V}$. Suppose that A does not satisfy Property-S. Then, for every integer b^*, $1 < b^* < 2n/3$, one of the following holds:*
1. *There exists a point $v \in \mathcal{V}$ contained in at least $\frac{2}{3}(n+1) - b^*$ ordinary lines;*
2. *$t_2(\mathcal{V}) \geq nb^*/2$.*

Proof. Since A violates Property-S, there exists a zero submatrix supported on rows $U \subseteq [m]$ and columns $W \subseteq [n]$ of the matrix A, where $|U| = a$ and $|W| = b$, such that

$$\frac{a}{m} + \frac{b}{n} > 1.$$

Let $X = [m] \setminus U$ and $Y = [n] \setminus W$ and note that $|X| = m - a$ and $|Y| = n - b$. Let the violating columns correspond to the set $\mathcal{V}_1 = \{v_1, \ldots, v_b\} \subset \mathcal{V}$. We consider two cases: when $b < b^*$, and when $b \geq b^*$.

Case 1: ($b < b^*$). We may assume that U is maximal, so every row in the submatrix $X \times W$ has at least one nonzero entry. Partition the rows of X into three parts: Let X_1, X_2 and X_3 be rows with one, two and three nonzero entries in columns of W respectively. We will get a lower bound on the number of ordinary lines containing exactly one point in $\mathcal{V}_1$ and one point in $\mathcal{V} \setminus \mathcal{V}_1$ by bounding the number of pairs $\{v_i, w\}$, with $v_i \in \mathcal{V}_1$ and $w \in \mathcal{V} \setminus \mathcal{V}_1$, that lie on special lines. Note that there are at most $b(n-b)$ such pairs, and each pair that does not lie on a special line determines an ordinary line.

Each row of X_1 gives two pairs of points $\{v_i, w_1\}$ and $\{v_i, w_2\}$ that lie on a special line, where $v_i \in \mathcal{V}_1$ and $w_1, w_2 \in \mathcal{V} \setminus \mathcal{V}_1$. Each row of X_2 gives two pairs of points $\{v_i, w\}$ and $\{v_j, w\}$, where $v_i, v_j \in \mathcal{V}_1$ and $w \in \mathcal{V} \setminus \mathcal{V}_1$ that lie on special lines. Each row of X_3 has all zero entries in the submatrix supported on $X \times Y$, so does not contribute any pairs. Recall that each pair of points on a special line appears exactly six times in the matrix. This implies that the number of pairs that lie on special lines with at least one point in $\mathcal{V}_1$ and one point in $\mathcal{V} \setminus \mathcal{V}_1$ is $\frac{2|X_1|+2|X_2|}{6} \leq \frac{2|X|}{6}$. Hence, the number of ordinary lines containing exactly one of $v_1, \ldots, v_b$ is then at least $b(n-b) - \frac{|X|}{3}$.

Recall that

$$1 < \frac{a}{m} + \frac{b}{n} = \left(1 - \frac{|X|}{m}\right) + \frac{b}{n}.$$

Substituting $m \leq n^2 - n$, from (1), we get

$$|X| < \frac{bm}{n} \leq b(n-1).$$

This gives that the number of ordinary lines containing exactly one point in $\mathcal{V}_1$ is at least

$$b(n-b) - \frac{|X|}{3} > \frac{2b}{3}n - \frac{3b^2-b}{3}.$$

We now have that there exists $v \in \mathcal{V}_1$ such that the number of ordinary lines containing v is at least

$$\left\lfloor \frac{2}{3}n - \frac{3b-1}{3} \right\rfloor \geq \left\lfloor \frac{2}{3}n - b^* + \frac{4}{3} \right\rfloor \geq \frac{2}{3}(n+1) - b^*.$$

Case 2: ($b \geq b^*$). We will determine a lower bound for $t_2(\mathcal{V})$ by counting the number of nonzero pairs of entries $A_{ij}, A_{ij'}$ with $j \neq j'$, that appear in the submatrix $U \times Y$. There are $\binom{n-b}{2}$ pairs of points in $\mathcal{V} \setminus \mathcal{V}_1$, each of which appears at most six times, therefore the number of pairs of such entries is at most $6\binom{n-b}{2}$. Each row of U has three pairs of nonzero entries, i.e. the number of pairs of entries equals $3a$. It follows that

$$3a \leq 6\binom{n-b}{2} \tag{2}$$

Recall equation (1) and that $\frac{a}{m} + \frac{b}{n} > 1$, which gives us

$$a > m\left(1 - \frac{b}{n}\right) = \left(n^2 - n - 2t_2(\mathcal{V})\right)\left(1 - \frac{b}{n}\right). \tag{3}$$

Combining (2) and (3), we get

$$\left(n^2 - n - 2t_2(\mathcal{V})\right)\left(1 - \frac{b}{n}\right) < 2\binom{n-b}{2}.$$

Solving for $t_2(\mathcal{V})$ gives us

$$t_2(\mathcal{V}) > \frac{nb}{2} \geq \frac{nb^*}{2}.$$ ◀

5 Proof of Theorem 8

We note here that the machinery developed so far is sufficient to prove both Theorems 8 and 11. Both proofs are based on similar ideas. We give the proof of Theorem 8 in this section.

The proof relies on Lemmas 25 and 26. Together, these lemmas imply that there must be a point with many ordinary lines containing it, or there are many ordinary lines in total. As mentioned in the proof overview, the theorem is then obtained by using an iterative argument removing a point with many ordinary lines through it, and then applying the same argument to the remaining points. We get the following easy corollary from Lemma 25 and Lemma 26.

▶ **Corollary 27.** *Let $\mathcal{V}$ be a set of n points in $\mathbb{C}^d$ not contained in a plane. Then one of the following holds:*

1. *There exists a point $v \in \mathcal{V}$ contained in at least $\frac{2}{3}n - \frac{7}{3}$ ordinary lines.*
2. *$t_2(\mathcal{V}) \geq \frac{3}{2}n$.*

Proof. Let A be the dependency matrix for $\mathcal{V}$. If A satisfies Property-S, then we are done by Lemma 25. Otherwise, let $b^* = 3$, and note that Lemma 26 gives us the statement of the corollary when $n \geq 5$. ◀

We are now ready to prove Theorem 8.

Proof of Theorem 8. If $t_2(\mathcal{V}) \geq \frac{3}{2}n$ then we are done. Else, by Corollary 27, we may assume there exists a point v_1 with at least $\frac{1}{3}(2n-7)$ ordinary lines and hence at most $\frac{1}{6}(n+4)$ special lines through it. Let $\mathcal{V}_1 = \mathcal{V} \setminus \{v_1\}$. If $\mathcal{V}_1$ is planar, then there are exactly $n-1$ ordinary lines through v_1. We note here that this is the only case where there exists fewer then $\frac{3}{2}n$ ordinary lines.

Suppose now that $\mathcal{V}_1$ is not planar. Again, by Corollary 27, there are either $\frac{3}{2}(n-1)$ ordinary lines in $\mathcal{V}_1$ or there exists a point $v_2 \in \mathcal{V}_1$ with at least $\frac{2}{3}(n-1) - \frac{7}{3} = \frac{1}{3}(2n-9)$ ordinary lines through it. In the former case, we get $\frac{3}{2}(n-1)$ ordinary lines in $\mathcal{V}_1$, at most $\frac{1}{6}(n+4)$ of which could contain v_1. This gives that the total number of ordinary lines in $\mathcal{V}$ is

$$t_2(\mathcal{V}) \geq \frac{3}{2}(n-1) - \frac{1}{6}(n+4) + \frac{1}{3}(2n-7) = \frac{1}{2}(4n-9).$$

When $n \geq 9$, we get that $t_2(\mathcal{V}) \geq \frac{3}{2}n$.

In the latter case there exists a point $v_2 \in \mathcal{V}_1$ with at least $\frac{1}{3}(2n-9)$ ordinary lines in $\mathcal{V}_1$ through it. Note that at most one of these could contain v_1, so we get at least $\frac{1}{3}(2n-7) + \frac{1}{3}(2n-9) - 1 = \frac{1}{3}(4n-19)$ ordinary lines through one of v_1 or v_2. Note also that the number of special lines through one of v_1 or v_2 is at most $\frac{1}{6}(n+4) + \frac{1}{6}(n+3) = \frac{1}{6}(2n+7)$.

Let $\mathcal{V}_2 = \mathcal{V}_1 \setminus \{v_2\}$. If $\mathcal{V}_2$ is contained in a plane, we get at least $n-3$ ordinary lines from each of v_1 and v_2 giving a total of $2n-6$ ordinary lines in $\mathcal{V}$. It follows that when $n \geq 12$, $t_2(\mathcal{V}) \geq \frac{3}{2}n$.

Otherwise $\mathcal{V}_2$ is not contained in a plane, and again Corollary 27 gives us two cases. If there are $\frac{3}{2}(n-2)$ ordinary lines in $\mathcal{V}_2$, then we get that the total number of ordinary lines is

$$t_2(\mathcal{V}) = \frac{3}{2}(n-2) - \frac{1}{6}(2n+7) + \frac{1}{3}(4n-19) = \frac{1}{2}(5n-21).$$

When $n \geq 11$, we get that $t_2(\mathcal{V}) \geq \frac{3}{2}n$.

Otherwise there exists a point v_3 with at least $\frac{2}{3}(n-2) - \frac{7}{3}$ ordinary lines through it. At most two of these could pass through one of v_1 or v_2, so we get $\frac{2}{3}(n-2) - \frac{7}{3} - 2 = \frac{1}{3}(2n-17)$ ordinary lines through v_3 in $\mathcal{V}$. Summing up the number of lines through one of v_1, v_2 and v_3, we get that

$$t_2(\mathcal{V}) \geq \frac{1}{3}(2n-17) + \frac{1}{3}(4n-19) = 2n-12.$$

When $n \geq 24$, we get that $t_2(\mathcal{V}) \geq \frac{3}{2}n$. ◀

6 Proof Idea of Theorem 10

We first give a more careful construction for the dependency matrix of a point set $\mathcal{V}$. Recall that we defined the dependency matrix in Definition 24 to contain a row for each collinear triple from a triple system constructed on each special line. The goal was to not have too many triples containing the same pair. In this section (Definition 33) we will give a construction of a dependency matrix that will have an additional property (captured in Item 4 of Lemma 31) which is used to obtain cancellation in the diagonal dominant argument.

We denote the argument of a complex number z by $\arg(z)$, and use the convention that for every complex number z, $\arg(z) \in (-\pi, \pi]$.

▶ **Definition 28** (angle between two complex numbers)**.** We define the *angle between two complex numbers a and b* to be the the absolute value of the argument of $a\bar{b}$, denoted by $\left|\arg\left(a\bar{b}\right)\right|$. Note that the angle between a and b equals the angle between b and a.

▶ **Definition 29** (co-factor)**.** Let v_1, v_2 and v_3 be three distinct collinear points in $\mathbb{C}^d$, and let a_1, a_2 and a_3 be the linear dependency coefficients among the three points. Define the *co-factor* of v_3 with respect to (v_1, v_2), denoted by $C_{(1,2)}(3)$, to be $\frac{a_1\overline{a_2}}{|a_1||a_2|}$. Notice that this is well defined with respect to the points, and does not depend on the choice of coefficients.

The following lemma will be used to show that "cancellations" must arise in a line containing four points (as mentioned earlier in the proof overview). We will later use this lemma as a black box to quantify the cancellations in lines with more than four points by applying it to random four tuples inside the line.

▶ **Lemma 30.** *Let v_1, v_2, v_3, v_4 be four collinear points in $\mathbb{C}^d$. Then at least one of the following hold:*

1. *The angle between $C_{(1,2)}(3)$ and $C_{(1,2)}(4)$ is at least $\pi/3$.*
2. *The angle between $C_{(1,3)}(4)$ and $C_{(1,3)}(2)$ is at least $\pi/3$.*
3. *The angle between $C_{(1,4)}(2)$ and $C_{(1,4)}(3)$ is at least $\pi/3$.*

Our final dependency matrix will be composed of blocks, each given by the following lemma. Roughly speaking, we construct a block of rows $A(l)$ for each special line l. The rows in $A(l)$ will be chosen carefully and will correspond to triples that will eventually give non trivial cancellations.

▶ **Lemma 31.** *Let l be a line in $\mathbb{C}^d$ and $\mathcal{V}_l = \{v_1, \ldots v_r\}$ be points on l with $r \geq 3$. Let V_l be the $r \times (d+1)$ matrix whose i^{th} row is the vector $(v_i, 1)$. Then there exists an $(r^2 - r) \times r$ matrix $A = A(l)$, which we refer to as the* dependency matrix *of l, such that the following hold:*

1. *$AV_l = 0$;*
2. *Every row of A has support of size three;*
3. *The support of every two columns of A intersects in exactly six locations;*
4. *If $r \geq 4$ then for at least $1/3$ of choices of $k \in [r^2 - r]$, there exists $k' \in [r^2 - r]$ such that following holds: For $k \in [r^2 - r]$, let R_k denote the rth row of A. Suppose $supp(R_k) = \{i, j, s\}$. Then $supp(R_{k'}) = \{i, j, t\}$ (for some $t \neq s$) and the angle between the co-factors $C_{(i,j)}(s)$ and $C_{(i,j)}(t)$ is at least $\pi/3$.*

Proof. Recall that Lemma 22 gives us a family of triples T_r on the set $[r]^3$. For every bijective map $\sigma : \mathcal{V}_l \to [r]$, construct a matrix A_σ in the following manner: Let T_l be the triple system on $\mathcal{V}_l^3$ induced by composing σ and T_r. For each triple $(v_i, v_j, v_k) \in T_l$, add a row with three non-zero entries in positions i, j, k corresponding to the linear dependency coefficients between v_i, v_j and v_k.

Note that for every σ, A_σ has $r^2 - r$ rows and r columns. Since the rows correspond to linear dependency coefficients, clearly we have $A_\sigma V_l = 0$ satisfying Property 1. Properties 2 and 3 follow from properties of the T_l from Lemma 22.

We will use a probabilistic argument to show that there exists a matrix A that has Property 4. Let Σ be the collection of all bijective maps from $[r]$ to the points $\mathcal{V}_l$, and let $\sigma \in \Sigma$ be a uniformly random element. Consider A_σ. Since every pair of points occurs in at least two distinct triples, for every row R_k of A_σ, there exists a row $R_{k'}$ such that the supports of R_k and $R_{k'}$ intersect in two entries. Suppose that R_k and $R_{k'}$ have supports contained in $\{i, j, s, t\}$. Suppose that σ maps $\{v_i, v_j, v_s, v_t\}$ to $\{1, 2, 3, 4\}$ and that $(1, 2, 3)$ and $(1, 2, 4)$ are triples in T_r. Without loss of generality, assume v_i maps to 1. Then by Lemma 30, the angle between at least one of the pairs $\{C_{(i,j)}(s), C_{(i,j)}(t)\}$, $\{C_{(i,s)}(j), C_{(i,s)}(t)\}$, $\{C_{(i,t)}(j), C_{(i,t)}(s)\}$ must be at least $\pi/3$. That is, given that v_i maps to 1, we have that the probability that R_k

satisfies Property 4 is at least 1/3. Then it is easy to see that

$\Pr(R_k \text{ satisfies Property 4}) \geq 1/3.$

Define the random variable X to be the number of rows satisfying Property 4, and note that we have

$$\mathbb{E}[X] \geq (r^2 - r)\frac{1}{3}.$$

It follows that there exists a matrix A in which at least 1/3 of the rows satisfy Property 4. ◀

Based on this new construction, we can give improved bound on the sum of the off-diagonal entries. The proof involves somewhat tedious calculations and can be found in the full version.

▶ **Lemma 32.** *There exists an absolute constant $c_0 > 0$ such that the following holds. Let l be a line in $\mathbb{C}^d$ and $\mathcal{V}_l = \{v_1, \dots v_r\}$ be points on l with $r \geq 4$. Let $A(l)$ be the dependency matrix for l, defined in Lemma 31, and A' a scaling of A such that the ℓ_2 norm of every row is α. Let $M = A'^* A'$.*

$$\sum_{i \neq j} |M_{ij}|^2 \leq 4(r^2 - r)\alpha^4 - c_0(r^2 - r)\alpha^2.$$

We are now ready to define the dependency matrix that we will use in the proof of Theorem 10.

▶ **Definition 33** (Dependency Matrix, second construction)**.** Let $\mathcal{V} = \{v_1, \dots v_n\}$ be a set of n points in $\mathbb{C}^d$ and let V be the $n \times (d+1)$ matrix whose i^{th} row is the vector $(v_i, 1)$. For each matrix $A(l)$, where $l \in \mathcal{L}_{\geq 3}(\mathcal{V})$, add $n - r$ column vectors of all zeroes, with length $r^2 - r$, in the column locations corresponding to points not in l, giving an $(r^2 - r) \times n$ matrix. Let A be the matrix obtained by taking the union of rows of these matrices for every $l \in \mathcal{L}_{\geq 3}(\mathcal{V})$. We refer to A as the *dependency matrix* of $\mathcal{V}$.

Note that this construction is a special case of the one given in Definition 24 and so satisfies all the properties mentioned there. In particular, we have $AV = 0$ and the number of rows in A equals $n^2 - n - 2t_2(\mathcal{V})$.

Proof Idea. We now briefly describe the proof idea for Theorem 10, which follows the proof of Theorem 8 closely. Given the dependency matrix A, if A satisfies Property-S, we are able to use matrix scaling along with the improved bound from Lemma 32. If the matrix does not satisfy Property-S, we use Lemma 26. This gives us the following corollary.

▶ **Corollary 34.** *There exists a constant $c_1 > 0$ and a positive integer n_0 such that the following holds. Let $\mathcal{V}$ be a set of $n \geq n_0$ points in $\mathbb{C}^d$ not contained in a plane. Then one of the following must hold:*

1. *There exists a point $v \in \mathcal{V}$ contained in at least $\frac{n}{2}$ ordinary lines.*
2. $t_2(\mathcal{V}) \geq \frac{3}{2}n + c_1 \sum_{r \geq 4}(r^2 - r)t_r(\mathcal{V})$.

To complete the proof, we again use a pruning argument. We use Corollary 34 to find a point with a large number of ordinary lines, "prune" this point, and then repeat this on the smaller set of points. We stop when either we can not find such a point, in which case Corollary 34 guarantees a large number of ordinary lines, or when we have accumulated enough ordinary lines. The assumption that no plane contains more than $n/2$ points guarantees that we are able to continue pruning until we find sufficiently many ordinary lines.

References

1. N. Alon. Perturbed identity matrices have high rank: Proof and applications. *Combinatorics, Probability and Computing*, 18(1-2):3–15, 2009.
2. B. Barak, Z. Dvir, A. Wigderson, and A. Yehudayoff. Fractional Sylvester–Gallai theorems. *Proceedings of the National Academy of Sciences*, 110(48):19213–19219, 2013.
3. P. Borwein and W. Moser. A survey of Sylvester's problem and its generalizations. *Aequationes Mathematicae*, 40(1):111–135, 1990.
4. R. K. Brayton, D. Coppersmith, and A. J. Hoffman. Self-orthogonal latin squares of all orders $n \neq 2, 3, 6$. *Bulletin of the American Mathematical Society*, 80, 1974.
5. J. Csima and E. T. Sawyer. There exist $6n/13$ ordinary points. *Discrete & Computational Geometry*, 9(2):187–202, 1993.
6. Z. Dvir, S. Saraf, and A. Wigderson. Improved rank bounds for design matrices and a new proof of Kelly's theorem. In *Forum of Mathematics, Sigma*, volume 2, page e4. Cambridge University Press, 2014.
7. N. Elkies, L. M. Pretorius, and K. Swanepoel. Sylvester–Gallai theorems for complex numbers and quaternions. *Discrete & Computational Geometry*, 35(3):361–373, 2006.
8. T. Gallai. Solution of problem 4065. *American Mathematical Monthly*, 51:169–171, 1944.
9. B. Green and T. Tao. On sets defining few ordinary lines. *Discrete & Computational Geometry*, 50(2):409–468, 2013.
10. A. J. W. Hilton. On double diagonal and cross latin squares. *Journal of the London Mathematical Society*, 2(4):679–689, 1973.
11. F. Hirzebruch. Arrangements of lines and algebraic surfaces. In *Arithmetic and geometry*, pages 113–140. Springer, 1983.
12. L. Kelly. A resolution of the Sylvester-Gallai problem of J.-P. Serre. *Discrete & Computational Geometry*, 1(1):101–104, 1986.
13. L. Kelly and W. Moser. On the number of ordinary lines determined by n points. *Canadian Journal of Mathematics*, 10:210–219, 1958.
14. E. Melchior. Über Vielseite der projektiven Ebene. *Deutsche Math*, 5:461–475, 1940.
15. Th. Motzkin. The lines and planes connecting the points of a finite set. *Transactions of the American Mathematical Society*, pages 451–464, 1951.
16. U. Rothblum and H. Schneider. Scalings of matrices which have prespecified row sums and column sums via optimization. *Linear Algebra and its Applications*, 114:737–764, 1989.
17. J.-P. Serre. Advanced problem 5359. *American Mathematical Monthly*, 73(1):89, 1966.
18. J. J. Sylvester. Mathematical question 11851. *Educational Times*, 59(98):256, 1893.

On Optimal 2- and 3-Planar Graphs*

Michael A. Bekos[1], Michael Kaufmann[2], and Chrysanthi N. Raftopoulou[3]

1 **Wilhelm-Schickhard-Institut für Informatik, Universität Tübingen, Tübingen, Germany**
bekos@informatik.uni-tuebingen.de

2 **Wilhelm-Schickhard-Institut für Informatik, Universität Tübingen, Tübingen, Germany**
mk@informatik.uni-tuebingen.de

3 **School of Applied Mathematical & Physical Sciences, NTUA, Athens, Greece**
crisraft@mail.ntua.gr

Abstract

A graph is k-planar if it can be drawn in the plane such that no edge is crossed more than k times. While for $k = 1$, *optimal* 1-planar graphs, i.e. those with n vertices and exactly $4n - 8$ edges, have been completely characterized, this has not been the case for $k \geq 2$. For $k = 2, 3$ and 4, upper bounds on the edge density have been developed for the case of simple graphs by Pach and Tóth, Pach et al. and Ackerman, which have been used to improve the well-known "Crossing Lemma". Recently, we proved that these bounds also apply to non-simple 2- and 3-planar graphs without homotopic parallel edges and self-loops.

In this paper, we completely characterize optimal 2- and 3-planar graphs, i.e., those that achieve the aforementioned upper bounds. We prove that they have a remarkably simple regular structure, although they might be non-simple. The new characterization allows us to develop notable insights concerning new inclusion relationships with other graph classes.

1998 ACM Subject Classification G.2.1 Combinatorics, G.2.2 Graph Theory

Keywords and phrases topological graphs, optimal k-planar graphs, characterization

Digital Object Identifier 10.4230/LIPIcs.SoCG.2017.16

1 Introduction

Topological graphs, i.e. graphs with a representation of the edges as Jordan arcs between corresponding vertex points in the plane, form a well-established subject in the field of geometric graph theory. Besides the classical problems on crossing numbers and crossing configurations [3, 19, 25], the well-known "Crossing Lemma" [2, 18] stands out as a prominent result. Researchers on graph drawing have followed a slightly different research direction, based on extensions of planar graphs that allow crossings in some restricted local configurations [7, 12, 15, 14, 17]. The main focus has been on *1-planar graphs*, where each edge can be crossed at most once, with early results dating back to Ringel [23] and Bodendiek et al. [8]. Extensive work on generation [24], characterization [16], recognition [11], coloring [9], page number [5], etc. has led to a very good understanding of properties of 1-planar graphs.

Pach and Tóth [22], Pach et al. [21] and Ackerman [1] bridged the two research directions by considering the more general class of *k-planar graphs*, where each edge is allowed to be

* This work is supported by the DFG grant Ka812/17-1.

crossed at most k times. In particular, Pach and Tóth provided significant progress, as they developed techniques for upper bounds on the number of edges of simple k-planar graphs, which subsequently led to upper bounds of $5n - 10$ [22], $5.5n - 11$ [21] and $6n - 12$ [1] for simple 2-, 3- and 4-planar graphs, respectively. An interesting consequence was the improvement of the leading constant in the "Crossing Lemma". Note that for general k, the current best bound on the number of edges is $4.1\sqrt{k}n$ [22].

Recently, we generalized the result and the bound of Pach et al. [21] to non-simple graphs, where non-homotopic parallel edges as well as non-homotopic self-loops are allowed [6]. Note that this non-simplicity extension is quite natural and not new, as for planar graphs, the density bound of $3n - 6$ still holds for such non-simple graphs.

In this paper, we now completely characterize optimal non-simple 2- and 3-planar graphs, i.e. those that achieve the bounds of $5n-10$ and $5.5n-11$ on the number of edges, respectively; refer to Theorems 9 and 17. In particular, we prove that the commonly known 2-planar graphs achieving the upper bound of $5n - 10$ edges, are in fact, the only optimal 2-planar graphs. Such graphs consist of a crossing-free subgraph where all not necessarily simple faces have size 5. At each face there are 5 more edges crossing in its interior. We correspondingly show that the optimal 3-planar graphs have a similar simple and regular structure where each planar face has size 6 and contains 8 additional crossing edges.

2 Preliminaries

Let G be a (not necessarily simple) *topological graph*, i.e. G is a graph drawn on the plane, so that the vertices of G are distinct points in the plane, its edges are Jordan curves joining the corresponding pairs of points, and:

(i) no edge passes through a vertex different from its endpoints,
(ii) no edge crosses itself and
(iii) no two edges meet tangentially.

Let $\Gamma(G)$ be such a drawing of G. The *crossing graph* $\mathcal{X}(G)$ of G has a vertex for each edge of G and two vertices of $\mathcal{X}(G)$ are connected by an edge if and only if the corresponding edges of G cross in $\Gamma(G)$. A connected component of $\mathcal{X}(G)$ is called *crossing component*. Note that the set of crossing components of $\mathcal{X}(G)$ defines a partition of the edges of G. For an edge e of G we denote by $\mathcal{X}(e)$ the crossing component of $\mathcal{X}(G)$ which contains e.

An edge e in $\Gamma(G)$ is called a *topological edge* (or simply *edge*, if it is clear in the context). Edge e is called *true-planar*, if it is not crossed in $\Gamma(G)$. The set of all true-planar edges of $\Gamma(G)$ forms the so-called *true-planar skeleton* of $\Gamma(G)$, which we denote by $\Pi(G)$. Since G is not necessarily simple, we will assume that $\Gamma(G)$ contains neither *homotopic parallel edges* nor *homotopic self-loops*, that is, both the interior and the exterior regions defined by any self-loop or by any pair of parallel edges contain at least one vertex. For a positive integer s, a cycle of length s is called *true-planar s-cycle* if it consists of true-planar edges of $\Gamma(G)$. If e is a true-planar edge, then $\mathcal{X}(e) = \{e\}$, while for a chord e of a true-planar s-cycle that has no vertices in its interior, it follows that all edges of $\mathcal{X}(e)$ are also chords of this s-cycle. Let $\mathcal{F}_s = \{v_1, v_2, \ldots, v_s\}$ be a facial s-cycle of $\Pi(G)$ with length $s \geq 3$. The order of the vertices (and subsequently the order of the edges) of $\mathcal{F}_s$ is determined by a walk along the boundary of $\mathcal{F}_s$ in clockwise direction. Since $\mathcal{F}_s$ is not necessarily simple, a vertex or an edge may appear more than once; see Fig. 1a. More general, a *region* in $\Gamma(G)$ is defined as a closed walk along non-intersecting segments of Jordan curves that are adjacent either at vertices or at crossing points of $\Gamma(G)$. The *interior* and the *exterior* of a connected region are defined as the topological regions to the right and to the left of the walk.

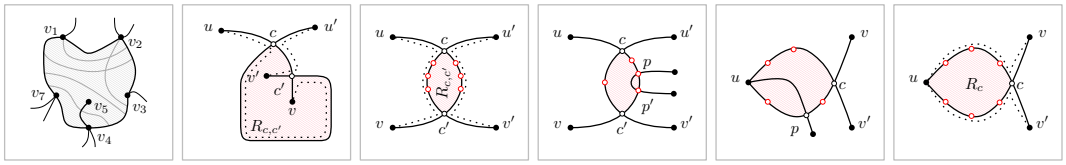


Figure 1 (a) A non-simple face $\{v_1, \ldots, v_7\}$, where v_6 is identified with v_4. Different configurations used in (b–d) Lemma 1, and (e–f) Lemma 2.

If every edge in $\Gamma(G)$ is crossed at most k times, $\Gamma(G)$ is called *k-planar*. A graph is *k-planar* if it has a k-planar drawing. An *optimal k-planar* graph is a k-planar graph with the maximum number of edges. In particular, we consider optimal 2- and 3-planar graphs achieving the best-known upper bounds of $5n - 10$ and $5.5n - 11$ edges. A k-planar drawing $\Gamma(G)$ of an optimal k-planar graph G is called *planar-maximal crossing-minimal* or PMCM-drawing, if and only if $\Gamma(G)$ has the maximum number of true-planar edges among all k-planar drawings of G and, subject to this, $\Gamma(G)$ has the minimum number of crossings.

Consider two edges (u, v) and (u', v') that cross at least twice in $\Gamma(G)$. Let c and c' be two crossing points of (u, v) and (u', v') that appear consecutively along (u, v) in this order from u to v (i.e., there is no other crossing point of (u, v) and (u', v') between c and c'). W.l.o.g. we assume that c and c' appear in this order along (u', v') from u' to v' as well. In Figs. 1b and 1c we have drawn two possible crossing configurations. First we drew (u, v) as an arc with u above v and the edge-segment of (u', v') between u and c to the right of (u, v). The edge-segment of (u', v') between c and c', starts at c and ends at c' either from the right (Fig. 1b) or from the left (Fig. 1c) of (u, v), yielding two crossing configurations.

▶ **Lemma 1.** *For $k \in \{2, 3\}$, let $\Gamma(G)$ be a PMCM-drawing of an optimal k-planar graph G in which two edges (u, v) and (u', v') cross more than once. Let c and c' be two consecutive crossings of (u, v) and (u', v') along (u, v), and let $R_{c,c'}$ be the region defined by the walk along the edge segment of (u, v) from c to c' and the one of (u', v') from c' to c. Then, $R_{c,c'}$ has at least one vertex in its interior and one in its exterior.*

Proof. Consider first the crossing configuration of Fig. 1b. Since crossings c and c' are consecutive along (u, v) and (u', v') does not cross itself, it follows that vertex u' lies in the exterior of $R_{c,c'}$, while vertex v' in the interior of $R_{c,c'}$. Hence, the lemma holds. Consider now the crossing configuration of Fig. 1c. Since c and c' are consecutive along (u, v), vertices u' and v' are in the exterior of $R_{c,c'}$. Assume now, to the contrary, that $R_{c,c'}$ contains no vertices in its interior. W.l.o.g. we further assume that (u, v) and (u', v') is a *minimal crossing pair* in the sense that, $R_{c,c'}$ cannot contain another region $R_{p,p'}$ defined by any other pair of edges that cross twice; for a counterexample see Fig. 1d. Let $nc(u, v)$ and $nc(u', v')$ be the number of crossings along (u, v) and (u', v') that are between c and c', respectively (red in Fig. 1c). Observe that by the "minimality" criterion of (u, v) and (u', v') we have $nc(u, v) = nc(u', v')$. We redraw edges (u, v) and (u', v') by exchanging their segments between c and c' and eliminate both crossings c and c' without affecting the k-planarity of G; see the dotted edges of Fig. 1c. This contradicts the crossing minimality of $\Gamma(G)$. ◀

▶ **Lemma 2.** *For $k \in \{2, 3\}$, let $\Gamma(G)$ be a PMCM-drawing of an optimal k-planar graph G in which two edges (u, v) and (u, v') incident to a common vertex u cross. Let c be the first crossing of them starting from u and let R_c be the region defined by the walk along the edge*

segment of (u, v) from u to c and the one of (u, v') from c to u. Then, R_c has at least one vertex in its interior and one in its exterior.

Proof Sketch. The proof is analogous to the one of Lemma 1; see Figs. 1e-1f and [20]. ◀

In our proofs by contradiction we usually deploy a strategy in which starting from an optimal 2- or 3-planar graph G, we modify G and its drawing $\Gamma(G)$ by adding and removing elements (vertices or edges) without affecting its 2- or 3-planarity. Then, the number of edges in the derived graph forces G to have either fewer or more edges than the ones required by optimality (contradicting the optimality or the 3-planarity of G, resp.). To deploy the strategy, we must ensure that we introduce neither homotopic parallel edges and self-loops nor self-crossing edges. We next show how to select and draw the newly inserted elements.

A Jordan curve $[u, v]$ connecting vertex u to v of $\Gamma(G)$ is called *potential edge* if and only if $[u, v]$ does not cross itself and is not a homotopic self-loop in $\Gamma(G)$, that is, either $u \neq v$ or $u = v$ and there is at least one vertex in the interior and the exterior of $[u, v]$. Note that u and v are not necessarily adjacent in G. However, since each topological edge $(u, v) \in E$ of G is represented by a Jordan curve in $\Gamma(G)$, it follows that edge (u, v) is by definition a potential edge of $\Gamma(G)$ among other potential edges that possibly exist. Furthermore, we say that vertices $v_1, v_2, \ldots, v_s$ define a *potential empty cycle* $\mathcal{C}_s$ in $\Gamma(G)$, if there exist potential edges $[v_i, v_{i+1}]$, for $i = 1, \ldots, s-1$ and potential edge $[v_1, v_s]$ of $\Gamma(G)$, which

(i) do not cross each other and

(ii) the walk along the curves between $v_1, v_2, \ldots, v_s, v_1$ defines a region in $\Gamma(G)$ that has no vertices in its interior.

Note that $\mathcal{C}_s$ might be non-simple.

▶ **Lemma 3.** *For $k \in \{2, 3\}$, let $\Gamma(G)$ be a PMCM-drawing of a k-planar graph G. Let also $\mathcal{C}_s$ be a potential empty cycle of length s in $\Gamma(G)$ and assume that κ edges of $\Gamma(G)$ are drawn completely in the interior of $\mathcal{C}_s$, while λ edges of $\Gamma(G)$ are crossing[1] the boundary of $\mathcal{C}_s$. Also, assume that if one focuses on $\mathcal{C}_s$ of $\Gamma(G)$, then μ pairwise non-homotopic edges can be drawn as chords completely in the interior of $\mathcal{C}_s$ without deviating k-planarity.*

(i) *If $\mu > \kappa + \lambda$, then G is not optimal.*

(ii) *If G is optimal and $\mu = \kappa + \lambda$, then all boundary edges of $\mathcal{C}_s$ exist[2] in $\Gamma(G)$.*

Proof. (i) If we could replace the $\kappa + \lambda$ edges of $\Gamma(G)$ that are either drawn completely in the interior of $\mathcal{C}_s$ or cross the boundary of $\mathcal{C}_s$ with the μ ones that one can draw exclusively in the interior of $\mathcal{C}_s$, then the lemma would trivially follow. However, to do so we need to ensure that this operation introduces neither homotopic parallel edges nor homotopic self-loops. Since the edges that we introduce are potential edges, it follows that no homotopic self-loops are introduced. We claim that homotopic parallel edges are not introduced either. In fact, if e and e' are two homotopic parallel edges, then both must be drawn completely in the interior of $\mathcal{C}_s$, which implies that e and e' are both newly-introduced edges; a contradiction, since we introduce μ pairwise non-homotopic edges. (ii) In the exchanging scheme that we just described, we drew μ edges as chords exclusively in the interior of $\mathcal{C}_s$. Of course, one can also draw the boundary edges of $\mathcal{C}_s$. Since G is optimal, these edges must exist in $\Gamma(G)$. ◀

[1] Note that the boundary edges of $\mathcal{C}_s$ are not necessarily present in $\Gamma(G)$.

[2] We say that a Jordan curve $[u, v]$ *exists* in $\Gamma(G)$ if and only if $[u, v]$ is homotopic to an edge in $\Gamma(G)$.

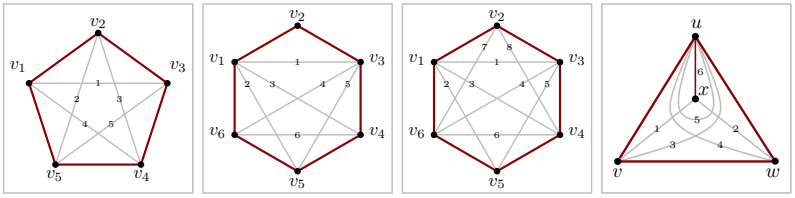


Figure 2 (a–c) A potential empty cycle $\mathcal{C}_s$ with (a) $s = 5$ and five chords with two crossings each, (b) $s = 6$ and six chords with at most two crossings each, and (c) $s = 6$ and eight chords with at most three crossings each. (d) Configuration used in the proof of Property 2.

3 Properties of optimal 2- and 3-planar graphs

In this section, we investigate properties of optimal 2- and 3-planar graphs. We prove that a PMCM-drawing $\Gamma(G)$ of an optimal 2- or 3-planar graph G can contain neither true-planar cycles of a certain length nor a pair of edges that cross twice. We use these properties to show that $\Gamma(G)$ is *quasi-planar*, i.e. it contains no 3 pairwise crossing edges.

Let R be a simple closed region that contains at least one vertex of G in its interior and one in its exterior. Let H_1 (H_2) be the subgraph of G whose vertices and edges are drawn entirely in the interior (exterior) of R. Note that potentially there exist edges that exit and enter R. We refer to H_1 and H_2 as the *compact subgraphs* of $\Gamma(G)$ defined by R. The following lemma bounds the number of edges in any compact subgraph of $\Gamma(G)$.

▶ **Property 1.** *Let $\Gamma(G)$ be a drawing of an optimal 2- or 3-planar graph G and let H be a compact subgraph of $\Gamma(G)$ on n' vertices defined by a closed region R. If $n' \geq 2$, H has at most $5n' - 6$ edges if G is optimal 2-planar, and at most $5.5n' - 6.5$ edges if G is optimal 3-planar. Further, there exists at least one edge of G crossing the boundary of R in $\Gamma(G)$.*

Proof. We prove this property for the class of 3-planar graphs; the proof for the class of 2-planar graphs is analogous. So, let $\Gamma(G)$ be a drawing of an optimal 3-planar graph $G = (V, E)$ with n vertices and m edges. Let H_1 and H_2 be two compact subgraphs of $\Gamma(G)$ defined by a closed region R. For $i = 1, 2$ let n_i and m_i be the number of vertices and edges of H_i. Suppose that $n_1 \geq 2$. In the absence of $\Gamma(H_2)$, drawing $\Gamma(H_1)$ might contain homotopic parallel edges or self-loops. To overcome this problem, we subdivide an edge-segment of the unbounded region of $\Gamma(H_1)$ by adding one vertex.[3] The derived graph, say H_1', has $n_1' = n_1 + 1$ vertices and $m_1' = m_1 + 1$ edges. Since H_1' has no homotopic parallel edges or self-loops and $n_1' \geq 3$, it follows that $m_1' \leq 5.5n_1' - 11$, which gives $m_1 \leq 5.5n_1 - 6.5$.

For the second part, assume to the contrary that no edge crosses the boundary of R. This implies that $m = m_1 + m_2$. We consider first the case where $n_1, n_2 \geq 2$. By the above we have that $m_1 \leq 5.5n_1 - 6.5$and $m_2 \leq 5.5n_2 - 6.5$. Since $n = n_1 + n_2$ and $m = m_1 + m_2$, it follows that $m \leq 5.5n - 13$; a contradiction to the optimality of G. Since a graph consisting only of two non-adjacent vertices cannot be optimal, it remains to consider the case where either $n_1 = 1$ or $n_2 = 1$. W.l.o.g. assume that $n_1 = 1$. Since $n_2 \geq 2$, it follows that $m_2 \leq 5.5n_2 - 6.5$, which implies $m \leq 5.5n - 12$; a contradiction to the optimality of G. ◀

For two compact subgraphs H_1 and H_2 defined by a closed region R, Property 1 implies that the drawings of H_1 and H_2 cannot be "separable". In other words, either there exists

[3] One can view this process as replacing $\Gamma(H_2)$ with a single vertex; thus no homotopic parallel edges exist in $\Gamma(H_1)$. Then we move this vertex towards the edge-segment we want to subdivide until it touches it.

an edge connecting a vertex of H_1 with a vertex of H_2, or there exists a pair of edges, one connecting vertices of H_1 and the other vertices of H_2, that cross in the drawing $\Gamma(G)$.

▶ **Property 2.** *In a PMCM-drawing $\Gamma(G)$ of an optimal 2-planar graph G there is no empty true-planar cycle of length three.*

Proof. Assume to the contrary that there exists an empty true-planar 3-cycle $\mathcal{C}$ in $\Gamma(G)$ on vertices u, v and w. Since G is connected and since all edges of $\mathcal{C}$ are true-planar, there is neither a vertex nor an edge-segment in $\mathcal{C}$, i.e., $\mathcal{C}$ is a chordless facial cycle of $\Pi(G)$. This allows us to add a vertex x in its interior and connect x to vertex u by a true-planar edge. Now vertices u, x, u, w and v define a potential empty cycle of length five, and we can draw five chords in its interior without violating 2-planarity and without introducing homotopic parallel edges or self-loops; refer to Fig. 2d. The derived graph G' has one more vertex than G and six more edges. Hence, if n and m are the number of vertices and edges of G respectively, then G' has $n' = n + 1$ vertices and $m' = m + 6$ edges. Then $m' = 5n' - 9$, which implies that G' has more edges than allowed; a contradiction. ◀

▶ **Property 3.** *The number of vertices of an optimal 3-planar graph G is even.*

Proof. Follows directly from the density bound of $5.5n - 11$ of G. ◀

▶ **Property 4.** *A PMCM-drawing $\Gamma(G)$ of an optimal 3-planar graph G has no true-planar cycle of odd length.*

Proof. Let $s \geq 1$ be an odd number and assume to the contrary that there exists a true-planar s-cycle $\mathcal{C}$ in $\Gamma(G)$. Denote by G_1 (G_2, resp.) the subgraph of G induced by the vertices of $\mathcal{C}$ and the vertices of G that are in the interior (exterior, resp.) of $\mathcal{C}$ in $\Gamma(G)$ without the chords of $\mathcal{C}$ that are in the exterior (interior, resp.) of $\mathcal{C}$ in $\Gamma(G)$. For $i = 1, 2$, observe that G_i contains a copy of $\mathcal{C}$. Let n_i and m_i be the number of vertices and edges of G_i that do not belong to $\mathcal{C}$. Based on graph G_i, we construct graph G'_i by employing two copies of G_i that share cycle $\mathcal{C}$. Observe that G'_i is 3-planar, because one copy of G_i can be embedded in the interior of $\mathcal{C}$, while the other one in its exterior. Hence, in this embedding, there exist neither homotopic self-loops nor homotopic parallel edges. Let n'_i and m'_i be the number of vertices and edges of G'_i that do not belong to $\mathcal{C}$. If G has n vertices and m edges, then by construction the following equalities hold:

(i) $n'_i = 2n_i + s$,
(ii) $m'_i = 2m_i + s$,
(iii) $n = n_1 + n_2 + s$, and
(iv) $m = m_1 + m_2 + s$.

We now claim that $n'_i \geq 3$. When $s \geq 3$ the claim clearly holds. Otherwise (i.e., $s = 1$), cycle $\mathcal{C}$ is degenerated to a self-loop which must contain at least one vertex in its interior and its exterior. Hence, the claim follows. Property 3 in conjunction with Eq. (i) implies that G'_i is not optimal, that is, $m'_i < 5.5n'_i - 11$. So, by Eq. (ii) it follows that $2m_i + s < 5.5(2n_i + s) - 11$. Summing up over i, we obtain that $2(m_1 + m_2 + s) < 5.5(2n_1 + 2n_2 + 2s) - 22$. Finally, from Eqs. (iii) and (iv) we conclude that $m < 5.5n - 11$; a contradiction to the optimality of G. ◀

▶ **Property 5.** *In a PMCM-drawing $\Gamma(G)$ of an optimal 2-planar graph G there is no pair of edges that cross twice with each other.*

Proof. Assume to the contrary that (u, u') and (v, v') cross twice in $\Gamma(G)$ at points c and c'. By 2-planarity no other edge of $\Gamma(G)$ crosses (u, u') and (v, v'). Let $R_{c,c'}$ be the region

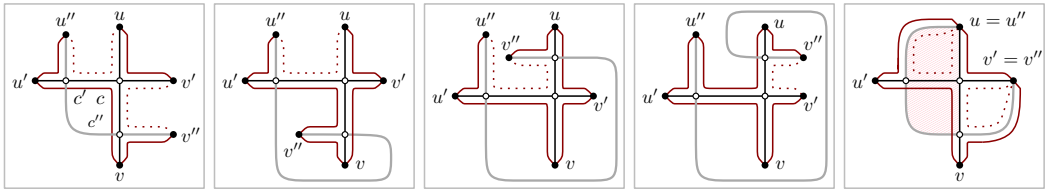


Figure 3 Crossing configurations for three mutually crossing edges. Potential edges are drawn solid red. Jordan curves that can either be potential edges or homotopic self-loops are drawn dotted red.

defined by the walk along the edge segment of (u, u') between c and c' and the edge segment of (v, v') between c' and c. As mentioned in the proof of Lemma 1, there exist two crossing configurations for (u, u') and (v, v'); see Figs. 1b and 1c. In the crossing configuration of Fig. 1b, vertices v and v' are in the interior of $R_{c,c'}$, while vertices u and u' in its exterior. Hence, $u \neq v$ and $u' \neq v'$ hold. We redraw (u, u') and (v, v') by exchanging the middle segments between c and c' and eliminate both crossings c and c' without affecting 2-planarity; see the dotted edges of Fig. 1b. Note that since $u \neq v$ and $u' \neq v'$ the two edges cannot be homotopic self-loops. Also, no homotopic parallel edges are introduced, since this would imply that at least one of the two edges already exists in $\Gamma(G)$ violating 2-planarity. Consider the crossing configuration of Fig. 1c. By Lemma 1, $R_{c,c'}$ has at least one vertex in its interior. By 2-planarity, no edge of G crosses the boundary of $R_{c,c'}$ contradicting Property 1. ◀

▶ **Property 6.** *In a PMCM-drawing $\Gamma(G)$ of an optimal 3-planar graph G there is no pair of edges that cross more than once with each other.*

Proof Sketch. As in the proof of Property 5, we show that if (u, v) and (u', v') cross three times, then either $\Gamma(G)$ is not crossing minimal or Property 1 is violated. The rest of the proof needs different arguments, as (u, v) and (u', v') may have one more crossing each; see [20]. ◀

Now assume that $\Gamma(G)$ contains three mutually crossing edges (u, v), (u', v') and (u'', v''). In Figs. 3a–3d we have drawn four the possible crossing configurations depending on the "direction" of the crossing of (u'', v'') along (u, v). Note that the endpoints of the three edges are not necessarily distinct (e.g., in Fig. 3e we illustrate the case where $u = u''$ and $v' = v''$ for the crossing configuration of Fig. 3a). For each crossing configuration, one can draw curves connecting the endpoints of (u, v), (u', v') and (u'', v'') (red colored in Figs. 3a–3d), which define a region that has no vertices in its interior. This region fully surrounds (u, v) and (u', v') and the two segments of (u'', v'') that are incident to vertices u'' and v''.

▶ **Claim 1.** *Each crossing configuration of Figs. 3b-3d induces at least 5 potential edges.*

Proof. All solid-drawn red curves of Figs. 3b–3d are indeed potential edges. ◀

▶ **Claim 2.** *The crossing configuration of Fig. 3a induces at least four potential edges.*

Proof. $[u', u'']$, $[u, v']$, $[u', v]$ and $[v, v'']$ are potential edges. ◀

▶ **Corollary 4.** *The configuration of Fig. 3a induces a potential empty cycle $\mathcal{C}$ of length ≥ 4. Each configuration of Figs. 3b–3d induces a potential empty cycle $\mathcal{C}$ of length ≥ 5.*

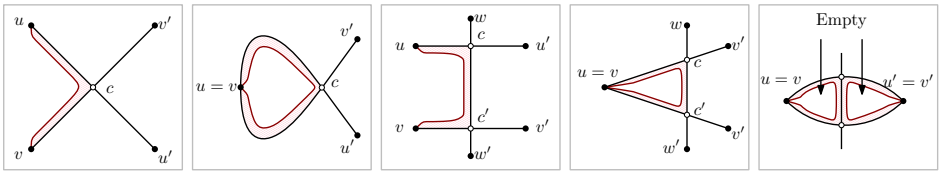


Figure 4 (a–b) vertices u and v form a corner pair; (c–d) vertices u and v form a side pair; (e) at least one of the two potential side-edges exists.

▶ **Claim 3.** *In the case where the crossing configuration of Fig. 3a induces exactly four potential edges, there exists at least one vertex in the interior of region $\mathcal{T}$ defined by the walk along the edge segment of (u, v) between c and c'', the edge segment of (u'', v'') between c'' and c' and the edge segment of (u', v') between c' and c.*

Proof. By Claim 2, $[u, u'']$, and $[v', v'']$ are homotopic self-loops. So, edges (u, v) and (u'', v'') are incident to a common vertex, namely, $u = u''$ and cross. By Lemma 2, $R_{c''}$ (red-shaded in Fig. 3e) has at least one vertex in its interior. Since $R_{c''}$ is the union of the interior of $\mathcal{T}$ and the homotopic self-loop $[u, u'']$, $\mathcal{T}$ contains at least one vertex in its interior. ◀

▶ **Property 7.** *A PMCM-drawing $\Gamma(G)$ of an optimal 2-planar graph G is quasi-planar.*

Proof. Assume to the contrary that (u, v), (u', v') and (u'', v'') mutually cross in $\Gamma(G)$; see Fig. 3. By Corollary 4, there is a potential empty cycle $\mathcal{C}$ of length at least 4. By 2-planarity, there is no other edge crossing (u, v), (u', v') or (u'', v''). So, the only edges that are drawn in the interior of $\mathcal{C}$ are (u, v) and (u', v'), while (u'', v'') is the only crossing the boundary of $\mathcal{C}$.

First, consider the case where $\mathcal{C}$ is of length ≥ 5. Since we can draw at least five chords completely in the interior of $\mathcal{C}$ as in Fig. 2a or 2b without violating its 2-planarity, it follows by Lemma 3.(i) (for $\kappa + \lambda = 3$ and $\mu \geq 5$) that G is not optimal; a contradiction. Finally, consider the case where $\mathcal{C}$ is of length four. In this case, we have the crossing configuration of Fig. 3a. By Claim 3 there is at least one vertex in the interior of region $\mathcal{T}$. More in general, let $G_{\mathcal{T}}$ be the compact subgraph of G that is completely drawn in the interior of region $\mathcal{T}$. Since edges (u, v), (u', v') and (u'', v'') have already two crossings, it follows that no edge of G crosses the boundary of $\mathcal{T}$ contradicting Property 1. ◀

▶ **Property 8.** *A PMCM-drawing $\Gamma(G)$ of an optimal 3-planar graph G is quasi-planar.*

Proof Sketch. This property is the analogue of Property 7 and its proof is given in [20]. ◀

Two not necessarily distinct vertices u and v of G form a *corner pair* if and only if an edge (u, u') crosses an edge (v, v') for some u' and v' in $\Gamma(G)$; see Fig. 4a. Let c be the crossing point of (u, u') and (v, v'). Then, a Jordan curve $[u, v]$ joining vertices u and v induces a region $R_{u,v}$ that is defined by the walk along the edge-segment of (u, u') from u to c, the edge segment of (v, v') from c to v and the curve $[u, v]$ from v to u. We call $[u, v]$ *corner edge* w.r.t. (u, u') and (v, v') if and only if $R_{u,v}$ has no vertices of $\Gamma(G)$ in its interior.

▶ **Property 9.** *In a PMCM-drawing $\Gamma(G)$ of an optimal k-planar graph G any corner edge $[u, v]$ is a potential edge.*

Proof. By the definition of potential edges, the property holds when $u \neq v$. Otherwise, $[u, v]$ is a self-loop; see Fig. 4b. If the property does not hold, then $[u, v]$ is a self-loop with no vertices either in its interior or in its exterior contradicting Lemma 2. ◀

We say that u and v form a *side pair* if and only if there exist edges (u, u') and (v, v') for some u' and v' such that they both cross a third edge (w, w') in $\Gamma(G)$ and $(u, u') \neq (v, v')$; see Figs. 4c–4d. Let c and c' be the crossing points of (u, u') and (v, v') with (w, w'). Assume w.l.o.g. that c and c' appear in this order along (w, w') from w to w'. Also assume that the edge-segment of (u, u') between u and c is on the same side of (w, w') as the edge-segment of (v, v') between v and c'; see Fig. 4c. Then, any Jordan curve $[u, v]$ joining u and v induces a region $R_{u,v}$ that is defined by the walk along the edge-segment of (u, u') from u to c, the edge segment of (w, w') from c to c', the edge segment of (v, v') from c' to v and the curve $[u, v]$ from v to u. We call $[u, v]$ *side-edge* w.r.t. (u, u') and (v, v') if and only if $R_{u,v}$ has no vertices of $\Gamma(G)$ in its interior. Since by Properties 7 and 8 edges (u, u') and (v, v') cannot cross with each other (as they both cross (w, w')), it follows that region $R_{u,v}$ is well-defined. Symmetrically we define region $R_{u',v'}$ and side-edge $[u', v']$ w.r.t. (u, u') and (v, v').

▶ **Property 10.** *In a PMCM-drawing $\Gamma(G)$ of an optimal k-planar graph G with $k \in \{2, 3\}$ at least one of the side-edges $[u, v]$, $[u', v']$ is a potential edge.*

Proof. Note that since edges (u, u'), (v, v') and (w, w') do not mutually cross, curves $[u, v]$ and $[u', v']$ cannot cross themselves. Assume to the contrary that neither $[u, v]$ nor $[u', v']$ are potential edges. This implies that $u = v$, $u' = v'$ and both $[u, v]$ and $[u', v']$ are self-loops that have no vertices in their interiors or their exteriors. Fig. 4e illustrates the case where both $[u, v]$ and $[u', v']$ are self-loops with no vertices in their interiors; the other cases are similar. It is not hard to see that (u, u') and (v, v') are homotopic side-edges; a contradiction. ◀

Edges (u, u') and (v, v') are called *side-apart* if and only if both side-edges $[u, v]$ and $[u', v']$ are potential edges.

4 Characterization of optimal 2-planar graphs

In this section we examine some more structural properties of optimal 2-planar graphs in order to derive their characterization (see Theorem 9).

▶ **Lemma 5.** *Let $\Gamma(G)$ be a PMCM-drawing of an optimal 2-planar graph G. Any edge that is crossed twice in $\Gamma(G)$ is a chord of a true-planar 5-cycle in $\Gamma(G)$.*

Proof. Let (u, v) be an edge that is crossed twice in $\Gamma(G)$ by (u', v') and (u'', v'') at points c and c'. By Property 5 edges (u', v') and (u'', v'') are not identical. We assume w.l.o.g. that c and c' appear in this order along (u, v) from vertex u to vertex v. We also assume that the edge-segment of (u', v') between u' and c is on the same side of edge (u, v) as the edge-segment of (u'', v'') between u'' and c'; refer to Fig. 5a. By Property 9 corner edges $[u, u']$, $[u, v']$, $[v, u'']$ and $[v, v'']$ are potential edges. By Property 10 at least one of side-edges $[u', u'']$ and $[v', v'']$ is a potential edge. Assume w.l.o.g. that $[v', v'']$ is a potential edge.

If $[u', u'']$ is a potential edge, vertices u, v', v'', v, u'' and u' define a potential empty cycle $\mathcal{C}$ on six vertices (shaded in gray in Fig. 5b). Edges (u, v), (u', v') and (u'', v'') are drawn in the interior of $\mathcal{C}$, and there exist at most two other edges that cross (u', v') or (u'', v''). In total there exist at most five edges that have an edge-segment within $\mathcal{C}$. However, in the interior of $\mathcal{C}$ one can draw six chords as in Fig. 2b without deviating 2-planarity. By Lemma 3.(i) for $\kappa + \lambda \leq 5$ and $\mu = 6$, it follows that G is not optimal; a contradiction.

If $[u', u'']$ is not a potential edge, $[u', u'']$ is a homotopic self-loop and vertices u, v', v'', v and u' define a potential empty cycle $\mathcal{C}$ on five vertices (gray-shaded in Fig. 5c). In the interior of $\mathcal{C}$ one can draw five chords as in Fig. 2a without deviating 2-planarity. By Lemma 3.(ii), for $\kappa + \lambda \leq 5$ and $\mu = 5$, all boundary edges of $\mathcal{C}$ exist in $\Gamma(G)$ and $\kappa + \lambda = 5$.

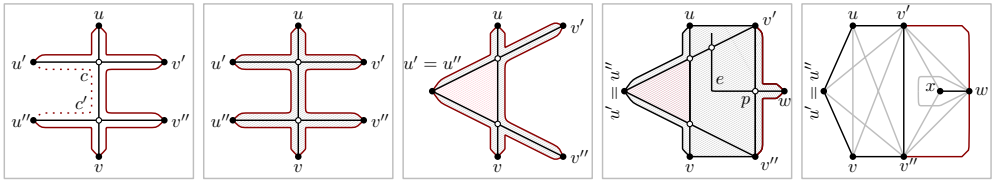


Figure 5 Different configurations used in Lemma 5.

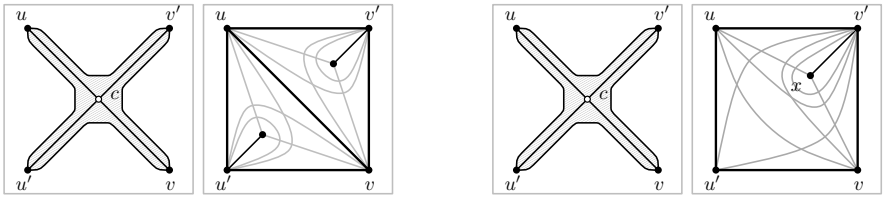


Figure 6 Different configurations used in: (a–b) Lemma 6, and (c–d) Lemma 14.

So, there exist two edges (other than (u, v)), say e and e', that cross (u', v') and (u'', v'') respectively.

If $\mathcal{C}$ is a true-planar 5-cycle in $\Gamma(G)$ the lemma holds. If not, e or e' crosses a boundary edge of $\mathcal{C}$. Suppose w.l.o.g. that e crosses (v', v'') of $\mathcal{C}$ at point p and let w and w' be the endpoints of e. Observe that e already has two crossings in $\Gamma(G)$. By 2-planarity, either the edge-segment of (w, w') between w and p or the one between w' and p is drawn completely in the exterior of $\mathcal{C}$. Suppose w.l.o.g. that this edge-segment is the former one. Then vertices v', w and v'' define a potential empty cycle $\mathcal{C}'$ on three vertices; see Fig. 5d. We proceed as follows: We remove edges (u, v), (u', v'), (u'', v''), e and e' and replace them with five chords drawn in the interior of $\mathcal{C}$ (as in Fig. 5e). The derived graph G' has the same number of edges as G. However, $\mathcal{C}'$ becomes a true-planar 3-cycle in G', contradicting Property 2. ◀

By Lemma 5, any edge of G that is crossed twice in $\Gamma(G)$ is a chord of a true-planar 5-cycle. The following lemma states that there are no edges with only one crossing in $\Gamma(G)$.

▶ **Lemma 6.** *Let $\Gamma(G)$ be a PMCM-drawing of an optimal 2-planar graph G. Then, every edge of $\Gamma(G)$ is either true-planar or has exactly two crossings.*

Proof. As shown in Lemma 5, for an edge e that is crossed twice in $\Gamma(G)$, both edges that cross e also have two crossings in $\Gamma(G)$. So, the crossing component $\mathcal{X}(e)$ consists exclusively of edges with two crossings. This implies that if (u, v) and (u', v') cross in $\Gamma(G)$ and (u, v) has only one crossing, then the same holds for (u', v'). Vertices u, v', v and u' define a potential empty cycle $\mathcal{C}$ on four vertices (gray-shaded in Fig. 6a). Since (u, v) and (u', v') have only one crossing each, the boundary of $\mathcal{C}$ exists in $\Gamma(G)$ and are true-planar edges. We proceed by removing (u', v'). $\mathcal{C}$ is split into two true-planar 3-cycles; see Fig. 6b. In both of them, we plug the 2-planar pattern of Fig. 2d. In total, we removed one edge and added two vertices and 12 edges, without creating any homotopic parallel edges or self-loops. So, if G has n vertices and m edges, the derived graph G' is 2-planar and has $n' = n + 2$ vertices and $m' = m + 11 = 5n' - 9$, i.e., G' has more edges than allowed; a contradiction. ◀

▶ **Lemma 7.** *The true-planar skeleton $\Pi(G)$ of a PMCM-drawing $\Gamma(G)$ of an optimal 2-planar graph is connected.*

Proof. Assume to the contrary that $\Pi(G)$ is not connected. Let H be a connected component of $\Pi(G)$. By Property 1 either there exists an edge (u, v) with $u \in H$ and $v \in G \setminus H$, or two crossing edges $e_1 \in H$ and $e_2 \in G \setminus H$. In the first case, (u, v) is not true-planar. By Lemma 5, there exists a true-planar 5-cycle with (u, v) as chord; a contradiction. In the second case, e_1 and e_2 belong to the same crossing component and by Lemma 5, there exists a true-planar 5-cycle with e_1 and e_2 as chords; a contradiction. ◀

▶ **Lemma 8.** *The true-planar skeleton $\Pi(G)$ of a PMCM-drawing $\Gamma(G)$ of an optimal 2-planar graph G contains only faces of length 5, each containing 5 crossing edges.*

Proof. Since $\Pi(G)$ is connected (by Lemma 7), all its faces are connected. By Lemmas 5 and 6, all crossing edges are chords of true-planar 5-cycles. We claim that $\Pi(G)$ has no chordless faces. First, $\Pi(G)$ has no chordless face of size ≥ 4, as otherwise one could add in its interior a chord, contradicting the optimality of G. By Property 2, $\Pi(G)$ contains no faces of length 3. Faces of length 1 or 2 correspond to homotopic self-loops and parallel edges. ◀

We are now ready to state the main theorem of this section.

▶ **Theorem 9.** *A graph G is optimal 2-planar if and only if G admits a drawing $\Gamma(G)$ without homotopic parallel edges and self-loops, such that the true-planar skeleton $\Pi(G)$ of $\Gamma(G)$ spans all vertices of G, it contains only faces of length 5 (that are not necessarily simple), and each face of $\Pi(G)$ has 5 crossing edges in its interior in $\Gamma(G)$.*

Proof. For the forward direction, consider an optimal 2-planar graph G. By Lemma 8, the true-planar skeleton $\Pi(G)$ of its 2-planar PMCM-drawing $\Gamma(G)$ contains only faces of length 5 each containing 5 crossing edges in its interior. Since the endpoints of two crossing edges are within a true-planar 5-cycle (by Lemmas 5 and 6) and since $\Pi(G)$ is connected (by Lemma 7), $\Pi(G)$ spans all vertices of G. So, the proof of this direction is complete.

For the reverse direction, denote by n, m and f the number of vertices, edges and faces of $\Pi(G)$. Since $\Pi(G)$ spans all vertices of G, it suffices to prove that G has exactly $5n-10$ edges. The fact that $\Pi(G)$ contains only faces of length 5 implies that $5f = 2m$. By Euler's formula for planar graphs, we have $m = 5(n-2)/3$ and $f = 2(n-2)/3$. Since each face of $\Pi(G)$ contains exactly 5 crossing edges, the total number of edges of G equals $m+5f = 5n-10$. ◀

5 Characterization of optimal 3-planar graphs

In this section we explore structural properties of optimal 3-planar graphs. Following similar arguments as in Section 4 we derive their characterizations (see Theorem 17).

▶ **Lemma 10.** *Let $\Gamma(G)$ be a PMCM-drawing of an optimal 3-planar graph G, and suppose that there exists a potential empty cycle $\mathcal{C}$ of 6 vertices in $\Gamma(G)$, such that the potential boundary edges of $\mathcal{C}$ exist in $\Gamma(G)$. Let $E_{\mathcal{C}}$ be the set of edge-segments within $\mathcal{C}$. If the following conditions C.1 and C.2 hold, then $\mathcal{C}$ is an empty true-planar 6-cycle in $\Gamma(G)$ and all edges with edge-segments in $E_{\mathcal{C}}$ are drawn as chords in its interior.*

(C.1) *$|E_{\mathcal{C}}| \leq 8$, and,*

(C.2) *every edge-segment of $E_{\mathcal{C}}$ has at least one crossing in the interior of $\mathcal{C}$.*

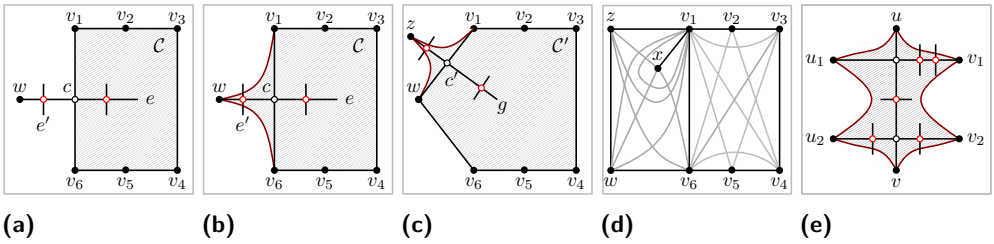


Figure 7 Different configurations used in (a–d) Lemma 10, (e) Lemma 11.

Proof Sketch. We start with the following observation: If e is an edge of G, then due to 3-planarity at most one edge-segment of e belongs to $E_{\mathcal{C}}$; if $E_{\mathcal{C}}$ contains at least two edge-segments of e, then we claim that e has at least four crossings. By C.2 each of the two edge-segments of e contributes one crossing to e. Since $\mathcal{C}$ is empty and contains two edge-segments of e, edge e exists and enters $\mathcal{C}$. Hence, e has two more crossings, summing up to a total of at least four.

Let $v_1, \ldots, v_6$ be the vertices of $\mathcal{C}$. If all edges with edge-segments in $E_{\mathcal{C}}$ completely lie in $\mathcal{C}$, then $\mathcal{C}$ is a true-planar 6-cycle and the lemma trivially holds. Otherwise, there is at least one edge e with an edge-segment in $E_{\mathcal{C}}$, that crosses a boundary edge of $\mathcal{C}$. W.l.o.g. we can assume that e crosses (v_1, v_6) of $\mathcal{C}$ at point c (refer to Fig. 7a). If w and w' are the two endpoints of e, then by the observation we made at the beginning of the proof it follows that either the edge-segment of (w, w') between w and c or the one between c and w' is drawn completely in the exterior of $\mathcal{C}$ (as otherwise e would have at least two edge-segments in $E_{\mathcal{C}}$). W.l.o.g. assume that this is the edge-segment between w and c. Then, corner edges $[v_1, w]$ and $[w, v_6]$ are potential edges (by Property 9).

Recall that e has one crossing in the interior of $\mathcal{C}$ (follows from C.2) and one more crossing with edge (v_1, v_6). By 3-planarity, e may have at most one more crossing, say with edge e'. Note that e' may or may not have an edge-segment in $E_{\mathcal{C}}$. Vertices w, v_1, …, v_6 define a potential empty cycle $\mathcal{C}'$ on 7 vertices; see Fig. 7b. The set $E_{\mathcal{C}'}$ of edge-segments within $\mathcal{C}'$ contains all edge-segments of $E_{\mathcal{C}}$ (i.e., $E_{\mathcal{C}} \subseteq E_{\mathcal{C}'}$) plus at most two additional edge-segments: the one defined by edge (v_1, v_6), and possibly an edge-segment of e'. Hence $|E_{\mathcal{C}'}| \leq 10$. We now make some observations in the form of claims, which we formally prove in [20].

▶ **Claim 4.** *A PMCM-drawing $\Gamma(G)$ of an optimal 3-planar graph G is quasi-planar.*

▶ **Claim 5.** *At least one edge with an edge-segment in $E_{\mathcal{C}'}$ crosses one edge of $\mathcal{C}'$.*

By Claim 5, there is an edge g that crosses a boundary edge, say $[w, v_1]$, of $\mathcal{C}'$ at point c'; refer to Fig. 7c.

▶ **Claim 6.** *All boundary edges of $\mathcal{C}'$ exist in $\Gamma(G)$; g has one crossing in the interior of $\mathcal{C}'$.*

We follow an analogous approach to the one we used for expanding $\mathcal{C}$ (that has 6 vertices) to $\mathcal{C}'$ (that has 7 vertices). We can find an endpoint of g, say z, such that w, z, v_1, v_2, …, v_6 define a potential empty cycle $\mathcal{C}''$ on 8 vertices. Furthermore, the set $E_{\mathcal{C}''}$ of edge-segments within $\mathcal{C}''$ has at most 12 elements (at most two more than $E_{\mathcal{C}'}$). We proceed by removing all edges with an edge-segment in $E_{\mathcal{C}''}$ and split $\mathcal{C}''$ into two true-planar cycles of length 6 and 4, by adding true-planar chord (v_1, v_6); see Fig. 7d. In the interior of the 6-cycle, we add 8 crossing edges as in Fig. 2c. In the interior of the 4-cycle, we add a vertex x with a true planar edge (v_1, x). Vertices v_1, x, v_1, v_6, w and z define a new potential empty cycle on 6 vertices, allowing us to add 8 more crossing edges. In total, we removed at most 12

edges, added a vertex and 18 edges. If n and m are the number of vertices and edges of G, then the derived graph G' has $n' = n + 1$ vertices and $m' \geq m + 6$ edges. The last equation gives $m' \geq 5.5n' - 10.5$, i.e. G' has more edges than allowed; a contradiction. ◀

Let (u, v) be an edge of G that is crossed by two edges (u_1, v_1) and (u_2, v_2) in $\Gamma(G)$ at points c_1 and c_2. By Property 6 edges (u_1, v_1) and (u_2, v_2) are not identical. We assume w.l.o.g. that c_1 and c_2 appear in this order along (u, v) from u to v. We also assume that the edge-segment of (u_1, v_1) between u_1 and c is on the same side of edge (u, v) as the edge-segment of (u_2, v_2) between u_2 and c_2; refer to Fig. 7e. Vertices u_1, u_2 and v_1, v_2 define two side pairs. By Property 10, at least one of side-edges $[u_1, u_2]$ and $[v_1, v_2]$ is a potential edge of $\Gamma(G)$. Recall that if both side-edges $[u_1, u_2]$ and $[v_1, v_2]$ are potential edges of $\Gamma(G)$, then edges (u_1, v_1) and (u_2, v_2) are called side-apart.

▶ **Lemma 11.** *Let $\Gamma(G)$ be a PMCM-drawing of an optimal 3-planar graph G. If (u, v) is crossed by side-apart edges (u_1, v_1) and (u_2, v_2) in $\Gamma(G)$, then (u, v) is a chord of an empty true-planar 6-cycle.*

Proof. Refer to Fig. 7e. Since (u_1, v_1) and (u_2, v_2) are side-apart, side-edges $[u_1, u_2]$ and $[v_1, v_2]$ are potential edges. By Property 9, corner edges $[u, u_1]$, $[u, v_1]$, $[u, u_2]$ and $[v, v_2]$ are potential edges. Hence, vertices u, v_1, v_2, v, u_2 and u_1 define a potential empty cycle $\mathcal{C}$ on six vertices (gray-shaded in Fig. 7e). Edges (u, v), (u_1, v_1) and (u_2, v_2) are drawn completely in the interior of $\mathcal{C}$ and there exist at most five other edges either drawn in the interior of $\mathcal{C}$ or crossing its boundary: at most one that crosses (u, v), and at most four others that cross (u_1, v_1) and (u_2, v_2). Since we can draw eight chords in the interior of $\mathcal{C}$ as in Fig. 2c, by Lemma 3.(ii), for $\kappa + \lambda \leq 8$ and $\mu = 8$, all boundary edges of $\mathcal{C}$ exist in $\Gamma(G)$. Furthermore $\kappa + \lambda = 8$ must hold. Note that the set $E_{\mathcal{C}}$ of edge-segments within $\mathcal{C}$ contains only edge-segments of these $\kappa + \lambda$ edges. Also, these 8 edges have exactly one edge-segment within $\mathcal{C}$ that is crossed in the interior of $\mathcal{C}$. Hence, C.1 and C.2 of Lemma 10 are satisfied and there exists an empty true-planar 6-cycle that has (u, v) as chord. ◀

▶ **Lemma 12.** *Let $\Gamma(G)$ be a PMCM-drawing of an optimal 3-planar graph G. If e is crossed by two side-apart edges in $\Gamma(G)$, $\mathcal{X}(e)$ consists of chords of an empty true-planar 6-cycle.*

Proof. The lemma follows by the observation that since e is a chord of an empty true-planar 6-cycle (by Lemma 11), all edges of $\mathcal{X}(e)$ are also chords of this 6-cycle. ◀

▶ **Lemma 13.** *Let $\Gamma(G)$ be a PMCM-drawing of an optimal 3-planar graph G. Any edge that is crossed three times in $\Gamma(G)$ is a chord of an empty true-planar 6-cycle in $\Gamma(G)$.*

Proof Sketch. We argue that the preconditions C.1 and C.2 of Lemma 10 are fulfilled, which implies the presence of the empty true-planar 6-cycle in $\Gamma(G)$. For more details refer to [20]. ◀

We next consider edges of G that have two or fewer crossings in $\Gamma(G)$.

▶ **Lemma 14.** *Let $\Gamma(G)$ be a PMCM-drawing of an optimal 3-planar graph G and let $\mathcal{X}$ be a crossing component of $\Gamma(G)$. Then, there is at least one edge in $\mathcal{X}$ that has three crossings.*

Proof. Assume to the contrary that there exists a crossing component $\mathcal{X}$ where all edges have at most two crossings. Assume first that $\mathcal{X}$ does not contain an edge with two crossings. Then, $|\mathcal{X}| = 2$. W.l.o.g. assume that $\mathcal{X} = \{e, e'\}$. The four endpoints of e and e' define a potential empty cycle $\mathcal{C}$ on 4 vertices; see Fig. 6c. Since e and e' have only one crossing each, the boundary of $\mathcal{C}$ exist in $\Gamma(G)$ and is true-planar. Note that there are no other edges

passing through the interior of $\mathcal{C}$. We proceed by removing e and e' and replace them with the 3-planar pattern of Fig. 6d, i.e., we add a vertex x in the interior of $\mathcal{C}$ and true-planar edge (v', x). Vertices u, v', x, v', v and u' define a potential empty cycle on six vertices, and we can add 8 edges in its interior as in Fig. 2c. If G has n vertices and m edges, the derived graph G' has $n' = n + 1$ vertices and $m' = m - 2 + 8$ edges. Then, G' is 3-planar and has $m' = 5.5n' - 10.5$ edges, i.e., G' has more edges than allowed by 3-planarity; a contradiction.

Assume now that there exists an edge $(u, v) \in \mathcal{X}$ which has two crossings with edges (u_1, v_1) and (u_2, v_2). By Lemma 11, (u_1, v_1) and (u_2, v_2) are not side-apart. Since all edges in $\mathcal{X}$ have at most two crossings, adopting the proof of Lemma 5 we can prove that the endpoints of (u, v), (u', v') and (u'', v'') define a potential empty cycle $\mathcal{C}$ on five vertices, with at most five edges passing through its interior. We proceed by redrawing these five edges as chords of $\mathcal{C}$ (as in Fig. 2a) and all its boundary edges are true-planar in the new drawing. The derived graph is optimal, since it has at least as many edges as G. Observe, however, that $\mathcal{C}$ becomes a true-planar 5-cycle in the new drawing; a contradiction to Property 3. ◀

The proofs of Lemmas 15 and 16 are similar to the ones of Lemmas 7 and 8; see also [20].

▶ **Lemma 15.** *The true planar skeleton $\Pi(G)$ of a PMCM-drawing $\Gamma(G)$ of an optimal 3-planar graph is connected.*

▶ **Lemma 16.** *The true-planar skeleton $\Pi(G)$ of a PMCM-drawing $\Gamma(G)$ of an optimal 3-planar graph G contains only faces of length 6, each containing 8 crossing edges in $\Gamma(G)$.*

We say that a chord of a $2s$-cycle is a *middle chord* if the two paths along the cycle connecting its endpoints both have length s. We now state the main theorem of this section.

▶ **Theorem 17.** *A graph G is optimal 3-planar if and only if G admits a drawing $\Gamma(G)$ without homotopic parallel edges and self-loops, such that the true-planar skeleton $\Pi(G)$ of $\Gamma(G)$ spans all vertices of G, it contains only faces of length 6 (that are not necessarily simple), and each face of $\Pi(G)$ has 8 crossing edges in its interior in $\Gamma(G)$ such that one of the middle chords is missing.*

Proof. For the forward direction, consider an optimal 3-planar graph G. By Lemma 16, the true-planar skeleton $\Pi(G)$ of its 3-planar PMCM-drawing $\Gamma(G)$ contains only faces of length 6, each of which has 8 edges in its interior in $\Gamma(G)$. By Property 8, one of the three middle chords of each face of $\Pi(G)$ cannot be present. Since the endpoints of two crossing edges are within a true-planar 6-cycle (by Lemmas 13 and 14) and since $\Pi(G)$ is connected (by Lemma 15), $\Pi(G)$ spans all vertices of G, which completes the proof of this direction.

For the reverse direction, denote by n, m and f the number of vertices, edges and faces of $\Pi(G)$. Since $\Pi(G)$ spans all vertices of G, it suffices to prove that G has exactly $5.5n - 11$ edges. The fact that $\Pi(G)$ contains only faces of length 6 implies that $6f = 2m$. By Euler's formula, we have $m = 3(n-2)/2$ and $f = (n-2)/2$. Since each face of $\Pi(G)$ contains exactly 8 crossing edges, the total number of edge of G equals to $m + 8f = 5.5n - 11$. ◀

6 Further Insights and Open Problems

The following corollaries are consequences of our new characterizations; for details see [20]. The definitions of bar 1-visibility and fan-planarity can be found in [13, 10] and [17, 7].

▶ **Corollary 18.** *Simple 3-planar graphs have at most $5.5n - 11.5$ edges.*

▶ **Corollary 19.** *Simple optimal 2-planar graphs admit bar 1-visibility representations.*

▶ **Corollary 20.** *Simple optimal 2-planar graphs are optimal fan-planar.*

Our characterizations naturally lead to many open questions; we only name a few.

- What is the complexity of the recognition problem for optimal 2- and 3-planar graphs?
- What is the exact upper bound on the number of edges of simple optimal 3-planar graphs? We conjecture that they do not have more than $5.5n - 15$ edges.
- Theorems 9 and 17 imply that optimal 2- and 3-planar graphs have a fully triangulated planar subgraph. Can this property be proved for optimal 4-planar or more in general for optimal k-planar graphs? Proving this property would be useful to derive better density bounds for $k \geq 4$.
- By Properties 7 and 8, optimal 2- and 3-planar graphs are quasi-planar. Angelini et al. [4] proved that every simple k-planar graph is $(k+1)$-quasi planar for $k \geq 3$ (i.e., it can be drawn with no $k+1$ pairwise crossing edges). Our results about optimal 2-planar and even more about optimal 3-planar graphs give indications that the result by Angelini et al. [4] may hold also for $k = 2$.
- We found a RAC drawing (i.e., a drawing in which all crossing edges form right angles) with at most one bend per edge for the optimal 2-planar graph having the dodecahedron as its true-planar structure. Is this generalizable to all simple optimal 2-planar graphs?

Acknowledgment. The authors thank F. Montecchiani for useful discussions on the bar 1-visibility of optimal 2-planar graphs.

References

1. E. Ackerman. On topological graphs with at most four crossings per edge. *CoRR*, 1509.01932, 2015.
2. M. Ajtai, V. Chvátal, M. Newborn, and E. Szemerédi. Crossing-free subgraphs. In *Theory and Practice of Combinatorics*, pages 9–12. North-Holland Mathematics Studies, 1982.
3. N. Alon and P. Erdős. Disjoint edges in geometric graphs. *Discrete & Computational Geometry*, 4:287–290, 1989. doi:10.1007/BF02187731.
4. P. Angelini, M. A. Bekos, F.-J. Brandenburg, G. Da Lozzo, G. Di Battista, W. Didimo, G. Liotta, F. Montecchiani, and I. Rutter. On the relationship between k-planar and k-quasi planar graphs. *CoRR*, 1702.08716, 2017.
5. M. A. Bekos, T. Bruckdorfer, M. Kaufmann, and C. N. Raftopoulou. 1-planar graphs have constant book thickness. In *ESA*, volume 9294 of *LNCS*, pages 130–141. Springer, 2015. doi:10.1007/978-3-662-48350-3_12.
6. M. A. Bekos, M. Kaufmann, and C. N. Raftopoulou. On the density of non-simple 3-planar graphs. In *Graph Drawing*, volume 9801 of *LNCS*, pages 344–356. Springer, 2016. doi:10.1007/978-3-319-50106-2_27.
7. C. Binucci, E. Di Giacomo, W. Didimo, F. Montecchiani, M. Patrignani, A. Symvonis, and I. G. Tollis. Fan-planarity: Properties and complexity. *Theor. Comp. Sci.*, 589:76–86, 2015. doi:10.1016/j.tcs.2015.04.020.
8. R. Bodendiek, H. Schumacher, and K. Wagner. Über 1-optimale Graphen. *Mathematische Nachrichten*, 117(1):323–339, 1984.
9. O. Borodin. A new proof of the 6 color theorem. *J. of Graph Theory*, 19(4):507–521, 1995. doi:10.1002/jgt.3190190406.
10. F.-J. Brandenburg. 1-visibility representations of 1-planar graphs. *J. Graph Algorithms Appl.*, 18(3):421–438, 2014. doi:10.7155/jgaa.00330.
11. F.-J. Brandenburg. Recognizing optimal 1-planar graphs in linear time. *CoRR*, 1602.08022, 2016.

12 O. Cheong, S. Har-Peled, H. Kim, and H.-S. Kim. On the number of edges of fan-crossing free graphs. *Algorithmica*, 73(4):673–695, 2015. doi:10.1007/s00453-014-9935-z.

13 A. M. Dean, W. S. Evans, E. Gethner, J. D. Laison, M. A. Safari, and W. T. Trotter. Bar k-visibility graphs. *J. Graph Algorithms Appl.*, 11(1):45–59, 2007.

14 E. Di Giacomo, W. Didimo, G. Liotta, H. Meijer, and S. K. Wismath. Planar and quasi-planar simultaneous geometric embedding. *Comput. J.*, 58(11):3126–3140, 2015. doi:10.1093/comjnl/bxv048.

15 W. Didimo, P. Eades, and G. Liotta. Drawing graphs with right angle crossings. *Theor. Comp. Sci.*, 412(39):5156–5166, 2011.

16 S. Hong, P. Eades, G. Liotta, and S.-H. Poon. Fáry's theorem for 1-planar graphs. In *COCOON*, volume 7434 of *LNCS*, pages 335–346. Springer, 2012. doi:10.1007/978-3-642-32241-9_29.

17 M. Kaufmann and T. Ueckerdt. The density of fan-planar graphs. *CoRR*, 1403.6184, 2014.

18 T. Leighton. *Complexity Issues in VLSI. Foundations of Computing Series.* MIT Press., 1983.

19 L. Lovász, J. Pach, and M. Szegedy. On Conway's thrackle conjecture. *Discrete & Computational Geometry*, 18(4):369–376, 1997. doi:10.1007/PL00009322.

20 A. Bekos M. M. Kaufmann, and N. Raftopoulou C.˙ On optimal 2- and 3-planar graphs. *CoRR*, 1703.06526, 2017.

21 J. Pach, R. Radoičić, G. Tardos, and G. Tóth. Improving the crossing lemma by finding more crossings in sparse graphs. *Discrete & Computational Geometry*, 36(4):527–552, 2006. doi:10.1007/s00454-006-1264-9.

22 J. Pach and G. Tóth. Graphs drawn with few crossings per edge. *Combinatorica*, 17(3):427–439, 1997. doi:10.1007/BF01215922.

23 G. Ringel. Ein Sechsfarbenproblem auf der Kugel. *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg (in German)*, 29:107–117, 1965.

24 Y. Suzuki. Re-embeddings of maximum 1-planar graphs. *SIAM J. Discrete Math.*, 24(4):1527–1540, 2010. doi:10.1137/090746835.

25 P. Turán. A note of welcome. *J. of Graph Theory*, 1(1):7–9, 1977. doi:10.1002/jgt.3190010105.

Reachability in a Planar Subdivision with Direction Constraints*

Daniel Binham[1], Pedro Machado Manhães de Castro[2], and Antoine Vigneron[3]

1 **Visual Computing Center, KAUST, Thuwal, Saudi Arabia**
ringscore@yahoo.com
2 **Centro de Informática, Universidade Federal de Pernambuco, Recife, Brazil**
pmmc@cin.ufpe.br
3 **School of Electrical and Computer Engineering, UNIST, Ulsan, Republic of Korea**
antoine@unist.ac.kr

Abstract

Given a planar subdivision with n vertices, each face having a cone of possible directions of travel, our goal is to decide which vertices of the subdivision can be reached from a given starting point s. We give an $O(n \log n)$-time algorithm for this problem, as well as an $\Omega(n \log n)$ lower bound in the algebraic computation tree model. We prove that the generalization where two cones of directions per face are allowed is NP-hard.

1998 ACM Subject Classification I.3.5 Computational Geometry and Object Modeling

Keywords and phrases Design and analysis of geometric algorithms, Path planning, Reachability

Digital Object Identifier 10.4230/LIPIcs.SoCG.2017.17

1 Introduction

We consider a motion planning problem where a point robot moves within a planar subdivision, with constraints on its direction of travel. Within each face of the subdivision, there is a cone of possible directions of travel, and we want to decide which vertices are reachable from a given starting position. (See Figure 1a.)

This type of constraints appear, for instance, for motion planning in the presence of flows [6]. In this model, a vehicle moves within a flow field, say wind or current. If the speed of the vehicle is less than the speed of the flow, then it can only travel in a cone of directions, with axis parallel to the direction of the current, and whose angle depends on the ratio between the speed of the robot and the speed of the current. (See Figure 1b.)

Our results. Our main result is an $O(n \log n)$-time algorithm to compute all the vertices that are reachable from a given source point s, where n is the size of the input subdivision, and each face has a cone of possible directions of travel (Section 6). We also give a matching $\Omega(n \log n)$ lower bound in the algebraic computation tree model. This result is based on Ben-Or's topological lower bound [1], and holds even in the special case where only one direction of travel is given for each face. Finally, we prove that the generalization where

* D. Binham was supported by KAUST base funding, and A. Vigneron was supported by the 2016 Research Fund (1.160054.01) of UNIST (Ulsan National Institute of Science and Technology).

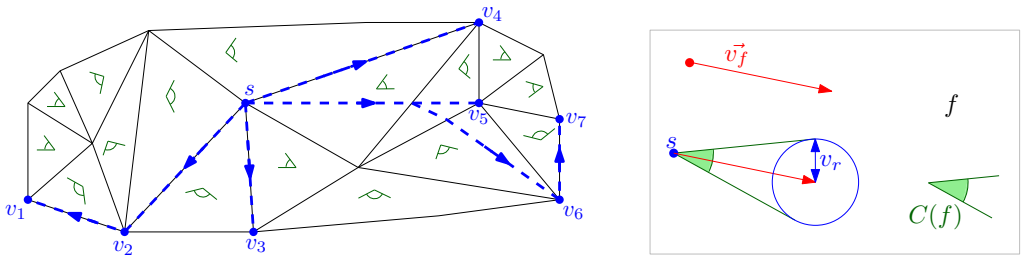


Figure 1 (a) The input to our reachability problem is the triangulation, the cone of direction in each face, and the starting point s. The output is the set of reachable vertices $\{v_1, v_2, v_3, v_4, v_5, v_6, v_7\}$. (b) Within the region f, the velocity of the flow is $\vec{v_f}$ and the control speed of the robot is v_r. The blue circle represents the points that can be reached from s in unit time. Hence, the possible directions of travel are given by the cone $C(f)$.

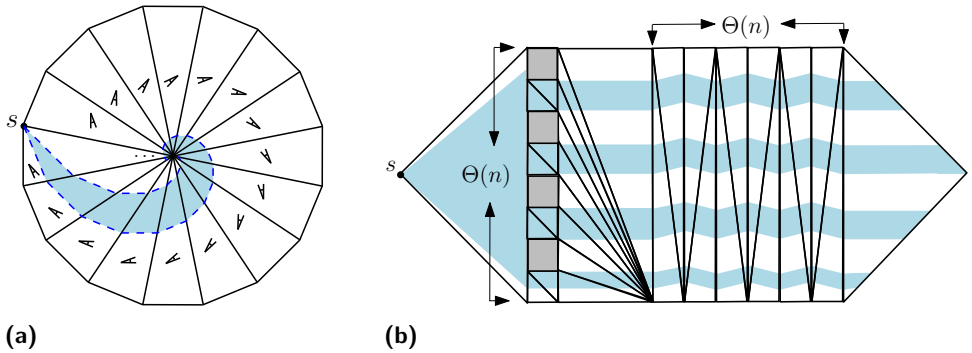


Figure 2 (a) The reachable region (shaded) is a spiral formed by an infinite sequence of blocks. (b) The reachable region has quadratic complexity, without any spiral.

two cones of directions per face are allowed is NP-hard. Our proof is a reduction from the partition problem, and it even holds when only two directions of travel are allowed throughout the subdivision.

A natural approach to compute all reachable vertices would be to construct the reachable region piece by piece, handling one face at a time. Unfortunately, this algorithm would not necessarily terminate as the direction constraints may force a path to follow a spiral with arbitrarily many edges (Figure 2a). Even when there are no such spirals, the complexity of the reachable region can still be quadratic (Figure 2b). So in order to achieve a near-linear running time, our algorithm uses efficient data structures to implicitly encode the boundary of the part of the reachable region that has already been constructed. This data structure is used to propagate a boundary path in $O(\log n)$ time, if it follows a previously constructed boundary. More details can be found in Section 6.

Comparison with previous work. The most directly related problem is to find a descending path (that is, a path that never goes up) between two points on a terrain. This is a special case of reachability under direction constraints: After projecting the terrain to horizontal, we get an instance of our problem where each face has a cone of possible directions which is a halfplane. De Berg and Van Kreveld [5] gave a data structure that can answer descending path reachability queries between two points in $O(\log n)$ time, after $O(n \log n)$ preprocessing time. Another related paper [4] shows how to compute a collection of n paths of steepest

descent in $O(n \log n)$ time [4]. This work uses a data structure similar to our data structure for recording beams (Section 6), but it uses it in a different way as it proceeds by sweeping a horizontal plane over the whole terrain. This approach cannot be applied to our problem, as there is no notion of elevation. Recently, Cheng and Jin [2] gave the first FPTAS for finding a shortest descending path between two points.

The problem of planning the movement in the presence of a flow was studied by Reif and Sun [6]. A point robot can apply a control velocity, with bounded norm, and each face of the triangulation has a flow of constant velocity (Figure 1b). They give an approximation algorithm for finding a shortest path between two points. However, it only applies when the control velocity is larger than the flow velocity, meaning that all directions of travel are possible. [1] Hence, their algorithm cannot be used to solve our problem, although it has the advantage of providing an approximate shortest path.

Sun and Reif also considered another anisotropic motion planning problem, where a wheeled robot travels on a terrain [7]. The mechanical constraints such as friction and steepness imply that some directions of travel are forbidden, and the speed depends on the direction. They present an approximation algorithm for the single source shortest path problem. This algorithm places Steiner points along the edges and searches the induced graph. The case where some directions of travel are forbidden is only briefly described [7, Theorem 4] and no time bound is given for this case. In any case, the number of Steiner points depends on several extra parameters, such as the minimum angle in the triangulation, or the length of the longest edge.

Cheng et al. considered approximate shortest path problems in anisotropic environments where the cost function within each face is a convex distance function [3]. This model allows different costs for different directions of travel, but again all directions must be allowed, so it does not solve our problem. They give an approximation algorithm whose running time depends on the ratio between the largest and the smallest speed in any direction, and the dependency on the input size is cubic.

In summary, we propose the first provably efficient algorithm to compute a path between two points in a planar subdivision, when there is one cone of possible directions of travel per face. Previous work on path planning with direction constraints either considered a special case where each cone of direction is a halfplane [2, 5], or did not provide any time bound [6, 7]. Unlike the algorithms in other anisotropic models [2, 3, 6, 7], our algorithm does not return an approximate shortest path, but it handles more general direction constraints, and it runs in near-linear time, regardless of the geometry of the input.

2 Notation and preliminary

Problem statement. We are given a planar triangulation $\mathcal{S}$ with n vertices. More precisely, $\mathcal{S}$ is a simplicial complex in $\mathbb{R}^2$. Each face f of $\mathcal{S}$ is a triangle, and is associated with a cone $C(f)$ of possible directions of travel. This cone is specified by a leftmost (clockwise) and rightmost (counterclockwise) direction $d_\ell(f) \in \mathbb{R}^2$ and $d_r(f) \in \mathbb{R}^2$. We assume that $C(f)$ is convex: Its opening angle is at most π. A direction (or vector) d is in $C(f)$ if $d = \lambda d_\ell(f) + \mu d_r(f)$ for some $\lambda, \mu \geq 0$. In addition, halfplanes bounded by lines through $(0, 0)$ are considered to be cones, as well as the whole plane $\mathbb{R}^2$, which is called the *full* cone of directions.

[1] The algorithm by Reif and Sun [6, Section 5] requires that the parameter ρ_r is larger than 1, which means that the control velocity is always larger than the flow velocity. In other words, all directions of travel are allowed. The speed depends on the direction, but it is always positive.

We denote by $\overline{pq}$ the directed closed line segment from point p to point q. We will abuse notation so that $\overline{pq} \in C(f)$ means that the vector $\overrightarrow{pq}$ is in $C(f)$. A segment $\overline{pq}$ in a face f is *feasible* if $\overline{pq} \in C(f)$. Let s and t be two points in this subdivision. A *feasible path* from s to t is a polyline whose edges are feasible segments. Given a starting point s, our goal is to find all the vertices of $\mathcal{S}$ that can be reached by a feasible path.

Model of computation. We assume that, given a point p on the boundary of a face f of $\mathcal{S}$, we can compute in constant time the points q and q' along the boundary of f that are in directions $d_\ell(f)$ and $d_r(f)$. In addition, we assume that we have at our disposal a constant-time logarithm function. It will help us handle spirals efficiently: We will be able to compute in $O(\log n)$ time the exit point of a spiral (Figure 11).

Notation. An *interval* is a closed segment along an edge of $\mathcal{S}$. We allow an interval to be a single point in the interior of an edge of $\mathcal{S}$, but vertices of $\mathcal{S}$ are not called intervals. So any interval u is contained in a unique edge of $\mathcal{S}$. This edge is denoted by $\text{edge}(u)$. A *full interval* is an edge of $\mathcal{S}$. In other words, a full interval u is such that $\text{edge}(u) = u$. An *extreme interval* is an interval $\overline{vq}$ such that v is a vertex of $\mathcal{S}$. In particular, a full interval is an extreme interval with respect to both of its endpoints. A *free interval* is an interval that is not extreme. In other words, a free interval is contained in the interior of an edge of $\mathcal{S}$.

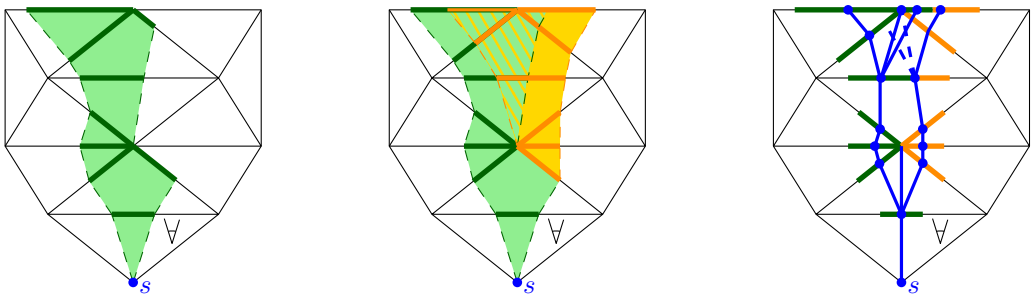
Let u be an interval on the boundary of a face f. We denote by $R(u, f)$ the set of points on the boundary of f that are reachable from u along a direction in $C(f)$. More precisely, $R(u, f)$ is the intersection of $u + C(f)$ with the boundary of f. An *image vertex* of (u, f) is a vertex of f that lies in $R(u, f)$. An *image interval* of (u, f) is a maximal segment of $R(u, f)$. In degenerate cases, an image interval can be a single point in the interior of an edge, but we do not count vertices of $\mathcal{S}$ as image intervals. The list of image vertices and intervals is denoted by $L(u, f)$. An interval u is *non-propagating* if $R(u, f) \subseteq u$. Otherwise, it is *propagating*. We may also say that u *propagates* into f.

Let π be a feasible path from s to t. As the cones of directions are convex, we can replace any subpath of π contained in a face f with a single edge $\overline{pq}$. So a path π can be specified by a sequence $p_1 f_1 p_2 f_2 \dots f_\ell p_{\ell+1}$ where $\overline{p_i p_{i+1}} \in C(f_i)$ and $f_i \neq f_{i+1}$ for all i. As this is the only relevant type of path for our problem, in order to alleviate notation, we will assume that all paths are of this form.

3 Overview

In this section, we present a brief description of our results and the approach used to prove them. We start with the algorithms.

The naive approach would be to compute the whole reachable region block by block, by recursively propagating intervals along the boundary of the faces of the subdivision. (See Figure 3.) One difficulty with this approach is that the blocks partially overlap, so the algorithm would do a lot of double-work, and it is not clear how to use planarity arguments in the analysis. So instead of constructing the whole reachable region, we construct the *skeleton* Ske (Figure 3c), which is a tree connecting the midpoints of the reachable intervals. While constructing this tree, we will prove that any new edge that crosses a previously constructed edge can be *pruned* without affecting the set of reachable nodes computed by the algorithm. Hence, we can ensure that the skeleton is a tree properly embedded in the plane, and we will be able to use planarity arguments in our proofs. For instance, it is easy to see that the



(a) (b) (c)

Figure 3 (a) Naive approach to compute the reachable region block by block. (b) After propagating another branch, some blocks overlap. (c) Our approach using the skeleton (thick, blue). Two edges are pruned (dashed), so that the skeleton remains a tree embedded in the plane.

skeleton can have only one branching node (i.e. degree at least 3) within each face of the subdivision, and thus the skeleton has a linear number of branchings.

We present in Section 4 a description of our first algorithm (Algorithm 1) to construct the skeleton edge by edge. In Section 5, we prove several properties of the skeleton constructed by Algorithm 1. Algorithm 1 is inefficient because it could enter an infinite loop when it encounters a spiral (Figure 2a), and even without spirals, the tree could have a quadratic number of edges (Figure 2b). Note that in both cases, the issue arises from long paths that cross the same sequence of edges: in the case of a spiral, the subsequence is repeated periodically, and in the second example, we have long, horizontal paths crossing the same sequence of edges.

We present in Section 6 an algorithm (Algorithm 2) that overcomes this difficulty using efficient data structures for handling *parallel beams*, that is, paths in the skeleton that cross the same sequence of edges of the triangulation. The idea is the following. Consider a path that crosses the sequence of edges $(e_1, e_2, \ldots, e_m)$ in their interiors. If x_1 is the coordinate of the path along e_1, then the coordinate x_m along e_m is given by a linear map, whose coefficients can be easily determined from the geometry of the faces and the cones of directions. We record these linear functions in a binary tree over $(e_1, \ldots, e_m)$, so that we can implicitly construct in $O(\log m)$ time any path through a subsequence $(e_i, \ldots, e_j)$, given its starting point x_i.

We record such data structures for the left side and the right side of each beam. (See Figure 4.) When a new beam B follows parallel and to the right of an already constructed beam C, it forms a new *tunnel*, which is the empty region between these two beams. We update the data structure in $O(\log m)$ time by first appending a subtree associated with C to the data structure for B. Then we create a single node for the tunnel, which is sufficient for our purpose, as any new beam entering the tunnel can only go parallel to B and C within this tunnel.

As we shall see, Algorithm 2 runs in $O(n \log n)$ time using this data structure. For instance, in the example of Figure 2b, the data structure for each one of the $\Theta(n)$-long horizontal beams can be constructed in $O(\log n)$ time, given the data structure for the beam immediately above it. The analysis relies on several observation on the structure of the skeleton. For instance, we show that it has $O(n)$ maximal tunnels and maximal beams, and that the total number of nodes in our data structures is $O(n)$. Spirals are handled in the same way as tunnels, as they can be seen as a special type of tunnels.

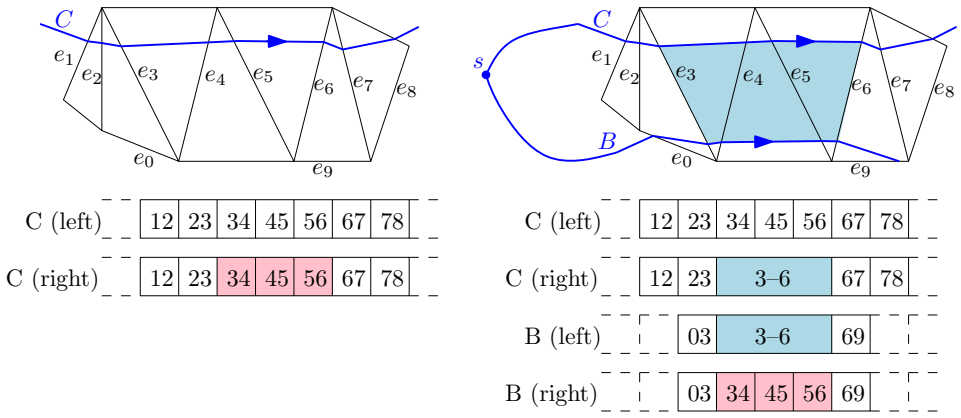


Figure 4 Data structure for beams. One node is created for the right side of C and the left side of B to represent the tunnel (blue). A subtree of the data structure for the right side of C (red) is deleted and inserted into the data structure for the right side of B.

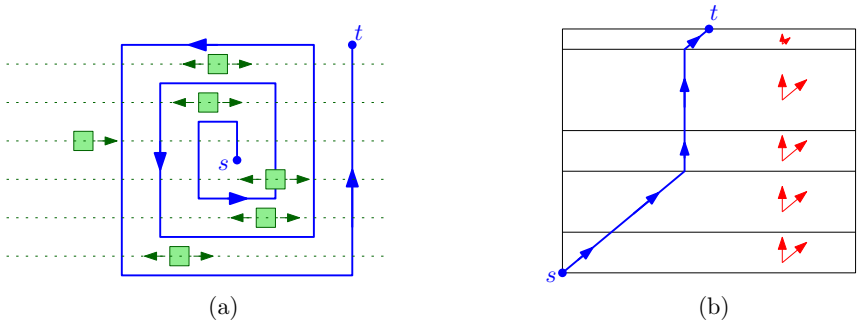


Figure 5 (a) $\Omega(n \log n)$ lower bound. (b) NP-hardness proof.

Finally, we show two hardness results. (Detailed proofs are omitted due to space limitation.) We first prove an $\Omega(n \log n)$ lower bound on our problem using Ben-Or's technique [1]. Figure 5a gives an outline of our construction: The direction constraints force the path to follow a spiral with $\Theta(n)$ edges. Then we place $\Theta(n)$ obstacles that can move left or right, so that the target point t can only be reached if no obstacle overlaps with the spiral. This problem has $n^{\Theta(n)}$ connected components, and hence it requires an algebraic computation tree of depth $\Omega(n \log n)$. Then we prove that the reachability problem where two cones of directions per face are allowed is NP-hard. In fact, our construction only requires that each face allows the directions $(0, 1)$ and $(1, 1)$. Our proof is a reduction to the partition problem: Given a set of integers, can it be partitioned into two subsets with same sum? The reduction is given in Figure 5b. The heights of the rectangles are the input integers, and the target point t is at the midpoint of the top edge. It can only be reached if the instance of the partition problem is positive.

4 First algorithm

In this section, we present our first algorithm (Algorithm 1), which recursively propagates reachable intervals or vertices to other reachable intervals or vertices that lie on the boundary of the same face. As Algorithm 1 constructs intervals one by one, it does not terminate if it encounters an infinite spiral. In Section 6, we present a faster version of this algorithm

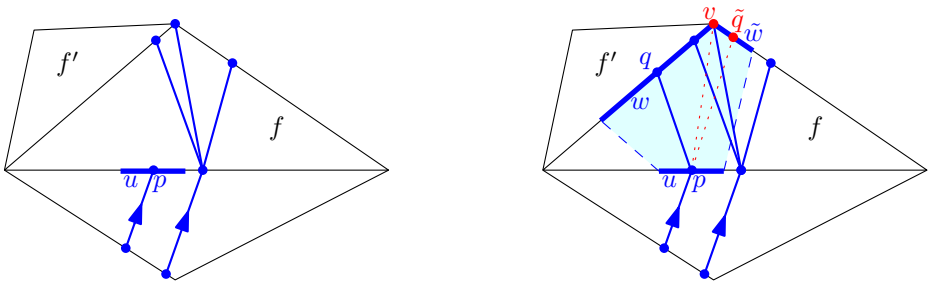


Figure 6 Propagating a pair (u, f), when u is an interval. (Left) The skeleton before propagating (u, f). (Right) When we propagate (u, f), the pair (w, f') is created, and v and $\tilde{w}$ are pruned.

(Algorithm 2) that computes all reachable vertices of $\mathcal{S}$ in $O(n \log n)$ time, using efficient data structures that allow us to implicitly construct a beam or a whole spiral in $O(\log n)$ time.

Algorithm 1 draws a directed tree Ske, called the Skeleton, whose nodes lie on edges of $\mathcal{S}$. In particular, these nodes are either vertices of $\mathcal{S}$ that are found to be reachable by our algorithm, or midpoints of reachable segments of edges of $\mathcal{S}$.

Description. Algorithm 1 *propagates* recursively pairs (u, f), where u is an interval or a vertex on the boundary of a face f. By propagating, we mean that we create children of the pair (u, f) that are of the form (w, f'), where w is an image vertex or interval in $L(u, f)$, and f' is a face other than f. Some of these pairs will be *pruned*, and thus not created. The pairs that have been created, but not yet propagated, are stored in a set $\mathcal{A}$, and are called *active pairs*. After being propagated, a pair is *inactive*. Each pair (u, f) is processed as follows.

First assume that u is an interval. (See Figure 6.) We compute the list $L(u, f)$ of image vertices and intervals. Then for each vertex or interval w in this list, we check whether it needs to be pruned as follows. Let $p = \text{mid}(u)$ and $q = \text{mid}(w)$. We prune w if at least one of the three conditions below is met (Figure 7).

- **(i)** $\overline{pq} \cap \text{Ske} \neq \{p\}$.
- **(ii)** u and w are intervals, and $\text{edge}(u) = \text{edge}(w)$.
- **(iii)** u and w are intervals, $\text{edge}(u) \neq \text{edge}(w)$, and there is a node q_0 of Ske in the interior of $\text{edge}(w)$ such that $\overline{q_0 q} \in C(f_0)$, where f_0 is the face other than f bounded by $\text{edge}(w)$.

Condition (i) ensures that Ske has no self-crossing and no cycle. Condition (ii) will be needed in Algorithm 2 (Section 6), so that we do not need to propagate an interval backwards. Condition (iii) will also be needed in Algorithm 2, in order to ensure that the skeleton cannot enter two-way tunnels. (See Lemma 7.) We will prove later that our pruning scheme is correct in the sense that, when Algorithm 1 terminates, it always outputs all the reachable vertices. However, for some input, it may not terminate, as it may enter infinite spirals, and thus some reachable vertices may never be visited.

If w is not pruned, we insert $\overline{pq}$ into Ske. Then we create all pairs (w, f') such that $w \subset f'$ and $f' \neq f$, and we insert them into the set $\mathcal{A}$ of active pairs. If w is an interval, there is at most one such face f'.

Now suppose that u is a vertex (Figure 8). We still compute the list $L(u, f)$ of image vertices and intervals. If $L(u, f)$ contains an interval w along the edge opposite to u, we apply pruning condition (i). So we prune w if $\overline{pq} \cap \text{Ske} \neq \{p\}$. Again, if w is not pruned, we insert $\overline{pq}$ into Ske, and insert into $\mathcal{A}$ the pair (w, f') if f is adjacent to a face f' along $\text{edge}(w)$. Then we handle each vertex $v \in L(u, f)$ such that $v \neq u$ (if any) as follows. Let

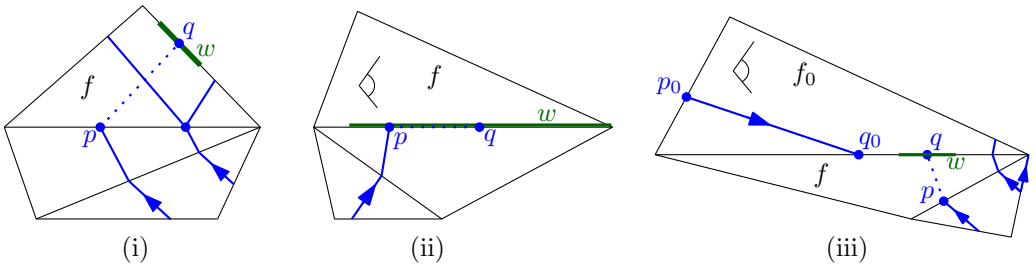


Figure 7 The three pruning conditions. The interval w corresponding to q is pruned, and hence the edge $\overline{pq}$ (dotted) is not constructed.

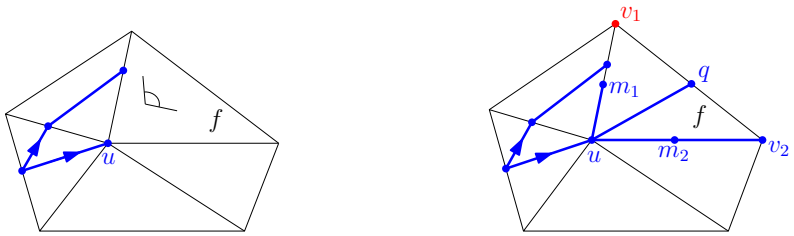


Figure 8 Propagating a pair (u, f), when u is a vertex. (Left) The skeleton before dequeuing (u, f). (Right) When we propagate (u, f), only v_1 is pruned, and the edges $\overline{uq}$, $\overline{um_1}$, $\overline{um_2}$ and $\overline{m_2v_2}$ are inserted into Ske.

m denote the midpoint of $\overline{uv}$. We apply pruning condition (i), so if $\overline{um} \cap \text{Ske} \neq \{u\}$, we prune $\overline{uv}$, and we are done with v. On the other hand, if $\overline{uv}$ is not pruned, then we insert the edge $\overline{um}$ into Ske, and if there is a face f' adjacent to f along $\overline{uv}$, we insert into $\mathcal{A}$ the pair $(\overline{uv}, f')$.

After this, still assuming that $\overline{uv}$ was not pruned, we try to extend the skeleton Ske further to v. So we apply pruning condition (i) to $\overline{mv}$. If $\overline{mv} \cap \text{Ske} \neq \{m\}$, we prune v. Otherwise, we insert the edge $\overline{mv}$ into Ske, and we insert into $\mathcal{A}$ the pair (v, f') for each face $f' \neq f$ that is adjacent to v.

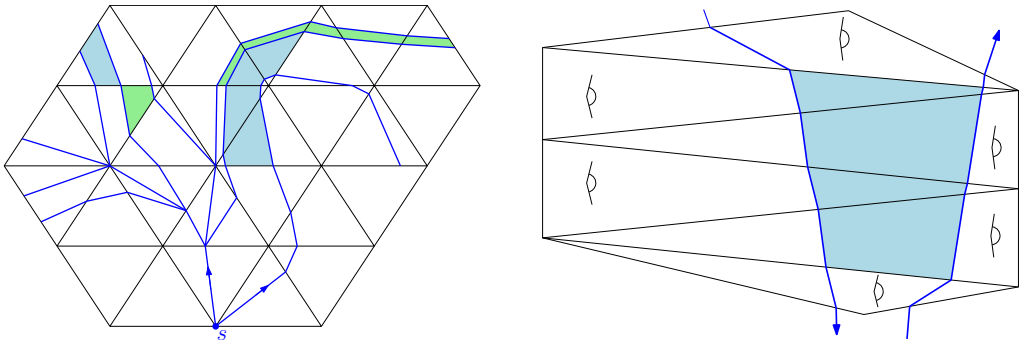
Proof of correctness. We can prove that Algorithm 1 is correct in the following sense:

▶ **Theorem 1.** *If Algorithm 1 terminates, then the reachable vertices are nodes of* Ske.

The condition that the algorithm terminates is necessary, as otherwise it could extend a spiral indefinitely and then some reachable vertices would not be visited. The proof is omitted due to space limitation. The idea is to show that, whenever we prune a vertex or an edge, which results in a pair (w, f') not being created, then some other pair (w', f') must have been created earlier such that $R(w, f') \subset R(w', f')$. It means that (w, f') can be safely pruned. Our proof is based on proving this type of invariants carefully through case analysis. It also uses the property below.

▶ **Lemma 2.** *At any time during the execution of Algorithm 1,* Ske *is a tree embedded in the plane.*

Proof. Pruning condition (i) ensures that Ske does not contain any cycle or crossing edges. When propagating a pair (u, f) where u is an interval, pruning condition (ii) ensures that the new node q is different from $p = \text{mid}(u)$. When u is a vertex of $\mathcal{S}$, then our algorithm does not attempt to propagate u into itself, so it does not create a duplicate node either. ◀



(a) (b)

Figure 9 (a) Four one-way tunnels (shaded). (b) A two-way tunnel. All these tunnels are maximal.

5 Properties of the skeleton

In this section, we consider the properties of the skeleton Ske, at any time during the execution of Algorithm 1. These properties will be needed in Section 6, in order to analyze and prove correctness of Algorithm 2. The proofs of the lemmas in this section are omitted due to space limitation.

A *branching node* of Ske is a node with outdegree at least 2. An edge of Ske that is incident to a branching node or a vertex of $\mathcal{S}$, is called a *special* edge. The other edges of Ske are called *transversal* edges. Hence, a transversal edge connects two points in the interiors of two different edges of $\mathcal{S}$, and is incident to at most one other edge at each endpoint. A *beam* is a subpath made of transversal edges. If it is not contained in any other beam, we say that it is a *maximal beam.*

▶ **Lemma 3.** *There are $O(n)$ branching nodes in* Ske.

▶ **Lemma 4.** *There are $O(n)$ special edges and maximal beams in* Ske.

The *combinatorial structure* of a transversal edge from a point in an edge e of $\mathcal{S}$ to a point along another edge e' is the pair (e, e'). Similarly, the combinatorial structure of a beam is the sequence of edges of $\mathcal{S}$ that it traverses. Two beams are *parallel* if they have the same combinatorial structures. A *one-way tunnel* is formed by two parallel beams such that there is no other edge of Ske in the space delimited by the two beams and the first and last edge of $\mathcal{S}$ that they meet. (Figure 9.) A *two-way tunnel* is analogous, but one sequence is equal to the other sequence reversed. A *tunnel* is either a one-way tunnel or a two-way tunnel. A tunnel is *maximal* if it is not contained in any other tunnel.

We now show that there is a linear number of maximal tunnels. We prove it by charging tunnels to either special edges, or *corners*: A corner of a face f is a pair (v, f) where v is a vertex of f. (See Figure 10.)

▶ **Lemma 5.** *At any time during the course of Algorithm 1, there are $O(n)$ maximal tunnels.*

A *spiral* is a one-way tunnel that is bounded by the same beam B on both sides. Therefore, the beam B, or its subsequence that bounds the spiral, is periodic: it has combinatorial structure $(e_1, e_2, \ldots, e_\ell)$ with $e_{i+p} = e_i$ for all $1 \leq i \leq i + p \leq \ell$ and some $p < \ell$. If a spiral can be extended indefinitely, that is, we can extend the sequence $(e_1, e_2, \ldots, e_\ell)$ into an arbitrarily long sequence with period p, then we say that the spiral is an *infinite spiral.*

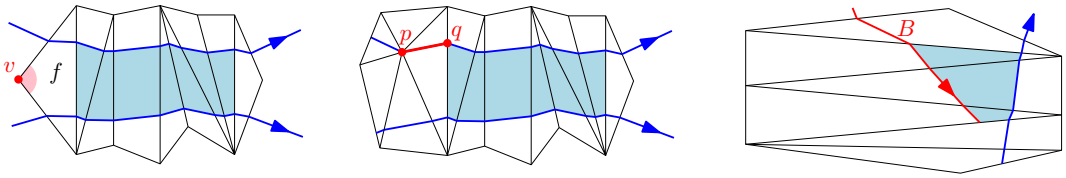


Figure 10 Proof of Lemma 5. (Left) The tunnel (shaded) is charged to the corner (v, f). (Middle) The tunnel is charged to the special edge $\overline{pq}$. (Right) The tunnel is charged to beam B.

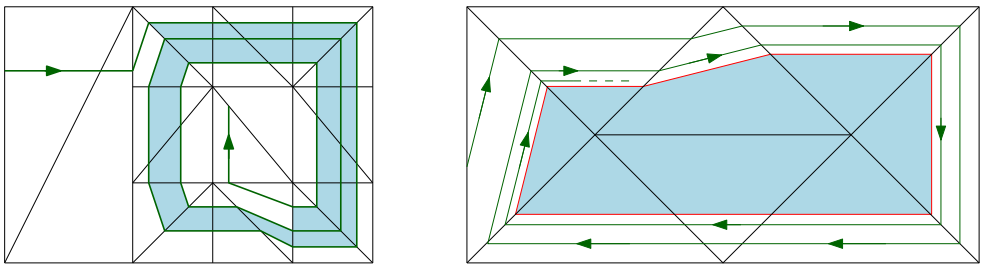


Figure 11 (Left) A finite spiral (shaded). (Right) An infinite spiral that does not converge to a vertex, as it never enters the shaded region.

Otherwise, we say that it is a *finite spiral*. Figure 2a shows an infinite spiral that converges to a single vertex. Figure 11 shows a finite spiral, and an infinite spiral that does not converge to a vertex.

We will see later that our algorithm handles finite spirals in the same way as a regular tunnel (that is, a tunnel which is not a spiral). Infinite spirals behave differently, but the lemma below shows that any beam that enters an infinite spiral cannot exit it, and hence Algorithm 2 will interrupt this beam. (See Figure 12a.)

▶ **Lemma 6.** *Suppose that Algorithm 1 extends an edge $\overline{pq}$ of the skeleton* Ske *inside an infinite spiral. Then the subtree of* Ske *rooted at p is a single beam, which remains within this spiral.*

The lemma below shows that, while we construct the skeleton, the only way to enter a tunnel is to enter a one-way tunnel from its entrance and towards its exit. It means that one-way tunnels will act as one-way gates, and two-way tunnels will act as barriers during the execution of the algorithm. The proof relies on pruning condition (iii), and is illustrated in Figure 12b.

▶ **Lemma 7.** *During the course of Algorithm 1, no new skeleton edge can be created inside an existing tunnel, except if the tunnel is a one-way tunnel, and the new edge has the same combinatorial type as an edge of the tunnel.*

Left-turn and right-turn function. We introduce the left-turn and right-turn functions associated with the combinatorial structure σ of a beam B. Intuitively, the left-turn function f_σ returns the exit point of a path that, from a given starting point, traverses the edge sequence σ, and turns left as much as possible. Similarly, the right-turn function corresponds to an extreme right-turning path. A more precise description follows.

So let $\sigma = (e_1, \ldots, e_\ell)$ denote the combinatorial structure of the beam B. We denote by $x_1, x_2, \ldots, x_\ell$ the coordinates along these edges. We assume that the edges are oriented in a consistent manner, so that the left hand side of B corresponds to smaller values of x_i and

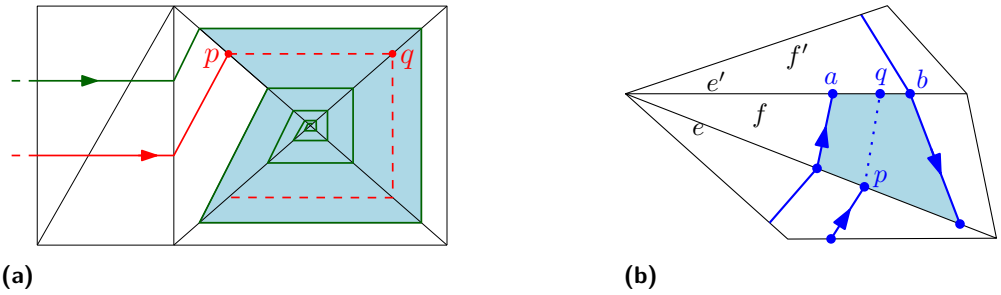


Figure 12 (a) Proof of Lemma 6. The red beam enters an infinite tunnel (shaded), and cannot get out. (b) Proof of Lemma 7. Edge $\overline{pq}$ gets pruned by condition (iii).

the right hand side corresponds to larger values. We assume that the range of x_i is $[0, 1]$, and thus the endpoints of e_i have coordinates $x_i = 0$ and $x_i = 1$.

Then a beam parallel to B is completely defined by its first interval, which lies along e_1. More precisely, we denote by $[a_i, b_i]$ the interval along e_i corresponding to this beam $B(a_1, b_1)$. We assume without loss of generality that $x_i(a_i) \leq x_i(b_i)$, that is, a_i lies on the left-hand side of B. Then $x_\ell(a_\ell)$ is a piecewise linear function $f_\sigma(x_1)$ of x_1, with at most one flat patch and one non-flat patch. More precisely, f_σ is given by two coefficients a_σ and b_σ and an interval $[c_\sigma, d_\sigma]$. The values of $a_\sigma, b_\sigma, c_\sigma, d_\sigma$ depend on the shapes and the cones of directions of the faces spanned by σ. When $x_1 \in [c_\sigma, d_\sigma]$, we have $f_\sigma(x_1) = a_\sigma x_1 + b_\sigma$. When $x_1 \leq c_\sigma$, then $f_\sigma(x_1) = f_\sigma(c_\sigma)$. When $x_1 > d_\sigma$, then there is no beam corresponding to this value of x_1.

Similarly, we define the right-turn function g_σ that gives the right endpoint of the interval along e_ℓ as a function of $x_1(b_1)$. Given a beam whose combinatorial structures σ is the concatenation $\sigma_1.\sigma_2$ of two sequence σ_1 and σ_2, the functions f_σ and g_σ can be determined in constant time, given the functions f_{σ_1}, g_{σ_1}, f_{σ_2} and g_{σ_2}.

6 Faster algorithm

In this section, we present an $O(n \log n)$ time algorithm (Algorithm 2) to compute all the vertices that are reachable from s. Algorithm 2 is based on Algorithm 1, and the speed-up comes from the fact that a new beam parallel to an existing beam can be implicitly constructed in logarithmic time, using appropriate data structures. The differences with Algorithm 1 are the following.

- The skeleton is built in a depth-first manner, always starting with the leftmost subtree, and going from left to right at each branching. So Algorithm 2 starts by constructing a leftmost turning path of Ske, and it constructs each maximal beam in one go.
- When constructing a new transversal edge, if an edge with the same combinatorial structure has been constructed earlier, the new beam follows parallel to a previously constructed beam, forming a new tunnel. Then a procedure called Construct Tunnel extends the new beam in one go, for as long as this tunnel can be extended. We will see that this procedure can be implemented to run in $O(\log n)$ time.
- If a beam begins to form an infinite spiral, or enters one, then it is interrupted, that is, the corresponding active pair never gets propagated. When all active pairs correspond to infinite spirals, Algorithm 2 halts.

On the other hand, similarly as in Algorithm 1, a special edge, or a transversal edge such that no other edge with the same combinatorial structure has been constructed before,

is constructed in constant time. As there are $O(n)$ different combinatorial structures for transversal edges, this contributes $O(n)$ to the running time. We will argue that CONSTRUCT TUNNEL is called only $O(n)$ times, and hence overall running time is $O(n \log n)$.

6.1 Data structures for beams

Let B be a beam with combinatorial structure $\sigma = (e_1, \ldots, e_m)$. Then we represent the left-turn and right-turn functions of any subsequence of σ using two balanced binary trees $T_\ell(B)$ and $T_r(B)$. These trees could be, for instance, Red-Black trees, as we will need to be able to perform insertion, deletion, join and split operations in $O(\log m)$ time [8]. We also record the first interval $[a_1, b_1]$ of B, that is, the interval of e_1 that was propagated to create the beam B.

First suppose that B does not bound any tunnel. Then $T_\ell(B)$ and $T_r(B)$ are two copies of the same binary tree representing the subsequences of σ in a hierarchical manner, and recording both the left-turn and the right-turn functions. Each node corresponds to a subsequence $\sigma_{ij} = (e_i, \ldots, e_j)$ of σ, and records the two indices i and j, as well as the functions $f_{\sigma_{ij}}$ and $g_{\sigma_{ij}}$. The root corresponds to the sequence σ of the whole beam B, and the leaves correspond to the sequences $\sigma_{i(i+1)}$ where $1 \leq i \leq m-1$. The subsequence corresponding to an internal node is the concatenation of the sequences stored at its children, which are consecutive.

If the beam B bounds a tunnel on its left side, starting from edge e_i to edge e_j, we replace the subtree of $T_\ell(B)$ corresponding to $\sigma_{ij} = (e_i, e_{i+1}, \ldots, e_j)$ with a single node that records the functions $f_{\sigma_{ij}}$ and $g_{\sigma_{ij}}$. If B bounds several tunnels on its left side, we do the same for each tunnel, as they correspond to disjoint subsequences of σ. The tree $T_r(B)$ is constructed in a similar way, except that it deals with tunnels on the right side of B.

Each tunnel is bounded by two parallel beams B and C, each one of them being recorded as one node in our data structure. Suppose that B lies on the left and C lies on the right of the tunnel. Then the node of $T_r(B)$ and the node of $T_\ell(C)$ corresponding to this tunnel record a pointer to each other.

As each node of a beam is the midpoint of the corresponding left- and right-turning path, this data structure allows us to implicitly trace a new beam that appears immediately to the left or to the right of B, for as long as it remains parallel to B, in logarithmic time.

If a beam B bounds an infinite spiral, then the infinite periodic subsequence is represented by a single node that we call a *sink* node. By Lemma 6, a beam that enters an infinite spiral cannot get out, so our algorithm will stop extending such beams when it encounters a sink node.

6.2 Procedure Construct Tunnel

Suppose that we are extending a beam B, and the next transversal edge to be constructed in this beam is $\overline{pq}$. Assume that an edge with the same combinatorial structure has been constructed earlier. Then there should exist a beam C adjacent to $\overline{pq}$. Without loss of generality, we assume that C lies to the right of $\overline{pq}$, and that it has combinatorial structure $\sigma = \gamma.(e_1, \ldots, e_m)$, where e_1 is the edge containing p.[2] The procedure CONSTRUCT TUNNEL will consider a constant number of cases below, which can be handled in logarithmic time using our data structure.

[2] When several beams cross e_1, we will show in the full version of this paper how to identify the beam C: we will use a pointer from each vertex v of $\mathcal{S}$ to the closest beam crossing each edge incident to v.

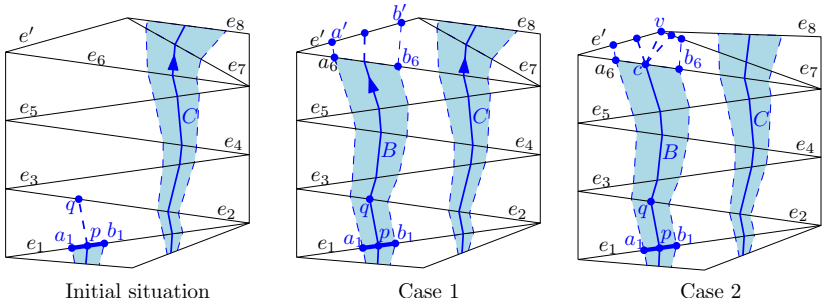


Figure 13 Result after calling Construct Tunnel once in Cases 1 and 2.

▶ **Lemma 8.** *Let N denote the total number, over all beams D constructed during the course of the algorithm, of nodes in the trees $T_\ell(D)$ or $T_r(D)$. The procedure* Construct Tunnel *runs in $O(\log N)$ time.*

We now describe the procedure Construct Tunnel.

Case 1. In this case, we assume that B follows the left side of C, until B and C split at some edge e_i. That is, the combinatorial structure of B is $\alpha.(e_1, \ldots, e_i, e')$ where $e' \neq e_{i+1}$, and α is the sequence of edges crossed by B before it reaches e_1. (See Figure 13.) So the goal is to identify this index i, and to update the data structures accordingly.

In order to simplify the presentation, we first assume that C did not bound any tunnel before we started to expand B on its left side. Let a_i and b_i denote the ith vertex along the left- and right-turning paths of B, starting from a_1 and b_1, respectively. We now want to find the last index i such that a_i and b_i lie on e_i. Our data structures $T_\ell(C)$ and $T_r(C)$ allows us to find it in $O(\log N)$ time as follows. Without loss of generality, we only consider a_i and use $T_\ell(C)$.

We find the last index i by traversing $T_\ell(C)$ from the leaf recording (e_1, e_2) towards the root, and going then going down towards the leaf recording (e_i, e_{i+1}). (Essentially, we are doing exponential search.) The left-turn function (e_1, e_2), together with the coordinates of a_1, allow us to determine whether $a_2 \in e_2$—more precisely, we check whether a_1 is in the domain of $f_{(e_1,e_2)}$. If not, then we are done. Otherwise, we move to the parent of the node recording (e_1, e_2). If (e_1, e_2) was a left child, then its parent records (e_1, e_2, e_3) or (e_1, e_2, e_3, e_4), so we use the corresponding left-turn function to determine whether $a_3 \in e_3$ or $a_4 \in e_4$. If (e_1, e_2) is a right child, then we move to the node at the same level and immediately to the right of its parent. This node records the sequence $(e_2, \ldots, e_j)$ where $j \leq 6$, and we can determine in constant time from the position of a_2 computed earlier whether a_j lies on e_j. We repeat this process $O(\log N)$ time, until it fails. Then we know that (e_i, e_{i+1}) is stored at a descendent of the current node, and we find it by traversing the tree downwards.

After finding this last index i, we cut from $T_\ell(C)$ the nodes corresponding to the sequence $(e_1, \ldots, e_i)$ and append them to $T_\ell(B)$. This can be done by performing two split and two join operations, which takes $O(\log N)$ time using Red-Black trees [8]. We then insert a single node into $T_\ell(C)$ that records the functions $f_{(e_1,\ldots,e_6)}$ and $g_{(e_1,\ldots,e_6)}$, whose coefficients we can also compute in $O(\log N)$ time using our data structure. Then we make a copy of this node and append it to $T_r(B)$. Finally, we record in each of these two nodes a pointer to the other. So overall, we have updated the data structures for B and C in $O(\log N)$ time.

The procedure above still applies when beam C bounds one or several tunnels on its left side which are then split by B. The nodes of $T_\ell(C)$ corresponding to these tunnels are moved

to $T_\ell(B)$ in the same way as the other nodes, which represent single edges of the skeleton. The cross-pointers between these nodes do not need to be updated. Thus, the data structure can be updated in $O(\log N)$ time.

Case 2. In this case, we assume that B follows the left side of C, until B reaches an edge e_i and branches. This case is similar to Case 1, except for the termination condition. In Case 1, the beam B was extended until it quit following C from the left. In Case 2, we extend B until it branches, and hence the left- and right-turning paths of B must be separated by a vertex v of the subdivision $\mathcal{S}$. (See Figure 13.) This vertex v can be found in $O(\log N)$ time using the data structures $T_\ell(C)$ and $T_\ell(B)$, which allow us to trace the left- and right-turning paths from a_1 and b_1. After v has been found, we know that B stops at the edge e_i opposite from v, and we update the data structures in the same way as in Case 1.

Case 3. In this case, we assume that B follows the left side of C, until B is interrupted because its next tentative edge gets pruned. So we assume that this edge $\overline{p_1q_1}$ has combinatorial structure (e_1, e_2), and it gets pruned due to an edge $\overline{p_2q_2}$ of Ske. We denote by e' the third edge of the face bounded by e_1 and e_2.

First assume that $p_2 \in e_1$ and $q_2 \in e_2$. Then we can prove that q_1 and q_2 must be at the midpoint of an edge of $\mathcal{S}$. We will show in the full version of this paper how to identify this case in constant time by augmenting the subdivision $\mathcal{S}$.

Now assume that $\overline{p_2q_2} \subset e_2$. In this case, $\overline{p_2q_2}$ is an edge of $\mathcal{S}$, so it can be identified in $O(\log N)$ time as in Case 1.

Suppose that $p_2 \in e_2$ and $q_2 \in e_1$. Then C forms a reversed tunnel with the beam D containing $\overline{p_2q_2}$. This tunnel is narrower along e_1, and since $\overline{p_1q_1}$ follows parallel to B, it cannot cross $\overline{p_2q_2}$, a contradiction.

The case where $p_2 \in e_1$ and $q_2 \in e'$ is similar to the case where $p_2 \in e_1$ and $q_2 \in e_2$. The case where $p_2 \in e'$ and $q_2 \in e_2$ is similar to the case where $p_2 \in e_1$ and $q_2 \in e_2$. The case where $p_2 \in e'$ and $q_2 \in e_1$ is similar to the case where $p_2 \in e_2$ and $q_2 \in e_1$. The case where $p_2 \in e_2$ and $q_2 \in e'$ is similar to the case where $p_2 \in e_2$ and $q_2 \in e_1$.

Case 4. In this case, we assume that B follows the left side of C, until we reach the terminal node of C. Our data structure allows us to identify this case in $O(\log N)$, by checking that the new section of B has the same combinatorial structure as C until we reach the end of C. If the terminal node is a sink, then B enters an infinite spiral and thus we stop extending it.

Case 5. The beams B and C are equal, and hence we start a spiral. So B has a combinatorial structure of the form $\alpha.\beta$, where $\beta = (e_1, e_2, \ldots, e_p)$, before the current call to Construct Tunnel. If the spiral is finite, then we are going to extend it into $\alpha.\beta^k.(e_1, \ldots, e_i)$, where $k \geq 0$ is the number of full turns made by the spiral, and i is the number of edges in the last (partial) turn.

Using our data structure, we compute f_σ and g_σ, which are linear functions. So $(f_\sigma)^k(a_1)$ and $(f_\sigma)^k(b_1)$ are geometric progressions, whose expression can be determined in constant time. If $x_1(a_1)$ is not in the domain of $(f_\sigma)^k$ for some k, then the spiral is finite, and we can find in constant time the first such index k, which is the number of full turns of the spiral. Similarly, we can find whether $x_1(b_1)$ leaves $(g_\sigma)^k$ for some k.

If the spiral is finite, then we append a node to $T_r(B)$ and $T_\ell(C)$ corresponding to the sequence $\beta^k.(e_1, \ldots, e_i)$ of the spiral. If the spiral is infinite, we stop propagating B, and append a sink node at the end of $T_\ell(B)$ and $T_r(B)$.

Case 6. In this case, we assume that B and C form a two-way tunnel, starting from $\overline{pq}$. For any node of $T_\ell(C)$ or $T_r(C)$ that corresponds to a sequence σ, we know the description of the left- and right-turn functions f_σ and g_σ. So we can also get in constant time their inverses f_σ^{-1} and g_σ^{-1}. Therefore, we can follow beam C backwards in the same way as in Cases 1 to 4, and Case 6 can be handled in the same way.

6.3 Main result

The main result of this paper is the theorem below. Due to space limitation, its proof is omitted.

▶ **Theorem 9.** *The vertices that are reachable from s are nodes of the skeleton computed by Algorithm 2. Therefore, we can compute all the reachable vertices in $O(n \log n)$ time.*

References

1 Michael Ben-Or. Lower bounds for algebraic computation trees. In *Proc. 15th ACM Symposium on Theory of Computing*, pages 80–86, 1983. doi:10.1145/800061.808735.

2 Siu-Wing Cheng and Jiongxin Jin. Approximate shortest descending paths. In *Proc. 24th ACM-SIAM Symposium on Discrete Algorithms*, pages 144–155, 2013.

3 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.

4 Mark de Berg, Herman J. Haverkort, and Constantinos P. Tsirogiannis. Implicit flow routing on terrains with applications to surface networks and drainage structures. In *Proc. 22nd ACM-SIAM Symposium on Discrete Algorithms*, pages 285–296, 2011.

5 Mark de Berg and Marc J. van Kreveld. Trekking in the alps without freezing or getting tired. *Algorithmica*, 18(3):306–323, 1997. doi:10.1007/PL00009159.

6 John H. Reif and Zheng Sun. Movement planning in the presence of flows. *Algorithmica*, 39(2):127–153, 2004. doi:10.1007/s00453-003-1079-5.

7 Zheng Sun and John H. Reif. On finding energy-minimizing paths on terrains. *IEEE Transactions on Robotics*, 21(1):102–114, 2005. doi:10.1109/TRO.2004.837232.

8 Robert Endre Tarjan. *Data Structures and Network Algorithms*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1983.

Fine-Grained Complexity of Coloring Unit Disks and Balls*

Csaba Biró[1], Édouard Bonnet[2], Dániel Marx[3], Tillmann Miltzow[4], and Paweł Rzążewski[5]

1 **Department of Mathematics, University of Louisville, Louisville, KY, USA**
csaba.biro@louisville.edu
2 **Institute for Computer Science and Control, Hungarian Academy of Sciences (MTA SZTAKI), Budapest, Hungary**
edouard.bonnet@dauphine.fr
3 **Institute for Computer Science and Control, Hungarian Academy of Sciences (MTA SZTAKI), Budapest, Hungary**
dmarx@cs.bme.hu
4 **Institute for Computer Science and Control, Hungarian Academy of Sciences (MTA SZTAKI), Budapest, Hungary**
t.miltzow@gmail.com
5 **Institute for Computer Science and Control, Hungarian Academy of Sciences (MTA SZTAKI), Budapest, Hungary; and**
Faculty of Mathematics and Information Science, Warsaw University of Technology, Warsaw, Poland
p.rzazewski@mini.pw.edu.pl

Abstract

On planar graphs, many classic algorithmic problems enjoy a certain "square root phenomenon" and can be solved significantly faster than what is known to be possible on general graphs: for example, Independent Set, 3-Coloring, Hamiltonian Cycle, Dominating Set can be solved in time $2^{O(\sqrt{n})}$ on an n-vertex planar graph, while no $2^{o(n)}$ algorithms exist for general graphs, assuming the Exponential Time Hypothesis (ETH). The square root in the exponent seems to be best possible for planar graphs: assuming the ETH, the running time for these problems cannot be improved to $2^{o(\sqrt{n})}$. In some cases, a similar speedup can be obtained for 2-dimensional geometric problems, for example, there are $2^{O(\sqrt{n}\log n)}$ time algorithms for Independent Set on unit disk graphs or for TSP on 2-dimensional point sets.

In this paper, we explore whether such a speedup is possible for geometric coloring problems. On the one hand, geometric objects can behave similarly to planar graphs: 3-Coloring can be solved in time $2^{O(\sqrt{n})}$ on the intersection graph of n unit disks in the plane and, assuming the ETH, there is no such algorithm with running time $2^{o(\sqrt{n})}$. On the other hand, if the number ℓ of colors is part of the input, then no such speedup is possible: Coloring the intersection graph of n unit disks with ℓ colors cannot be solved in time $2^{o(n)}$, assuming the ETH. More precisely, we exhibit a smooth increase of complexity as the number ℓ of colors increases: If we restrict the number of colors to $\ell = \Theta(n^\alpha)$ for some $0 \leqslant \alpha \leqslant 1$, then the problem of coloring the intersection graph of n unit disks with ℓ colors

- can be solved in time $\exp\left(O(n^{\frac{1+\alpha}{2}}\log n)\right) = \exp\left(O(\sqrt{n\ell}\log n)\right)$, and
- cannot be solved in time $\exp\left(o(n^{\frac{1+\alpha}{2}})\right) = \exp\left(o(\sqrt{n\ell})\right)$, unless the ETH fails.

* Supported by the ERC grant PARAMTIGHT: "Parameterized complexity and the search for tight complexity results", no. 280152.

33rd International Symposium on Computational Geometry (SoCG 2017).
Editors: Boris Aronov and Matthew J. Katz; Article No. 18; pp. 18:1–18:16

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

More generally, we consider the problem of coloring d-dimensional unit balls in the Euclidean space and obtain analogous results showing that the problem

- can be solved in time $\exp\left(O(n^{\frac{d-1+\alpha}{d}}\log n)\right) = \exp\left(O(n^{1-1/d}\ell^{1/d}\log n)\right)$, and
- cannot be solved in time $\exp\left(n^{\frac{d-1+\alpha}{d}-\epsilon}\right) = \exp\left(O(n^{1-1/d-\epsilon}\ell^{1/d})\right)$ for any $\epsilon > 0$, unless the ETH fails.

1998 ACM Subject Classification G.2.2 Graph Theory, F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases unit disk graphs, unit ball graphs, coloring, exact algorithm

Digital Object Identifier 10.4230/LIPIcs.SoCG.2017.18

1 Introduction

There are many examples of 2-dimensional geometric problems that are NP-hard, but can be solved significantly faster than the general case of the problem: for example, there are $2^{O(\sqrt{n}\log n)}$ time algorithms for TSP on 2-dimensional point sets or for Independent Set on the intersection graph of unit disks in the plane [27, 21, 1], while only $2^{O(n)}$ time algorithms are known for these problems on general metrics or on arbitrary graphs. There is evidence that these running times are essentially best possible: under the Exponential Time Hypothesis (ETH) of Impagliazzo, Paturi, and Zane [16], the $2^{O(\sqrt{n}\log n)}$ time algorithms for these 2-dimensional problems cannot be improved to $2^{o(\sqrt{n})}$, and the $2^{O(n)}$ algorithms for the general case cannot be improved to $2^{o(n)}$. Thus running times with a square root in the exponent seems to be the natural complexity behavior of many 2-dimensional geometric problems. There is a similar "square root phenomenon" for planar graphs, where running times of the form $2^{O(\sqrt{n})}$, $2^{O(\sqrt{k})}\cdot n^{O(1)}$, or $n^{O(\sqrt{k})}$ are known for a number of problems [4, 6, 5, 15, 11, 12, 7, 9, 8, 28, 14, 10, 17, 18, 2, 24, 25, 21]. More generally, for d-dimensional geometric problems, running times of the from $2^{O(n^{1-1/d})}$ or $n^{O(k^{1-1/d})}$ appear naturally, and Marx and Sidiropoulos [22] showed that, assuming the ETH, this form of running time is essentially best possible for some problems.

In this paper, we explore whether such a speedup is possible for geometric coloring problems. Deciding whether an n-vertex graph has an ℓ-coloring can be done in time $\ell^{O(n)}$ by brute force, or in time $2^{O(n)}$ using dynamic programming. On planar graphs, we can decide 3-colorability significantly faster in time $2^{O(\sqrt{n})}$, for example, by observing that planar graphs have treewidth $O(\sqrt{n})$. Let us consider now the problem of coloring the intersection graph of a set of unit disks in the 2-dimensional plane, that is, assigning a color to each disk such that if two disks intersect, then they receive different colors. For a constant number of colors, geometric objects can behave similarly to planar graphs: 3-Coloring can be solved in time $2^{O(\sqrt{n})}$ on the intersection graph of n unit disks in the plane and, assuming the ETH, there is no such algorithm with running time $2^{o(\sqrt{n})}$. However, while every planar graph is 4-colorable, unit disks graphs can contain arbitrary large cliques, and hence the ℓ-colorability is a meaningful question for larger, non-constant, values of ℓ as well. We show that if the number ℓ of colors is part of the input and can be up to $\Theta(n)$, then, surprisingly, no speedup is possible: Coloring the intersection graph of n unit disks with ℓ colors cannot be solved in time $2^{o(n)}$, assuming the ETH. What happens between these two extremes of constant number of colors and $\Theta(n)$ colors? Our main 2-dimensional result exhibits a smooth increase of complexity as the number ℓ of colors increases.

▶ **Theorem 1.** *For any fixed $0 \leqslant \alpha \leqslant 1$, the problem of coloring the intersection graph of n unit disks with $\ell = \Theta(n^\alpha)$ colors*

- *can be solved in time $2^{O(n^{\frac{1+\alpha}{2}} \log n)} = 2^{O(\sqrt{n\ell} \log n)}$, and*
- *cannot be solved in time $2^{o(n^{\frac{1+\alpha}{2}})} = 2^{o(\sqrt{n\ell})}$, unless the ETH fails.*

Let us remark that when we express the running time as a function of *two* parameters (number n of disks and number ℓ of colors) it is not obvious what we mean by claiming that a running time is "best possible." In the statement of Theorem 1, we follow Fomin et al. [13], who studied the complexity of a two-parameter clustering problem in a similar way: We restrict the parameter ℓ to be $\Theta(n^\alpha)$ for some fixed α, and determine the complexity under this restriction as a univariate function of n.

The proof is not very specific to disks and can be easily adapted to, say, axis-parallel unit squares or other fat objects. However, it seems that the requirement of fatness is essential for this type of complexity behavior as, for example, the coloring of the intersection graphs of line segments (of arbitrary lengths) does not admit any speedup compared to the $2^{O(n)}$ algorithm, even for a constant number of colors.

▶ **Theorem 2.** *There is no $2^{o(n)}$ time algorithm for* 6-Coloring *the intersection graph of line segments in the plane, unless the ETH fails.*

How does the complexity change if we look at the generalization of the coloring problem into higher dimensions? It is known for some problems that if we generalize the problem from two dimensions to d dimensions, then the square root in the exponent of the running time changes to a $1 - 1/d$ power, which makes the running time closer and closer to the running time of the brute force as d increases [22]. This may suggest that the d-dimensional generalization of Theorem 1 should have $(n\ell)^{1-1/d}$ in the exponent instead of $\sqrt{n\ell}$. Actually, this is not exactly what happens:[1] the correct exponent seems to be $n^{1-1/d}$ times $\ell^{1/d}$. That is, as d increases, the running time becomes less and less sensitive to the number of colors and approaches $2^{O(n)}$, even for constant number of colors.

▶ **Theorem 3.** *For any fixed $0 \leqslant \alpha \leqslant 1$ and dimension $d \geqslant 2$, the problem of coloring the intersection graph of n unit balls in the d-dimensional Euclidean space with $\ell = \Theta(n^\alpha)$ colors*

- *can be solved in time $2^{O\left(n^{\frac{d-1+\alpha}{d}} \log n\right)} = 2^{O\left(n^{1-1/d}\ell^{1/d} \log n\right)}$, and*
- *cannot be solved in time $2^{n^{\frac{d-1+\alpha}{d}-\epsilon}}$ for any $\epsilon > 0$, unless the ETH fails.*

Techniques. The upper bounds of Theorems 1 and 3 follow fairly easily using standard techniques. Clearly, the problem of coloring unit disks with ℓ colors makes sense only if every point of the plane is contained in at most ℓ disks: otherwise the intersection graph would contain a clique of size larger than ℓ and we would immediately know that there is no ℓ-coloring. On the other hand, if every point is contained in at most ℓ of the n unit disks, then it is known that there is a balanced separator of size $O(\sqrt{n\ell})$ [23, 27, 26]. By finding such a separator and trying every possible coloring on the disks of the separator, we can branch into $\ell^{O(\sqrt{n\ell})}$ smaller instances (here it is convenient to generalize the problem into the list coloring problem, where certain colors are forbidden on certain disks). This recursive

[1] The astute reader can quickly realize that $2^{O((n\ell)^{1-1/d})}$ is certainly not the correct answer when, say, $\ell = \Theta(n)$ and $d = 3$: then $2^{O((n\ell)^{1-1/d})} = 2^{O(n^{4/3})}$ is worse than the running time $2^{O(n)}$ possible even for general graphs!

procedure has the running time claimed in Theorem 1. We can use higher-dimensional separation theorems and a similar approach to prove the upper bound of Theorem 3.

For the lower bound, the first observation is that instances with the following structure seem to be the hardest: the set of disks consists of g^2 groups forming a $g \times g$-grid and each group consists of ℓ pairwise intersecting disks such that disks in group (i, j) can intersect disks only from those other groups that are adjacent to (i, j) in the $g \times g$-grid. Note that this instance has $n = g^2\ell$ disks. As a sanity check, let us observe that the $g\ell$ disks in any given row have $\ell^{g\ell}$ possible different colorings, hence we can solve the problem by a dynamic programming algorithm that sweeps the instance row by row in time in $2^{O(g\ell \log \ell)} = 2^{O(\sqrt{n\ell} \log \ell)}$, which is consistent with the upper bound of Theorem 1. We introduce the Partial d-grid Coloring problem as a slight generalization of such grid-like instances where some of the $g \times g$ groups can be missing.

To prove that instances of this form cannot be solved significantly faster, we reduce from a restricted version of satisfiability where g^2k variables are partitioned into g^2 groups forming a $g \times g$-grid and there are two types of constraints: clauses of size at most 3 where each variable comes from the same group and equality constraints forcing two variables from two adjacent groups to be equal. It is not very difficult to show that any 3-SAT instance with $O(gk)$ variables and $O(gk)$ clauses can be embedded into such a problem, hence the ETH implies that the problem cannot be solved in time $2^{o(gk)}$. We reduce such instances of 3-SAT to the coloring problem by representing each group of k variables with a group of $\ell = O(k)$ disks and make the following correspondence between truth assignments and colorings: if the i-th variable of the group is true, then we represent it by giving color $2i - 1$ to the $(2i - 1)$-st disk and color $2i$ to the $2i$-th disk, and we represent false by swapping these two colors. Then we implement gadgets that enforce the meaning of the clauses and the equality constraints. This way, we create an equivalent instance with $O(g^2)$ groups of $\ell = O(k)$ disks in each group, hence an algorithm with running time $2^{o(g\ell)} = 2^{o(gk)}$ would violate ETH, which is what we wanted to show.

The d-dimensional lower bound of Theorem 3 goes along the same lines, but we first prove a lower bound for a d-dimensional version of 3-SAT, where there are g^d groups of variables of size k each, arranged into a $g \times \cdots \times g$-grid. Based on earlier results by Marx and Sidiropoulos [22], we prove an almost tight lower bound for this d-dimensional 3-SAT by embedding a 3-SAT instance with roughly $g^{d-1}k$ variables and clauses into the d-dimensional $g \times \cdots \times g$-grid. Then the reduction from this problem to coloring unit balls in d-dimensional space is very similar to the 2-dimensional case.

2 Intermediate problems

In this section, we introduce two technical problems, which will serve as an intermediate step in our hardness reductions. Let us start with some notation and definitions. For an integer n, we denote by $[n]$ the set $\{1, 2, \ldots, n\}$. For a set S, we denote by 2^S the family of all subsets of S. For a fixed dimension d and $i \in [d]$, we denote by e_i the d-dimensional vector, whose i-th coordinate is equal to 1 and all remaining coordinates are equal to 0. For two positive integers g, d, we denote by $R[g, d]$ the *d-dimensional grid*, i.e., a graph whose vertices are all vectors from $[g]^d$, and two vertices are adjacent if they differ on exactly one coordinate, and exactly by one (on that coordinate). In other words, a and a' are adjacent if $a = a' \pm e_i$ for some $i \in [d]$. We will often refer to vertices of a grid as *cells*.

Problem: d-GRID 3-SAT
Input: A d-dimensional grid $G = R[g, d]$, a positive integer k, a function $\zeta : v \in V(G) \mapsto \{v_1, v_2, \ldots, v_k\}$ mapping each cell v to k fresh boolean variables, and a set $\mathcal{C}$ of constraints of two kinds:
clause constraints: for a cell v, a set $\mathcal{C}(v)$ of pairwise variable-disjoint disjunctions of at most 3 literals on $\zeta(v)$;
equality constraints: for adjacent cells v and w, a set $\mathcal{C}(v, w)$ of pairwise variable-disjoint constraints of the form $v_i = w_j$ (with $i, j \in [k]$).
Question: Is there an assignment of the variables such that all constraints are satisfied?

Not all variables need to appear in some constraint. The size of the instance $I = (G, k, \zeta, \mathcal{C})$ of d-GRID 3-SAT is the total number of variables, i.e., $g^d k$.

Problem: PARTIAL d-GRID COLORING
Input: An induced subgraph G of the d-dimensional grid $R[g, d]$, a positive integer ℓ, and a function $\rho : v \in V(G) \mapsto \{p_1^v, p_2^v, \ldots, p_\ell^v\} \in ([\ell]^d)^\ell$ mapping each cell v to a set of ℓ points in $[\ell]^d$.
Question: Is there an ℓ-coloring of all the points such that:
- two points in the same cell get different colors;
- if v and w are adjacent in G, say, $w = v + e_i$ (for some $i \in [d]$), and $p \in \rho(v)$ and $q \in \rho(w)$ receive the same color, then $p[i] \leqslant q[i]$ where $a[i] := a \cdot e_i$ is the i-th coordinate of a?

Here the size of the instance is the total number of points, i.e., $|V(G)|\ell \leqslant g^d \ell$.

3 Two-Dimensional Lower Bounds

In this section, we discuss how to obtain a lower bound for the complexity of coloring unit disk graphs. We do it using a three-step reduction and the intermediate problems introduced in the previous section. Thanks to introducing these two intermediate steps, our construction is easy to generalize to higher dimensions (see Section 4).

First we reduce 3-SAT to 2-GRID 3-SAT.

▶ **Theorem 4.** *For any $0 \leqslant \alpha \leqslant 1$ there is no algorithm solving* 2-GRID 3-SAT *with total size n and $k = \Theta(n^\alpha)$ variables per cell in time $2^{o(\sqrt{nk})} = 2^{o(n^{\frac{1+\alpha}{2}})}$, unless the ETH fails.*

The next step is reducing 2-GRID 3-SAT to PARTIAL 2-GRID COLORING. This step is the most important part of the proof.

▶ **Theorem 5.** *For any $0 \leqslant \alpha \leqslant 1$, there is no $2^{o(\sqrt{n\ell})}$ algorithm solving* PARTIAL 2-GRID COLORING *on a total of n points and $\ell = \Theta(n^\alpha)$ points in each cell (that is n/ℓ cells), unless the ETH fails.*

Proof. We present a reduction from 2-GRID 3-SAT to PARTIAL 2-GRID COLORING. Let $I = (G, k, \zeta, \mathcal{C})$ be an instance of 2-GRID 3-SAT, where $G = R[g, 2]$ and each cell contains k variables. We think of G as embedded in the plane in a natural way, with edges being horizontal or vertical segments. We construct an equivalent instance $J = (F, \ell, \rho)$ of PARTIAL

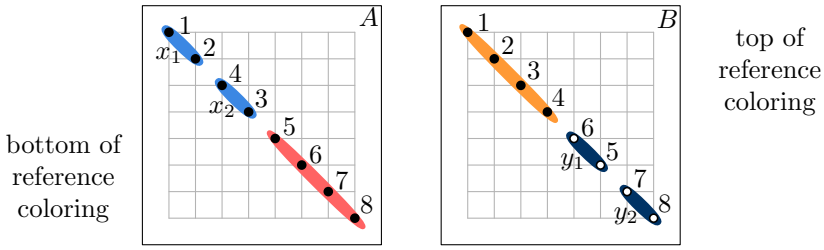


Figure 1 Cells of even parity contain the bottom half of the reference coloring as in cell A and cells of odd parity contain the top part of the reference coloring, as in cell B.

2-GRID COLORING with $|V(F)| = \Theta(|V(G)|) = \Theta(g^2)$ and $\ell := 4k$ points per cell, where F is an induced subgraph of $R[g', 2]$ with $g' = \Theta(g)$.

First, we will explain the most basic building blocks of our construction, i.e., standard cells, reference cell, variable-assignment cells, local reference cells, and wires. Then we are ready to give an overview of the whole reduction. We finish with an elaborate explanation of more complicated gadgets and proof of their correctness.

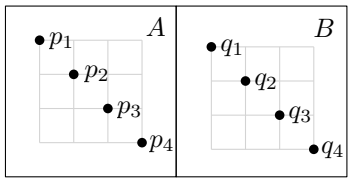
Standard cells. A *standard cell* is a cell where the points $p_1, \ldots, p_\ell$ are on the main diagonal, that is $p_i = (i, i)$ for every $i \in [\ell]$ (see cells A and B of Figure 2a). When we talk about the ordering of the points in a standard cell, we always mean the left-to-right (or equivalently, top-to-bottom) ordering. Standard cells will be used for the basic pieces of the construction, i.e., variable-assignment cells, local reference cells, and wires (see below).

Reference coloring. Later in the construction we will choose one standard cell $\bar{R}$, which will be given a special function. We will refer to the coloring of $\bar{R}$ as the *reference coloring*. For each $i \in [\ell]$, we define the color i to be the color used for the point p_i in $\bar{R}$. Now, saying that a point somewhere else has color i, has an absolute meaning; it means using the same color as used for point p_i in $\bar{R}$.

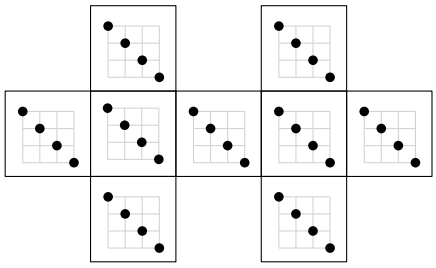
Variable-assignment cells. For each cell $v = (i, j) \in V(G)$, we introduce in F a standard cell $A(v) = (\delta i, \delta j)$, where δ is a large constant. The cells $A(v)$ for $v \in V(G)$ are responsible for encoding the truth assignment of variables in $\zeta(v)$. Therefore we call them *variable-assignment cells*. We will partition variable-assignment cells into two types. The cell $A(v)$ for $v = (i, j)$ of I is called *even* if $i + j$ is even. Otherwise $A(v)$ is *odd*. Note that if v and w are adjacent cells in I, then $A(v)$ and $A(w)$ have different parity.

As each variable-assignment cell contains $\ell = 4k$ points, there are $\ell! = 2^{O(\ell \log \ell)}$ ways to color these points with ℓ colors. We will only make use of $2^{\ell/4}$ colorings among those. In our construction, we will make sure that each variable-assignment cell receives one of the *standard colorings*. If the cell $A(v)$ is even, the coloring φ of $A(v)$ is standard if $\{\varphi(p_{2i-1}), \varphi(p_{2i})\} = \{2i - 1, 2i\}$ for $i \in [k]$ and $\varphi(p_i) = i$ for $i \in [4k] \setminus [2k]$. If the cell $A(v)$ is odd, its standard colorings φ are the ones with $\varphi(p_i) = i$ for $i \in [2k]$ and $\{\varphi(p_{2i-1}), \varphi(p_{2i})\} = \{2i - 1, 2i\}$ for $i \in [2k] \setminus [k]$. The choice of the particular standard coloring for the points in $A(v)$ defines the actual assignment of variables in $\zeta(v)$. If $A(v)$ is even, then for each $i \in [k]$, we interpret the coloring in the following way:

$p_{2i-1} \mapsto 2i - 1$, $p_{2i} \mapsto 2i$ as setting the variablev_i to true;
$p_{2i-1} \mapsto 2i$, $p_{2i} \mapsto 2i - 1$ as setting the variablev_i to false.



(a) If two standard cells are adjacent, they must have the same coloring.



(b) Wires can be used to create many copies of the same cell.

Figure 2 Construction and usage of wires.

If $A(v)$ is odd, for each $i \in [k]$, we interpret it in that way:

$p_{2k+2i-1} \mapsto 2i-1$, $p_{2k+2i} \mapsto 2i$as setting the variable$v_i$to true;
$p_{2k+2i-1} \mapsto 2i$, $p_{2k+2i} \mapsto 2i-1$as setting the variable$v_i$to false.

Observe that in even (odd, respectively) cells $A(v)$ the assignment of variables is only encoded by the coloring of the first (last, respectively) $2k$ points in $A(v)$. The colors of the remaining points are exactly the same as in the reference coloring, so each cell contains exactly one half of the reference coloring.

Local reference cells. For all $i, j \in [g-1]$, we introduce a new standard cell $R(i,j) = (\delta i + \delta/2, \delta j + \delta/2)$, called a *local reference cell*. Moreover, we set the reference $\bar{R}$ to be $R(1,1)$. In the construction, we will ensure that the coloring of each local reference cell is exactly the same, i.e., is exactly the reference coloring.

Consider the variable-assignment cell $A(v)$ for $v = (i,j)$. We say that a local reference cell $R(i',j')$ is associated with $A(v)$, if $j - j' \in \{0,1\}$ and $i - i' \in \{0,1\}$. Note that each variable-assignment cell has one, two, or four associated local reference cells. Moreover, if v, w are adjacent cells of I, then $A(v)$ and $A(w)$ share at least one associated local reference cell.

Wires. If two standard cells are adjacent, then they must be colored in the same way; thus having a path of standard cells, allows us to transport the information from one cell to another. Let us prove that claim. Let A and B be two adjacent standard cells, such that A is left of B (see Figure 2a; the argument is similar if the cells are vertically adjacent).

Let $p_1, \ldots, p_\ell$ be the points of the cell A and $q_1, \ldots, q_\ell$ be the points of the cell B. Note that the color of q_1 is necessarily equal to the color of p_1, because the x-coordinates of points $p_2, p_3, \ldots, p_\ell$ exceed the x-coordinate of q_1. Inductively, we can show that for every $i \geqslant 2$, the color of q_i is the same as the color of p_i. Indeed, the colors used for $p_{i+1}, p_{i+2}, \ldots, p_\ell$ are not available for q_i, because these points are too close to q_i. On the other hand, by the inductive assumption, all colors used on $p_1, p_2, \ldots, p_{i-1}$ are already used for points $q_1, q_2, \ldots, q_{i-1}$. Thus the only possible choice for the color of q_i is the color of p_i.

Observe that the use of wires allows us to create many copies of the same cell (see Fig. 2b). We say two cells are the same, if the point configuration and their coloring must be necessarily the same.

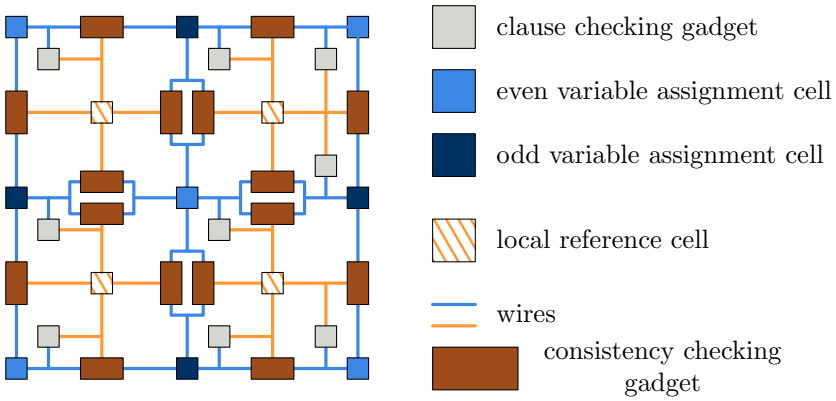


Figure 3 Illustration of the instance J. Each blue square represents a cell $A(v)$ corresponding to the cell v of I (light blue cells represent even cells and dark blue ones represent odd cells). The orange squares are local reference cells, which contain the reference coloring. Gray and brown squares represent, respectively, clause-checking and consistency gadgets.

Overview of the construction. Before we move on to describe more complicated gadgets, we explain the overview of the construction. Figure 3 presents the arrangement of the cells in F. For each variable-assignment cell $A(v)$, we introduce a *clause-checking gadget*, which is responsible for ensuring that all clauses in $\mathcal{C}(v)$ are satisfied. This gadget requires an access to the reference coloring, which can attain from the local reference cells (we can choose any of the local reference cells associated with $A(v)$). For each edge vw of G, we introduce a *consistency gadget*. In fact, for inner edges of G (i.e., the ones not incident with the outer face) we introduce two consistency gadgets, one for each face incident with vw. This gadget is responsible for ensuring the consistency on three different levels:

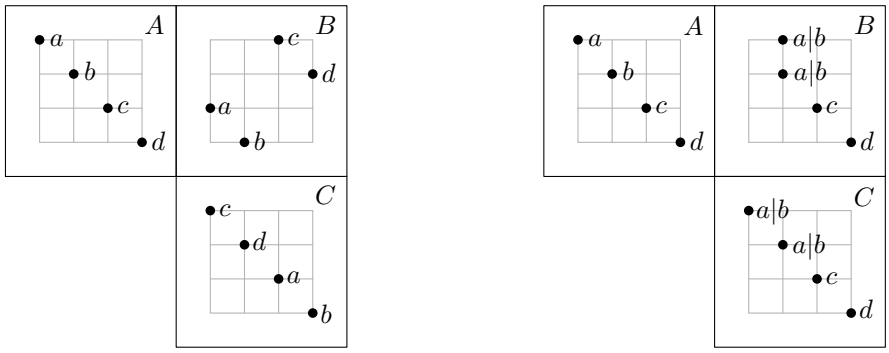
- to force all equality constraints $\mathcal{C}(v, w)$ to be satisfied,
- to ensure that each of $A(v)$ and $A(w)$ receives one of the standard colorings,
- to ensure that the local reference cell contains exactly the reference coloring.

This gadget also requires access to the reference coloring, so we join it with the appropriate local reference cell (see Fig. 3).

To join the variable-assignment cells and local reference cells with appropriate gadgets, we will use wires. Notice that each cell A can interact with at most four other cells, which may not be enough, if we want to attach several gadgets to A. However, since wires allow us to create an exact copy of A, we can attach any constant number of gadgets to A, adding only a constant number of additional cells. Moreover, we can do it in a way that ensures that no two gadgets interact with each other (anywhere but on A). Thus, when we say that we attach some gadget to a cell, we will not discuss how exactly we do this.

Every gadget uses only a constant number of cells. Thus, making the constant δ large enough and using wires, we can make sure that different gadgets do not interact with each other (except for the shared cells). The total size of the construction is clearly increased only by a constant factor.

Permuting points and colors. Recall that when describing wires, we have not used the second coordinate of the points $p_1, \ldots, p_\ell$ and $q_1, \ldots, q_\ell$. In fact, those coordinates can be chosen at our convenience, and the argument supporting the claim in the paragraphs on the wires would still work. Combining this observation horizontally and vertically, we can force any permutation of the colors (see Figure 4a). The gadget is realized as follows. Let σ be



(a) The coloring of C is the coloring of A with the permutation $\sigma = (3, 4, 1, 2)$ applied.

(b) In the cell C, colors a and b are now interchangeable.

Figure 4 Permutation gadget (left) and forgetting gadget (right), attached to cells A and C.

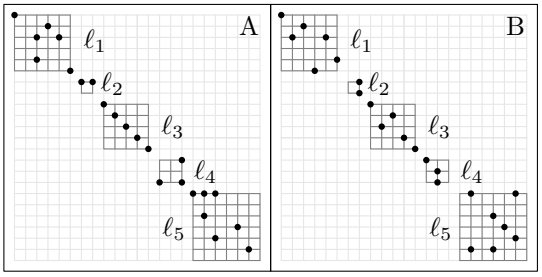
our target permutation. To the right of a standard cell A, we put a cell B. We place the points in B at the positions of 1's in the permutation matrix of σ. Below the cell B, we put a standard cell C. Is is straightforward to verify that in any feasible coloring of those three cells, for every $i \in [\ell]$, the points p_i and $q_{\sigma(i)}$ have the same color, where p_i (resp. q_i) is the point in (i, i) in the cell A (resp. cell C).

Forgetting color assignment. Besides permuting points and colors, it is also possible to forget the color assignment of some points. Figure 4b shows a forgetting gadget attached to standard cells A and C. In the cell A we have the coloring from left to right a, b, c, d. In the cell C, the first two points can be colored either a, b or b, a. In particular, if A is an even variable-assignment cell, then by looking at C we cannot distinguish anymore whether the variable was set to true or to false. Thus, using a forgetting gadget attached to two standard cells, we may force equality of colors of some corresponding points, while giving some freedom of choosing the others. This concept will be used in the next paragraph.

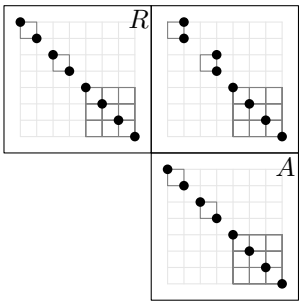
Parallelism. As we may have hinted in the previous paragraph, subparts of a given cell can act independently. In particular, this means that we can choose to forget any subset of information but preserve the rest. It is important to note that this is a more general phenomenon. Let $\ell_1, \ldots, \ell_t$ be positive integers summing up to ℓ. Consider an arrangement of cells where the points of each cell are all contained in the same square boxes of side lengths respectively $\ell_1, \ldots, \ell_t$, along the diagonal as shown in Figure 5a. For each $h \in [t]$, the h-th box (of side length ℓ_h) contains exactly ℓ_h points.

One may observe that a slight generalization of the argument given in the paragraph on wires shows that if A and B are adjacent cells with the same box-structure, i.e., each has points grouped in t boxes of sizes $\ell_1, \ldots, \ell_t$, then for each $h \in [t]$, the set of colors used on points in h-th box in A is exactly the same as the set of colors used in h-th box in B (see Figure 5a).

We point out that the combination of this observation and the forgetting gadget attached to a local reference cell and a variable-assignment cell A can be used to ensure that A receives one of the standard colorings (see Fig. 5b). The construction of the forgetting gadget varies depending on the parity of A. In general the gadget preserves the colors of $2k$ points containing the copy of one half of the reference coloring, and allows any permutation of colors within two-element boxes representing the variables. We will use a similar approach to check several clauses in parallel within the same group of a constant number of cells.



(a) The sets of colors used within corresponding boxes of A and B are equal.



(b) If R contains the reference coloring, then A receives one of standard colorings (for an even cell).

Figure 5 Boxes in adjacent cells with the same box-structure act independently from each other.

Clause gadget. We detail how a disjunction of three literals is encoded (see the left part of Figure 6). Clauses with fewer literals are just a simplification of what comes next. First, we will explain how to express a clause C, whose variables x_1, x_2, x_3 are contained in a (6×6)-box of a variable-assignment cell A. In the next paragraph we will show how to check several variable-disjoint clauses in one constant-size gadget. In general, in what follows, one should think of the coordinates that we will specify as coordinates within a box part of the cell, rather than as coordinates in the cell. The same applies to the colors, we should always look at the set of colors appearing in the particular box. Obviously, the clause-checking gadget needs to interact with variable-assignment encoding the values of x_1, x_2, x_3. For simplicity of notation assume that x_1 is encoded by coloring points p_1, p_2 with colors $1, 2$; x_2 is encoded by coloring points p_3, p_4 with colors $3, 4$ and; x_3 is encoded by coloring points p_5, p_6 with colors $5, 6$. Our clause-checking gadget needs also an access to the reference coloring contained in the cell R. This is necessary to be able to distinguish between colors e.g. 1 and 2, and thus between setting x_1 to true or to false.

First consider cells S, T, and U. The cell R contains the reference coloring and we force the order of the colors in cell T to be from top to bottom $1, 3, 5, 2, 4, 6$, using the permutation gadget. Consider now cell U. It has one point at position $(3, 3)$ and 5 points superimposed at position $(6, 6)$. Now, because of cell T, the point p can only have a color $c \in \{1, 3, 5\}$. All the other colors should be given to the 5 superimposed points. Then, consider cells A and B.

The cell A contains the variable assignment. Recall that for each variable we use two points. If a variable occupying rows $2i-1$ and $2i$ in the cell A occurs positively in C, then we place in cell B a point in row $2i-1$ to the left of the box (say, the third column) and a point in the row $2i$ to the right of the box (the sixth column); if the variable appears negatively, we do the opposite: we place in cell B a point in the row $2i-1$ to the right of the box (sixth column) and a point in row $2i$ to the left of the box (third column). By construction, the colors to the right are not available to the point p. Therefore, the point p (and henceforth the whole set of cells) can be colored if and only if at least one literal is set to true by the truth assignment.

Checking clauses in parallel. Consider the cell v of 2-GRID 3-SAT. Let $C_1, \ldots, C_f$ be the clauses of $\mathcal{C}(v)$ and recall that these clauses are pairwise variable-disjoint. Let σ be a permutation of points in $A(v)$, such that the $2|C_1|$ points encoding the variables of C_1 appear on positions $1, 2, \ldots, 2|C_1|$, the $2|C_1|$ points encoding the variables of C_2 appear on positions

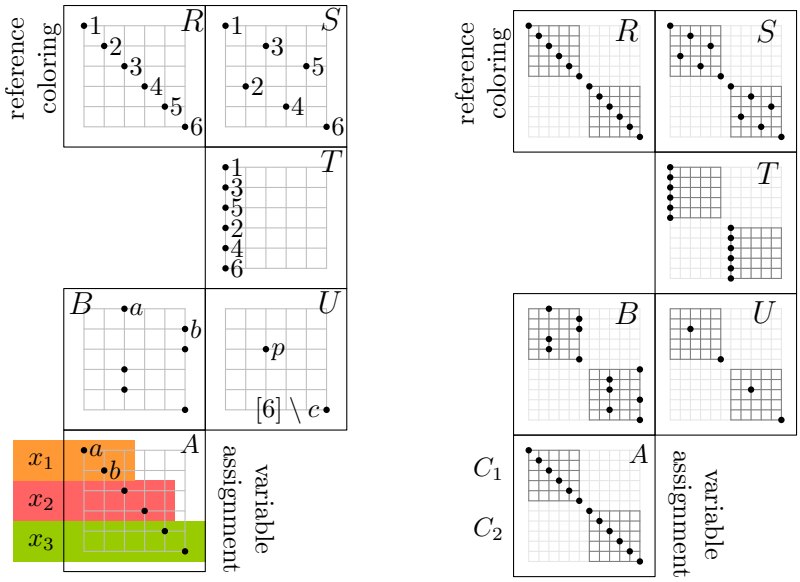


Figure 6 Illustration of the clause-checking gadget. To the left, one clause $x_1 \vee \neg x_2 \vee x_3$ is represented. To the right, two clauses are checked in parallel.

$2|C_1|+1, 2|C_1|+2, \ldots, 2|C_1|+2|C_2|$ and so on. The points encoding variables which do not appear in any clause from $\mathcal{C}(v)$ and the points which do not encode any variable (i.e., the points carrying a half of the reference coloring) appear on the last position, in any order.

We introduce a new standard cell A, and using a permutation gadget we ensure that it contains the copies of points of $A(v)$ in the permutation σ. In the same way we introduce a standard cell R, which contains the reference coloring with the permutation σ applied. An illustration on how two clauses can be checked simultaneously is shown on the right part of Figure 6. Observe that since the clauses in $\mathcal{C}(v)$ are pairwise variable-disjoint, one clause-checking gadget is enough to ensure the satisfiability of all clauses in $\mathcal{C}(v)$.

Thus, for each cell $A(v)$ and its associated local reference cell R, we introduce a clause-checking gadget corresponding to the clauses in $\mathcal{C}(v)$, and join it with $A(v)$ and R.

Equality check. Let A be a cell of J and let the points p_{2i-1}, p_{2i} (p_{2j-1}, p_{2j} for $2i < 2j-1$) in the cell A encode the variable x (y, respectively). Suppose we want to make sure that always $x = y$. This is equivalent to saying that in any proper coloring φ, we have $\varphi(p_{2i-1}) + 1 = \varphi(p_{2i})$ whenever $\varphi(p_{2j-1}) + 1 = \varphi(p_{2j})$.

Such an equivalence of two variables can be expressed by two clauses $C_1 = x \vee \neg y$ and $C_2 = \neg x \vee y$. Thus, if we have an access to the reference coloring, we can ensure the equivalence using the clause-checking gadget. Observe that C_1 and C_2 are not variable-disjoint, so in fact we need to use two clause-checking gadgets. However, two clause-checking gadgets are enough to ensure the equivalence of any set of pairwise-disjoint pairs of variables represented in the single cell. Observe that A does not have to be a variable-assignment cell (i.e., does not have to carry a half of the reference coloring). In fact, we will use the equality checks for cells where each pair of points p_{2i-1}, p_{2i} corresponds to some variable, encoded in an analogous way as in variable-assignment cells.

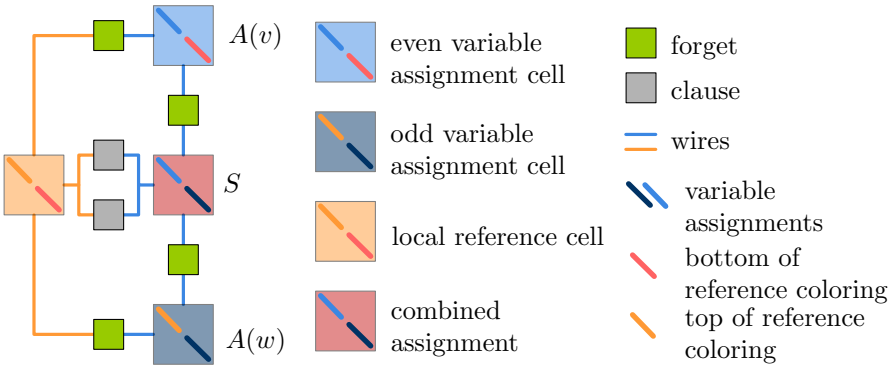


Figure 7 Overview of the consistency gadget. The clause gadgets serve to realize the equality constraints $\mathcal{C}(v, w)$.

Consistency gadget. The last gadget, called the consistency gadget, will join every three cells $A(v), A(w), R$, where $A(v)$ and $A(w)$ are variable-assignment cells corresponding to adjacent cells v and w of I, and a R is a local reference cell associated with both $A(v)$ and $A(w)$. This gadget is responsible for ensuring that colorings of these three cells are consistent, that is:

- each cell $A(v)$, $A(w)$ is colored with a standard coloring,
- the equality constraints $\mathcal{C}(v, w)$ in the 2-GRID 3-SAT instance I are satisfied,
- R has exactly the reference coloring.

Suppose that $A(v)$ is even, $A(w)$ is odd, and v is above w in I (all other cases are symmetric). We denote the points of $A(v)$ by $p_1, p_2, \ldots, p_\ell$, the points of $A(w)$ by $q_1, q_2, \ldots, q_\ell$, and the points by R by $r_1, r_2, \ldots, r_\ell$ (going from top-left to bottom-right). First, we introduce two forgetting gadgets and attach one of them to R and $A(v)$, and the other one to R and $A(w)$. The first gadget ensures that in every coloring φ we have

- $\{\varphi(p_{2i-1}), \varphi(p_{2i})\} = \{\varphi(r_{2i-1}), \varphi(r_{2i})\}$ for $i \in [k]$,
- $\varphi(p_{2i-1}) = \varphi(r_{2i-1})$ and $\varphi(p_{2i}) = \varphi(r_{2i})$ for $i \in [2k] \setminus [k]$.

The second one ensures that in every coloring φ we have

- $\varphi(q_{2i-1}) = \varphi(r_{2i-1})$ and $\varphi(q_{2i}) = \varphi(r_{2i})$ for $i \in [k]$,
- $\{\varphi(q_{2i-1}), \varphi(q_{2i})\} = \{\varphi(r_{2i-1}), \varphi(r_{2i})\}$ for $i \in [2k] \setminus [k]$.

We also introduce a new standard cell S. Let $s_1, s_2, \ldots, s_\ell$ be the points in S. With two additional forgetting gadgets, one attached to S and $A(v)$, and the other one attached to S and $A(w)$, we ensure that in every coloring φ we have:

- $\varphi(s_{2i-1}) = \varphi(p_{2i-1})$ and $\varphi(s_{2i}) = \varphi(p_{2i})$ for $i \in [k]$,
- $\varphi(s_{2i-1}) = \varphi(q_{2i-1})$ and $\varphi(s_{2i}) = \varphi(q_{2i})$ for $i \in [2k] \setminus [k]$.

Note that the cell S contains the information about the values of all variables in $\zeta(v)$ (first $2k$ points) and in $\zeta(w)$ (second $2k$ points). Now consider the set of equality constraints $\mathcal{C}(v, w)$, recall that each of them is of the form $v_i = w_j$. Thus we want to ensure that in every coloring φ, we have $\varphi(s_{2i-1}) + 1 = \varphi(s_{2i})$ if and only if $\varphi(s_{2k+2j-1}) + 1 = \varphi(s_{2k+2j})$. We can easily do it by performing the equality check on S, using two clause gadgets and R as a reference coloring. The whole consistency gadget is displayed schematically in Figure 7.

Is is straightforward to observe that if I is satisfiable, then J can be colored with ℓ colors, in a way described above. The opposite implication follows from the claims below.

▶ **Claim 6.** *The coloring of each $R(i,j)$ for $i,j \in [g-1]$ is exactly the same as the coloring of $\bar{R} = R(1,1)$.*

Proof. To show this, we will prove that the coloring of $R(i,j)$ is the same as the coloring of $R(i-1,j)$ for each $2 \leqslant i \leqslant g-1$ and $j \in [g-1]$. The case for $R(i,j-1)$ is analogous, and the claim follows inductively.

Let $v = (i,j)$ and $w = (i,j+1)$ be the cells of I. Note that v and w are adjacent and $A(v)$ and $A(w)$ are associated with both $R(i-1,j)$ and $R(i,j)$. Without loss of generality assume that v is even and w is odd. For $f \in [\ell]$, by p_f, q_f, r_f, and r'_f we denote, respectively, the points of $A(v)$, $A(w)$, $R(i-1,j)$, and $R(i,j)$. By the correctness of forget gadget, we know that for every coloring φ, we have: $\varphi(r_f) = \varphi(q_f) = \varphi(r'_f)$ for all $f \in [2k]$, and $\varphi(r_f) = \varphi(p_f) = \varphi(r'_f)$ for all $f \in [4k] \setminus [2k]$. This proves the claim. ◀

▶ **Claim 7.**

1. *The coloring of each $A(v)$ is one of the standard colorings.*
2. *For each pair of adjacent cells v, w of I, all local constraints $\mathcal{C}(v,w)$ are satisfied.*
3. *For each cell v of I, all constraints $\mathcal{C}(v)$ are satisfied.*

The claim follows directly from Claim 6 and the correctness of forget, clause-checking, and consistency gadgets.

Now, observe that the total number of points in F is $n = O(g^2\ell) = O(n')$, where $n' = g^2k$ is the total size of I. Thus, the existence of an algorithm solving J in time $2^{o(\sqrt{n\ell})}$ could be used to solve I in time $2^{o(\sqrt{n'k})}$, which, by Theorem 4, contradicts the ETH. ◀

Now, to prove the lower bound in Theorem 1, we need to show a reduction from Partial 2-grid Coloring to the problem of coloring unit disk graphs. This reduction is fairly standard and uses a well-known approach [20, Theorems 1 and 3].

4 Higher Dimensional Lower Bounds

The following result is a generalization of Theorem 4 to higher dimensions.

▶ **Theorem 8.** *For any integer $d \geqslant 3$ and reals $\epsilon > 0$ and $0 \leqslant \alpha \leqslant 1$, there is no algorithm solving d-grid 3-Sat with total size n and $k = \Theta(n^\alpha)$ variables per cell in time $2^{n^{\frac{d-1+\alpha}{d}-\epsilon}} = 2^{O(n^{1-1/d-\epsilon}k^{1/d})}$, unless the ETH fails.*

After establishing the hardness of d-grid 3-Sat, we can proceed to showing the hardness of Partial d-grid Coloring.

▶ **Theorem 9.** *For any integer $d \geqslant 3$, and reals $0 \leqslant \alpha \leqslant 1$ and $\epsilon > 0$, there is no $2^{n^{1-1/d-\epsilon}\ell^{1/d}}$ algorithm solving Partial d-grid Coloring on a total of n points and $\ell = \Theta(n^\alpha)$ points in each cell, unless the ETH fails.*

The final step in proving the lower bound in Theorem 3 is reducing Partial d-grid Coloring to ℓ-Coloring of intersection graph of d-dimensional unit balls. It is very similar to the one in Theorem 1 (see also [22, Theorem 3.1.]).

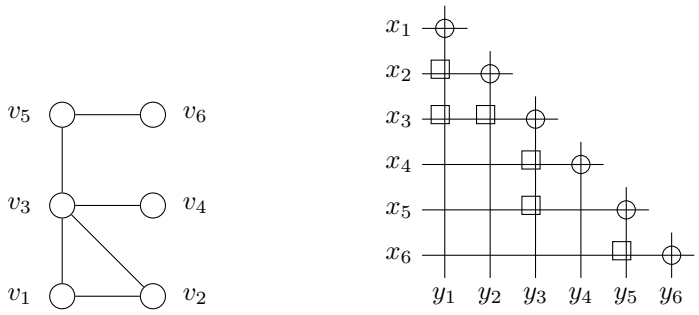


Figure 8 A graph G (left) and a high-level construction of G' (right). Circles denote equality gadgets and squares denote inequality gadgets.

5 Segments

In this section, we show that fatness is indeed necessary to obtain subexponential-time algorithm for coloring. We prove that a subexponential algorithm for coloring intersection graphs of segments (i.e., convex non-fat objects) with 6 colors would contradict the ETH.

Our construction works even if we use only horizontal or vertical segments. This class is known as 2-Dir. Note that if all segments are parallel, the intersection graph is an interval graph and, as such, can be colored in polynomial time. Moreover, we can even assume that the representation of the input graph is given. This is an important assumption, since the recognition of 2-Dir graphs is NP-complete (see Kratochvíl and Matoušek [19]).

Sketch of Proof of Theorem 2. We reduce from 3-coloring of graphs with maximum degree at most 4. Let G be a graph with n vertices and $m = \Theta(n)$ edges. It is a folklore result that, assuming the ETH, there is no algorithm solving this problem in time $2^{o(n)}$ (see for instance Lemma 1 in [3]). We construct a 2-Dir graph G', such that G is 3-colorable if and only if G' is 6-colorable.

Let the vertex set of G be $V = \{v_1, v_2 \ldots, v_n\}$. For each vertex v_i we introduce two segments: a horizontal one, called x_i, and a vertical one, called y_i, so that they form a half of a $n \times n$ grid (see Figure 8). Using appropriate gadgets we ensure that each x_i can only receive colors $\{1, 2, 3\}$, while each y_i can only receive colors $\{4, 5, 6\}$.

Each color $c \in \{1, 2, 3\}$ will be identified with the color $c + 3$. Thus, we want to ensure that in any feasible 6-coloring f of G' we have:

1. $f(x_i) + 3 = f(y_i)$ for all $i \in [n]$,
2. $f(x_i) + 3 \neq f(y_j)$ for all $i > j$ such that $v_i v_j$ is an edge of G.

This is achieved by using constant-size *equality gadgets* and *inequality gadgets*. At the crossing point of x_i and y_i, we put an equality gadget (represented by a circle on Figure 8). Moreover, for each edge $v_i v_j$ of G, we put an inequality gadget at the crossing point of x_i and y_j, $i > j$ (represented by a square on Figure 8).

The number of vertices of G' is $n' = \Theta(n)$, so the theorem follows. ◀

References

1 Jochen Alber and Jirí Fiala. Geometric separation and exact solutions for the parameterized independent set problem on disk graphs. *J. Algorithms*, 52(2):134–151, 2004. doi:10.1016/j.jalgor.2003.10.001.

2 Rajesh Hemant Chitnis, MohammadTaghi Hajiaghayi, and Dániel Marx. Tight bounds for Planar Strongly Connected Steiner Subgraph with fixed number of terminals (and extensions). In *SODA 2014 Proc.*, pages 1782–1801, 2014. doi:10.1137/1.9781611973402.129.

3 Marek Cygan, Fedor V. Fomin, Alexander Golovnev, Alexander S. Kulikov, Ivan Mihajlin, Jakub W. Pachocki, and Arkadiusz Socała. Tight lower bounds on graph embedding problems. *CoRR*, abs/1602.05016, 2016. URL: http://arxiv.org/abs/1602.05016.

4 Erik D. Demaine, Fedor V. Fomin, Mohammad Taghi Hajiaghayi, and Dimitrios M. Thilikos. Bidimensional parameters and local treewidth. *SIAM J. Discrete Math.*, 18(3):501–511, 2004. doi:10.1137/S0895480103433410.

5 Erik D. Demaine, Fedor V. Fomin, Mohammad Taghi Hajiaghayi, and Dimitrios M. Thilikos. Fixed-parameter algorithms for (k, r)-Center in planar graphs and map graphs. *ACM Transactions on Algorithms*, 1(1):33–47, 2005. doi:10.1145/1077464.1077468.

6 Erik D. Demaine, Fedor V. Fomin, Mohammad Taghi Hajiaghayi, and Dimitrios M. Thilikos. Subexponential parameterized algorithms on bounded-genus graphs and H-minor-free graphs. *J. ACM*, 52(6):866–893, 2005. doi:10.1145/1101821.1101823.

7 Erik D. Demaine and Mohammad Taghi Hajiaghayi. Fast algorithms for hard graph problems: Bidimensionality, minors, and local treewidth. In *GD 2014 Proc.*, pages 517–533, 2004. doi:10.1007/978-3-540-31843-9_57.

8 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.

9 Erik D. Demaine and MohammadTaghi Hajiaghayi. Linearity of grid minors in treewidth with applications through bidimensionality. *Combinatorica*, 28(1):19–36, 2008. doi:10.1007/s00493-008-2140-4.

10 Frederic Dorn, Fedor V. Fomin, Daniel Lokshtanov, Venkatesh Raman, and Saket Saurabh. Beyond bidimensionality: Parameterized subexponential algorithms on directed graphs. In *STACS 2010 Proc.*, pages 251–262, 2010. doi:10.4230/LIPIcs.STACS.2010.2459.

11 Frederic Dorn, Fedor V. Fomin, and Dimitrios M. Thilikos. Subexponential parameterized algorithms. *Computer Science Review*, 2(1):29–39, 2008. doi:10.1016/j.cosrev.2008.02.004.

12 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.

13 Fedor V. Fomin, Stefan Kratsch, Marcin Pilipczuk, Michal Pilipczuk, and Yngve Villanger. Tight bounds for parameterized complexity of cluster editing with a small number of clusters. *J. Comput. Syst. Sci.*, 80(7):1430–1447, 2014. doi:10.1016/j.jcss.2014.04.015.

14 Fedor V. Fomin, Daniel Lokshtanov, Venkatesh Raman, and Saket Saurabh. Subexponential algorithms for partial cover problems. *Inf. Process. Lett.*, 111(16):814–818, 2011. doi:10.1016/j.ipl.2011.05.016.

15 Fedor V. Fomin and Dimitrios M. Thilikos. Dominating sets in planar graphs: Branch-width and exponential speed-up. *SIAM J. Comput.*, 36(2):281–309, 2006. doi:10.1137/S0097539702419649.

16 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001. doi:10.1006/jcss.2001.1774.

17 Philip N. Klein and Dániel Marx. Solving Planar k-Terminal Cut in $O(n^{c\sqrt{k}})$ time. In *ICALP 2012 Proc.*, pages 569–580, 2012. doi:10.1007/978-3-642-31594-7_48.

18 Philip N. Klein and Dániel Marx. A subexponential parameterized algorithm for Subset TSP on planar graphs. In *SODA 2014 Proc.*, pages 1812–1830, 2014. doi:10.1137/1.9781611973402.131.

19 J. Kratochvíl and J. Matoušek. Intersection graphs of segments. *Journal of Combinatorial Theory, Series B*, 62(2):289–315, 1994. doi:10.1006/jctb.1994.1071.

20 Dániel Marx. Efficient approximation schemes for geometric problems? In *ESA 2005 Proc.*, pages 448–459, 2005. doi:10.1007/11561071_41.

21 Dániel Marx and Michal Pilipczuk. Optimal parameterized algorithms for planar facility location problems using voronoi diagrams. In Nikhil Bansal and Irene Finocchi, editors, *ESA 2015 Proc.*, volume 9294 of *LNCS*, pages 865–877. Springer, 2015. doi:10.1007/978-3-662-48350-3_72.

22 Dániel Marx and Anastasios Sidiropoulos. The Limited Blessing of Low Dimensionality: When 1-1/D is the Best Possible Exponent for D-dimensional Geometric Problems. In *Proceedings of the Thirtieth Annual Symposium on Computational Geometry*, SOCG 2014 Proc., pages 67:67–67:76, New York, NY, USA, 2014. ACM. doi:10.1145/2582112.2582124.

23 Gary L. Miller, Shang-Hua Teng, William Thurston, and Stephen A. Vavasis. Separators for sphere-packings and nearest neighbor graphs. *J. ACM*, 44(1):1–29, January 1997. doi:10.1145/256292.256294.

24 Marcin Pilipczuk, Michał Pilipczuk, Piotr Sankowski, and Erik Jan van Leeuwen. Subexponential-time parameterized algorithm for Steiner Tree on planar graphs. In *STACS 2013 Proc.*, pages 353–364, 2013. doi:10.4230/LIPIcs.STACS.2013.353.

25 Marcin Pilipczuk, Michal Pilipczuk, Piotr Sankowski, and Erik Jan van Leeuwen. Network sparsification for steiner problems on planar and bounded-genus graphs. In *FOCS 2014 Proc.*, pages 276–285. IEEE Computer Society, 2014. doi:10.1109/FOCS.2014.37.

26 W. D. Smith and N. C. Wormald. Geometric separator theorems. available online at https://www.math.uwaterloo.ca/~nwormald/papers/focssep.ps.gz.

27 W. D. Smith and N. C. Wormald. Geometric separator theorems and applications. In *Proceedings of the 39th Annual Symposium on Foundations of Computer Science*, FOCS 1998 Proc., pages 232–243, Washington, DC, USA, 1998. IEEE Computer Society. URL: http://dl.acm.org/citation.cfm?id=795664.796397.

28 Dimitrios M. Thilikos. Fast sub-exponential algorithms and compactness in planar graphs. In *ESA 2011 Proc.*, pages 358–369, 2011. doi:10.1007/978-3-642-23719-5_31.

Anisotropic Triangulations via Discrete Riemannian Voronoi Diagrams*†

Jean-Daniel Boissonnat[1], Mael Rouxel-Labbé[2], and Mathijs Wintraecken[3]

1 **INRIA Sophia Antipolis Méditerranée, Valbonne, France**
2 **INRIA Sophia Antipolis Méditerranée, Valbonne, France; and GeometryFactory, Valbonne, France**
3 **INRIA Sophia Antipolis Méditerranée, Valbonne, France**

Abstract

The construction of anisotropic triangulations is desirable for various applications, such as the numerical solving of partial differential equations and the representation of surfaces in graphics. To solve this notoriously difficult problem in a practical way, we introduce the discrete Riemannian Voronoi diagram, a discrete structure that approximates the Riemannian Voronoi diagram. This structure has been implemented and was shown to lead to good triangulations in $\mathbb{R}^2$ and on surfaces embedded in $\mathbb{R}^3$ as detailed in our experimental companion paper.

In this paper, we study theoretical aspects of our structure. Given a finite set of points $\mathcal{P}$ in a domain Ω equipped with a Riemannian metric, we compare the discrete Riemannian Voronoi diagram of $\mathcal{P}$ to its Riemannian Voronoi diagram. Both diagrams have dual structures called the discrete Riemannian Delaunay and the Riemannian Delaunay complex. We provide conditions that guarantee that these dual structures are identical. It then follows from previous results that the discrete Riemannian Delaunay complex can be embedded in Ω under sufficient conditions, leading to an anisotropic triangulation with curved simplices. Furthermore, we show that, under similar conditions, the simplices of this triangulation can be straightened.

1998 ACM Subject Classification Computational Geometry and Object Modeling

Keywords and phrases Riemannian Geometry, Voronoi diagram, Delaunay triangulation

Digital Object Identifier 10.4230/LIPIcs.SoCG.2017.19

1 Introduction

Anisotropic triangulations are triangulations whose elements are elongated along prescribed directions. Anisotropic triangulations are known to be well suited when solving PDE's [10, 19, 24]. They can also significantly enhance the accuracy of a surface representation if the anisotropy of the triangulation conforms to the curvature of the surface [15].

Many methods to generate anisotropic triangulations are based on the notion of Riemannian metric and create triangulations whose elements adapt locally to the size and anisotropy prescribed by the local geometry. The numerous theoretical and practical results [1] of the Euclidean Voronoi diagram and its dual structure, the Delaunay triangulation, have pushed authors to try and extend these well-established concepts to the anisotropic setting.

* A full version of the paper is available at `https://arxiv.org/abs/1703.06487`.
† The first and third authors have received funding from the European Research Council under the European Union's ERC Grant Agreement number 339025 GUDHI (Algorithmic Foundations of Geometric Understanding in Higher Dimensions).

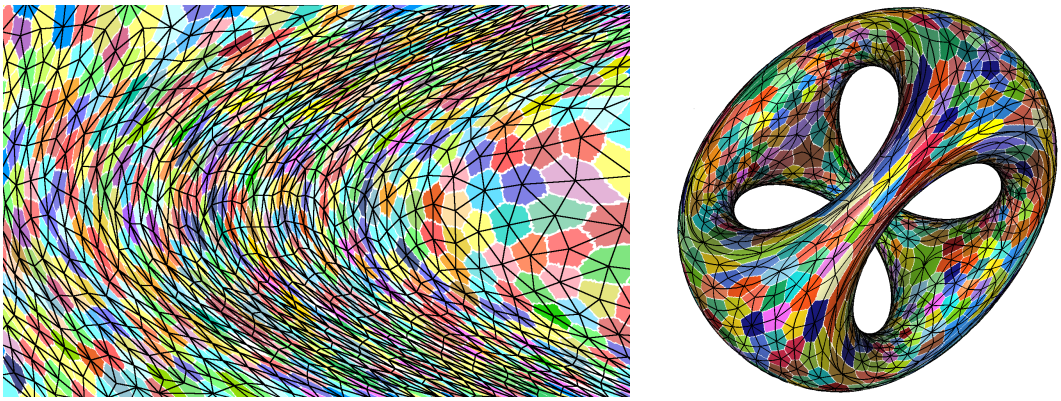


Figure 1 Left, the discrete Riemannian Voronoi diagram (colored cells with bisectors in white) and its dual complex (in black) realized with straight simplices of a two-dimensional domain endowed with a hyperbolic shock-based metric field. Right, the discrete Riemannian Voronoi diagram and the dual complex realized with curved simplices of the "chair" surface endowed with a curvature-based metric field [23].

Labelle and Shewchuk [17] and Du and Wang [12] independently introduced two anisotropic Voronoi diagrams whose anisotropic distances are based on a discrete approximation of the Riemannian metric field. Contrary to their Euclidean counterpart, the fact that the dual of these anisotropic Voronoi diagrams is an embedded triangulation is not immediate, and, despite their strong theoretical foundations, the anisotropic Voronoi diagrams of Labelle and Shewchuk and Du and Wang have only been proven to yield, under certain conditions, a good triangulation in a two-dimensional setting [6, 7, 9, 12, 17].

Both these anisotropic Voronoi diagrams can be considered as an approximation of the exact Riemannian Voronoi diagram, whose cells are defined as $V_g(p_i) = \{x \in \Omega \mid d_g(p_i, x) \leq d_g(p_j, x), \forall p_j \in \mathcal{P} \backslash p_i\}$, where $d_g(p, q)$ denotes the geodesic distance. Their main advantage is to ease the computation of the anisotropic diagrams. However, their theoretical and practical results are rather limited. The exact Riemannian Voronoi diagram comes with the benefit of providing a more favorable theoretical framework and recent works have provided sufficient conditions for a point set to be an embedded Riemannian Delaunay complex [2, 14, 18]. We approach the Riemannian Voronoi diagram and its dual Riemannian Delaunay complex with a focus on both practicality and theoretical robustness. We introduce the discrete Riemannian Voronoi diagram, a discrete approximation of the (exact) Riemannian Voronoi diagram. Experimental results, presented in our companion paper [23], have shown that this approach leads to good anisotropic triangulations for two-dimensional domains and surfaces, see Figure 1.

We introduce in this paper the theoretical side of this work, showing that our approach is theoretically sound in all dimensions. We prove that, under sufficient conditions, the discrete Riemannian Voronoi diagram has the same combinatorial structure as the (exact) Riemannian Voronoi diagram and that the dual discrete Riemannian Delaunay complex can be embedded as a triangulation of the point set, with either curved or straight simplices. Discrete Voronoi diagrams have been independently studied, although in a two-dimensional isotropic setting by Cao et al. [8].

2 Riemannian geometry

In the main part of the text we consider an (open) domain Ω in $\mathbb{R}^n$ endowed with a Riemannian metric g, which we shall discuss below. We assume that the metric g is Lipschitz continuous. The structures of interest will be built from a finite set of points $\mathcal{P}$, which we call *sites*.

2.1 Riemannian metric

A *Riemannian metric field* g, defined over Ω, associates a *metric* $g(p) = G_p$ to any point p of the domain. This means that for any $v, w \in \mathbb{R}^n$ we associate an inner product $\langle v, w \rangle_g = v^t g(p) w$, in a way that smoothly depends on p. Using a Riemannian metric, we can associate lengths to curves and define the geodesic distance d_g as the minimizer of the lengths of all curves between two points. When the map $g : p \mapsto G$ is constant, the metric field is said to be *uniform*. In this case, the distance between two points x and y in Ω is $d_G(x, y) = \|x - y\|_G = \sqrt{(x-y)^t G (x-y)}$.

Most traditional geometrical objects can be generalized using the geodesic distance. For example, the geodesic (closed) ball centered on $p \in \Omega$ and of radius r is given by $B_g(p, r) = \{x \in \Omega \mid d_g(p, x) \leq r\}$. In the following, we assume that $\Omega \subset \mathbb{R}^n$ is endowed with a Lipschitz continuous metric field g.

We define the *metric distortion* between two distance functions $d_g(x, y)$ and $d_{g'}(x, y)$ to be the function $\psi(g, g')$ such that for all x, y in a small-enough neighborhood we have: $1/\psi(g, g')\, d_g(x, y) \leq d_{g'}(x, y) \leq \psi(g, g')\, d_g(x, y)$. Observe that $\psi(g, g') \geq 1$ and $\psi(g, g') = 1$ when $g = g'$. Our definition generalizes the concept of distortion between two metrics $g(p)$ and $g(q)$, as defined by Labelle and Shewchuk [17] (see Appendix B of the full version of this paper, [4]).

2.2 Geodesy

Let $v \in \mathbb{R}^n$. From the unique geodesic γ satisfying $\gamma(0) = p$ with initial tangent vector $\dot{\gamma} = v$, one defines the *exponential map* through $\exp(v) = \gamma(1)$. The *injectivity radius* at a point p of Ω is the largest radius for which the exponential map at p restricted to a ball of that radius is a diffeomorphism. The injectivity radius ι_Ω of Ω is defined as the infimum of the injectivity radii at all points. For any $p \in \Omega$ and for a two-dimensional linear subspace H of the tangent space at p, we define the *sectional curvature* K at p for H as the Gaussian curvature at p of the surface $\exp_p(H)$.

In the theoretical studies of our algorithm, we will assume that the injectivity radius of Ω is strictly positive and its sectional curvatures are bounded.

2.3 Power protected nets

Controlling the quality of the Delaunay and Voronoi structures will be essential in our proofs. For this purpose, we use the notions of net and of power protection.

Power protection of point sets. Power protection of simplices is a concept formally introduced by Boissonnat, Dyer and Ghosh [2]. Let σ be a simplex whose vertices belong to $\mathcal{P}$, and let $B_g(\sigma) = B_g(c, r)$ denote a circumscribing ball of σ where $r = d_g(c, p)$ for any vertex p of σ. We call c the circumcenter of σ and r its circumradius.

For $0 \leq \delta \leq r$, we associate to $B_g(\sigma)$ the dilated ball $B_g^{+\delta}(\sigma) = B(c, \sqrt{r^2 + \delta^2})$. We say that σ is *δ-power protected* if $B_g^{+\delta}(\sigma)$ does not contain any point of $\mathcal{P} \setminus \mathrm{Vert}(\sigma)$ where $\mathrm{Vert}(\sigma)$

denotes the vertex set of σ. The ball $B_g^{+\delta}$ is the *power protected* ball of σ. Finally, a point set $\mathcal{P}$ is δ-power protected if the Delaunay ball of its simplices are δ-power protected.

Nets. To ensure that the simplices of the structures that we shall consider are well shaped, we will need to control the density and the sparsity of the point set. The concept of net conveys these requirements through *sampling* and *separation* parameters.

The sampling parameter is used to control the density of a point set: if Ω is a bounded domain, $\mathcal{P}$ is said to be an *ε-sample set* for Ω with respect to a metric field g if $d_g(x, \mathcal{P}) < \varepsilon$, for all $x \in \Omega$. The sparsity of a point set is controlled by the separation parameter: the set $\mathcal{P}$ is said to be *μ-separated* with respect to a metric field g if $d_g(p, q) \geq \mu$ for all $p, q \in \mathcal{P}$. If $\mathcal{P}$ is an ε-sample that is μ-separated, we say that $\mathcal{P}$ is an *(ε, μ)-net*.

3 Riemannian Delaunay triangulations

Given a metric field g, the *Riemannian Voronoi diagram* of a point set $\mathcal{P}$, denoted by $\mathrm{Vor}_g(\mathcal{P})$, is the Voronoi diagram built using the geodesic distance d_g. Formally, it is a partition of the domain in *Riemannian Voronoi cells* $\{V_g(p_i)\}$, where $\mathrm{V}_g(p_i) = \{x \in \Omega \mid d_g(p_i, x) \leq d_g(p_j, x), \forall p_j \in \mathcal{P} \setminus p_i\}$.

The Riemannian Delaunay complex of $\mathcal{P}$ is an abstract simplicial complex, defined as the nerve of the Riemannian Voronoi diagram, that is the set of simplices $\mathrm{Del}_g(\mathcal{P}) = \{\sigma \mid \mathrm{Vert}(\sigma) \in \mathcal{P}, \cap_{p\in\sigma} \mathrm{V}_g(p) \neq 0\}$. There is a straightforward duality between the diagram and the complex, and between their respective elements.

In this paper, we will consider both abstract simplices and complexes, as well as their geometric realization in $\mathbb{R}^n$ with vertex set $\mathcal{P}$. We now introduce two realizations of a simplex that will be useful, one curved and the other one straight.

The *straight realization* of a n-simplex σ with vertices in $\mathcal{P}$ is the convex hull of its vertices. We denote it by $\bar{\sigma}$. In other words,

$$\bar{\sigma} = \{x \in \Omega \subset \mathbb{R}^n \mid x = \sum_{p\in\sigma} \lambda_p(x)\, p, \lambda_p(x) \geq 0, \sum_{p\in\sigma} \lambda_p(x) = 1\}. \tag{1}$$

The *curved realization*, noted $\tilde{\sigma}$ is based on the notion of Riemannian center of mass [16, 13]. Let y be a point of $\bar{\sigma}$ with barycentric coordinate $\lambda_p(y), p \in \sigma$. We can associate the energy functional $\mathcal{E}_y(x) = \frac{1}{2} \sum_{p\in\sigma} \lambda_p(y) d_g(x, p)^2$. We then define the curved realization of σ as

$$\tilde{\sigma} = \{\tilde{x} \in \Omega \subset \mathbb{R}^n \mid \tilde{x} = \operatorname{argmin} \mathcal{E}_{\bar{x}}(x), \bar{x} \in \bar{\sigma}\}. \tag{2}$$

The edges of $\tilde{\sigma}$ are geodesic arcs between the vertices. Such a curved realization is well defined provided that the vertices of σ lie in a sufficiently small ball according to the following theorem of Karcher [16].

▶ **Theorem 1** (Karcher). *Let the sectional curvatures K of Ω be bounded, that is $\Lambda_- \leq K \leq \Lambda_+$. Let us consider the function $\mathcal{E}_y$ on B_ρ, a geodesic ball of radius ρ that contains the set $\{p_i\}$. Assume that $\rho \in \mathbb{R}^+$ is less than half the injectivity radius and less than $\pi/4\sqrt{\Lambda_+}$ if $\Lambda_+ > 0$. Then $\mathcal{E}_y$ has a unique minimum point in B_ρ, which is called the* center of mass.

Given an (abstract) simplicial complex $\mathcal{K}$ with vertices in $\mathcal{P}$, we define the straight (resp., curved) realization of $\mathcal{K}$ as the collection of straight (resp., curved) realizations of its simplices, and we write $\bar{\mathcal{K}} = \{\bar{\sigma}, \sigma \in \mathcal{K}\}$ and $\tilde{\mathcal{K}} = \{\tilde{\sigma}, \sigma \in \mathcal{K}\}$.

We will consider the case where $\mathcal{K}$ is $\mathrm{Del}_g(\mathcal{P})$. A simplex of $\overline{\mathrm{Del}}_g(\mathcal{P})$ will simply be called a straight Riemannian Delaunay simplex and a simplex of $\widetilde{\mathrm{Del}}_g(\mathcal{P})$ will be called a curved

Riemannian Delaunay simplex, omitting "realization of". In the next two sections, we give sufficient conditions for $\overline{\mathrm{Del}}_g(\mathcal{P})$ and $\widetilde{\mathrm{Del}}_g(\mathcal{P})$ to be embedded in Ω, in which case we will call them the straight and the curved Riemannian *triangulations* of $\mathcal{P}$.

3.1 Sufficient conditions for $\widetilde{\mathrm{Del}}_g(\mathcal{P})$ to be a triangulation of $\mathcal{P}$

It is known that $\widetilde{\mathrm{Del}}_g(\mathcal{P})$ is embedded in Ω under sufficient conditions. We give a short overview of these results. As in Dyer et al. [13], we define the non-degeneracy of a simplex $\tilde{\sigma}$ of $\widetilde{\mathrm{Del}}_g(\mathcal{P})$.

▶ **Definition 2.** The curved realization $\widetilde{\sigma}$ of a Riemannian Delaunay simplex σ is said to be non-degenerate if and only if it is homeomorphic to the standard simplex.

Sufficient conditions for the complex $\widetilde{\mathrm{Del}}_g(\mathcal{P})$ to be embedded in Ω were given in [13]: a curved simplex is known to be non-degenerate if the Euclidean simplex obtained by lifting the vertices to the tangent space at one of the vertices via the exponential map has sufficient quality compared to the bounds on sectional curvature. Here, good quality means that the simplex is well shaped, which may be expressed either through its fatness (volume compared to longest edge length) or its thickness (smallest height compared to longest edge length).

Let us assume that, for each vertex p of $\mathrm{Del}_g(\mathcal{P})$, all the curved Delaunay simplices in a neighborhood of p are non-degenerate and patch together well. Under these conditions, $\widetilde{\mathrm{Del}}_g(\mathcal{P})$ is embedded in Ω. We call $\widetilde{\mathrm{Del}}_g(\mathcal{P})$ the *curved Riemannian Delaunay triangulation* of $\mathcal{P}$.

3.2 Sufficient conditions for $\overline{\mathrm{Del}}_g(\mathcal{P})$ to be a triangulation of $\mathcal{P}$

Assuming that the conditions for $\widetilde{\mathrm{Del}}_g(\mathcal{P})$ to be embedded in Ω are satisfied, we now give conditions such that $\overline{\mathrm{Del}}_g(\mathcal{P})$ is also embedded in Ω. The key ingredient will be a bound on the distance between a point of a simplex $\tilde{\sigma}$ and the corresponding point on the associated straight simplex $\bar{\sigma}$ (Lemma 3). This bound depends on the properties of the set of sites and on the local distortion of the metric field. When this bound is sufficiently small, $\overline{\mathrm{Del}}_g(\mathcal{P})$ is embedded in Ω as stated in Theorem 4.

▶ **Lemma 3.** *Let σ be an n-simplex of $\mathrm{Del}_g(\mathcal{P})$. Let $\bar{x}$ be a point of $\bar{\sigma}$ and $\widetilde{x}$ the associated point on $\widetilde{\sigma}$ (as defined in Equation 1). If the geodesic distance d_g is close to the Euclidean distance $d_{\mathbb{E}}$, i.e. the distortion $\psi(g, g_{\mathbb{E}})$ is bounded by ψ_0, then $|\widetilde{x} - \bar{x}| \leq \sqrt{2 \cdot 4^3(\psi_0 - 1)\varepsilon^2}$.*

We now apply Lemma 3 to the facets of the simplices of $\widetilde{\mathrm{Del}}_g(\mathcal{P})$. The altitude of the vertex p in a simplex τ is noted $D(p, \tau)$.

▶ **Theorem 4.** *Let $\mathcal{P}$ be a δ-power protected (ε, μ)-net with respect to g on Ω. Let σ be any n-simplex of $\mathrm{Del}_g(\mathcal{P})$ and p be any vertex of σ. Let τ be a facet of σ opposite of vertex p. If, for all $\widetilde{x} \in \widetilde{\tau}$, we have $|\widetilde{x} - \bar{x}| \leq D(p_i, \sigma)$ ($\bar{x}$ is defined in Equation 1), then $\overline{\mathrm{Del}}_d(\mathcal{P})$ is embedded in Ω.*

The condition $|\widetilde{x} - \bar{x}| \leq D(p_i, \sigma)$ is achieved for a sufficiently dense sampling according to Lemma 3 and the fact that the distortion $\psi_0 = \psi(g, g_{\mathbb{E}})$ goes to 1 when the density increases. The complete proofs of Lemma 3 and Theorem 4 can be found in[4, Appendix F].

4 Discrete Riemannian structures

Although Riemannian Voronoi diagrams and Delaunay triangulations are appealing from a theoretical point of view, they are very difficult to compute in practice despite many studies [21]. To circumvent this difficulty, we introduce the discrete Riemannian Voronoi diagram. This discrete structure is easy to compute (see our companion paper [23] for details) and, as will be shown in the following sections, it is a good approximation of the exact Riemannian Voronoi diagram. In particular, their dual Delaunay structures are identical under appropriate conditions.

We assume that we are given a dense triangulation of the domain Ω we call the *canvas* and denote by $\mathcal{C}$. The canvas will be used to approximate geodesic distances between points of Ω and to construct the discrete Riemannian Voronoi diagram of $\mathcal{P}$. This bears some resemblance to the graph-induced complex of Dey et al. [11]. Notions related to the canvas will explicitly carry *canvas* in the name (for example, an edge of $\mathcal{C}$ is a *canvas edge*). In our analysis, we shall assume that the canvas is a dense triangulation, although weaker and more efficient structures can be used (see Section 9 and [23]).

4.1 The discrete Riemannian Voronoi Diagram

To define the discrete Riemannian Voronoi diagram of $\mathcal{P}$, we need to give a unique color to each site of $\mathcal{P}$ and to color the vertices of the canvas accordingly. Specifically, each canvas vertex is colored with the color of its closest site.

▶ **Definition 5** (Discrete Riemannian Voronoi diagram)**.** Given a metric field g, we associate to each site p_i its *discrete cell* $\mathrm{V}_g^{\mathrm{d}}(p_i)$ defined as the union of all canvas simplices with at least one vertex of the color of p_i. We call the set of these cells the *discrete Riemannian Voronoi diagram* of $\mathcal{P}$, and denote it by $\mathrm{Vor}_g^{\mathrm{d}}(\mathcal{P})$.

Observe that contrary to typical Voronoi diagrams, our discrete Riemannian Voronoi diagram is not a partition of the canvas. Indeed, there is a one canvas simplex-thick overlapping since each canvas simplex $\sigma_{\mathcal{C}}$ belongs to all the Voronoi cells whose sites' colors appear in the vertices of $\sigma_{\mathcal{C}}$. This is intentional and allows for a straightforward definition of the complex induced by this diagram, as shown below.

4.2 The discrete Riemannian Delaunay complex

We define the *discrete Riemannian Delaunay complex* as the set of simplices $\mathrm{Del}_g^{\mathrm{d}}(\mathcal{P}) = \{\sigma \mid \mathrm{Vert}(\sigma) \in \mathcal{P}, \cap_{p\in\sigma} \mathrm{V}_g^{\mathrm{d}}(p) \neq 0\}$. Using a triangulation as canvas offers a very intuitive way to construct the discrete complex since each canvas k-simplex σ of $\mathcal{C}$ has $k+1$ vertices $\{v_0, \ldots, v_k\}$ with respective colors $\{c_0, \ldots, c_k\}$ corresponding to the sites $\{p_{c_0}, \ldots, p_{c_k}\} \in \mathcal{P}$. Due to the way discrete Voronoi cells overlap, a canvas simplex $\sigma_{\mathcal{C}}$ belongs to each discrete Voronoi cell whose color appears in the vertices of σ. Therefore, the intersection of the discrete Voronoi cells $\{V_g^{\mathrm{d}}(p_i)\}$ whose colors appear in the vertices of σ is non-empty and the simplex σ with vertices $\{p_i\}$ thus belongs to the discrete Riemannian Delaunay complex. In that case, we say that the canvas simplex $\sigma_{\mathcal{C}}$ *witnesses* (or is a witness of) σ. For example, if the vertices of a canvas 3-simplex $\tau_{\mathcal{C}}$ have colors yellow–blue–blue–yellow, then the intersection of the discrete Voronoi cells of the sites p_{yellow} and p_{blue} is non-empty and the one-simplex σ with vertices p_{yellow} and p_{blue} belongs to the discrete Riemannian Delaunay complex. The canvas simplex $\tau_{\mathcal{C}}$ thus witnesses the (abstract, for now) edge between p_{yellow} and p_{blue}.

Figure 2 illustrates a canvas painted with discrete Voronoi cells, and the witnesses of the discrete Riemannian Delaunay complex.

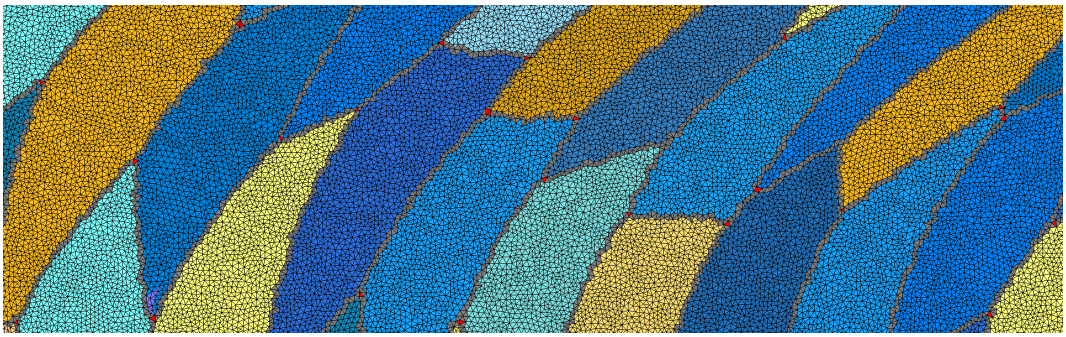


Figure 2 A canvas (black edges) and a discrete Riemannian Voronoi diagram drawn on it. The canvas simplices colored in red are witnesses of Voronoi vertices. The canvas simplices colored in grey are witnesses of Voronoi edges. Canvas simplices whose vertices all have the same color are colored with that color.

▶ Remark. If the intersection $\bigcap_{i=0\dots k} \mathrm{V}^{\mathrm{d}}_g(p_{c_i})$ is non-empty, then the intersection of any subset of $\{\mathrm{V}^{\mathrm{d}}_g(p_{c_i})\}_{i=0\dots k}$ is non-empty. In other words, if a canvas simplex $\sigma_{\mathcal{C}}$ witnesses a simplex σ, then for each face τ of σ, there exists a face $\tau_{\mathcal{C}}$ of $\sigma_{\mathcal{C}}$ that witnesses τ. As we assume that there is no boundary, the complex is pure and it is sufficient to only consider canvas n-simplices whose vertices have all different colors to build $\mathrm{Del}^{\mathrm{d}}_g(\mathcal{P})$.

Similarly to the definition of curved and straight Riemannian Delaunay complexes, we can define their discrete counterparts we respectively denote by $\widetilde{\mathrm{Del}}^{\mathrm{d}}_g(\mathcal{P})$ and $\overline{\mathrm{Del}}^{\mathrm{d}}_g(\mathcal{P})$. We will now exhibit conditions such that these complexes are well-defined and embedded in Ω.

5 Equivalence between the discrete and the exact structures

We first give conditions such that $\mathrm{Vor}^{\mathrm{d}}_g(\mathcal{P})$ and $\mathrm{Vor}_g(\mathcal{P})$ have the same combinatorial structure, or, equivalently, that the dual Delaunay complexes $\mathrm{Del}_g(\mathcal{P})$ and $\mathrm{Del}^{\mathrm{d}}_g(\mathcal{P})$ are identical. Under these conditions, the fact that $\mathrm{Del}^{\mathrm{d}}_g(\mathcal{P})$ is embedded in Ω will immediately follow from the fact that the exact Riemannian Delaunay complex $\mathrm{Del}_g(\mathcal{P})$ is embedded (see Sections 3.1 and 3.2). It thus remains to exhibit conditions under which $\mathrm{Del}^{\mathrm{d}}_g(\mathcal{P})$ and $\mathrm{Del}_g(\mathcal{P})$ are identical.

Requirements will be needed on both the set of sites in terms of density, sparsity and protection, and on the density of the canvas. The central idea in our analysis is that power protection of $\mathcal{P}$ will imply a lower bound on the distance separating two non-adjacent Voronoi objects (and in particular two Voronoi vertices). From this lower bound, we will obtain an upper bound on the size on the cells of the canvas so that the combinatorial structure of the discrete diagram is the same as that of the exact one. The density of the canvas is expressed by $e_{\mathcal{C}}$, the length of its longest edge.

The main result of this paper is the following theorem.

▶ **Theorem 6.** *Assume that $\mathcal{P}$ is a δ-power protected (ε, μ)-net in Ω with respect to g. Assume further that ε is sufficiently small and δ is sufficiently large compared to the distortion between $g(p)$ and g in an ε-neighborhood of p. Let $\{\lambda_i\}$ be the eigenvalues of $g(p)$ and ℓ_0 a value that depends on ε and δ (Precise bounds for ε, δ and l_0 are given in the proof). Then, if $e_{\mathcal{C}} < \min_{p\in\mathcal{P}} \left[\min_i \left(\sqrt{\lambda_i}\right) \min\{\mu/3, \ell_0/2\}\right]$, $\mathrm{Del}^{d}_g(\mathcal{P}) = \mathrm{Del}_g(\mathcal{P})$.*

The rest of the paper will be devoted to the proof of this theorem. Our analysis is divided into two parts. We first consider in Section 6 the most basic case of a domain of $\mathbb{R}^n$ endowed

with the Euclidean metric field. The result is given by Theorem 7. The assumptions are then relaxed and we consider the case of an arbitrary metric field over Ω in Section 7. As we shall see, the Euclidean case already contains most of the difficulties that arise during the proof and the extension to more complex settings will be deduced from the Euclidean case by bounding the distortion.

6 Equality of the Riemannian Delaunay complexes in the Euclidean setting

In this section, we restrict ourselves to the case where the metric field is the Euclidean metric $g_{\mathbb{E}}$. To simplify matters, we initially assume that geodesic distances are computed exactly on the canvas. The following theorem gives sufficient conditions to have equality of the complexes.

▶ **Theorem 7.** *Assume that $\mathcal{P}$ is a δ-power protected (ε, μ)-net of Ω with respect to the Euclidean metric field $g_{\mathbb{E}}$. Denote by $\mathcal{C}$ the canvas, a triangulation with maximal edge length $e_{\mathcal{C}}$. If $e_{\mathcal{C}} < \min\left\{\mu/16, \delta^2/64\varepsilon\right\}$, then $\mathrm{Del}^d_{\mathbb{E}}(\mathcal{P}) = \mathrm{Del}_{\mathbb{E}}(\mathcal{P})$.*

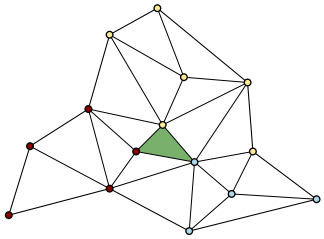
We shall now prove Theorem 7 by enforcing the two following conditions which, combined, give the equality between the discrete Riemannian Delaunay complex and the Riemannian Delaunay complex:

1. for every Voronoi vertex in the Riemannian Voronoi diagram $v = \cap_{\{p_i\}} V_g(p_i)$, there exists at least one canvas simplex with the corresponding colors $\{c_{p_i}\}$;
2. no canvas simplex witnesses a simplex that does not belong to the Riemannian Delaunay complex (equivalently, no canvas simplex has vertices whose colors are those of non-adjacent Riemannian Voronoi cells).

Condition 2 is a consequence of the separation of Voronoi objects, which in turn follows from power protection. The separation of Voronoi objects has previously been studied, for example by Boissonnat et al. [2]. Although the philosophy is the same, our setting is slightly more difficult and the results using power protection are new and use a more geometrical approach (see [4, Appendix C]).

6.1 Sperner's Lemma

Rephrasing Condition 1, we seek requirements on the density of the canvas $\mathcal{C}$ and on the nature of the point set $\mathcal{P}$ such that there exists at least one canvas n-simplex of $\mathcal{C}$ that has exactly the colors $c_0, \ldots, c_d$ of the vertices $p_0, \ldots, p_d$ of a simplex σ, for all $\sigma \in \mathrm{Del}_g(\mathcal{P})$. To prove the existence of such a canvas simplex, we employ Sperner's lemma [25], which is a discrete analog of Brouwer's fixed point theorem. We recall this result in Theorem 8 and illustrate it in a two-dimensional setting (inset).



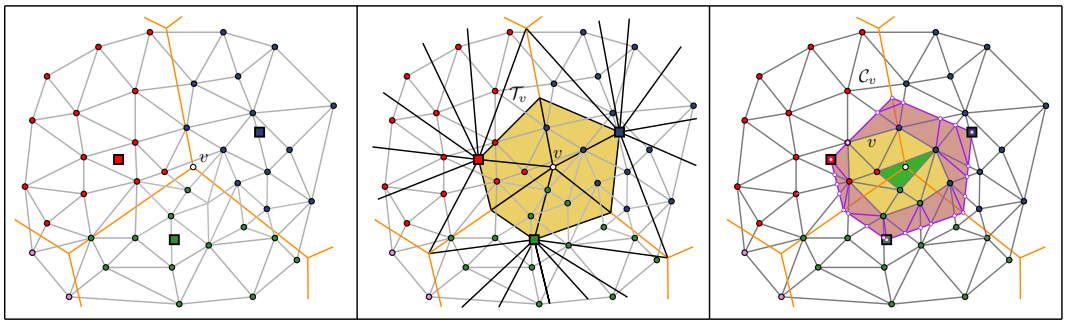


Figure 3 Illustration of the construction of $\mathcal{C}_v$. The Riemannian Voronoi diagram is drawn with thick orange edges and the sites are colored squares. The canvas is drawn with thin gray edges and colored circular vertices. The middle frame shows the subdivision of the incident Voronoi cells with think black edges and the triangulation $\mathcal{T}_v$ is drawn in yellow. On the right frame, the set of simplices $\mathcal{C}_v$ is colored in purple (simplices that do not belong to $\mathcal{C}$) and in dark yellow (simplices that belong to $\mathcal{C}$).

▶ **Theorem 8** (Sperner's Lemma). *Let $\sigma = (p_0, \dots, p_n)$ be an n-simplex and let T_σ denote a triangulation of the simplex. Let each vertex $v' \in T_\sigma$ be colored such that the following conditions are satisfied:*

- *The vertices p_i of σ all have different colors.*
- *If a vertex p' lies on a k-face $(p_{i_0}, \dots p_{i_k})$ of σ, then p' has the same color as one of the vertices of the face, that is p_{i_j}.*

Then, there exists an odd number of simplices in T_σ whose vertices are colored with all $n+1$ colors. In particular, there must be at least one.

We shall apply Sperner's lemma to the canvas $\mathcal{C}$ and show that for every Voronoi vertex v in the Riemannian Voronoi diagram, we can find a subset $\mathcal{C}_v$ of the canvas that fulfills the assumptions of Sperner's lemma, hence obtaining the existence of a canvas simplex in $\mathcal{C}_v$ (and therefore in $\mathcal{C}$) that witnesses σ_v. Concretely, the subset $\mathcal{C}_v$ is obtained in two steps:

- We first apply a barycentric subdivision of the Riemannian Voronoi cells incident to v. From the resulting set of simplices, we extract a triangulation $\mathcal{T}_v$ composed of the simplices incident to v (Section 6.2).
- We then construct the subset $\mathcal{C}_v$ by overlaying the border of $\mathcal{T}_v$ and the canvas (Section 6.3).

We then show that if the canvas simplices are small enough – in terms of edge length – then $\mathcal{C}_v$ is the triangulation of a simplex that satisfies the assumptions of Sperner's lemma.

The construction of $\mathcal{C}_v$ is detailed in the following sections and illustrated in Figure 3: starting from a colored canvas (left), we subdivide the incident Voronoi cells of v to obtain $\mathcal{T}_v$ (middle), and deduce the set of canvas simplices $\mathcal{C}_v$ which forms a triangulation that satisfies the hypotheses of Sperner's lemma, thus giving the existence of a canvas simplex (in green, right) that witnesses the Voronoi vertex within the union of the simplices, and therefore in the canvas.

6.2 The triangulation $\mathcal{T}_v$

For a given Voronoi vertex v in the Euclidean Voronoi diagram $\mathrm{Vor}_{\mathbb{E}}(\mathcal{P})$ of the domain Ω, the initial triangulation $\mathcal{T}_v$ is obtained by applying a combinatorial barycentric subdivision of the Voronoi cells of $\mathrm{Vor}_{\mathbb{E}}(\mathcal{P})$ that are incident to v: to each Voronoi cell V incident to v, we associate to each face F of V a point c_F in F which is not necessarily the geometric barycenter. We randomly associate to c_F the color of any of the sites whose Voronoi cells intersect to

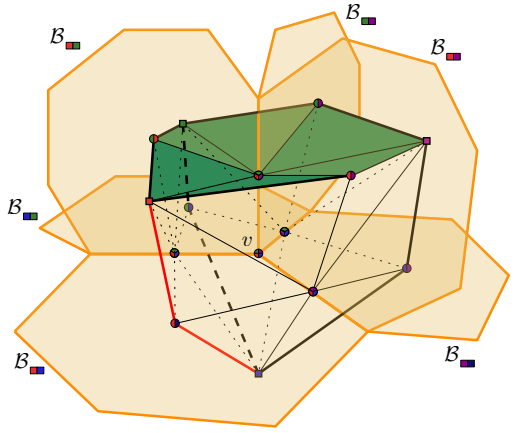


Figure 4 The triangulation $\mathcal{T}_v$ in 3D. A face (in green) and an edge (in red) of $\sigma_{\mathcal{S}}$.

give F. For example, in a two-dimensional setting, if the face F is a Voronoi edge that is the intersection of V_{red} and V_{blue}, then c_F is colored either red or blue. Then, the subdivision of V is computed by associating to all possible sequences of faces $\{F_0, F_1, \dots F_{n-1}, F_n\}$ such that $F_0 \subset F_1 \cdots \subset F_n = V$ and $\dim(F_{i+1}) = \dim(F_i) + 1$ the simplex with vertices $\{c_{F_0}, c_{F_1}, \dots, c_{F_{n-1}}, c_{F_n}\}$. These barycentric subdivisions are allowed since Voronoi cells are convex polytopes.

Denote by Σ_V the set of simplices obtained by barycentric subdivision of V and $\Sigma_v = \{\cup\Sigma_V \mid v \in V\}$. The triangulation $\mathcal{T}_v$ is defined as the star of v in Σ_v, that is the set of simplices in Σ_v that are incident to v. $\mathcal{T}_v$ is illustrated in Figure 4 in dimension 3. As shall be proven in Lemma 9, $\mathcal{T}_v$ can be used to define a combinatorial simplex that satisfies the assumptions of Sperner's lemma.

$\mathcal{T}_v$ as a triangulation of an n-simplex

By construction, the triangulation $\mathcal{T}_v$ is a triangulation of the (Euclidean) Delaunay simplex σ_v dual of v as follows. We first perform the standard barycentric subdivision on this Delaunay simplex σ_v. We then map the barycenter of a k-face τ of σ_v to the point c_{F_i} on the Voronoi face F_i, where F_i is the Voronoi dual of the k-face τ. This gives a piecewise linear homeomorphism from the Delaunay simplex σ_v to the triangulation $\mathcal{T}_v$. We call the image of this map the simplex $\sigma_{\mathcal{S}}$ and refer to the images of the faces of the Delaunay simplex as the faces of $\sigma_{\mathcal{S}}$. We can now apply Sperner's lemma.

▶ **Lemma 9.** *Let $\mathcal{P}$ be a δ-power protected (ε, μ)-net. Let v be a Voronoi vertex in the Euclidean Voronoi diagram, $\mathrm{Vor}_{\mathbb{E}}(\mathcal{P})$, and let Σ_v be defined as above. The simplex $\sigma_{\mathcal{S}}$ and the triangulation $\mathcal{T}_v$ satisfy the assumptions of Sperner's lemma in dimension n.*

Proof. By the piecewise linear map that we have described above, $\mathcal{T}_v$ is a triangulation of the simplex $\sigma_{\mathcal{S}}$. Because by construction the vertices c_{F_i} lie on the Voronoi duals F_i of the corresponding Delaunay face τ, c_{F_i} has the one of the colors of of the Delaunay vertices of τ. Therefore, $\sigma_{\mathcal{S}}$ satisfies the assumptions of Sperner's lemma and there exists an n-simplex in $\mathcal{T}_v$ that witnesses v and its corresponding simplex σ_v in $\mathrm{Del}_g(\mathcal{P})$. ◀

6.3 Building the triangulation $\mathcal{C}_v$

Let p_i be the vertices of the k-face $\tau_\mathcal{S}$ of $\sigma_\mathcal{S}$. In this section we shall assume not only that $\tau_\mathcal{S}$ is contained in the union of the Voronoi cells of $V(p_i)$, but in fact that $\tau_\mathcal{S}$ is a distance $8e_\mathcal{C}$ removed from the boundary of $\cup V(p_i)$, where $e_\mathcal{C}$ is the longest edge length of a simplex in the canvas. We will now construct a triangulation $\mathcal{C}_v$ of $\sigma_\mathcal{S}$ such that:

- $\sigma_\mathcal{S}$ and its triangulation $\mathcal{C}_v$ satisfy the conditions of Sperner's lemma,
- the simplices of $\mathcal{C}_v$ that have no vertex that lies on the boundary $\partial\sigma_\mathcal{S}$ are simplices of the canvas $\mathcal{C}$.

The construction goes as follows. We first intersect the canvas $\mathcal{C}$ with $\sigma_\mathcal{S}$ and consider the canvas simplices $\sigma_{\mathcal{C},i}$ such that the intersection of $\sigma_\mathcal{S}$ and $\sigma_{\mathcal{C},i}$ is non-empty. These simplices $\sigma_{\mathcal{C},i}$ can be subdivided into two sets, namely those that lie entirely in the interior of $\sigma_\mathcal{S}$, which we denote by $\sigma^{\text{int}}_{\mathcal{C},i}$, and those that intersect the boundary, denoted by $\sigma^{\partial}_{\mathcal{C},i}$.

The simplices $\sigma^{\text{int}}_{\mathcal{C},i}$ are added to the set $\mathcal{C}_v$. We intersect the simplices $\sigma^{\partial}_{\mathcal{C},i}$ with $\sigma_\mathcal{S}$ and triangulate the intersection. Note that $\sigma^{\partial}_{\mathcal{C},i} \cap \sigma_\mathcal{S}$ is a convex polyhedron and thus triangulating it is not a difficult task. The vertices of the simplices in the triangulation of $\sigma^{\partial}_{\mathcal{C},i} \cap \sigma_\mathcal{S}$ are colored according to which Voronoi cell they belong to. Finally, the simplices in the triangulation of $\sigma^{\partial}_{\mathcal{C},i} \cap \sigma_\mathcal{S}$ are added to the set $\mathcal{C}_v$.

Since $\mathcal{T}_v$ is a triangulation of $\sigma_\mathcal{S}$, the set $\mathcal{C}_v$ is by construction also a triangulation of $\sigma_\mathcal{S}$. This triangulation trivially gives a triangulation of the faces $\tau_\mathcal{S}$. Because we assume that $\tau_\mathcal{S}$ is contained in the union of its Voronoi cells, with a margin of $8e_\mathcal{C}$ we now can draw two important conclusions:

- The vertices of the triangulation of each face $\tau_\mathcal{S}$ have the colors of the vertices p_i of $\tau_\mathcal{S}$.
- None of the simplices in the triangulation of $\sigma^{\partial}_{\mathcal{C},i} \cap \sigma_\mathcal{S}$ can have $n+1$ colors, because every such simplex must be close to one face $\tau_\mathcal{S}$, which means that it must be contained in the union of the Voronoi cells $V(p_i)$ of the vertices of $\tau_\mathcal{S}$.

We can now invoke Sperner's lemma; $\mathcal{C}_v$ is a triangulation of the simplex $\sigma_\mathcal{S}$ whose every face has been colored with the appropriate colors (since $\sigma_\mathcal{S}$ triangulated by $\mathcal{T}_v$ satisfies the assumptions of Sperner's lemma, see Lemma 9). This means that there is a simplex $\mathcal{C}_v$ that is colored with $n+1$ colors. Because of our second observation above, the simplex with these $n+1$ colors must lie in the interior of $\sigma_\mathcal{S}$ and is thus a canvas simplex.

We summarize by the following lemma:

▶ **Lemma 10.** *If every face $\tau_\mathcal{S}$ of $\sigma_\mathcal{S}$ with vertices p_i is at distance $8e_\mathcal{C}$ from the boundary of the union of its Voronoi cells $\partial(\cup V(p_i))$, then there exists a canvas simplex in $\mathcal{C}_v$ such that it is colored with the same vertices as the vertices of $\sigma_\mathcal{S}$.*

The key task that we now face is to guarantee that faces $\tau_\mathcal{S}$ indeed lie well inside of the union of the appropriate Voronoi regions. This requires first and foremost power protection. Indeed, if a point set is power protected, the distance between a Voronoi vertex c and the Voronoi faces that are not incident to c, which we will refer to from now on as *foreign* Voronoi faces, can be bounded, as shown in the following Lemma:

▶ **Lemma 11.** *Suppose that c is the circumcenter of a δ-power protected simplex σ of a Delaunay triangulation built from an ε-sample, then all foreign Voronoi faces are at least $\delta^2/8\varepsilon$ far from c.*

The proof of this Lemma is given in the full version of this paper (see [4, Section C.2]).

In almost all cases, this result gives us the distance bound we require: we can assume that vertices $\{c_{F_0}, c_{F_1}, \ldots, c_{F_{n-1}}, c_{F_n}\}$ which we used to construct $\mathcal{T}_v$, are well placed, meaning

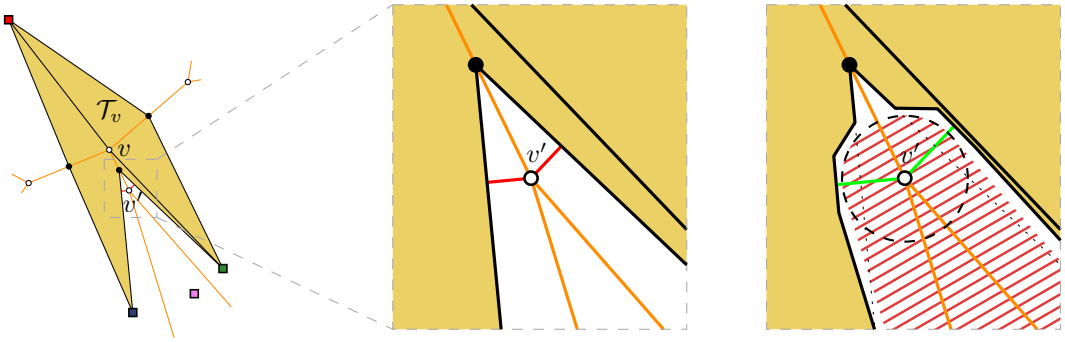


Figure 5 The point v' can be arbitrarily close to $\mathcal{T}_v$, as shown by the red segments (left and center). After piecewise linear deformation, this issue is resolved, as seen by the green segments (right).

that there is a minimum distance between these vertices and foreign Voronoi objects. However it can still occur that foreign Voronoi objects are close to a face $\tau_{\mathcal{S}}$ of $\sigma_{\mathcal{S}}$. This occurs even in two dimensions, where a Voronoi vertex v' can be very close to a face $\tau_{\mathcal{S}}$ because of obtuse angles, as illustrated in Figure 5.

Thanks to power protection, we know that v' is removed from foreign Voronoi objects. This means that we can deform $\sigma_{\mathcal{S}}$ (in a piecewise linear manner) in a neighborhood of v' such that the distance between v' and all the faces of the deformed $\sigma_{\mathcal{S}}$ is lower bounded.

In general the deformation of $\sigma_{\mathcal{S}}$ is performed by "radially pushing" simplices away from the foreign Voronoi faces of v with a ball of radius $r = \min\left\{\mu/16, \delta^2/64\varepsilon\right\}$. The value $\mu/16$ is chosen so that we do not move any vertex of σ_v (the dual of v): indeed, $\mathcal{P}$ is μ-separated and thus $d_{\mathbb{E}}(p_i, p_j) > \mu$. The value $\delta^2/64\varepsilon$ is chosen so that $\sigma_{\mathcal{S}}$ and its deformation stay isotopic (no "pinching" can happen), using Lemma 11. In fact it is advisable to use a piecewise linear version of "radial pushing", to ensure that the deformation of $\sigma_{\mathcal{S}}$ is a polyhedron. This guarantees that we can triangulate the intersection, see Chapter 2 of Rourke and Sanderson [22]. After this deformation we can follow the steps we have given above to arrive at a well-colored simplex.

▶ **Lemma 12.** *Let $\mathcal{P}$ be a δ-power protected (ε, μ)-net. Let v be a Voronoi vertex of the Euclidean Voronoi diagram $\mathrm{Vor}_{\mathbb{E}}(\mathcal{P})$, and $\mathcal{T}_v$ as defined above. If the length $e_{\mathcal{C}}$ of the longest canvas edge is bounded as follows: $e_{\mathcal{C}} < r = \min\left\{\mu/16, \delta^2/64\varepsilon\right\}$, then there exists a canvas simplex that witnesses v and the corresponding simplex σ_v in $\mathrm{Del}_{\mathbb{E}}(\mathcal{P})$.*

Conclusion

So far, we have only proven that $\mathrm{Del}_g(\mathcal{P}) \subseteq \mathrm{Del}_g^{\mathrm{d}}(\mathcal{P})$. The other inclusion, which corresponds to Condition 2 mentioned above, is much simpler: as long as a canvas edge is shorter than the smallest distance between a Voronoi vertex and a foreign face of the Riemannian Voronoi diagram, then no canvas simplex can witness a simplex that is not in $\mathrm{Del}_g(\mathcal{P})$. Such a bound is already given by Lemma 11 and thus, if $e_{\mathcal{C}} < \delta^2/8\varepsilon$ then $\mathrm{Del}_g^{\mathrm{d}}(\mathcal{P}) \subseteq \mathrm{Del}_g(\mathcal{P})$. Observe that this requirement is weaker than the condition imposed in Lemma 12 and it was thus already satisfied. It follows that $\mathrm{Del}_g^{\mathrm{d}}(\mathcal{P}) = \mathrm{Del}_g(\mathcal{P})$ if $e_{\mathcal{C}} < \min\left\{\mu/16, \delta^2/64\varepsilon\right\}$, which concludes the proof of Theorem 7.

▶ Remark. Assuming that the point set is a δ-power protected (ε, μ)-net might seem like a strong assumption. However, it should be observed that any non-degenerate point set can be

seen as a δ-power protected (ε, μ)-net, for a sufficiently large value of ε and sufficiently small values of δ and μ. Our results are therefore always applicable but the necessary canvas density increases as the quality of the point set worsens (Lemma 12). In our practical companion paper [23, Section 7], we showed how to generate δ-power protected (ε, μ)-nets for given values of ε, μ and δ.

7 Extension to more complex settings

In the previous section, we have placed ourselves in the setting of an (open) domain endowed with the Euclidean metric field. To prove Theorem 6, we need to generalize Theorem 7 to more general metrics, which will be done in the two following subsections.

The common path to prove $\mathrm{Del}_g^{\mathrm{d}}(\mathcal{P}) = \mathrm{Del}_g(\mathcal{P})$ in all settings is to assume that $\mathcal{P}$ is a power protected net with respect to the metric field. We then use the stability of entities under small metric perturbations to take us back to the now solved case of the domain Ω endowed with an Euclidean metric field. Separation and stability of Delaunay and Voronoi objects has previously been studied by Boissonnat et al. [2, 3], but our work lives in a slightly more complicated setting. Moreover, our proofs are generally more geometrical and sometimes simpler. For completeness, the extensions of these results to our context are detailed in the full version of this paper [4, Appendices C and E].

We now detail the different intermediary settings. For completeness, the full proofs are included in the appendices.

7.1 Uniform metric field

We first consider the rather easy case of a non-Euclidean but uniform (constant) metric field over an (open) domain. The square root of a metric gives a linear transformation between the base space where distances are considered in the metric and a *metric space* where the Euclidean distance is used (see [4, Appendix B.1]). Additionally, we show that a δ-power protected (ε, μ)-net with respect to the uniform metric is, after transformation, still a δ-power protected (ε, μ)-net but with respect to the Euclidean setting [4, Lemma 26], bringing us back to the setting we have solved in Section 6. Bounds on the power protection, sampling and separation coefficients, and on the canvas edge length can then be obtained from the result for the Euclidean setting, using Theorem 12. These bounds can be transported back to the case of uniform metric fields by scaling these values according to the smallest eigenvalue of the metric [4, Theorem 40].

7.2 Arbitrary metric field

The case of an arbitrary metric field over Ω is handled by observing that an arbitrary metric field is locally well-approximated by a uniform metric field. It is then a matter of controlling the distortion.

We first show that, for any point $p \in \Omega$, density separation and power protection are locally preserved in a neighborhood U_p around p when the metric field g is approximated by the constant metric field $g' = g(p)$ [4, Lemmas 27 and 39]: if $\mathcal{P}$ is a δ-power protected (ε, μ)-net with respect to g, then $\mathcal{P}$ is a δ'-power protected (ε', μ')-net with respect to g'. Previous results can now be applied to obtain conditions on δ', ε', μ' and on the (local) maximal length of the canvas such that $\mathrm{Del}_g^{\mathrm{d}}(\mathcal{P}) = \mathrm{Del}_g(\mathcal{P})$ (see [4, Lemma 41]).

These local triangulations can then be stitched together to form a triangulation embedded in Ω. The (global) bound on the maximal canvas edge length is given by the minimum of

the local bounds, each computed through the results of the previous sections. This ends the proof of Theorem 6.

Once the equality between the complexes is obtained, conditions giving the embeddability of the discrete Karcher Delaunay triangulation and the discrete straight Delaunay triangulation are given by previous results that we have established in Sections 3.1 and 3.2 respectively.

8 Extensions of the main result

Approximate geodesic computations. Approximate geodesic distance computations can be incorporated in the analysis of the previous section by observing that computing inaccurately geodesic distances in a domain Ω endowed with a metric field g can be seen as computing exactly geodesic distances in Ω with respect to a metric field g' that is close to g [4, Section H.3].

General manifolds. The previous section may also be generalized to an arbitrary smooth n-manifold $\mathcal{M}$ embedded in $\mathbb{R}^m$. We shall assume that, apart from the metric induced by the embedding of the domain in Euclidean space, there is a second metric g defined on $\mathcal{M}$. Let $\pi_p : \mathcal{M} \to T_p\mathcal{M}$ be the orthogonal projection of points of $\mathcal{M}$ on the tangent space $T_p\mathcal{M}$ at p. For a sufficiently small neighborhood $U_p \subset T_p\mathcal{M}$, π_p is a local diffeomorphism (see Niyogi [20]).

Denote by $\mathcal{P}_{T_p}$ the point set $\{\pi_p(p_i), p_i \in \mathcal{P}\}$ and $\mathcal{P}_{U_p}$ the restriction of $\mathcal{P}_{T_p}$ to U_p. Assuming that the conditions of Niyogi et al. [20] are satisfied (which are simple density constraints on ε compared to the reach of the manifold), the pullback of the metric with the inverse projection $(\pi_p^{-1})^* g$ defines a metric g_p on U_p such that for all $q, r \in U_p$, $d_{g_p}(q, r) = d_g(\pi_p^{-1}(q), \pi_p^{-1}(r))$. This implies immediately that if $\mathcal{P}$ is a δ-power protected (ε, μ)-net on $\mathcal{M}$ with respect to g then $\mathcal{P}_{U_p}$ is a δ-power protected (ε, μ)-net on U_p. We have thus a metric on a subset of a n-dimensional space, in this case the tangent space, giving us a setting that we have already solved. It is left to translate the sizing field requirement from the tangent plane to the manifold $\mathcal{M}$ itself. Note that the transformation π_p is completely independent of g. Boissonnat et al. [2, Lemma 3.7] give bounds on the metric distortion of the projection on the tangent space. This result allows to carry the canvas sizing field requirement from the tangent space to $\mathcal{M}$.

9 Implementation

The construction of the discrete Riemannian Voronoi diagram and of the discrete Riemannian Delaunay complex has been implemented for $n = 2, 3$ and for surfaces of $\mathbb{R}^3$. An in-depth description of our structure and its construction as well as an empirical study can be found in our practical paper [23]. We simply make a few observations here.

The theoretical bounds on the canvas edge length provided by Theorems 6 and 7 are far from tight and thankfully do not need to be honored in practice. A canvas whose edge length are about a tenth of the distance between two seeds suffices. This creates nevertheless unnecessarily dense canvasses since the density does not in fact need to be equal everywhere at all points and even in all directions. This issue is resolved by the use of anisotropic canvasses.

Our analysis was based on the assumption that all canvas vertices are painted with the color of the closest site. In our implementation, we color the canvas using a multiple-front

vector Dijkstra algorithm [5], which empirically does not suffer from the same convergence issues as the traditional Dijkstra algorithm, starting from all the sites. It should be noted that any geodesic distance computation method can be used, as long as it converges to the exact geodesic distance when the canvas becomes denser. The Riemannian Delaunay complex is built on the fly during the construction of the discrete Riemannian Voronoi diagram: when a canvas simplex is first fully colored, its combinatorial information is extracted and the corresponding simplex is added to $\mathrm{Del}_g(\mathcal{P})$.

Acknowledgments. We thank Ramsay Dyer for enlightening discussions.

References

1 F. Aurenhammer and R. Klein. Voronoi diagrams. In J. Sack and G. Urrutia, editors, *Handbook of Computational Geometry*, pages 201–290. Elsevier Science Publishing, 2000.

2 J.-D. Boissonnat, R. Dyer, and A. Ghosh. Delaunay triangulation of manifolds. *Foundations of Computational Mathematics*, pages 1–33, 2017.

3 J.-D. Boissonnat, R. Dyer, A. Ghosh, and S. Y. Oudot. Only distances are required to reconstruct submanifolds. *Comp. Geom. Theory and Appl.*, 2016. To appear.

4 J.-D. Boissonnat, M. Rouxel-Labbé, and M. Wintraecken. Anisotropic triangulations via discrete Riemannian Voronoi diagrams, 2017. URL: `https://arxiv.org/abs/1703.06487`.

5 M. Campen, M. Heistermann, and L. Kobbelt. Practical anisotropic geodesy. In *Proceedings of the Eleventh Eurographics/ACMSIGGRAPH Symposium on Geometry Processing*, SGP'13, pages 63–71. Eurographics Association, 2013.

6 G. D. Cañas and S. J. Gortler. Orphan-free anisotropic Voronoi diagrams. *Discrete and Computational Geometry*, 46(3), 2011.

7 G. D. Cañas and S. J. Gortler. Duals of orphan-free anisotropic Voronoi diagrams are embedded meshes. In *SoCG*, pages 219–228. ACM, 2012.

8 T. Cao, H. Edelsbrunner, and T. Tan. Proof of correctness of the digital Delaunay triangulation algorithm. *Comp. Geo.: Theory and Applications*, 48, 2015.

9 S.-W. Cheng, T. K. Dey, E. A. Ramos, and R. Wenger. Anisotropic surface meshing. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 202–211. Society for Industrial and Applied Mathematics, 2006.

10 E. F. D'Azevedo and R. B. Simpson. On optimal interpolation triangle incidences. *SIAM J. Sci. Statist. Comput.*, 10(6):1063–1075, 1989.

11 T. K. Dey, F. Fan, and Y. Wang. Graph induced complex on point data. *Computational Geometry*, 48(8):575–588, 2015.

12 Q. Du and D. Wang. Anisotropic centroidal Voronoi tessellations and their applications. *SIAM Journal on Scientific Computing*, 26(3):737–761, 2005.

13 R. Dyer, G. Vegter, and M. Wintraecken. Riemannian simplices and triangulations. Preprint: arXiv:1406.3740, 2014.

14 R. Dyer, H. Zhang, and T. Möller. Surface sampling and the intrinsic Voronoi diagram. *Computer Graphics Forum*, 27(5):1393–1402, 2008.

15 M. Garland and P. S. Heckbert. Surface simplification using quadric error metrics. In *ACM SIGGRAPH*, pages 209–216, 1997.

16 H. Karcher. Riemannian center of mass and mollifier smoothing. *Communications on Pure and Applied Mathematics*, 30:509–541, 1977.

17 F. Labelle and J. R. Shewchuk. Anisotropic Voronoi diagrams and guaranteed-quality anisotropic mesh generation. In *SCG'03: Proceedings of the Nineteenth Annual Symposium on Computational Geometry*, pages 191–200. ACM, 2003.

18 G. Leibon. *Random Delaunay triangulations, the Thurston-Andreev theorem, and metric uniformization*. PhD thesis, UCSD, 1999.

19 J.-M. Mirebeau. Optimal meshes for finite elements of arbitrary order. *Constructive approximation*, 32(2):339–383, 2010.

20 P. Niyogi, S. Smale, and S. Weinberger. Finding the homology of submanifolds with high confidence from random samples. *Discrete & Comp. Geom.*, 39(1-3), 2008.

21 G. Peyré, M. Péchaud, R. Keriven, and L. D. Cohen. Geodesic methods in computer vision and graphics. *Found. Trends. Comput. Graph. Vis.*, 2010.

22 C. Rourke and B. Sanderson. *Introduction to piecewise-linear topology*. Springer Science & Business Media, 2012.

23 M. Rouxel-Labbé, M. Wintraecken, and J.-D. Boissonnat. Discretized Riemannian Delaunay triangulations. In *Proc. of the 25th Intern. Mesh. Round.* Elsevier, 2016.

24 J. R. Shewchuk. What is a good linear finite element? Interpolation, conditioning, anisotropy, and quality measures, Manuscript 2002.

25 E. Sperner. Fifty years of further development of a combinatorial lemma. *Numerical solution of highly nonlinear problems*, pages 183–197, 1980.

An Approximation Algorithm for the Art Gallery Problem*†

Édouard Bonnet[1] and Tillmann Miltzow[2]

1 **Institute for Computer Science and Control, Hungarian Academy of Sciences (MTA SZTAKI), Budapest, Hungary**
edouard.bonnet@lamsade.dauphine.fr

2 **Institute for Computer Science and Control, Hungarian Academy of Sciences (MTA SZTAKI), Budapest, Hungary**
t.miltzow@gmail.com

Abstract

Given a simple polygon $\mathcal{P}$ on n vertices, two points x, y in $\mathcal{P}$ are said to be visible to each other if the line segment between x and y is contained in $\mathcal{P}$. The Point Guard Art Gallery problem asks for a minimum-size set S such that every point in $\mathcal{P}$ is visible from a point in S. The set S is referred to as guards. Assuming integer coordinates and a specific general position on the vertices of $\mathcal{P}$, we present the first $O(\log \mathrm{OPT})$-approximation algorithm for the point guard problem. This algorithm combines ideas in papers of Efrat and Har-Peled [18] and Deshpande et al. [15, 16]. We also point out a mistake in the latter.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases computational geometry, art-gallery, approximation algorithm

Digital Object Identifier 10.4230/LIPIcs.SoCG.2017.20

1 Introduction

Given a simple polygon $\mathcal{P}$ on n vertices, two points x, y in $\mathcal{P}$ are said to be visible to each other if the line segment between x and y is contained in $\mathcal{P}$. The point-guard art gallery problem asks for a minimum-size set S such that every point in $\mathcal{P}$ is visible from a point in S. The set S is referred to as guards.

Victor Klee introduced the art gallery problem to Václav Chvátal, who showed that $\lfloor n/3 \rfloor$ guards are always sufficient and sometimes necessary for a polygon with n vertices [11]. In 1978, Steve Fisk gave an elegant proof of the same result [21]. This constitutes the first combinatorial result related to the art gallery problem.

Related problems. A large amount of research is committed to the study of combinatorial and algorithmic aspects of the art gallery problem, as reflected by the following surveys [33, 34, 31]. This research is focused on the art gallery problem and its many variants, based on different definitions of visibility, restricted classes of polygons, different shapes and positions of guards, etc. The most natural definition of visibility is arguably the one we gave above. Other possible definitions are: x sees y if the axis-parallel rectangle spanned by x and y is contained in $\mathcal{P}$; x sees y if the line segment joining x to y intersects $\mathcal{P}$ at most c times,

* A full version of the paper is available at http://arxiv.org/abs/1607.05527.

† supported by the ERC grant PARAMTIGHT: "Parameterized complexity and the search for tight complexity results", no. 280152.

for some value of c; x sees y if there exists a straight-line path from x to y within $\mathcal{P}$ with at most c bends. Common shapes of polygons comprise: simple polygons, polygons with holes, simple orthogonal polygons, x-monotone polygons and star-shaped polygons. Common placements of guards include: vertex guards and point guards as defined above, but also edge-guard (guards are edges of the polygon), segment guards (guards are interior segments of the polygon) and perimeter guards (guards must be placed on the boundary of $\mathcal{P}$).

On the algorithmic side, very few variants are known to be solvable in polynomial time [30, 17] and most results are on approximating the minimum number of guards [15, 16, 23, 26, 27, 18, 28]. Many of the approximation algorithms are based on the fact that the range space defined by the visibility regions has bounded VC-dimension for simple polygons [24], combined with some algorithmic ideas of Clarkson [13, 8].

On the negative side, Eidenbenz et al. [19] showed NP-hardness and inapproximability for the most principal variants. In particular, they show that getting a PTAS for those variants is very unlikely, even on simple polygons. For polygons with holes, they even show that there is no $o(\log n)$-approximation algorithm, unless P=NP. Their reduction from Set Cover also implies that the art gallery problem is W[2]-hard on polygons with holes and that there is no $n^{o(k)}$ algorithm, to determine if k guards are sufficient, under the Exponential Time Hypothesis (ETH) [19, Sec.4]. Recently, the authors of the present paper show a similar result for simple polygons (i.e., without holes) [7].

Point Guard Art Gallery Problem. Notwithstanding the large amount of research on the art gallery problem, there is only one exact algorithmic result on the point guard variant. The result is not so well-known and attributed to Micha Sharir [18]: one can find in time $n^{O(k)}$ a set of k guards for the point guard variant, if it exists. This result is quite easy to achieve with some tools from real algebraic geometry [3] and seemingly hopeless to prove without this powerful machinery (see [4] for the very restricted case $k = 2$). Although the algorithm utilizes remarkably sophisticated tools, it uses almost no problem-specific insights and no better exact algorithm is known. Moreover, we recall that the papers [19, 7] suggest that there is no significantly better exact algorithm even for simple polygons.

Regarding *approximation* algorithms for the point guard variant, the results are similarly sparse. For general polygons, Deshpande et al. gave a randomized pseudo-polynomial time $O(\log n)$-approximation algorithm [15, 16]. However, we will see that their algorithm is not correct. Efrat and Har-Peled gave a randomized polynomial time $O(\log |OPT_{\text{grid}}|)$-approximation algorithm by restricting guards to a very fine grid [18]. However, they can not prove that their grid solution is indeed an approximation of an optimal guard placement. In this paper, we develop the ideas of Deshpande et al. in combination with the algorithm of Efrat and Har-Peled. Here, OPT denotes an optimal set of guards and OPT_{grid} an optimal set of guards that is restricted to some grid. Finally, let us mention that there exist approximation algorithms for monotone and rectilinear polygons [28], when the very restrictive structure of the polygon is exploited.

Lack of progress and motivation. Note that the art gallery problem can be seen as a geometric hitting set problem. In a hitting set problem, we are given a universe U and a set of subsets $S \subseteq 2^U$ and we are asked to find a smallest set $X \subseteq U$ such that $\forall s \in S\ \exists x \in X : x \in s$. Usually the set system is given explicitly or can be at least easily restricted to a set of polynomial size. In our case, the universe is the entire polygon (not just the boundary) and the set system is the set of visibility regions (given a point $x \in \mathcal{P}$, the visibility region $\text{Vis}(x)$ is defined as the set of points visible from x). The lack of progress

has come from the obvious yet crucial fact that the set system is *infinite* and that no one has found a way to restrict the universe to a finite set (see [12, 1] for some attempts). We also wish to quote a recent remark by Bhattiprolu and Har-Peled [5] both confirming that the point guard is the most principal variant and highlighting the challenge of finding an approximation algorithm: *"One of the more interesting versions of the geometric hitting set problem, is the art gallery problem, where one is given a simple polygon in the plane, and one has to select a set of points (inside or on the boundary of the polygon) that "see" the whole polygon. While much research has gone into variants of this problem [31], nothing is known as far as an approximation algorithm (for the general problem). The difficulty arises from the underlying set system being infinite, see [18] for some efforts in better understanding this problem."*

Besides theoretical considerations, there is a series of work to find efficient implementations to solve the art gallery problem in practice; see [14] for a survey on this large body of work. There as well the focus lies on the point guard variant. One of the key challenges is to find a discretization of the solution space, as was pointed out recently [22]: *"... a finite discretization whose existence in the AGP [(Art Gallery Problem)] is, to the best of our knowledge, still unknown and poses a key challenge w.r.t. software solving the AGP."* Although we cannot answer this question with respect to exact computation, we show that a fine enough grid is a sufficient discretization of the solution space with respect to constant-factor approximation; see Lemma 4. We also highlight certain fundamental problems related to solution-space discretization.

Our contribution. Recently Elbassioni showed how the framework of Brönnimann and Goodrich [8] can be extended to infinite range spaces, if one allows that some small δ-fraction of the ground set is not covered [20]. The main application of his paper is to yield an approximation algorithm for a variant of the point guard art gallery problem when one is allowed to guard only *almost all* the polygon. We show here how to achieve the same asymptotic approximation factor, while guarding the whole polygon. However, we rely on two assumptions on the gallery, which we detail below.

▶ **Assumption 1** (Integer Vertex Representation)**.** *Vertices are given by integers, represented in binary.*

An *extension* of a polygon $\mathcal{P}$ is a line that goes through two vertices of $\mathcal{P}$.

▶ **Assumption 2** (General Position Assumption)**.** *No three extensions meet in a point of $\mathcal{P}$ which is* not *a vertex and no three vertices are collinear.*

Note that we allow that three (or more) extensions meet in a vertex or outside the polygon.

No three points lie on a line is a typical assumption in computational geometry and discrete geometry. Often this assumption is a pure technicality. In some cases, however, the result might in fact be wrong without this assumption. In our case, we do believe that Lemma 4 could be proven without Assumption 2, but it seems that some new ideas would be needed. See [2] for an example where the main result is that some general position assumption can be weakened. The idea of general position assumptions is that a small random perturbation of the point set yields the assumption with probability almost 1. In case that the points are given by integers small random perturbations, destroy the integer property. But random perturbations could be performed in the following way: first multiply all coordinates by some large constant $2^C \in \mathbb{N}$ and then add a random integer x with $-C \leqslant x \leqslant C$.

The integer representation assumption (Assumption 1) seems to be very strong as it gives us useful distance bounds not just between any two different vertices of the polygon, but also between any two objects that do not share a point (see Lemma 8). On the other hand, real computers work with binary numbers and cannot compute real numbers with arbitrary precision. The real-RAM model was introduced as a convenient theoretical framework to simplify the analysis of algorithms with numerical and/or geometrical flavors, see for instance [25, page 1]. Also note that Assumption 1 can be replaced by assuming that all coordinates are represented by rational numbers with specified nominator and denominator. (There could be other potentially more compact ways to specify rational numbers.) Multiplying all numbers with the smallest common multiple of the denominators takes polynomial time, makes all numbers integers and does not change the geometry of the problem.

▶ **Theorem 3.** *Under Assumptions 1 and 2, there is a randomized approximation algorithm with approximation factor $O(\log |OPT|)$ for* Point Guard Art Gallery *for simple polygons. The running time is polynomial in the size of the input.*

The main technical idea is to show the following lemma:

▶ **Lemma 4** (Global Visibility Containment)**.** *Let $\mathcal{P}$ be some (not necessarily simple) polygon. Under Assumptions 1 and 2, it holds that there exists a grid Γ and a guard set $S_{grid} \subseteq \Gamma$, which sees the entire polygon and $|S_{grid}| = O(|S|)$, where S is an optimal guard set.*

To be a bit more precise, let M be the largest appearing integer. Then the number of points in Γ is polynomial in M. This is potentially exponential in the size of the input. Thus algorithms that rely on storing all points of Γ explicitly do not have polynomial worst case running time. The algorithm of Efrat and Har-Peled [18] does *not* store every point of Γ explicitly and, with the lemma above, the algorithm gives an $O(\log |OPT|)$-approximation on the grid Γ.

While Lemma 4 tells us that we can restrict our attention to a finite grid, when considering constant factor approximation, the same is not known for exact computation. In particular, it is not known whether the Point Guard problem lies in NP. Recently, some researchers popularized an interesting complexity class, called $\exists\mathbb{R}$, being somewhere between NP and PSPACE [10, 32, 9, 29]. Many geometric problems, for which membership in NP is uncertain, have been shown to be complete for this class. This suggests that there might be indeed no polynomial sized witness for these problems as this would imply $NP = \exists\mathbb{R}$. The history of the art gallery problem suggests the possibility that the Point Guard problem is $\exists\mathbb{R}$-complete. If $NP \neq \exists\mathbb{R}$, then this would imply that there is indeed no hope to find a witness of polynomial size for the Point Guard problem.

Given a polygon $\mathcal{P}$, we will always assume that all coordinates of its vertices are given by *positive* integers in binary. (This can be achieved in polynomial time.) We denote by M the largest appearing integer and we denote by $\operatorname{diam}(\mathcal{P})$ the largest distance between any two points in $\mathcal{P}$. Note that $\operatorname{diam}(\mathcal{P}) < 2M$. We denote $L = 20M > 10$. Note that $\log L$ is linear in the input size. We define the grid

$$\Gamma = (L^{-12} \cdot \mathbb{Z}^2) \cap \mathcal{P}.$$

In other words, we scale the integer grid by L^{-12} and take all points of the grid within the polygon $\mathcal{P}$. Note that all vertices of $\mathcal{P}$ have integer coordinates and thus are included in Γ.

▶ **Theorem 5** (Efrat, Har-Peled [18])**.** *Given a simple polygon $\mathcal{P}$ with n vertices, one can spread a grid Γ inside $\mathcal{P}$, and compute an $O(\log OPT_{\text{grid}})$-approximation for the smallest*

subset of Γ *that sees* $\mathcal{P}$*. The expected running time of the algorithm is*

$$O(n\,OPT_{\text{grid}}^2 \log OPT_{grid}\ \log(n\,OPT_{\text{grid}})\ \log^2 \Delta),$$

where Δ *is the ratio between the diameter of the polygon and the grid size.*

The term OPT_{grid} refers to the optimum, when restricted to the grid Γ. For the solution S that is output by the algorithm of Efrat and Har-Peled the following holds $|S| = O(|OPT_{\text{grid}}| \log |OPT_{\text{grid}}|)$. However, Efrat and Har-Peled make no claim on the relation between $|S|$ and the actual optimum $|OPT|$. Note that the grid size equals $w = L^{-12}$, thus $\Delta \leqslant L^{12+1} = L^{13}$ and consequently $\log \Delta \leqslant 13 \log L$, which is polynomial in the size of the input.

Efrat and Har-Peled implicitly use the real-RAM as model of computation: elementary computations are expected to take $O(1)$ time and coordinates of points are given by real numbers. As we assume that coordinates are given by integers, the word-RAM or integer-RAM is a more appropriate model of computation. All we need to know about this model is that we can upper bound the time for elementary computations by a polynomial in the bit length of the involved numbers. Thus, going from the real-RAM to the word-RAM only adds a polynomial factor in the running time of the algorithm of Efrat and Har-Peled. Therefore, from the discussion above we see that it is sufficient to prove Lemma 4.

Organization. In Section 2, we describe the counterexample to the algorithm of Deshpande et al. [16]. This proves useful as a starting point of Section 3 in which we show Lemma 4. Due to space constraints, the detailed proofs of the lemmas can only be found in the full version [6]. Finally in Section 4, we indicate some remaining open questions.

2 Counterexample

In this section, we point out a mistake in the algorithm of Deshpande et al. [15, 16]. This mistake though constitutes an interesting starting point for our purpose.

The algorithm by Deshpande et al. can be described from a high level perspective as follows: maintain and refine a triangulation T of the polygon until every triangle $\Delta \in T$ satisfies the so-called *local visibility containment* property. The local visibility containment property of Δ certifies that every point $x \in \Delta$ can only see points that are also seen by the vertices of Δ. However, we will argue that it is impossible to attain the local visibility containment property with any finite triangulation; hence, the algorithm never stops.

Actually, we will show two lemmas, which describe fundamental issues with such an approach. Let $D \subseteq \mathcal{P}$ be a finite set of points in the polygon and $x \in \mathcal{P}$, then we denote by $D_x = \{\, d \in D \,:\, \text{dist}(d, x) \leqslant 1 \,\}$.

▶ **Lemma 6.** *There is a polygon* $\mathcal{P}$ *such that for any finite set* D*, there exists a point* x *such that* x *sees a point* p *that is not visible from* D_x*.*

Thus in case that each triangle in the triangulation by Deshpande et al. has diameter smaller than 1 Lemma 6 shows that the promised local visibility property cannot hold. We imagine that all vertices of the triangulation T form the set D. It was claimed that for each point x the triangle Δ containing x sees whatever x sees. Now, Lemma 6 says that even the larger set D_x cannot see everything that is seen by x. Thus in particular the triangle Δ cannot see everything seen by x.

The triangles of Deshpande et al. might be very large and thus not contained in D_x. The next Lemma addresses the issue of large triangles.

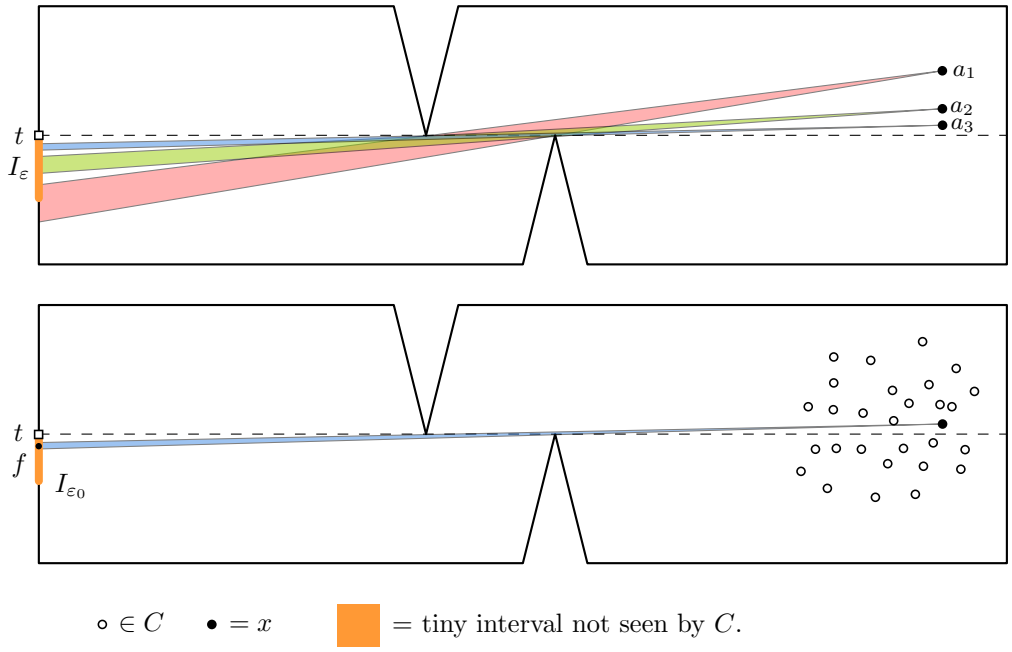


Figure 1 Illustration of the polygon with the property described in Lemma 6.

▶ **Lemma 7.** *Let $c \in \mathbb{N}$ be any constant. There exists a polygon $\mathcal{P}_c$ such that for any finite set of points D, there exists a point $x \in \mathcal{P}_c$ such that any subset $S \subseteq D$ of size $c-1$ cannot see the entire visibility region of x.*

Note that the point x depends on the set D. If we invoke Lemma 7 with $c = 4$ it refutes the algorithm of Deshpande et al. for good as follows. Consider the polygon $\mathcal{P}_4$ as in Lemma 7. For the purpose of contradiction suppose that there exists a triangulation T with the local visibility containment property. We denote by D the set of vertices of T. According to Lemma 7, there exists a point x such that any three points of D cannot see everything that x sees. In particular, the three vertices of the triangle Δ containing x cannot see everything that is seen by x. But T is supposed to have exactly this property — a contradiction.

Again, we want to mention that the paper of Deshpande et al. has ideas that helped to achieve the result of the present paper. In particular, we will show that the local visibility containment property does indeed hold most of the time.

Proof of Lemma 6. See Figure 1, for the definition of polygon $\mathcal{P}$ and the following description. We have two opposite reflex vertices with supporting line ℓ. The sequence of points $(a_i)_{i\in\mathbb{N}}$ are chosen closer and closer to ℓ on the right side of the polygon above ℓ. None of the a_i's can see t, as this would require to be actually on ℓ. Furthermore we denote by I_ε the *open* interval of length ε below t with endpoint t. The interval is indicated in orange in Figure 1. It is clear that for each $\varepsilon > 0$, there exists an i such that a_i sees at least part of I_ε. Let D be any finite set of points. Consider now any finite collection of points $C \subseteq D$ with distance at most 2 to the limit of the $(a_i)_{i\in\mathbb{N}}$. As we will choose x as one of the a_i's it holds $D_x \subseteq C$. For each point $p \in C$ exists an $\varepsilon(p)$ such that p sees nothing of the open interval $I_{\varepsilon(p)}$. Let $\varepsilon_0 = \min_{p\in C} \varepsilon(p)$. None of the points of C see anything of the open interval I_{ε_0}. Recall that the visibility of the a_i's come arbitrarily close to t. Thus, there is some a_k that sees a point f on interval I_{ε_0}. We define x to be a_k. Recall that the set D_x is contained in C. We conclude no point of D_x sees the point f, which is seen by x, as claimed. ◀

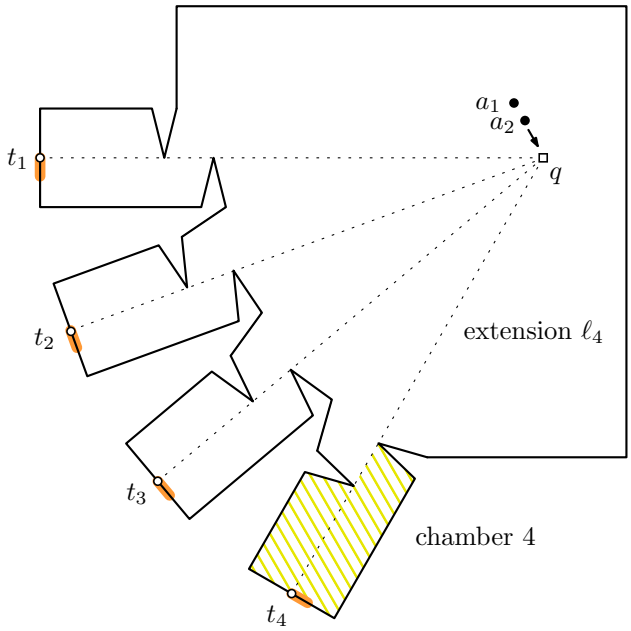


Figure 2 Illustration of the polygon with the property described in Lemma 7.

Proof of Lemmma 7. See Figure 2, for the following description of the polygon $\mathcal{P}_c$. We build $\mathcal{P}_c$ from c disjoint chambers with an entrance of opposite reflex vertices. The chambers are arranged in a way that all the extensions of the opposite reflex vertices meet in a common point q. In this way, we get c extensions $\ell_1, \ldots, \ell_c$. We denote by t_i the intersection of the extension ℓ_i with the i-th chamber. An important nuance in the construction is the fact that one can see into all the chambers *simultaneously* from points b arbitrarily close to q.

Again we construct a sequence $(a_i)_{i\in\mathbb{N}}$ such that it works as in the proof of Lemma 6, but for all chambers simultaneously. For this let a_i be any point with $\text{dist}(a_i, q) = 1/i$ and the property that it sees into each chamber, for all $i \in \mathbb{N}$. As in the proof of Lemma 6, it holds that each a_i sees a small interval close to t_j, for all $i \in N$ and $j \in \{1, \ldots, c\}$. In particular each such interval *approaches* t_j, for all $j = \{1, \ldots, c\}$.

Let $\varepsilon > 0$. We denote the *open* interval of length ε to the right of t_j on the boundary of $\mathcal{P}_c$ with $I_{\varepsilon,j}$ (indicated orange in the figure). No point $b \in \mathcal{P}$ can see two intervals $I_{\varepsilon,j}$ and $I_{\varepsilon,j'}$ *entirely*, for any $\varepsilon > 0$ and $j \neq j'$. Because to see the *whole* interval $I_{\varepsilon,j}$ requires to be in chamber j. However, no point can be in two chambers simultaneously. To avoid confusion, we want to point out that no a_i can see any interval $I_{\varepsilon,j}$ entirely. But for every $\varepsilon > 0$, there exists an i_0 such that a_{i_0} sees part of *all* $I_{\varepsilon,j}$ simultaneously.

Let $D \subseteq \mathcal{P}$ be any finite set. Then there exists some ε_0 such that any point $p \in D$ sees at most one entire interval $I_{\varepsilon_0,j}$ and nothing of any other interval $I_{\varepsilon_0,j'}$ for any $j' \neq j$. To see this consider first the case that p is contained in one of the chambers. Then the statement is clear as it cannot see any point of any other chamber. In the other case p is outside of any chamber and thus cannot see any interval $I_{\varepsilon,j}$ entirely for any j. Thus in this case there exists some $\varepsilon(p) > 0$ such that p sees nothing of $I_{\varepsilon,j}$ for any j and $0 < \varepsilon < \varepsilon(p)$. We choose $\varepsilon_0 = \min_{p\in D} \varepsilon(p)$.

By the definition of $(a_i)_{i\in\mathbb{N}}$ there exists some $x = a_k$ that sees at least one point $f_j \in I_{\varepsilon_0,j}$, for all $j = \{1, \ldots, c\}$. As no point of D sees two f_j simultaneously, we need at least c points of D to see $f_1, \ldots, f_c$. ◀

3 Detailed exposition of the proof

The details, which we skipped here due to space constraints, can be found in [6]. Nonetheless, all the ideas and crucial facts are highlighted in this conference version, and illustrated with a number of figures. In almost all figures, the proportions of some objects and distances may not reflect the reality of things. They are displayed this way to convey a message, albeit exaggeratedly.

Our high-level proof idea is that the local visibility containment property holds for every point x that is sufficiently far away from all extension lines. (We slightly tune the meaning of the local visibility containment property.) This constitutes the first step. (Recall that the extension of two vertices is the line that contains these vertices.) In a second step, we will show that a point in the gallery cannot be close to more than two extensions at the same time. We will add one vertex for each extension that x is close to. Recall that the vertices of the polygon are also in Γ.

The first step is much more tedious than the second one. A reason for that is that many observations that seem true at first sight turn out to be erroneous. Therefore, some extra care is needed for this step in the definitions of the concepts and in breaking a general situation to a distinction of more elementary cases. The crux is mainly to identify these elementary cases and to properly handle them. All the other proofs are elementary.

3.1 Benefit of Integer Coordinates

The integer coordinate assumption not only implies that the distance between any two vertices is at least 1 but it also gives useful lower bounds on distances between any two objects of interest that do not share a point. The next lemma lists all such lower bounds that we will need later. We denote by $\mathrm{dist}(u,v), \mathrm{dist}(u,\ell)$ and $\mathrm{dist}(\ell,\ell')$ the Euclidean distance between the points u and v, the point u and the line ℓ, and the lines ℓ and ℓ', respectively.

▶ **Lemma 8.** *Let $\mathcal{P}$ be a polygon with integer coordinates and L as defined above. Let v and w be vertices of $\mathcal{P}$, ℓ and ℓ' supporting lines of two vertices, and p and q intersections of supporting lines.*

1. *$dist(v,w) > 0 \Rightarrow dist(v,w) \geqslant 1$.*
2. *$dist(v,\ell) > 0 \Rightarrow dist(v,\ell) \geqslant L^{-1}$.*
3. *$dist(p,\ell) > 0 \Rightarrow dist(p,\ell) \geqslant L^{-5}$.*
4. *$dist(p,q) > 0 \Rightarrow dist(p,q) \geqslant L^{-4}$.*
5. *Let $\ell \neq \ell'$ be parallel. Then $dist(\ell,\ell') \geqslant L^{-1}$.*
6. *Let $\ell \neq \ell'$ be any two non-parallel supporting lines and α the smaller angle between them. Then holds $\tan(\alpha) \geqslant 8L^{-2}$.*
7. *Let $a \in \mathcal{P}$ be a point and ℓ_1 and ℓ_2 be some non-parallel lines with $dist(\ell_i,a) < d$, for $i = 1,2$. Then ℓ_1 and ℓ_2 intersect in a point p with $dist(a,p) \leqslant dL^2$.*

As these bounds are important for the intuition of the forthcoming ideas, we will give an example by proving Item 2.

Proof of Item 2. The distance d can be computed as

$$d = \frac{|(v-w_1)\cdot(w_2-w_1)^{\perp}|}{\|w_2-w_1\|_2} \geqslant \frac{1}{\mathrm{diam}(\mathcal{P})} \geqslant \frac{1}{L}.$$

Figure 3 illustrates how to derive this elementary formula. Here, $\cdot$ denotes the scalar product, $x^{\perp}$ is the vector x rotated by 90° counter-clockwise, and $\|x\|_2$ is the Euclidean norm of x.

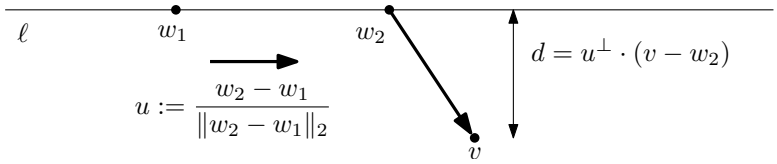


Figure 3 Computing the distance between a line and a vertex.

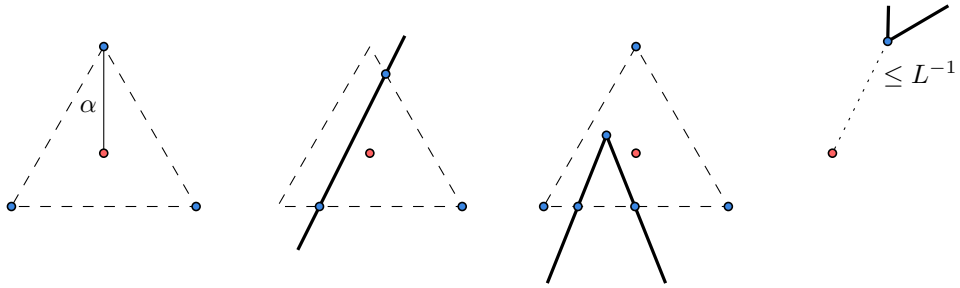


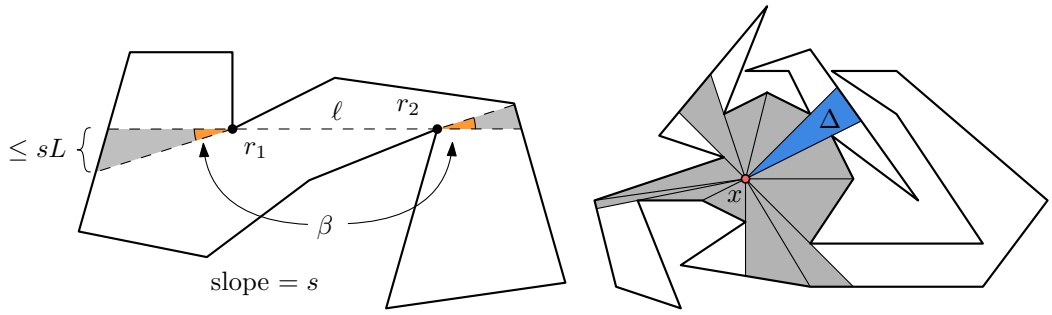
Figure 4 The red point indicates a point of the original optimal solution. The blue points indicate the surrounding grid points that we choose. The polygon is indicated by bold lines. From left to right, we have three cases: the interior case, the boundary case, and the corner case. To the very right, we indicate that in every case the vertices of $\mathcal{P}$ with distance less than L^{-1} are also included in α-grid$^*(x)$.

The numerator of this formula is at least 1 as it is a non-zero integer by assumption. The denominator is upper bounded by the diameter of $\mathcal{P}$, which is in turn upper bounded by L. ◀

3.2 Surrounding Grid Points

Given a point $x \in \mathcal{P}$ and a number α much smaller than the grid width, we will define α-grid(x) as a set of grid points around x, see Figure 4. The parameter α is an upper bound on the distance between x and α-grid(x). (We will chose later $\alpha = L^{-11}$.) In case that there exists a vertex v of $\mathcal{P}$ with distance dist$(x, v) \leqslant L^{-1}$, we define α-grid$^*(x) = \alpha$-grid$(x) \cup v$. Later, we will make use of the fact that $|\alpha\text{-grid}^*(x)| \leqslant 7$.

The following precise definition depends on the position of x and the value α. It is included for the interested reader, but not strictly needed to understand the remainder of the main body. Let c be a circle with radius α and center x. Then there exists a unique equilateral triangle $\Delta(x)$ inscribed c such that the lower side of $\Delta(x)$ is horizontal. We distinguish three cases. In the *interior* case, $\Delta(x)$ and $\partial\mathcal{P}$ are disjoint. In the *boundary* case, $\Delta(x)$ and $\partial\mathcal{P}$ have a non-empty intersection, but no vertex of $\mathcal{P}$ is contained in $\Delta(x)$. In the *corner* case, one vertex of $\mathcal{P}$ is contained in Δ. It is easy to see that this covers all the cases. We also say a point x is in the interior case, and so on. In the *interior case* α-grid(x) is defined as follows. Let v_1, v_2, v_3 be the vertices of Δ. Then the grid points g_i, which are closest to v_i, for all $i = 1, 2, 3$ form the surrounding grid points. In the *boundary case* α-grid(x) is defined as follows. Let S be the set of vertices of Δ and all intersection points of $\partial\mathcal{P}$ with $\partial\Delta(x)$. For each point $v \in S$, we define the grid point g_v closest to v and accordingly we define $G_S = \{\, g_v \,:\, v \in S \,\}$. Then α-grid$(x) = G_S$. In the *corner case* α-grid(x) is defined as follows. Let S be the set of vertices of Δ and all intersection points of $\partial\mathcal{P}$ with $\partial\Delta(x)$. For each point $v \in S$, we define the grid point g_v closest to v and accordingly we define $G_S = \{\, g_v \,:\, v \in S \,\}$. Then α-grid$(x) = G_S$. In any case, if there is a reflex vertex



(a) A polygon with two opposite reflex vertices and their s-bad region.

(b) The star triangle decomposition of the visibility region of x.

Figure 5

r with $\text{dist}(x, r) \leqslant L^{-1}$ then we include r in the set $\alpha\text{-grid}^*(x) = r \cup \alpha\text{-grid}(x)$ as well. We will usually denote the points in $\alpha\text{-grid}(x)$ with g_1, g_2 or just g.

3.3 Local Visibility Containment

Let s be a fixed parameter to be specified later ($s = L^{-9}$). For any extension ℓ we define an s-bad region, see the gray area in Figure 5a for an illustration. Note that the bad region consists of two connected components, each being a triangle. (There can be no vertex in the interior of the triangle, because of Lemma 8 Item 2.) The parameter $s = \tan(\beta)$ is indicated in the figure. Furthermore, for each point x, the visibility region can be decomposed into triangles as indicated in Figure 5b. The region is called bad region, because Lemma 9 does not hold for points in those regions.

Let Δ be some triangle of the visibility region of x (in blue in Figure 5b). In this section, we denote the defining vertices of Δ by r_1 and r_2.

Then the main lemma asserts that $\alpha\text{-grid}^*(x)$ sees Δ except if x is in an s-bad region of the vertices defining Δ.

▶ **Lemma 9** (Special Local Visibility Containment Property)**.** *Let r_1 and r_2 be two consecutive vertices in the clockwise order of the vertices visible from $x \in \mathcal{P}$ and let x be outside the s-bad region of the vertices r_1 and r_2 and Δ the triangle of the visibility region of x defined by r_1 and r_2. We make the following assumptions: $s \leqslant L^{-3}$, $\alpha \leqslant L^{-7}$ and $16L\alpha \leqslant s$. Then α-grid$^*(x)$ sees Δ.*

Important is the one-to-one correspondence between the triangles that cannot be seen and the extension line that we can make responsible for it.

The proof is structured in many cases. At first the triangle Δ is split into a *small triangle* (R_1) and a trapezoid (R_2), as indicated in Figure 7. We show separately, for R_1 and R_2 that $\alpha\text{-grid}^*(x)$ sees these two regions.

Another important case distinction is on whether Δ contains a point $g \in \alpha\text{-grid}^*(x)$, see Figure 6. In the first case g sees Δ as Δ is convex. In the other case, we can identify two points $g_1, g_2 \in \alpha\text{-grid}(x)$ to the left and right of Δ. For all what follows we are only concerned with the second case.

We are confronted with the situation that there might be a vertex v that is not in Δ, but obstructs the vision of g_1, g_2 in one way or another. Whenever this happens, we distinguish two cases: Either $\text{dist}(v, x) < L^{-1}$ or $\text{dist}(v, x) \geqslant L^{-1}$. In the first case $v \in \alpha\text{-grid}^*(x)$. To understand the case that v is "far" from x, one must realize that L^{-1} is huge compared to

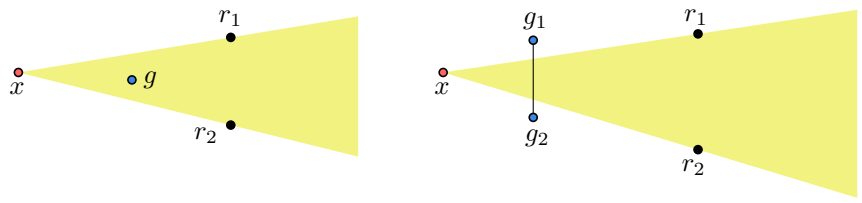


Figure 6 Left: The point g is contained in Δ and thus g sees Δ, as Δ is convex. Right: The line segment s cuts Δ.

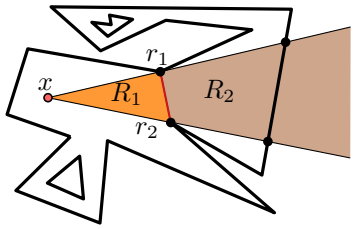


Figure 7 To show that each triangle of the visibility region is visible by α-grid$^*(x)$, we treat the small triangle R_1 and the trapezoid R_2 individually. In particular, as we do not make use of the finiteness of R_2, we just assume it is an infinite cone.

$\alpha = L^{-11}$. Thus x and $g \in \alpha$-grid(x) are affected by v in a very similar way. Unfortunately, not in exactly the same way. For instance x sees the segment seg(r_1, r_2), which has length at least one, and it is easy to show that certain points of $g_1, g_2 \in \alpha$-grid(x) see the entire segment, except a sub-segment of length at most L^{-2}, see Figure 8. This sub-segment is completely irrelevant, but we have to deal with it. These issues arise at various places. It makes forthcoming definitions more tedious and requires the proofs to be carried out with extra care.

To see that α-grid$^*(x)$ sees R_1 relies mainly on the insight, which we already mentioned above, that reflex vertices v, with dist$(x, v) \geqslant L^{-1}$ can only block a very small part of the visibility of α-grid(x) at the bottom of segment seg(r_1, r_2), as illustrated in Figure 8. For the case that there exists a reflex vertex v with dist$(x, v) < L^{-1}$, recall that v is included in α-grid$^*(x)$. Therefore, even if a reflex vertex obstructs the vision of g_2 onto seg(r_1, r_2), then g_2 can see the entire upper half of region R_1 and similarly, g_1 sees the entire lower part of R_1, as illustrated in Figure 8. Thus g_1 and g_2 see together the entire region R_1. Note that the argument does not rely on x being outside a bad region.

To prove that R_2 can be seen by α-grid$^*(x)$ is more demanding. As it seems not useful to use the boundedness of R_2, we just assume it to be an infinite cone and we show that α-grid$^*(x)$ sees this cone. Obviously, the part of $\partial\mathcal{P}$ "behind" seg(r_1, r_2) is not considered blocking. The crucial step to show that R_2 can be seen by α-grid$^*(x)$ is to show that the black region as indicated in Figure 9 does not exist. The idea is that this is implied if ray$_1$ and ray$_2$ diverge.

In other words if ray$_1$ and ray$_2$ never meet then the black region is empty. For this purpose, we make use of the fact that dist$(g_1, g_2) \approx \alpha$, for any $g_1, g_2 \in \alpha$-grid(x) by definition, while dist$(r_1, r_2) \geqslant 1$, because of integer coordinates. Thus intuitively, the distance of ray$_1$ and ray$_2$ is closer at its apex than at the segment seg(r_1, r_2). Indeed any two rays ray$_a$ and ray$_b$ will not intersect, if the following three conditions are met, see Figure 10.

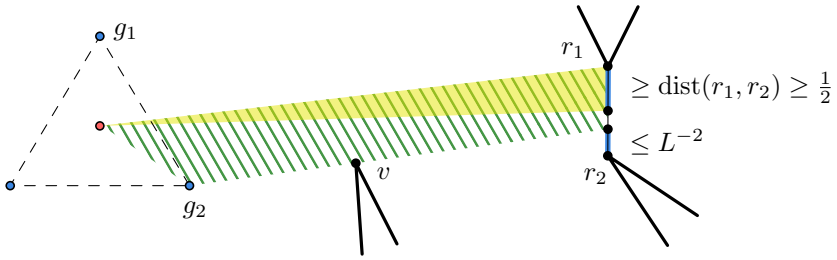


Figure 8 The point $g_2 \in \alpha\text{-grid}(x)$ sees the upper half of the Region R_1 as the green region is completely contained inside the polygon.

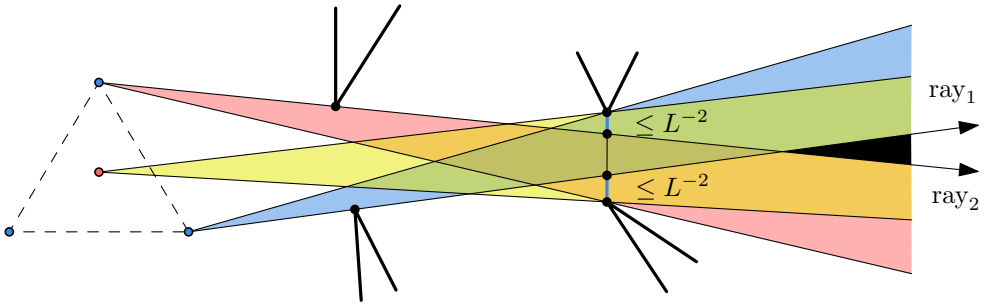


Figure 9 The visibility of the grid points $g \in \alpha\text{-grid}(x)$ can be blocked, but we can bound the amount by which it is blocked. The key idea to show that R_2 can be seen by $\alpha\text{-grid}(x)$ is to show that the region indicated in solid black is empty.

- The apex of ray_a and ray_b are "close".
- The "defining" points q_a, q_b are "far" apart.
- Both apices are outside of the s-bad region of q_a and q_b.

In order to invoke the last statement, we have to show that g_1 and g_2 are outside some appropriately defined bad regions. For this we use that x is outside the s-bad region of r_1 and r_2. The points where ray_1 and ray_2 intersect $\text{seg}(r_1, r_2)$ play the role of the defining points.

3.4 Global Visibility Containment

Given a minimum solution OPT, we describe a set $G \subseteq \Gamma$ of size $O(|OPT|)$ and we show that G sees the entire polygon, see Figure 12 for an illustration. For each $x \in OPT$, G contains $\alpha\text{-grid}^*(x)$. Furthermore if x is contained in an s-bad region, G contains at least one of the vertices defining this bad region. It is clear by the previous discussion that G sees the entire polygon, as the only part that is not seen by $\alpha\text{-grid}^*(x)$ are some small regions, which are entirely seen by the vertices bounding it.

It remains to show that there is no point in three bad regions. For this, we heavily rely on the integer coordinates and the general position assumption. Note that the integer coordinate assumption implies not just that the distance between any two vertices is at least 1 but also that the distance between any extension ℓ and a vertex v not on ℓ is at least L^{-1}. Also the angle between any two extensions is at least L^{-2}. (Recall that L is an upper bound on the diameter and the largest appearing integer.) These bounds and other bounds of this kind imply that if any three bad regions meet in the interior, then their extension lines must meet in a single point, see Figure 11. We exclude this by our general position assumption.

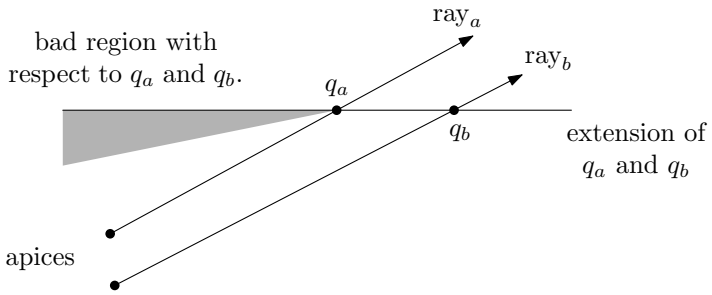


Figure 10 If the distance of the rays is closer at its apices than at q_a and q_b then we can conclude that the rays are diverging and never crossing.

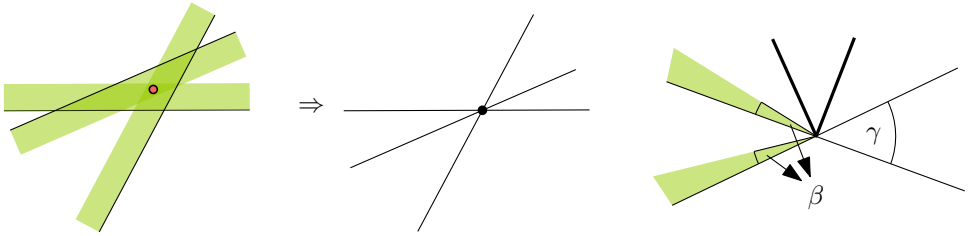


Figure 11 Three bad regions meeting in an interior point implies that the extensions must meet in a single point. No two bad regions intersect in the vicinity of a vertex, as they are defined by some angle $\beta \ll L^{-2}$. But the angle γ between any two extensions is at least L^{-2}.

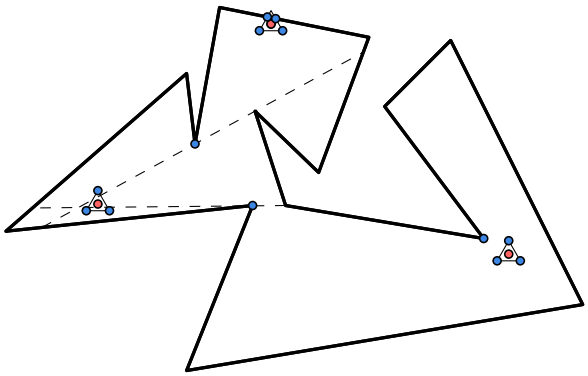


Figure 12 The red dots indicate the optimal solution. The blue dots indicate the set $G \subseteq \Gamma$ that are part of an approximate solution. The red dot on the top is in the interior case and four grid points are added around it. The red dot on the left is too close to two supporting lines and we add one of the reflex vertices of each of the supporting lines. The red dot to the right has distance less than L^{-1} to a reflex vertex, so we add that vertex to G as well.

Close to a vertex, we use a different argument: No two bad regions intersect in the vicinity of a vertex, as bad regions are defined by some angle β with $\tan(\beta) \ll L^{-2}$. But the angle γ between any two extensions is at least L^{-2}.

Recall that $|\alpha\text{-grid}^*(x)| \leqslant 7$. Together with the argument above follows that each x is in at most 2 bad regions and $|G| \leqslant (7+2)|OPT| = O(|OPT|)$.

4 Conclusion

We presented an $O(\log |OPT|)$-approximation algorithm for the Point Guard Art Gallery problem under two relatively mild assumptions. The most natural open question is whether Assumption 2 can be removed. We believe that this is possible but it will require some additional efforts and ideas. Another improvement of the result would be to achieve an approximation ratio of $O(\log n)$ for polygons with holes. This would match the currently best known algorithm for the Vertex Guard variant and the lower bound for both problems. In that respect, it is noteworthy that Lemma 4 does not require the polygon to be simple. One might also ask about the inapproximability of Point Guard Art Gallery for simple polygons. For the moment, the problem is only known to be inapproximable for a certain constant ratio (quite close to 1), unless P=NP. It would be interesting to get superconstant inapproximability under standard complexity theoretic assumptions or improved approximation algorithms.

References

1 Eyüp Serdar Ayaz and Alper Üngör. Minimal witness sets for art gallery problems. *EuroCG*, 2016.

2 János Barát, Vida Dujmovic, Gwenaël Joret, Michael S. Payne, Ludmila Scharf, Daria Schymura, Pavel Valtr, and David R. Wood. Empty pentagons in point sets with collinearities. *SIAM J. Discrete Math.*, 29(1):198–209, 2015.

3 Saugata Basu, Richard Pollack, and Marie-Francoise Roy. *Algorithms in real algebraic geometry*. Springer, 2007.

4 Patrice Belleville. Computing two-covers of simple polygons. Master's thesis, McGill University, 1991.

5 Vijay V. S. P. Bhattiprolu and Sariel Har-Peled. Separating a voronoi diagram via local search. In *SOCG*, pages 18:1–18:16, 2016.

6 Édouard Bonnet and Tillmann Miltzow. An approximation algorithm for the art gallery problem. *CoRR*, 1607.05527, 2016. URL: `http://arxiv.org/abs/1607.05527`.

7 Édouard Bonnet and Tillmann Miltzow. The parameterized hardness of the art gallery problem. In *ESA 2016*, pages 19:1–19:17, 2016. Arxiv identifier: 1603.08116.

8 Hervé Brönnimann and Michael T. Goodrich. Almost optimal set covers in finite VC-dimension. *Discrete & Computational Geometry*, 14(4):463–479, 1995. `doi:10.1007/BF02570718`.

9 John Canny. Some algebraic and geometric computations in PSPACE. In *STOC*, pages 460–467. ACM, 1988.

10 Jean Cardinal. Computational geometry column 62. *SIGACT News*, 46(4):69–78, December 2015. `doi:10.1145/2852040.2852053`.

11 Václav Chvátal. A combinatorial theorem in plane geometry. *Journal of Combinatorial Theory, Series B*, 18(1):39–41, 1975.

12 Kyung-Yong Chwa, Byung-Cheol Jo, Christian Knauer, Esther Moet, René van Oostrum, and Chan-Su Shin. Guarding art galleries by guarding witnesses. *Int. J. Comput. Geometry Appl.*, 16(2-3):205–226, 2006.

13 Kenneth L. Clarkson. Algorithms for polytope covering and approximation. In *WADS 1993*, pages 246–252, 1993. doi:10.1007/3-540-57155-8_252.

14 Pedro Jussieu de Rezende, Cid C. de Souza, Stephan Friedrichs, Michael Hemmer, Alexander Kröller, and Davi C. Tozoni. Engineering art galleries. *CoRR*, abs/1410.8720, 2014. URL: http://arxiv.org/abs/1410.8720.

15 Ajay Deshpande. A pseudo-polynomial time $O(\log^2 n)$-approximation algorithm for art gallery problems. Master's thesis, Department of Mechanical Engineering, Department of Electrical Engineering and Computer Science, MIT, 2006.

16 Ajay Deshpande, Taejung Kim, Erik D. Demaine, and Sanjay E. Sarma. A pseudopolynomial time $O(\log n)$-approximation algorithm for art gallery problems. In *WADS 2007*, pages 163–174, 2007. doi:10.1007/978-3-540-73951-7_15.

17 Stephane Durocher and Saeed Mehrabi. Guarding orthogonal art galleries using sliding cameras: algorithmic and hardness results. In *MFCS 2013*, pages 314–324. Springer, 2013.

18 Alon Efrat and Sariel Har-Peled. Guarding galleries and terrains. *Inf. Process. Lett.*, 100(6):238–245, 2006. doi:10.1016/j.ipl.2006.05.014.

19 Stephan Eidenbenz, Christoph Stamm, and Peter Widmayer. Inapproximability results for guarding polygons and terrains. *Algorithmica*, 31(1):79–113, 2001.

20 Khaled Elbassioni. Finding small hitting sets in infinite range spaces of bounded VC-dimension. *CoRR*, abs/1610.03812, 2016. accepted to SoCG 2017.

21 Steve Fisk. A short proof of Chvátal's watchman theorem. *J. Comb. Theory, Ser. B*, 24(3):374, 1978. doi:10.1016/0095-8956(78)90059-X.

22 Stephan Friedrichs, Michael Hemmer, James King, and Christiane Schmidt. The continuous 1.5d terrain guarding problem: Discretization, optimal solutions, and PTAS. *JoCG*, 7, 2016.

23 Subir Kumar Ghosh. Approximation algorithms for art gallery problems in polygons. *Discrete Applied Mathematics*, 158(6):718–722, 2010.

24 Alexander Gilbers and Rolf Klein. A new upper bound for the VC-dimension of visibility regions. *Computational Geometry*, 47(1):61–74, 2014.

25 Heuna Kim and Günter Rote. Congruence testing of point sets in 4-space. In *SoCG*, volume 51 of *LIPIcs*, pages 48:1–48:16, 2016. Arxiv identifier: 1603.07269.

26 James King. Fast vertex guarding for polygons with and without holes. *Comput. Geom.*, 46(3):219–231, 2013. doi:10.1016/j.comgeo.2012.07.004.

27 David G. Kirkpatrick. An $O(\log \log OPT)$-approximation algorithm for multi-guarding galleries. *Discrete & Computational Geometry*, 53(2):327–343, 2015. doi:10.1007/s00454-014-9656-8.

28 Erik A. Krohn and Bengt J. Nilsson. Approximate guarding of monotone and rectilinear polygons. *Algorithmica*, 66(3):564–594, 2013.

29 Jirí Matousek. Intersection graphs of segments and $\exists\mathbb{R}$. *CoRR*, 1406.2636, 2014.

30 Rajeev Motwani, Arvind Raghunathan, and Huzur Saran. Covering orthogonal polygons with star polygons: The perfect graph approach. *J. Comput. Syst. Sci.*, 40(1):19–48, 1990. doi:10.1016/0022-0000(90)90017-F.

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

32 Marcus Schaefer. *Complexity of Some Geometric and Topological Problems*, pages 334–344. Springer, 2010. doi:10.1007/978-3-642-11805-0_32.

33 Thomas C. Shermer. Recent results in art galleries. *IEEE*, 80(9):1384–1399, 1992.

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

Self-Approaching Paths in Simple Polygons*

Prosenjit Bose[1], Irina Kostitsyna†[2], and Stefan Langerman[3]

1 **School of Computer Science, Carleton University, Ottawa, Canada**
jit@scs.carleton.ca
2 **Computer Science Department, Université libre de Bruxelles (ULB), Brussels, Belgium**
irina.kostitsyna@ulb.ac.be
3 **Computer Science Department, Université libre de Bruxelles (ULB), Brussels, Belgium**
stefan.langerman@ulb.ac.be

Abstract

We study *self-approaching paths* that are contained in a simple polygon. A self-approaching path is a directed curve connecting two points such that the Euclidean distance between a point moving along the path and any future position does not increase, that is, for all points a, b, and c that appear in that order along the curve, $|ac| \geq |bc|$. We analyze the properties, and present a characterization of shortest self-approaching paths. In particular, we show that a shortest self-approaching path connecting two points inside a polygon can be forced to follow a general class of non-algebraic curves. While this makes it difficult to design an exact algorithm, we show how to find a self-approaching path inside a polygon connecting two points under a model of computation which assumes that we can calculate involute curves of high order.

Lastly, we provide an algorithm to test if a given simple polygon is self-approaching, that is, if there exists a self-approaching path for any two points inside the polygon.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases self-approaching path, simple polygon, shortest path, involute curve

Digital Object Identifier 10.4230/LIPIcs.SoCG.2017.21

1 Introduction

The problem of finding an optimal obstacle-avoiding path in a polygonal domain is one of the fundamental problems of computational geometry. Often a desired path has to conform to certain constraints. For example, a path may be required to be monotone [3], curvature-constrained [9], have no more than k links [15], etc. A natural requirement to consider is that a point moving along a desired path must be always getting closer to its destination. Such *radially monotone paths* appear, for example, in greedy geographic routing in a network setting [10], and beacon routing in a geometric setting [5]. A strengthening of a radially monotone path is a *self-approaching path* [12, 13, 1]: a point moving along a self-approaching path is always getting closer not only to its destination, but also to all the points on the path ahead of it. There are several reasons to prefer self-approaching paths over radially monotone paths. First, unlike for a radially monotone path, any subpath of a self-approaching path is self-approaching. Thus, if the destination is not known in advance and the desired path is required to be radially monotone, one would have to resort to using

* A full version of the paper is available at https://arxiv.org/abs/1703.06107.
† I. K. was supported in part by the NWO under project no. 612.001.106, and by F.R.S.-FNRS.

self-approaching paths. Second, the length of a radially monotone path can be arbitrarily large in comparison with the Euclidean distance between the source and the destination points, whereas self-approaching paths have a bounded detour [13]. These properties make self-approaching paths of interest to various applications, including network routing, graph drawing, and others.

In this paper we study self-approaching paths that are contained in a simple polygon. We consider the following questions:

- Given two points s and t inside a simple polygon P, does there exist a self-approaching s-t path inside P?
- Find the shortest self-approaching s-t path.
- Given a point s in a simple polygon P, what is the set of all points reachable from s with self-approaching paths?
- Given a point t, what is the set of all points from which t is reachable with a self-approaching path?
- Given a polygon P, test if it is self-approaching, *i.e.*, if there exists a self-approaching path between any two points in P.

Related work. Self-approaching curves were first introduced in the context of online searching for the kernel of a polygon [12]. They were further studied in [13] where, among other results, the authors prove that the length of any self-approaching curve connecting two points is not greater than 5.3331 times the Euclidean distance between the points. An equivalent definition of a self-approaching path is that for every point on the path there has to be a 90° angle containing the rest of the path. Aichholzer *et al.* [1] developed a generalization of self-approaching paths for an arbitrarily fixed angle α. A relevant class of paths is increasing-chords paths [17], which are self-approaching in both directions. The nice properties of self-approaching and increasing-chords paths, and their potential to be applied in network routing, were recognized by the graph drawing community. As a result, a number of papers have appeared in recent years on self-approaching and increasing-chords graphs [2, 8, 16].

This paper is organized in the following way. We introduce a few definitions and concepts in Section 2. In Section 3, we characterize a shortest self-approaching path between two points in a simple polygon. In Section 4 we present an algorithm to construct the shortest self-approaching path between two points if it exists, or to report that it does not exist, by assuming a model of computation in which we can solve certain transcendental equations. Finally, in Section 5 we present a linear-time algorithm to decide if a polygon is self-approaching, that is, if there is a self-approaching path between any two point of the polygon. Due to space limitations, some proofs are omitted. For the details refer to the full version of this paper [6].

2 Preliminaries

For two points p_1 and p_2 on a directed path π that starts at point s, we shall say that $p_1 <_\pi p_2$ if p_1 lies between s and p_2 along π. For a directed path π and two points $p_1 <_\pi p_2$ on it, denote the subpath from p_1 to p_2 by $\pi(p_1, p_2)$.

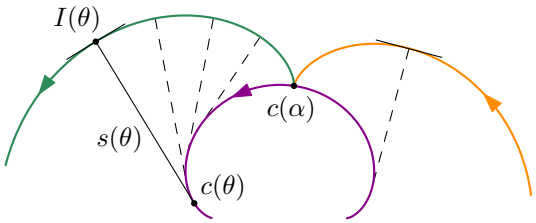


Figure 1 Curve $c(\theta)$ and two involutes. The arrows designate the direction of growth of the parameter θ. The involute on the left is defined by tangents pointing in the negative direction of c, and the involute on the right is defined by tangents pointing in the positive direction of c.

▶ **Definition 1.** A *self-approaching path* π in a continuous domain is a piece-wise smooth[1] oriented curve such that for any three points a, b, and c on it, such that $a <_\pi b <_\pi c$: $|ac| \geq |bc|$, where $|ac|$ and $|bc|$ are Euclidean distances.

Icking *et al.* [13] showed the following *normal property* of a self-approaching path, that we will be using extensively in this paper,

▶ **Lemma 2** (the normal property [13])**.** *An s-t path π is self-approaching if and only if any normal to π at any point $a \in \pi$ does not cross $\pi(a,t)$.*

▶ **Definition 3.** A normal h to a directed curve π at some point $a \in \pi$ defines two half-planes. Let the *positive half-plane* h^+ be the open half-plane which is congruent with the direction of π at point a.

We can rephrase the normal property in the following way.

▶ **Lemma 4** (the half-plane property)**.** *An s-t path π is self-approaching if and only if, for any normal h to π at any point $a \in \pi$, the subpath $\pi(a,t)$ lies completely in the positive half-plane h^+.*

▶ **Definition 5.** A *bend* of a self-approaching path π is a point of discontinuity of the first derivative of π.

▶ **Definition 6.** A *reachable region* $\mathcal{R}(s) \subseteq P$, for a given point s in a polygon P, is the set of all points $t \in P$ for which there exists a self-approaching s-t path $\pi \in P$.

▶ **Definition 7.** A *reverse-reachable region* $\mathcal{R}^{-1}(t) \subseteq P$, for a given point t in a polygon P, is a set of all points $s \in P$ for which there exists a self-approaching s-t path $\pi \in P$.

2.1 Involutes

Next we introduce involute curves of kth order that will appear later as parts of shortest self-approaching paths.

An involute of a convex curve c is a curve traced by the end point of an unwinding pull-taut string rolled on c. Consider a parameterization $\vec{c}(\theta)$ of the curve, and let c be oriented in the direction of growth of the parameter θ. The involute of c can be computed by the following formula:

$$\vec{I}(\theta) = \vec{c}(\theta) - s(\theta)\frac{\vec{c}\,'(\theta)}{|\vec{c}\,'(\theta)|},$$

[1] Some previous works do not require the curve to be smooth. However in this paper we will be mostly considering shortest self-approaching paths, and thus the requirement on smoothness is justified.

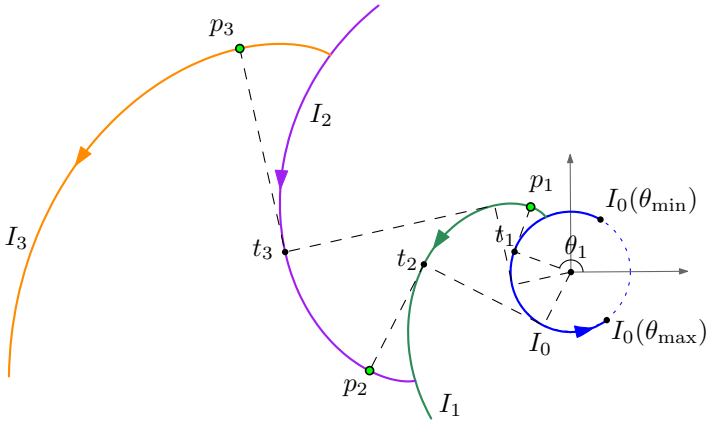


Figure 2 Circular arc $I_0(\theta)$, and three involutes $I_1(\theta)$, $I_2(\theta)$, and $I_3(\theta)$: for each i, $I_i(\theta)$ is an involute about $I_{i-1}(\theta)$ that passes through point p_i. The arrows designate the direction of growth of the parameter θ (they are not necessarily consistent with the direction of a self-approaching path).

where $s(\theta)$ is the length of the tangent segment $|\overline{c(\theta)I(\theta)}|$,

$$s(\theta) = \int_{\alpha}^{\theta} |\vec{c}\,'(t)|dt\,.$$

The constant α defines the point at which the involute I will start unwinding around c (see Fig. 1). The involute has two branches: the *positive* branch unwinds starting at point α in the direction of growth of θ, and the *negative* branch unwinds in the opposite direction. If the curve c is defined on the interval $[\theta_{\min}, \theta_{max}]$, then the positive branch of its involute is defined on the interval $[\alpha, \theta_{\max}]$, and the negative – on the interval $[\theta_{\min}, \alpha]$.

We define an *involute of order k* of a curve $c(\theta)$ to be an involute of one branch (that contains the point corresponding to a parameter α_k) of an involute of order $k-1$ of $c(\theta)$, with an involute of order 0 being the curve $c(\theta)$ itself,

$$\vec{I}_k(\theta) = \vec{I}_{k-1}(\theta) - s_k(\theta)\frac{\vec{I}'_{k-1}(\theta)}{|\vec{I}'_{k-1}(\theta)|}\,, \qquad \text{where} \qquad s_k(\theta) = \int_{\alpha_k}^{\theta} |\vec{I}'_{k-1}(t)|dt\,,$$

$$\vec{I}_0(\theta) = \vec{c}(\theta)\,.$$

In the following sections we will show that shortest self-approaching paths consist of straight-line segments, circular arcs, and involutes of circular arcs of some order. In the full version of this paper [6] we provide the details of the derivation of the following formula for an involute of a circle of order k:

$$I_k(\theta) = \sum_{0}^{\lfloor \frac{k}{2} \rfloor} (-1)^i a_{2i}(\theta) \begin{pmatrix} \cos\theta \\ \sin\theta \end{pmatrix} - \sum_{0}^{\lceil \frac{k}{2} \rceil - 1} (-1)^i a_{2i+1}(\theta) \begin{pmatrix} -\sin\theta \\ \cos\theta \end{pmatrix}\,,$$

where each involute I_i passes through a point $p_i(r_i, \varphi_i)$ for all $1 \le i \le k$,

$$a_i(\theta) = r_0\frac{\theta^i}{i!} + c_1\frac{\theta^{i-1}}{(i-1)!} + \cdots + c_i\,,$$

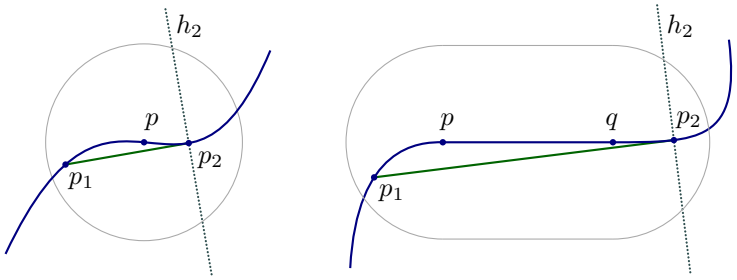


Figure 3 If a self-approaching path has an inflection point (or a segment) interior to P, then there exists a shortcut.

and the constants c_i can be found from the following equations:

$$r_i \cos(\theta_i - \varphi_i) = \sum_{0}^{\lfloor \frac{i}{2} \rfloor} (-1)^j a_{2j}(\theta_i), \qquad r_i \sin(\theta_i - \varphi_i) = \sum_{0}^{\lceil \frac{i}{2} \rceil - 1} (-1)^j a_{2j+1}(\theta_i)\,. \tag{1}$$

The length $|\overline{p_k t_k}|$ of the tangent segment equals $|a_k(\theta_k)|$.

3 Properties of a shortest self-approaching path

In this section we will prove the following properties of a shortest self-approaching path from s to t inside a simple polygon P:

- A shortest self-approaching path is unique.
- The shortest self-approaching path consists of straight segments, circular arcs and involutes to the latter pieces of the path.

We begin by proving several lemmas:

▶ **Lemma 8.** *For any two points $p_1 <_\pi p_2$ on a self-approaching s-t path π in $\mathbb{R}^2$, the perpendicular bisector of the straight-line segment $\overline{p_1 p_2}$ does not intersect the subpath $\pi(p_2, t)$.*

Proof. Let h^- be the half-plane defined by the perpendicular bisector of segment $\overline{p_1 p_2}$ that contains p_1. Assume there is a point q on the subpath $\pi(p_2, t)$ that is interior to h^-. Then $|p_1 q| < |p_2 q|$, which contradicts the definition of a self-approaching path. ◀

▶ **Lemma 9.** *Bends of a shortest self-approaching path in a simple polygon P form a subset of vertices of P.*

Thus, any point of a shortest self-approaching s-t path which is interior to P has a well-defined tangent. This point is an *inflection point*, if its tangent separates the self-approaching path in a small enough ε-neighborhood. We can also introduce a notion of an *inflection segment* for a path that contains a straight-line segment as a subpath. A straight-line segment of a path is an inflection segment if its supporting line separates the path in a small enough ε-neighborhood around the segment (refer to Fig. 3).

▶ **Lemma 10.** *A shortest self-approaching s-t path in a simple polygon P cannot have an inflection point (or an inflection segment) that is interior to P.*

Proof. Suppose a shortest self-approaching s-t path π has an inflection point p (or an inflection segment $\overline{pq}$) interior to P. Consider an ε-neighborhood of p (or $\overline{pq}$) for some small ε such that it is also interior to P, and it does not contain other inflection points. Choose a

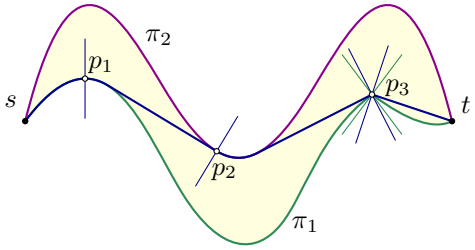


Figure 4 A geodesic bounded between two self-approaching s-t paths is also self-approaching.

point p_1 on subpath $\pi(s,p)$ close to p and draw a tangent through it to a subpath of $\pi(p,t)$ contained in the ε-neighborhood (refer to Fig. 3). Let p_2 be the tangent point. We can always choose p_1 such that the segment $\overline{p_1p_2}$ lies inside the ε-neighborhood. Let h_2 be the normal line to π drawn through p_2. Because π is self-approaching, the subpath $\pi(p_2,t)$ lies in the positive half-plane h_2^+. Therefore, none of the normal lines to $\overline{p_1p_2}$ intersects the subpath $\pi(p_2,t)$. Thus, $\pi(s,p_1) \oplus \overline{p_1p_2} \oplus \pi(p_2,t)$ is self-approaching and is shorter than π. ◀

Define the *inflection* points of a directed geodesic path γ from s to t as the first points of the inflection segments of γ, *i.e.*, the set of last points in the maximal subchains of γ with the same direction of turn.

▶ **Lemma 11.** *A shortest self-approaching path from s to t in a simple polygon P contains all the inflection points of the geodesic path from s to t.*

Proof. Consider an inflection segment $\overline{p_ip_j}$ of the geodesic path γ from s to t, p_i is one of its inflection points. Any shortest self-approaching path π intersects $\overline{p_ip_j}$. If the intersection point were not p_i, then π would contain an inflection point that is interior to P, but this would contradict Lemma 10. ◀

Consider two self-approaching paths π_1 and π_2 in a simple polygon P from s to t that do not have other points in common. Let γ be a geodesic path from s to t inside the area bounded by π_1 and π_2. Then, the following lemma holds.

▶ **Lemma 12.** *A geodesic path γ between two self-approaching paths π_1 and π_2 is also self-approaching.*

Proof. We use the fact that the geodesic lies inside of the convex hull of each side of the boundaries between which it is constrained, *i.e.*, $\gamma \subset CH(\pi_1)$ and $\gamma \subset CH(\pi_2)$.

Any point $p \in \gamma$ either lies on one of the paths π_1 and π_2 or on a straight line segment that is bitangent to the boundary (refer to Fig. 4).

Consider the case when p lies on π_1 or π_2, and is not a bend point (as point p_1 in the figure). Let, w.l.o.g., $p \in \pi_1$. The positive half-plane h^+ of the normal to π_1 at p contains the rest of the path $\pi_1(p,t)$. Therefore it contains the convex hull of $\pi_1(p,t)$, and the subpath $\gamma(p,t)$ of the geodesic.

When p lies on a path π_1 and is a bend point, the two normals to the path at p define two positive half-planes whose intersection contains the rest of the path from p to t. The two normals to the geodesic path at this point will lie in between the two normals to the boundary path (as in the figure for point p_3). Thus, the intersection of the two positive half-planes of the normals to the geodesic contains the convex hull of the subpath from s to t, and, therefore, the rest of the geodesic path $\gamma(p,t)$.

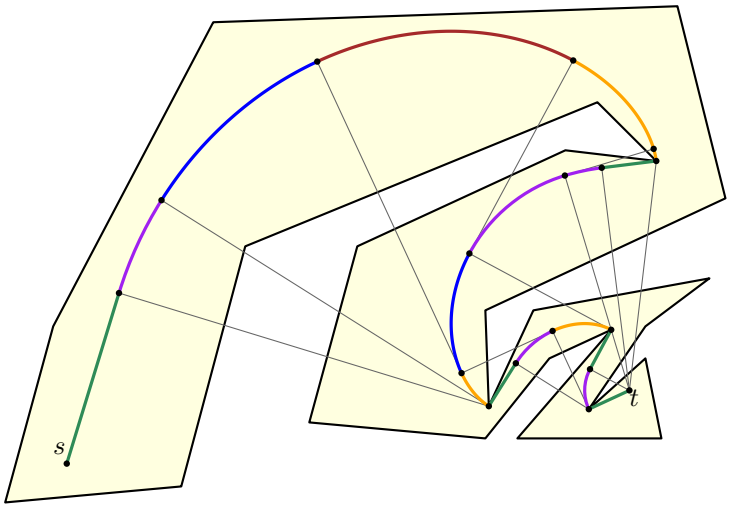


Figure 5 Shortest self-approaching path from s to t consists of straight-line segments, circular arcs, and involutes of a circle of some order. Straight segments are shown in green, circular arcs in purple, involutes of a circle of first order in orange, involutes of a circle of a second order in blue, and involutes of a circle of third order in brown.

In the case when p lies on a bitangent, consider its end point p_2. The normal to γ at p is parallel to the normal to γ at p_2. By one of the cases considered above, the positive half-plane at p_2 (or the intersection of two positive half-planes) will contain $\gamma(p_2, t)$, and, therefore, the positive half-plane of the normal to γ at p will contain the subpath $\gamma(p, t)$.

Thus, by the half-plane property, γ is self-approaching. ◀

As a corollary to this lemma, for two self-approaching paths from s to t, a path, composed of geodesics in the areas bounded by subpaths of the two paths between each pair of consecutive intersection points, is also self-approaching. In other words, let $s = p_0, p_1, \ldots, p_k, p_{k+1} = t$ be all the intersection points of π_1 and π_2 in the order they appear on π_1 and π_2. Observe that the intersection points must appear in the same order along the both paths, otherwise there would exist three points on one of these paths which would violate the definition of a self-approaching path. Let γ_i be the geodesic from p_i to p_{i+1} in the area between the two subpaths $\pi_1(p_i, p_{i+1})$ and $\pi_2(p_i, p_{i+1})$. Then,

▶ **Lemma 13.** *The concatenation of the geodesics $\gamma = \gamma_0 \oplus \gamma_1 \oplus \cdots \oplus \gamma_k$ is self-approaching.*

Proof. By a similar argument as in Lemma 12, for any normal to γ_i at point p, its positive half-plane either contains the convex hull of $\pi_1(p, t)$, or it contains the convex hull of $\pi_2(p, t)$. In both cases, that implies that the subpath $\gamma(p, t)$ lies in the positive half-plane of the normal. Therefore, γ is self-approaching. ◀

The next theorem is a direct corollary of Lemma 13.

▶ **Theorem 14.** *A shortest self-approaching s-t path is unique.*

Figure 5 shows an example of a shortest self-approaching path inside a polygon. In the following theorem we give its characterization.

▶ **Theorem 15.** *A shortest self-approaching s-t path in a simple polygon consists of straight-line segments, circular arcs, and circle involutes of some order.*

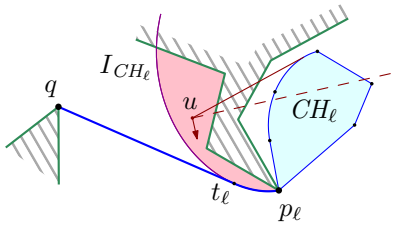


Figure 6 Illustration to Theorem 15.

Proof. Let $p_1, p_2, \ldots, p_k$ be the points of the shortest self-approaching s-t path π^* in the order from s to t, in which the path touches the boundary of P. Consider the last segment $\pi^*(p_k, t)$. It is a straight-line segment. Otherwise it could be shortened in the following way. Consider the last segment $\overline{qt}$ of a geodesic path from s to t, and extend it in the direction from t to q until intersecting path π^*; denote the intersection point as q'. (Note, that it exists, as the extension of $\overline{qt}$ beyond q until intersecting the boundary of P separates s from t.) Then, π^* can be shortened by replacing $\pi^*(q', t)$ by the segment $\overline{q't}$.

Now, suppose that all the segments $\pi^*(p_i, p_{i+1})$ consist of straight-line segments, circular arcs, or involutes of a circle of some order for all $i > \ell$ for some ℓ. We will show, that then, the segment $\pi^*(p_{\ell-1}, p_\ell)$ consists of straight-line segments, circular arcs, and/or involutes.

Denote $CH_\ell = CH(\pi^*(p_\ell, t))$. Let, w.l.o.g., π^* touch the boundary of the polygon at point p_ℓ on its left side (refer to Fig. 6). Then construct an involute I_{CH_ℓ} of the convex hull CH_ℓ starting at point p_ℓ with the tangent point moving in clockwise direction around CH_ℓ until the first intersection point of the involute with the boundary of P. The area D_ℓ on the concave side of the involute that it cuts off of the polygon P is a "dead" region for any self-approaching path that ends with the subpath $\pi^*(p_{\ell,t})$ (red area in the Fig. 6). In other words, for any point $u \in D_\ell$, any path connecting u to p_ℓ will have a normal that intersects CH_ℓ, and therefore the subpath $\pi^*(p_\ell, t)$. To show that, consider any piecewise-smooth path π_u from u to p_ℓ. Parameterize π_u for some parameter $\tau \in [0, 1]$, where $\pi_u(0) = u$ and $\pi_u(1) = p_\ell$. Consider the distance function $d_u(\tau)$ from a point moving along π_u to the involute I_{CH_ℓ}. This function will be piecewise smooth as both of the paths are piecewise-smooth. As a point, moving along π_u, has to eventually coincide with p_ℓ, there exists parameter τ' at which the distance function is decreasing, and therefore, the angle between a tangent vector to π_u at the point $u' = \pi_u(\tau')$ and a tangent from u' to the convex hull CH_ℓ is greater than $90°$. Therefore, a positive half-plane of the normal to π_u at point u' does not fully contain the convex hull CH_ℓ, and therefore, the path $\pi_u \oplus \pi^*(p_\ell, t)$ is not self-approaching.

Now, consider a geodesic path from s to p_ℓ in the region $P \backslash D_\ell$, and consider its last segment qp_ℓ, where q is the last point before p_ℓ that belongs to the boundary of P. This segment can be a straight-line segment, or a straight-line segment $\overline{qt_\ell}$ followed by a piece of the involute I_{CH_ℓ}, where $\overline{qt_\ell}$ is tangent to I_{CH_ℓ}. If segment qp_ℓ is not on π^*, then, by a similar argument as above, we can show that π^* can be shortened. Extend the segment $\overline{qt_\ell}$ beyond the point q until the intersection q' with π^*. Then, π^* can be shortened if the subpath $\pi^*(q'p_\ell)$ is replaced by the segment qp_ℓ of the geodesic.

The boundary of the convex hull CH_ℓ consists of straight-line segments and pieces of the subpath $\pi^*(p_\ell, t)$, which we assumed were straight segments, arcs, and circle involutes. Therefore, the segment qp_ℓ of the geodesic path also consists of straight segments, circular arcs, and circle involutes, possibly, of one order higher than the following subpath. Therefore, the shortest self-approaching path consists of straight-line segments, circular arcs, and circle involutes of some order, that is not higher than the number of bends on the path. ◀

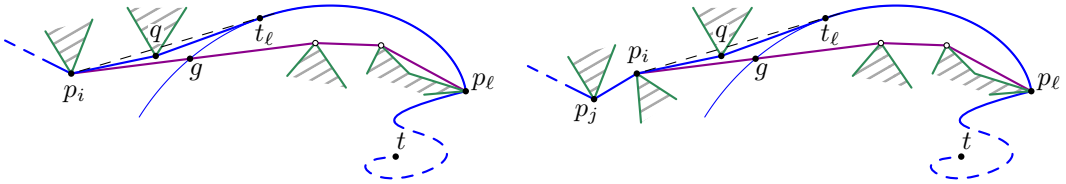


Figure 7 A subpath of a shortest self-approaching s-t path π^* (in blue) between two consecutive inflection points of the geodesic path γ (in purple) from s to t is geodesically convex. The last bend q of π^* before the vertex p_ℓ does not necessarily belong to γ.

In the last proof, the point q of the last segment qp_ℓ of the geodesic path from s to p_ℓ in $P \backslash D_\ell$ does not necessarily belong to the geodesic path from s to t. Consider an example in Fig. 7. In it, several vertices of the geodesic path γ are in the dead region (on the concave side of the involute). The tangent line from the last vertex (p_i in the left example, and p_j in the right example) of γ before p_ℓ that is not in the dead region intersects the boundary of the polygon. Angle $\angle p_i g t_\ell$, where g is the intersection point of γ with the involute, is an obtuse angle. This follows from the fact that the intersection angle between the straight-line segment $\overline{p_i p_\ell}$ and the tangent to the involute at the intersection point must not be greater than 90°, otherwise the point p_ℓ would not lie in the positive half-plane of the normal to the involute at the intersection point. Then, the total turn angle of the self-approaching path π^* from p_i to t_ℓ is less than 90°, and thus, the subpath $\pi^*(p_i, t_\ell)$ consists of straight-line segments. Let the previous inflection point of γ before p_ℓ be p_j, and the next inflection point of γ on or after p_ℓ be p_k. It follows then that the subpath $\pi^*(p_j, p_k)$ is *geodesically convex*, that is, the shortest path between any two points on $\pi^*(p_j, p_k)$ lies completely on one (and the same side) of the path. We obtain the following lemma.

▶ **Lemma 16.** *A shortest self-approaching s-t path in a simple polygon P consists of geodesically convex paths between inflection points of the geodesic from s to t.*

▶ **Theorem 17.** *A shortest self-approaching s-t path in a simple polygon P with n vertices consists of $O(n^2)$ segments. There exists a simple polygon P and two points s and t in it, such that the shortest self-approaching from s to t has $\Omega(n^2)$ segments.*

4 Existence of a self-approaching path

In this section we consider the question of testing whether, for given points s and t in a polygon P, they can be connected with a self-approaching path. In Theorem 15 we proved that a shortest self-approaching path can consist of involutes of a circle of high order, and in Section 2 we showed that such an involute is defined by a system of transcendental equations. In [14] Laczkovich proved a strengthening of Richardson's theorem, which states that in general the statement $\exists x : f(x) = 0$ is undecidable, where $f(x)$ is an expression generated by the rational numbers, the variable x, the operations of addition, multiplication, composition, and the sine function. Equations (1) describing the involutes are a special case of the class of expressions in Laczkovich's theorem. Nevertheless, it strongly suggests that an involute of a circle of order higher than one cannot be computed.

Next, we show an algorithm to test whether there exists a self-approaching path connecting two points s and t, and if so, to compute the shortest path, under the assumption that we can solve Equations (1). Subsequently, it is possible to release this assumption, and modify the algorithm to build an approximate solution, given that the shortest self-approaching path from s-to-t exists and there is a small leeway around it free of the polygon boundary points.

4.1 Shortest path algorithm

The proof of Theorem 15 is constructive. Assume that we can solve equations of the form as Equations (1) for an involute of order k in time $O(f(k))$, and evaluate the formula of the involute of order k for a given parameter θ in time $O(g(k))$. Then, we can decide if two points s and t can be connected by a self-approaching path, and we can construct the shortest path between the points. The outline of the algorithm is:

- Starting at t, move backwards along a geodesic s-t path γ. Maintain the convex hull CH of the final part of the shortest self-approaching path π^* to the destination t built so far.
- At every bend point p_ℓ:
  - Calculate the appropriate branch of an involute I_{CH} of the convex hull CH. If I_{CH} intersects the opposite boundary of the polygon, thus separating s from t, report that a self-approaching path from s to t does not exist and terminate the algorithm.
  - Otherwise, find a geodesic path γ_ℓ from the preceding inflection point of γ to p_ℓ in $P \backslash I_{CH}$, and add its last segment qp_ℓ as a prefix to π^*.
  - Update the convex hull CH. Repeat for the new bend point q, until s is reached. Report the found path π^*.

To obtain an algorithm with an optimal running time, there are a few considerations to take into account when constructing the shortest path. First, instead of unnecessarily constructing the whole involute I_{CH} until the intersection point with the boundary of P, and then discarding the part of it under the tangent line from q, its segments can be built one by one as needed up to the tangent point. Second, to optimally test if I_{CH} intersects the opposite boundary of the polygon, we can maintain a shortest path tree that will allow us to build funnels from the opposite sides of the polygon boundary. Third, it is not necessary to construct the whole geodesic γ_ℓ to be able to compute its last segment qp_ℓ. Instead, we can move backwards along γ, vertex by vertex, until we reach a point from which the tangent to I_{CH} can be constructed (possibly with adding new points along it).

Let the edges of P be oriented in counter-clockwise order. We shall call the two ends of an edge e, the *front*-point, and the *end*-point.

Next, we present the details of the algorithm.

Initialization step. Compute the shortest path tree SPT_s with root s [11], and preprocess it to answer the lowest common ancestor query [4]. Compute the geodesic γ from s to t, and store γ as a stack of vertices. Let the first and the last segments of γ be $\overline{sp'}$ and $\overline{p''t}$ respectively. Extend $\overline{sp'}$ beyond s until intersection with ∂P at some point a, and extend $\overline{p''t}$ beyond t until intersection with ∂P at some point b. This can be done in $O(\log n)$ time with a ray-shooting query after linear-time preprocessing of the polygon [7]. Let L be the chain of the boundary of P from b to a in counter-clockwise order, we shall call it the *left* chain. Similarly, let the *right* chain R be the chain of the boundary of P from a to b in counter-clockwise order. Initialize π^* and CH with the last segment $\overline{p''t}$ of γ, and pop the point t from γ.

The main loop. Let p_ℓ be the point on top of the stack γ, before the beginning of the current iteration of the loop. Let π^* touch ∂P in point p_ℓ on its left side (the case when π^* touches ∂P on its right side is equivalent). Let CH be the convex hull of the already built subpath $\pi^*(p_\ell, t)$.

Pop p_ℓ from the top of the stack γ. Consider the previous segment $\overline{p_i p_\ell}$ of γ (point p_i is currently on top of the stack γ).

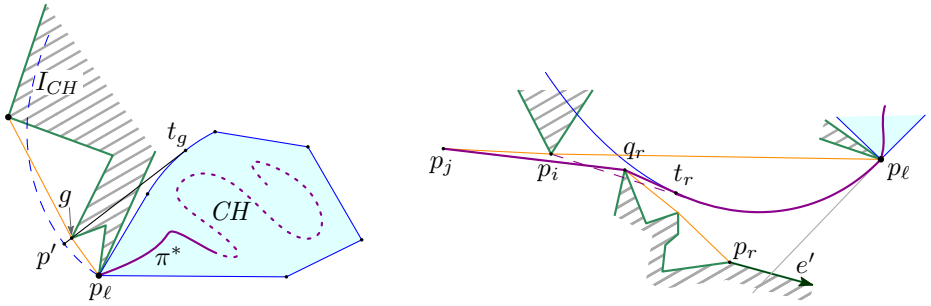


Figure 8 Illustration for Case 2 of the algorithm.

Case 1. If the angle between $\overline{p_\ell p_i}$ and the tangent in the clockwise direction to CH at point p_ℓ form an angle that is not less than 90°, then $\overline{p_i p_\ell}$ lies on the shortest self-approaching path from s to t; append $\pi^*(p_\ell, t)$ with $\overline{p_i p_\ell}$ in the front, and update the convex hull.

Case 2. If the angle between $\overline{p_\ell p_i}$ and the tangent in the clockwise direction to CH form an angle that is less than 90°, we need to calculate the involute I_{CH} of the convex hull for the tangent point moving clockwise around the boundary of CH starting at p_ℓ. We first will determine until which point to calculate I_{CH}.

First, we check whether the points on the geodesic path γ before p_ℓ lie in the dead region defined by I_{CH}. To do that without explicitly constructing I_{CH} first, for each point g on top of the stack γ, we construct a tangent line to CH which is leaving it on its right side (refer to Fig. 8 (left)). Let t_g be the tangent point on CH, and let $t_g \in I_g$ for some involute segment I_g on the boundary of the convex hull. Let $\overline{gt_g}$ intersect I_{CH} at point p'. We know that the length of the segment $\overline{p't_g}$ is equal to the length of the boundary of the convex hull CH from t_g to p_ℓ. Thus, to check whether g lies in the dead region we can compare the length of the segment $\overline{gt_g}$ to the length of the boundary of CH from t_g to g. If g does lie in the dead region, we simply remove it from the top of the stack γ, and proceed. If at some moment γ becomes empty, *i.e.*, the point s lies in the dead region, we report that a self-approaching path from s to t does not exist and terminate the algorithm.

Now, let p_i be the first point on γ before p_ℓ that does not lie in the dead region of I_{CH}. As in Fig. 7, the tangent segment from p_i to the involute may intersect the right chain of the boundary of P. Moreover, the right chain of the boundary of P may intersect I_{CH}. To test and account for that case, we do the following. Let $\vec{\tau} = \vec{I}'_{CH}(p_\ell)$ be the tangent vector to I_{CH} at point p_ℓ. Run a ray shooting query from p_ℓ in the direction $-\vec{\tau}$. Let it intersect an edge e' of R, and denote its front-point as p_r (refer to Fig. 8 (right)). Then, find a vertex p_j in the shortest path tree SPT_s that is the lowest common ancestor of p_ℓ and p_r. Let γ_ℓ and γ_r be the two shortest paths from p_j to p_ℓ and to p_r respectively. Paths γ_ℓ and γ_r form two convex chains. If γ_r does not intersect I_{CH}, then either a common tangent to γ_ℓ and I_{CH}, or a common tangent to γ_r and I_{CH}, will belong to π^*. To be able to compute the common tangents, we now explicitly construct I_{CH} segment by segment until a certain point. Let $p'p''$ be the last segment of I_{CH} constructed so far (with the curve orientation from p' to p''). We stop the construction of I_{CH} when the segment $\overline{p'p_j}$ makes a left turn with respect to the tangent vector $-\vec{\tau}$, where $\vec{\tau} = \vec{I}'_{CH}(p')$.

Whether γ_r intersects I_{CH} can be found during the computation of the common tangent. If it does, report that s and t cannot be connected with a self-approaching path and terminate the algorithm.

Let q_ℓ and q_r be the two tangent points on γ_ℓ and γ_r respectively of the common tangent lines with I_{CH}. One of the points q_ℓ and q_r, or both, will be equal to p_j.

- If $p_j = q_\ell = q_r$, then append π^* with $\overline{p_j t_j} \oplus I_{CH}(t_j, p_\ell)$, where t_j is the tangent point on I_{CH}.
- If $p_j = q_r \neq q_\ell$, then append π^* with $\gamma(p_j, q_\ell) \oplus \overline{q_\ell t_\ell} \oplus I_{CH}(t_\ell, p_\ell)$, where t_ℓ is the tangent point on I_{CH} of the common tangent with γ_ℓ.
- If $p_j = q_\ell \neq q_r$, then append π^* with $\gamma(p_j, q_r) \oplus \overline{q_r t_r} \oplus I_{CH}(t_r, p_\ell)$, where t_r is the tangent point on I_{CH} of the common tangent with γ_r.

Remove the points from γ until p_j is on top of the stack, and update CH. Iterate over the main loop until γ is empty, and return π^*.

In the full version of this paper [6] we discuss how to compute common tangents between a chain of involute segments of order $\leq k$ of size n and a polygonal chain of size m in $O(\log(m+n) + g(k)\log m + f(k))$ time, and between two chains of involutes of sizes n and m in $O(\log(m+n) + g(k)\log m + f(k))$ time. We also show how to test if two chains intersect in $O(\log(m+n) + (g(k)+f(k))\log m)$ time.

Maintaining *CH*. At the end of each iteration of the main algorithm, we need to update the convex hull of the subpath of the shortest self-approaching path built so far. This can involve finding a tangent from a point to a chain of involutes, or finding a common tangent of two chains of involutes.

Moreover, we want to be able to optimally calculate the length of a boundary from the current point p_ℓ to some point t_g. For that, associate two values $\text{dist}_{cw}(u)$ and $\text{dist}_{ccw}(u)$ to each end point of a segment on CH that will contain the distance to p_ℓ (up to some constant that will be equal for all the points) along the boundary in clockwise and counter-clockwise direction, respectively. Moreover, for two points u and v on CH, the length of the boundary between them can be calculated by $\text{dist}_{ccw}(v) - \text{dist}_{ccw}(u)$, if the chain of CH between u and v in counter-clockwise order does not contain p_ℓ. This fact will allow us to maintain the values in the points unchanged when updating the convex hull.

At every iteration of the algorithm, the distance from some tangent point t_g on an involute segment $p'p''$ to p_ℓ in the clockwise direction can be computed by formula $s(t_g) = \text{length}_I(t_g, p'') + \text{dist}_{cw}(p'') - \text{dist}_{cw}(p_\ell)$, where $\text{length}_I(t_g, p'')$ is the arc length of the involute from point t_g to p''. Analogously, the distance from t_g to p_ℓ in the counter-clockwise direction can be computed by taking $s(t_g) = \text{length}_I(t_g, p') + \text{dist}_{ccw}(p') - \text{dist}_{ccw}(p_\ell)$.

When updating the convex hull after extending the path π^*, we calculate the lengths of the tangent segments and the new involute arcs, and set the values $\text{dist}_{cw}(u)$ and $\text{dist}_{ccw}(u)$ to the new points of CH relatively to the values of the points remaining on CH. This will take $f(k)$ time to compute the arc length per segment of an involute of order k.

Taking these considerations into account, we conclude with the following theorem:

▶ **Theorem 18.** *The algorithm above constructs a shortest self-approaching path from s to t or reports that it does not exist in $O(K + \frac{n \log K}{\sqrt{K}}(g(\sqrt{K}) + f(\sqrt{K})))$ running time, where K is the size of the output, $f(k)$ is the time it takes to compute an involute of order k, and $g(k)$ is the time it takes to evaluate an involute of order k at a given point.*

5 Self-approaching polygon

A polygon is self-approaching, if for any two points there exists a self-approaching path connecting them.

▶ **Theorem 19.** *Polygon P is self-approaching if and only if for any disk D centered at any point $p \in P$, the intersection $D \cap P$ has one connected component.*

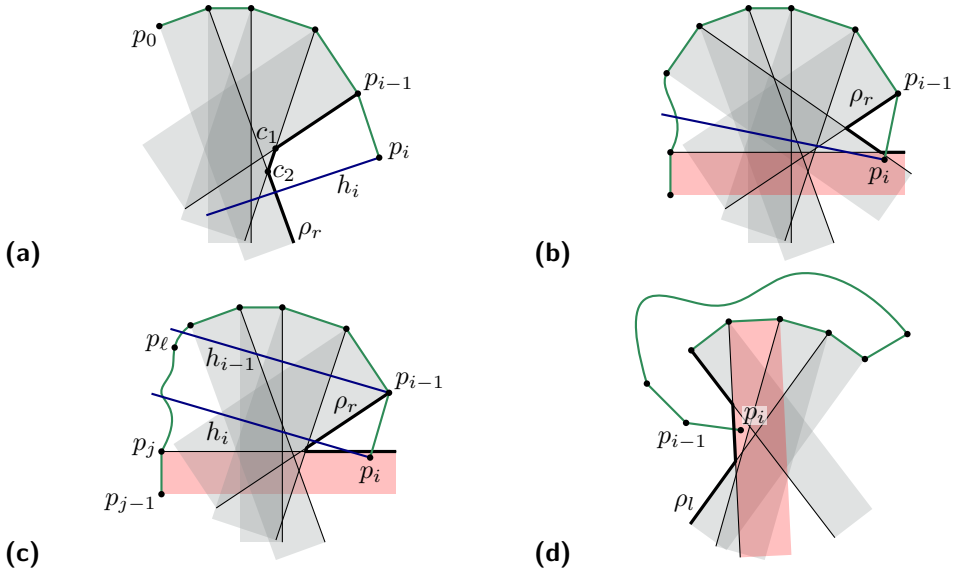


Figure 9 The illustration for Theorem 21.

Recall that a path is increasing-chord if it is self-approaching in both directions.

▶ **Corollary 20.** *Any self-approaching polygon is also increasing-chord.*

Next, we present an algorithm to test whether a given simple polygon P is self-approaching. Observe that from the proof of Theorem 19 the following property holds: the polygon P is self-approaching if and only if an area bounded between the two normals to e at its two end points in the right half-plane of e is free of ∂P, for all edges e on the boundary of P directed in counter-clockwise order. We call this area the *half-strip* of e. We will use this property to test efficiently if the polygon is self-approaching.

Let P be given as a set of points $p_0, p_1, \ldots, p_{n-1}$ in counter-clockwise order around the boundary. We will start at p_0, move along the boundary in counter-clockwise order and maintain the union of all the half-strips of the edges visited so far. More precisely, we will maintain the left and the right sides, ρ_l and ρ_r, of the hour-glass shape that is the union of the half-strips; ρ_l and ρ_r are convex polygonal chains (refer to Fig. 9). Store the segments of ρ_l and ρ_r as two lists, the last segments in the lists are infinite rays.

At every iteration of the algorithm, perform the following steps. Let p_i be the current point of the polygon P. The chain ρ_r contains the right side of the union of all the half-strips up to point p_{i-1}. Consider the next boundary segment $\overline{p_{i-1}p_i}$, and a perpendicular ray h_i at the point p_i (refer to Fig. 9 (a)). To update the chain ρ_r, do the following: Traverse ρ_r, and for every its segment $\overline{c_j c_{j+1}}$,

- if $\overline{p_{i-1}p_i}$ intersects $\overline{c_j c_{j+1}}$, then report that P is not self-approaching and terminate;
- if h_i intersects $\overline{c_j c_{j+1}}$, calculate the intersection point c', and replace the first elements of the list ρ_r up to $\overline{c_j c_{j+1}}$ with two segments, $\overline{p_i c'}$ and $\overline{c' c_{j+1}}$; repeat for the next point p_{i+1}.

Traverse the boundary of polygon P twice in counter-clockwise order, and then repeat the same algorithm traversing the boundary of P twice in clockwise order. If none of the segments $\overline{p_{i-1}p_i}$ intersected a segment of ρ_r, report that P is self-approaching.

▶ **Theorem 21.** *Given a simple polygon P with n vertices, the presented algorithm tests in $O(n)$ time if it is self-approaching.*

Proof. Consider the counter-clockwise traversal of the boundary of P. There are two cases when the boundary segment $\overline{p_{i-1}p_i}$ intersects ρ_r. In the first case, $\overline{p_{i-1}p_i}$ intersects ρ_r, and h_i does not intersect it (refer to Fig. 9 (b)). Let us call it the intersection of type 1. In the second case, $\overline{p_{i-1}p_i}$ intersects ρ_r after h_i intersects it (refer to Fig. 9 (c)). Let us call it the intersection of type 2. Moreover, the boundary segment $\overline{p_{i-1}p_i}$ may intersect ρ_l (refer to Fig. 9 (d)). Let us call it the intersection of type 3.

When traversing the polygon counter-clockwise, the presented algorithm will recognize the first type of the intersection, but not the second or third type. In case of the second type, during one iteration, the algorithm stops traversing ρ_r after finding the intersection point of h_i, and thus will not find the intersection of the segment with ρ_r. And in case of the third type, the algorithm does not check for intersection with ρ_l at all.

Nevertheless, we will prove, that by repeating the checks above twice and in two directions, counter-clockwise from p_0 to p_{n-1}, and clockwise from p_{n-1} to p_0, the algorithm will correctly decide if the polygon is self-approaching or not.

Case 0. If the polygon is self-approaching, then none of the segments will intersect ρ_r or ρ_l. The algorithm will traverse the polygon twice, then twice in clockwise direction, and report that it is self-approaching.

Case 1. If only the first intersection type occurs, then the algorithm will traverse the boundary of P until the first violation of the half-strip property, correctly report that the polygon is not self-approaching, and terminate.

Case 2. Suppose that the second intersection type occurs. Consider the first segment $\overline{p_{i-1}p_i}$, such that both h_i and $\overline{p_{i-1}p_i}$ intersect ρ_r. Let $\overline{p_{i-1}p_i}$ intersect the normal to some preceding segment $\overline{p_{j-1}p_j}$ at the point p_j. As the ray h_i intersects ρ_r before $\overline{p_{i-1}p_i}$ does, it also intersects the polygon boundary between the points p_j and p_{i-1}. And, therefore, the ray h_{i-1} perpendicular to $\overline{p_{i-1}p_i}$ at the point p_{i-1} also intersects the polygon boundary between the points p_j and p_{i-1}. Then, consider the behavior of the algorithm during the backwards traversal. Let p_ℓ for $j \leq \ell < i-1$ be the first point on the left side of the ray h_{i-1}. Then the segment $\overline{p_{\ell+1}, p_\ell}$ intersects h_{i-1}, and either the segment $\overline{p_{\ell+1}, p_\ell}$ intersects the left chain ρ_l or there was another segment before $\overline{p_{\ell+1}, p_\ell}$ that intersected ρ_l. Note, that because the intersection of $\overline{p_{i-1}p_i}$ and ρ_r was the first violation of the half-strip property in counter-clockwise order, the intersection of $\overline{p_{\ell+1}, p_\ell}$ and ρ_l cannot be of the second type, otherwise p_{i-1} would already lie on the right side of a normal to $\overline{p_{\ell+1}, p_\ell}$ at the point $\overline{p_{\ell+1}}$. Therefore, this intersection can only be of type one, and the algorithm will recognize it during the backwards traversal.

Case 3. Suppose that the third intersection type occurs. Then, there will be a segment $\overline{p_{j+1}, p_j}$ (where $j \geq i$), for which the first or the second intersection type occurs when traversing the polygon in the opposite direction, and thus either case 1 or case 2 applies.

Thus, we only need to explicitly check for the first intersection type. The running time of the algorithm is $O(n)$. At every iteration, the number of segments removed from the list ρ_r is equal to half the number of tests for intersections the algorithm makes, and the number of segments added back is at most 2. Therefore, the total number of segments that can be removed from ρ_r over one traversal of the boundary is not more than $2n$. Similarly, the total number of segments that can be removed from ρ_l over one traversal of the boundary is not more than $2n$. Therefore, the algorithm performs $O(n)$ intersection tests. ◀

Acknowledgements. This work was begun at the CMO-BIRS Workshop on Searching and Routing in Discrete and Continuous Domains, October 11–16, 2015.

References

1 O. Aichholzer, F. Aurenhammer, C. Icking, R. Klein, E. Langetepe, and G. Rote. Generalized self-approaching curves. *Discrete Applied Mathematics*, 109(1-2):3–24, 2001. doi:10.1016/S0166-218X(00)00233-X.

2 S. Alamdari, T. M. Chan, E. Grant, A. Lubiw, and V. Pathak. Self-approaching Graphs. In *20th International Symposium on Graph Drawing (GD)*, pages 260–271, 2012. doi:10.1007/978-3-642-36763-2_23.

3 E. M. Arkin, R. Connelly, and J. S. B. Mitchell. On monotone paths among obstacles with applications to planning assemblies. In *5th Annual Symposium on Computational Geometry (SCG)*, pages 334–343. ACM Press, 1989. doi:10.1145/73833.73870.

4 M. A. Bender and M. Farach-Colton. The LCA Problem Revisited. In *Latin American Symposium on Theoretical Informatics*, pages 88–94, 2000. doi:10.1007/10719839_9.

5 M. Biro, J. Iwerks, I. Kostitsyna, and J. S. B. Mitchell. Beacon-Based Algorithms for Geometric Routing. In *13th Algorithms and Data Structures Symposium (WADS)*, pages 158–169. Springer, 2013. doi:10.1007/978-3-642-40104-6_14.

6 P. Bose, I. Kostitsyna, and S. Langerman. Self-approaching paths in simple polygons. Preprint, http://arxiv.org/abs/1703.06107, 2017.

7 B. Chazelle, H. Edelsbrunner, M. Grigni, L. Guibas, J. Hershberger, M. Sharir, and J. Snoeyink. Ray shooting in polygons using geodesic triangulations. *Algorithmica*, 12(1):54–68, 1994. doi:10.1007/BF01377183.

8 H. Dehkordi, F. Frati, and J. Gudmundsson. Increasing-Chord Graphs On Point Sets. In *22nd International Symposium on Graph Drawing*, pages 464–475. Springer, 2014. doi:10.1007/978-3-662-45803-7_39.

9 L. E. Dubins. On Curves of Minimal Length with a Constraint on Average Curvature, and with Prescribed Initial and Terminal Positions and Tangents. *American Journal of Mathematics*, 79(3):497–516, 1957. doi:10.2307/2372560.

10 J. Gao and L. Guibas. Geometric algorithms for sensor networks. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 370(1958):27–51, 2012. doi:10.1098/rsta.2011.0215.

11 L. Guibas, J. Hershberger, D. Leven, M. Sharir, and R. Tarjan. Linear-time algorithms for visibility and shortest path problems inside triangulated simple polygons. *Algorithmica*, 2(1-4):209–233, 1987. doi:10.1007/BF01840360.

12 C. Icking and R. Klein. Searching for the kernel of a polygon – a competitive strategy. In *11th Annual Symposium on Computational Geometry (SCG)*, pages 258–266. ACM Press, 1995. doi:10.1145/220279.220307.

13 C. Icking, R. Klein, and E. Langetepe. Self-approaching curves. *Mathematical Proceedings of the Cambridge Philosophical Society*, 125(3):441–453, 1999. doi:10.1017/S0305004198003016.

14 M. Laczkovich. The removal of π from some undecidable problems involving elementary functions. *Proceedings of the American Mathematical Society*, 131(07):2235–2241, 2003. doi:10.1090/S0002-9939-02-06753-9.

15 J. S. B. Mitchell, C. Piatko, and E. M. Arkin. Computing a shortest k-link path in a polygon. In *33rd Annual Symposium on Foundations of Computer Science*, pages 573–582. IEEE, 1992. doi:10.1109/SFCS.1992.267794.

16 M. Nöllenburg, R. Prutkin, and I. Rutter. On self-approaching and increasing-chord drawings of 3-connected planar graphs. *Journal of Computational Geometry*, 7(1):47–69, 2016. doi:10.20382/jocg.v7i1a3.

17 G. Rote. Curves with increasing chords. *Mathematical Proceedings of the Cambridge Philosophical Society*, 115(01):1, 1994. doi:10.1017/S0305004100071875.

Maximum Volume Subset Selection for Anchored Boxes

Karl Bringmann[1], Sergio Cabello*[2], and Michael T. M. Emmerich[3]

1 **Max Planck Institute for Informatics, Saarland Informatics Campus, Saarbrücken, Germany**
2 **Department of Mathematics, IMFM, Ljubljana, Slovenia; and Department of Mathematics, FMF, University of Ljubljana, Ljubljana, Slovenia**
3 **Leiden Institute of Advanced Computer Science (LIACS), Leiden University, Leiden, The Netherlands**

Abstract

Let B be a set of n axis-parallel boxes in $\mathbb{R}^d$ such that each box has a corner at the origin and the other corner in the positive quadrant of $\mathbb{R}^d$, and let k be a positive integer. We study the problem of selecting k boxes in B that maximize the volume of the union of the selected boxes. The research is motivated by applications in skyline queries for databases and in multicriteria optimization, where the problem is known as the hypervolume subset selection problem. It is known that the problem can be solved in polynomial time in the plane, while the best known running time in any dimension $d \geq 3$ is $\Omega\big(\binom{n}{k}\big)$. We show that:

- The problem is NP-hard already in 3 dimensions.
- In 3 dimensions, we break the bound $\Omega\big(\binom{n}{k}\big)$, by providing an $n^{O(\sqrt{k})}$ algorithm.
- For any constant dimension d, we give an efficient polynomial-time approximation scheme.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases geometric optimization, subset selection, hypervolume indicator, Klee's measure problem, boxes, NP-hardness, PTAS

Digital Object Identifier 10.4230/LIPIcs.SoCG.2017.22

1 Introduction

An *anchored box* is an orthogonal range of the form $\textsc{box}(p) := [0, p_1] \times \ldots \times [0, p_d] \subset \mathbb{R}^d_{\geq 0}$, spanned by the point $p \in \mathbb{R}^d_{>0}$. This paper is concerned with the problem Volume Selection: Given a set P of n points in $\mathbb{R}^d_{>0}$, select k points in P maximizing the volume of the union of their anchored boxes. That is, we want to compute

$$\textsc{VolSel}(P, k) := \max_{S \subseteq P,\, |S|=k} \textsc{vol}\Big(\bigcup_{p \in S} \textsc{box}(p) \Big),$$

as well as a set $S^* \subseteq P$ of size k realizing this value. Here, vol denotes the usual volume.

Motivation

This geometric problem is of key importance in the context of multicriteria optimization and decision analysis, where it is known as the *hypervolume subset selection problem (HSSP)*

* Supported by the Slovenian Research Agency, program P1-0297 and project L7-5459.

[2, 3, 4, 24, 12, 13]. In this context, the points in P correspond to solutions of an optimization problem with d objectives, and the goal is to find a small subset of P that "represents" the set P well. The quality of a representative subset $S \subseteq P$ is measured by the volume of the union of the anchored boxes spanned by points in S; this is also known as the *hypervolume indicator* [34]. Note that with this quality indicator, finding the optimal size-k representation is equivalent to our problem VolSel(P, k). In applications, such bounded-size representations are required in archivers for non-dominated sets [23] and for multicriteria optimization algorithms and heuristics [3, 10, 7].[1] Besides, the problem has recently received attention in the context of skyline operators in databases [17].

In 2 dimensions, the problem can be solved in polynomial time [2, 13, 24], which is used in applications such as analyzing benchmark functions [2] and efficient postprocessing of multiobjective algorithms [12]. A natural question is whether efficient algorithms also exist in dimension $d \geq 3$, and thus whether these applications can be pushed beyond two objectives.

In this paper, we answer this question negatively, by proving that Volume Selection is NP-hard already in 3 dimensions. We then consider the question whether the previous $\Omega(\binom{n}{k})$ bound can be improved, which we answer affirmatively in 3 dimension. Finally, in any constant dimension, we improve the best-known $(1 - 1/e)$-approximation to an efficient polynomial-time approximation scheme (EPTAS). See Section 1.2 for details.

1.1 Further Related Work

Klee's Measure Problem

To compute the volume of the union of n (not necessarily anchored) axis-aligned boxes in $\mathbb{R}^d$ is known as Klee's measure problem. The fastest known algorithm takes time[2] $O(n^{d/2})$, which can be improved to $O(n^{d/3}\text{polylog}(n))$ if all boxes are cubes [15]. By a simple reduction [8], the same running time as on cubes can be obtained on anchored boxes, which can be improved to $O(n \log n)$ for $d \leq 3$ [6]. These results are relevant to this paper because Klee's measure problem on anchored boxes (spanned by the points in P) is a special case of Volume Selection (by calling VolSel$(P, |P|)$).

Chan [14] gave a reduction from k-Clique to Klee's measure problem in $2k$ dimensions. This proves NP-hardness of Klee's measure problem when d is part of the input (and thus d can be as large as n). Moreover, since k-Clique has no $f(k) \cdot n^{o(k)}$ algorithm under the Exponential Time Hypothesis [16], Klee's measure problem has no $f(d) \cdot n^{o(d)}$ algorithm under the same assumption. The same hardness results also hold for Klee's measure problem on anchored boxes, by a reduction in [8] (NP-hardness was first proven in [11]).

Finally, we mention that Klee's measure problem has a very efficient randomized $(1 \pm \varepsilon)$-approximation algorithm in time $O(n \log(1/\delta)/\varepsilon^2)$ with error probability δ [9].

Known Results for Volume Selection

As mentioned above, 2-dimensional Volume Selection can be solved in polynomial time; the initial $O(kn^2)$ algorithm [2] was later improved to $O((n-k)k + n \log n)$ [13, 24]. In higher dimensions, by enumerating all size-k subsets and solving an instance of Klee's measure problem on anchored boxes for each one, there is an $O\big(\binom{n}{k} k^{d/3}\text{polylog}(k)\big)$ algorithm. For

[1] We remark that in these applications the anchor point is often not the origin, however, by a simple translation we can move our anchor point from $(0, \ldots, 0)$ to any other point in $\mathbb{R}^d$.

[2] In O-notation, we always assume d to be a constant, and $\log(x)$ is to be understood as $\max\{1, \log(x)\}$.

small $n-k$, this can be improved to $O(n^{d/2}\log n + n^{n-k})$ [10]. Volume Selection is NP-hard when d is part of the input, since the same holds already for Klee's measure problem on anchored boxes. However, this does not explain the exponential dependence on k for constant d.

Since the volume of the union of boxes is a submodular function (see, e.g., [31]), the greedy algorithm for submodular function maximization [27] yields a $(1-1/e)$-approximation of VolSel(P,k). This algorithm solves $O(nk)$ instances of Klee's measure problem on at most k anchored boxes, and thus runs in time $O(nk^{d/3+1}\text{polylog}(k))$. Using [9], this running time improves to $O(nk^2\log(1/\delta)/\varepsilon^2)$, at the cost of decreasing the approximation ratio to $1-1/e-\varepsilon$ and introducing an error probability δ. See [20] for related results in 3 dimensions.

A problem closely related to Volume Selection is Convex Hull Subset Selection: Given n points in $\mathbb{R}^d$, select k points that maximize the volume of their convex hull. For this problem, NP-hardness was recently announced in the case $d=3$ [28].

1.2 Our Results

In this paper we push forward the understanding of Volume Selection. We prove that Volume Selection is NP-hard already for $d=3$ (Section 3). Previously, NP-hardness was only known when d is part of the input and thus can be as large as n. Moreover, this establishes Volume Selection as another example for problems that can be solved in polynomial time in the plane but are NP-hard in three or more dimensions (see also [5, 26]).

In the remainder, we focus on the regime where $d\geq 3$ is a constant and $k\ll n$. All known algorithms (explicitly or implicitly) enumerate all size-k subsets of the input set P and thus take time $\Omega\big(\binom{n}{k}\big)=n^{\Omega(k)}$. In 3 dimensions, we break this time bound by providing an $n^{O(\sqrt{k})}$ algorithm (Section 4). To this end, we project the 3-dimensional Volume Selection to a 2-dimensional problem and then use planar separator techniques.

Finally, in Section 5 we design an EPTAS for Volume Selection. More precisely, we give a $(1-\varepsilon)$-approximation algorithm running in time $O((n/\varepsilon^d)(\log n+k+2^{O(\varepsilon^{-2}\log 1/\varepsilon)^d}))$, for any constant dimension d. Note that the "combinatorial explosion" is restricted to d and ε; for any constant d,ε the algorithm runs in time $O(n(k+\log n))$. This improves the previously best-known $(1-1/e)$-approximation, even in terms of running time.

2 Preliminaries

All boxes considered in the paper are axis-parallel and anchored at the origin. For points $p=(p_1,\dots,p_d)$, $q=(q_1,\dots,q_d)\in\mathbb{R}^d$, we say that p *dominates* q if $p_i\geq q_i$ for all $1\leq i\leq d$. For $p=(p_1,\dots,p_d)\in\mathbb{R}^d_{>0}$, we let $\text{box}(p):=[0,p_1]\times\ldots\times[0,p_d]$. Note that $\text{box}(p)$ is the set of all points $q\in\mathbb{R}^d_{\geq 0}$ that are dominated by p. A *point set* P is a set of points in $\mathbb{R}^d_{>0}$. We denote the union $\bigcup_{p\in P}\text{box}(p)$ by $\mathcal{U}(P)$. The usual Euclidean volume is denoted by vol. With this notation, we set

$$\mu(P):=\text{vol}(\mathcal{U}(P))=\text{vol}\Big(\bigcup_{p\in P}\text{box}(p)\Big)=\text{vol}\Big(\bigcup_{p\in P}[0,p_1]\times\ldots\times[0,p_d]\Big).$$

We study Volume Selection: Given a point set P of size n and $0\leq k\leq n$, compute

$$\text{VolSel}(P,k):=\max_{S\subseteq P,\,|S|=k}\mu(S).$$

Note that we can relax the requirement $|S|=k$ to $|S|\leq k$ without changing this value.

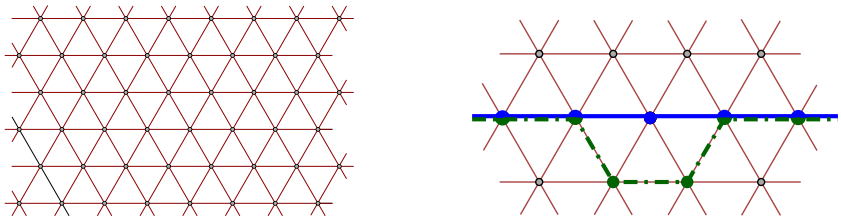


Figure 1 Left: triangular grid Γ. Right: choosing the parity of paths.

3 Hardness in 3 dimensions

We consider the following decision variant of 3-dimensional Volume Selection: Given a triple (P, k, V), where P is a set of points in $\mathbb{R}^3_{>0}$, k is a positive integer and V is a positive real value, is there a subset $Q \subseteq P$ of k points such that $\mu(Q) \geq V$?

We are going to show that the problem is NP-complete. First, we show that an intermediate problem about selecting a large independent set in a given induced subgraph of the triangular grid is NP-hard. Then we argue that this problem can be embedded using boxes whose points lie in two parallel planes. One plane is used to define the triangular-grid-like structure and the other is used to encode the subset of vertices that describe the induced subgraph of the grid.

3.1 Triangular grid

Let Γ be the infinite graph with vertex set and edge set (see Figure 1):

$$V(\Gamma) = \{(i + j \cdot 1/2, j \cdot \sqrt{3}/2) \mid i, j \in \mathbb{N}\},$$
$$E(\Gamma) = \{ab \mid a, b \in V(\Gamma), \text{ the Euclidean distance between } a \text{ and } b \text{ is exactly } 1\}.$$

We use the problem Independent Set on Induced Triangular Grid: Given a pair (A, ℓ), where A is a subset of $V(\Gamma)$ and ℓ is a positive integer, is there a subset $B \subseteq A$ of ℓ vertices such that no two vertices of B are connected by an edge of $E(\Gamma)$?

▶ **Lemma 3.1.** Independent Set on Induced Triangular Grid *is NP-complete.*

Proof Sketch. Garey and Johnson [19] show that the problem Vertex Cover is NP-complete for planar graphs of degree at most 3, which implies that Independent Set is NP-complete for planar graphs of degree at most 3.

Given a planar graph G of degree at most 3, we construct an orthogonal drawing of G on a square grid of polynomial size [29, 30] and transform it into a drawing of G on Γ. Rescaling and rerouting, we get a graph H that is an induced subgraph of Γ, and a subdivision of G where each edge of G is path in H with an even number of interior vertices. See Figure 1, right, to see how to choose the parity of the path. If $\alpha(G)$ is the size of the largest independent set in G, and each edge uv of G is represented by a path with $2k_{uv}$ internal vertices, then $\alpha(H) = \alpha(G) + \sum_{uv \in E(G)} k_{uv}$. Indeed, we can obtain H from G by repeatedly replacing an edge by a 3-edge path, and any such replacement increases the size of the largest independent set by exactly 1. ◀

3.2 The point set

Let $m \geq 3$ be an arbitrary integer and consider the point set P_m defined by $P_m = \{(x, y, z) \in \mathbb{N}^3 \mid x + y + z = m\}$, see Figure 2. Standard induction shows that the set P_m has $(m-1)(m-2)/2$ points and that $\mu(P_m) = m(m-1)(m-2)/6$.

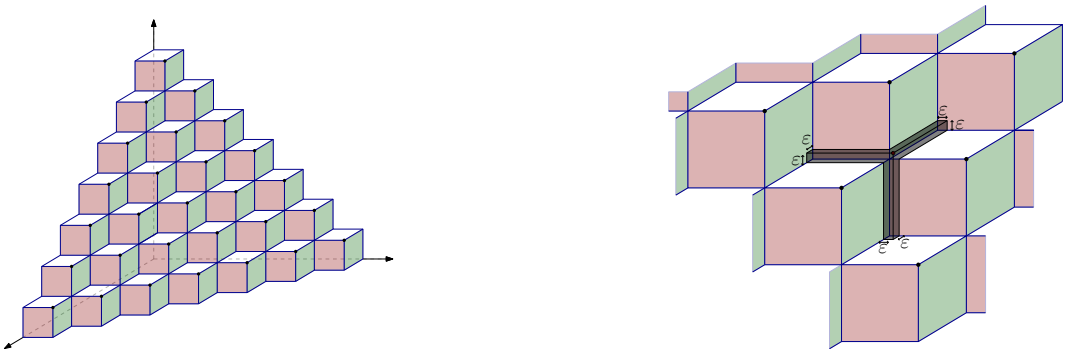


Figure 2 Left: the point set P_m and the boxes $\textsc{box}(p)$, with $p \in P_m$. Right: the point $q = p + \Delta_\varepsilon$ and the set $\textsc{diff}(q)$.

Consider the real number $\varepsilon = 1/4m^2$, and define the vector $\Delta_\varepsilon = (\varepsilon, \varepsilon, \varepsilon)$. Note that ε is much smaller than 1. For each point $p \in P_{m-1}$, consider the point $p + \Delta_\varepsilon$, see Figure 2, right. Let us define the set $Q_m = \{p + \Delta_\varepsilon \mid p \in P_{m-1}\}$. It is clear that Q_m has $|P_{m-1}| = (m-2)(m-3)/2$ points, for $m \geq 3$. The points of Q_m lie on the plane $x + y + z = m - 1 + 3\varepsilon$. For each point q of Q_m define

$$\textsc{diff}(q) \;=\; \mathcal{U}\big(P_m \cup \{q\}\big) \setminus \mathcal{U}\big(P_m\big) \;=\; \Big(\bigcup_{p \in P_m \cup \{q\}} \textsc{box}(p)\Big) \setminus \Big(\bigcup_{p \in P_m} \textsc{box}(p)\Big).$$

Note that $\textsc{diff}(q)$ is the union of 3 boxes of size $\varepsilon \times \varepsilon \times 1$ and a cube of size $\varepsilon \times \varepsilon \times \varepsilon$, see Figure 2, right. The sets and the parameter ε are selected to have the following properties.

▶ **Lemma 3.2.** *The following holds.*

- *If $Q' \subseteq Q_m$ and the sets $\textsc{diff}(q)$, for all $q \in Q'$, are pairwise disjoint, then $\mu(P_m \cup Q') = \mu(P_m) + |Q'| \cdot (3\varepsilon^2 + \varepsilon^3)$.*
- *If $Q' \subseteq Q_m$ and Q' contains two points q_0 and q_1 such that $\textsc{diff}(q_0)$ and $\textsc{diff}(q_1)$ intersect, then $\mu(P_m \cup Q') < \mu(P_m) + |Q'| \cdot (3\varepsilon^2 + \varepsilon^3)$.*
- *If P' is a subset of P_m such that $P_m \setminus P'$ is non-empty, then $\mu(P' \cup Q_m) < \mu(P_m)$.*

3.3 The reduction

We can define naturally a graph T_m on the set Q_m by using the intersection of the sets $\textsc{diff}(\cdot)$. The vertex set of T_m is Q_m, and two points $q, q' \in Q_m$ define an edge qq' of T_m if and only if $\textsc{diff}(q)$ and $\textsc{diff}(q')$ intersect, see Figure 3. Simple geometry shows that T_m is isomorphic to a part of the triangular grid Γ, up to scaling. Thus, choosing m large enough, we can get an arbitrarily large portion of the triangular grid Γ. Note that a subset of vertices $Q' \subseteq Q_m$ is independent in T_m if and only if the sets $\{\textsc{diff}(q) \mid q \in Q'\}$ are pairwise disjoint.

▶ **Theorem 3.3.** *The problem* Volume Selection *is NP-complete in* 3 *dimensions.*

Proof. Consider an instance (A, ℓ) to Independent Set on Induced Triangular Grid, where A is a subset of the vertices of the triangular grid Γ and ℓ is an integer. Take m large enough so that T_m is isomorphic to an induced subgraph of Γ that contains A. For each vertex v of T_m let $\psi_\Gamma(v)$ be the corresponding vertex of Γ. For each subset B of A, let $Q_m(B)$ be the subset of T_m that corresponds to B, that is, $Q_m(B) = \{q \in Q_m \mid \psi_\Gamma(q) \in B\}$.

Consider the set of points $P = P_m \cup Q_m(A)$, the parameter $k = (m-1)(m-2)/2 + \ell$, and the value $V = \frac{m(m-1)(m-2)}{6} + \ell \cdot (3\varepsilon^2 + \varepsilon^3)$. Then we can show that (A, ℓ) is a yes

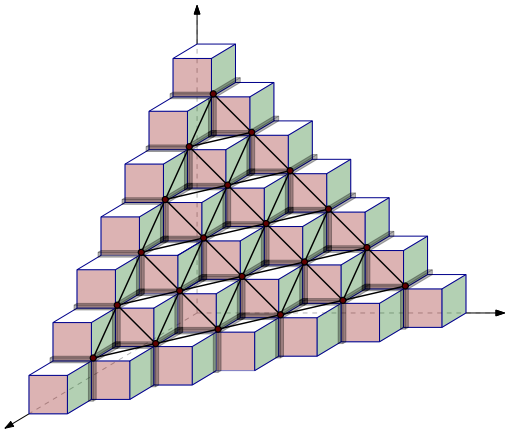


Figure 3 The graph T_m for $m = 9$.

instance for Independent Set on Induced Triangular Grid if and only if (P, k, V) is a yes instance for Volume Selection.

If (A, ℓ) is a yes instance for Independent Set on Induced Triangular Grid, there is a subset $B \subseteq A$ of ℓ independent vertices in Γ. This implies that $Q_m(B)$ is an independent set in T_m, that is, the sets $\{\text{DIFF}(q) \mid q \in Q_m(B)\}$ are pairwise disjoint. Lemma 3.2 then implies that

$$\mu(P_m \cup Q_m(B)) \;=\; \mu(P_m) + |B| \cdot (3\varepsilon^2 + \varepsilon^3) \;=\; \frac{m(m-1)(m-2)}{6} + \ell \cdot (3\varepsilon^2 + \varepsilon^3) \;=\; V.$$

Therefore $P_m \cup Q_m(B)$ is a subset of P with $|P_m| + |B| = (m-1)(m-2)/2 + \ell = k$ points such that $\mu(P_m \cup Q_m(B)) = V$ and thus (P, k, V) is a yes instance for Volume Selection.

Assume now that (P, k, V) is a yes instance for Volume Selection. This means that P contains a subset Q of k points such that

$$\mu(Q) \;\geq\; V \;=\; \frac{m(m-1)(m-2)}{6} + \ell \cdot (3\varepsilon^2 + \varepsilon^3) \;=\; \mu(P_m) + \ell \cdot (3\varepsilon^2 + \varepsilon^3) \;>\; \mu(P_m).$$

Because of Lemma 3.2, it must be that P_m is contained in Q, as otherwise we would have $\mu(Q) < \mu(P_m)$. Since we have $P_m \subset Q$ and $P = P_m \cup Q_m(A)$, we obtain that Q is $P_m \cup Q_m(B)$ for some $B \subseteq A$. Moreover, $|B| = k - |P_m| = \ell$. By Lemma 3.2, if $Q_m(B)$ is not an independent set in T_m, we have

$$\mu(Q) \;=\; \mu(P_m \cup Q_m(B)) \;<\; \mu(P_m) + \ell(3\varepsilon^2 + \varepsilon) \;=\; V,$$

which contradicts the assumption that $\mu(Q) \geq V$. Thus it must be that $Q_m(B)$ is an independent set in T_m. It follows that $B \subset A$ has size ℓ and is an independent set in Γ, and thus (A, ℓ) is a yes instance for Independent Set on Induced Triangular Grid. ◀

4 Exact Algorithm in 3 Dimensions

In this section we design an algorithm to solve Volume Selection in 3 dimensions in time $n^{O(\sqrt{k})}$. The main insight is that, for an optimal solution Q^*, the boundary of $\mathcal{U}(Q^*)$ is a planar graph with $O(k)$ vertices, and therefore has a balanced separator with $O(\sqrt{k})$ vertices. We would like to guess the separator, break the problem into two subproblems, and solve each of them recursively. This basic idea leads to a few technical challenges to take care of.

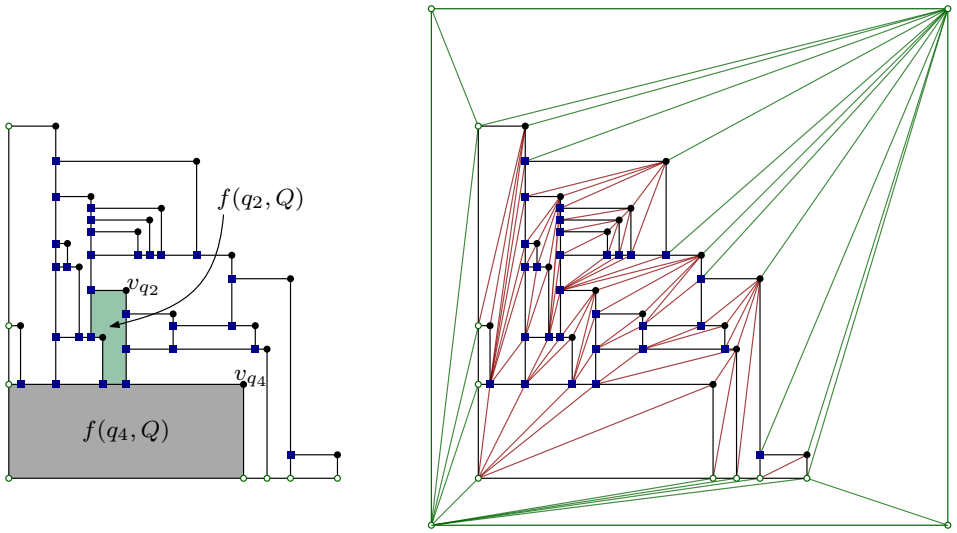


Figure 4 The graphs $G(Q)$ (left) and $T(Q)$ (right).

One obstacle is that subproblems should be really independent because we do not want to double count some covered parts. Essentially, a separator in the graph-theory sense does not imply independent subproblems in our context. Another technicality is that some of the subproblems that we encounter recursively cannot be solved optimally; we can only get a lower bound to the optimal value. However, for the subproblems that define the optimal solution at the higher level of the recursion, we do compute an optimal solution.

Let P be a set of n points in the positive quadrant of $\mathbb{R}^3$. Through our discussion, we will assume that P is fixed and thus drop the dependency on P and n from the notation. We can assume that no point of P is dominated by another point of P. Using an infinitesimal perturbation of the points, we can assume that all points have all coordinates different. Let M be the largest x- or y-coordinate in P, thus $M = \max\{p_x, p_y \mid p \in P\}$. We define σ to be the square in $\mathbb{R}^2$ defined by $[-1, M+1] \times [-1, M+1]$. It has side length $M+2$.

For each subset Q of P, consider the projection of $\mathcal{U}(Q)$ onto the xy-plane. This defines a plane graph, which we denote by $G(Q)$; see Figure 4, left. We consider $G(Q)$ as a geometric, embedded graph where each vertex is a point and each edge is a horizontal or vertical straight-line segment on the xy-plane. The projection of each point $q \in Q$ defines a vertex, which we denote by v_q. Each vertex $q \in Q$ defines a bounded face $f(q, Q)$ in $G(Q)$. This is the projection of the face on the boundary of $\mathcal{U}(Q)$ contained in the plane $\{(x, y, z) \in \mathbb{R}^3 \mid z = q_z\}$. In fact, each bounded face of $G(Q)$ is $f(q, Q)$ for some $q \in Q$. We triangulate each bounded face $f(q, Q)$ of $G(Q)$ *canonically*, see Figure 4 right. We add all possible edges from the top rightmost vertex v_q, then all possible edges from the bottom leftmost vertex, and finally all edges from the left bottom-most vertex. This is the canonical triangulation of the face $f(q, Q)$, and we apply it to each bounded face of $G(Q)$. The outer face of $G(Q)$ may also have many vertices. We place on top the square σ, with vertices $\{-1, M+1\}^2$, and triangulate in some systematic way. Let $T(Q)$ be the resulting geometric, embedded graph, see Figure 4, right. The graph $T(Q)$ is a triangulation of the square σ with internal vertices. It is easy to see that $G(Q)$ and $T(Q)$ have $O(|Q|)$ vertices and edges.

A polygonal domain is a subset of the plane defined by a polygon where we remove the interior of some polygons, which form holes. A polygonal domain D is ***Q*-compliant** if its boundary is contained in the edge set of $T(Q)$. Note that a Q-compliant polygonal domain has $O(|Q|)$ edges because the graph $T(Q)$ has $O(|Q|)$ edges.

We are going to use dynamic programming based on planar separators of $T(Q^*)$ for an optimal solution Q^*. A ***valid tuple*** to define a subproblem is a tuple (S, D, ℓ), where $S \subset P$, D is an S-compliant polygonal domain, and ℓ is a positive integer. The tuple (S, D, ℓ) models a subproblem where the points of S are already selected to be part of the feasible solution, D is a S-compliant domain so that we only care about the volume inside the cylinder $D \times \mathbb{R}$, and we can still select ℓ points from $P \cap (D \times \mathbb{R})$. We have two different values associated to each valid tuple, depending on which subsets Q of vertices from $P \cap D$ can be selected:

$$\begin{aligned}\Phi_{\text{free}}(S, D, \ell) &= \max\{\text{VOL}(\mathcal{U}(S \cup Q) \cap (D \times \mathbb{R})) \mid Q \subset P \cap (D \times \mathbb{R}),\ |Q| \leq \ell\}.\\ \Phi_{\text{comp}}(S, D, \ell) &= \max\{\text{VOL}(\mathcal{U}(S \cup Q) \cap (D \times \mathbb{R})) \mid Q \subset P \cap (D \times \mathbb{R}),\ |Q| \leq \ell,\\ &\qquad D \text{ is } (S \cup Q)\text{-compliant}\}.\end{aligned}$$

Obviously, for all valid tuples (S, D, ℓ) we have $\Phi_{\text{comp}}(S, D, \ell) \leq \Phi_{\text{free}}(S, D, \ell)$. On the other hand, we are interested in the valid tuple $(\emptyset, \sigma, k)$, for which we have $\Phi_{\text{free}}(\emptyset, \sigma, k) = \Phi_{\text{comp}}(\emptyset, \sigma, k)$.

We would like to get a recursive formula for $\Phi_{\text{free}}(S, D, \ell)$ or $\Phi_{\text{comp}}(S, D, \ell)$ using planar separators. More precisely, we would like to use a separator in $T(S \cup Q^*)$ for an optimal solution, and then branch on all possible such separators. However, none of the two definitions seem good enough for this. If we would use $\Phi_{\text{free}}(S, D, \ell)$, then we divide into domains that may have too much freedom and the interaction between subproblems gets complex. If we would use $\Phi_{\text{comp}}(S, D, \ell)$, then merging the problems becomes an issue. Thus, we take a mixed route where we argue that, for the valid tuples that are relevant for finding the optimal solution, we actually have $\Phi_{\text{free}} = \Phi_{\text{comp}}$.

A ***valid partition*** π of (S, D, ℓ) is a collection of valid tuples $\pi = \{(S_1, D_1, \ell_1), \ldots, (S_t, D_t, \ell_t)\}$ such that

- $S_1 = \cdots = S_t = S \cup S_0$ for some set $S_0 \subset P \cap D$;
- $|S_0| = O\left(\sqrt{|S| + \ell}\right)$;
- the domains $D_1, \ldots, D_t$ have pairwise disjoint interiors and $D = \bigcup_i D_i$;
- $\ell = |S_0| + \sum_i \ell_i$; and
- $\ell_i \leq 2\ell/3$ for each $i = 1, \ldots, t$.

Let $\Pi(S, D, \ell)$ be the family of valid partitions for the tuple (S, D, ℓ). We remark that different valid partitions may have different cardinality.

▶ **Lemma 4.1.** *For each valid tuple (S, D, ℓ) we have*

$$\begin{aligned}\Phi_{\text{free}}(S, D, \ell) &\geq \max_{\pi \in \Pi(S,D,\ell)} \sum_{(S', D', \ell') \in \pi} \Phi_{\text{free}}(S', D', \ell'),\\ \Phi_{\text{comp}}(S, D, \ell) &\leq \max_{\pi \in \Pi(S,D,\ell)} \sum_{(S', D', \ell') \in \pi} \Phi_{\text{comp}}(S', D', \ell').\end{aligned}$$

Proof Sketch. For the first inequality, we show that, for each $\pi \in \Pi(S, D, \ell)$, joining solutions to the subproblems $\Phi_{\text{free}}(\cdot)$ defined by $\{(S', D', \ell') \mid (S', D', \ell') \in \pi\}$ gives a feasible solution for the problem $\Phi_{\text{free}}(S, D, \ell)$.

For the second inequality, we consider an optimal solution $Q^* \subseteq P \cap D$ with at most ℓ points for the problem $\Phi_{\text{comp}}(S, D, \ell)$. The triangulation $T(S \cup Q^*)$ is a 3-connected planar graph and the boundary of D is contained in $T(S \cup Q^*)$ because D is $(S \cup Q^*)$-compliant. We now use the cycle-separator theorem of Miller [25] to split the vertices of Q^*: There is a cycle γ in $T(S \cup Q^*)$ of length $O(\sqrt{|S| + \ell})$ such that the interior of γ has at most $2|Q^*|/3$ vertices of Q^* and the exterior of γ has at most $2|Q^*|/3$ vertices of Q^*. Using this

cycle separator we can build a valid partition $\pi_\gamma \in \Pi(S, D, \ell)$ such that $Q^* \cap D'$ is a feasible solution to each $(S', D', \ell') \in \pi_\gamma$. For the correctness argument, we use an easy monotonicity property of being Q-compliant, which we skip in this short version. We then have

$$\Phi_{\text{comp}}(S, D, \ell) \;\leq \sum_{(S', D', \ell') \in \pi_\gamma} \Phi_{\text{comp}}(S', D', \ell'),$$

and the second inequality follows. ◀

Our dynamic programming algorithm closely follows the inequalities of Lemma 4.1. Specifically, we define for each valid tuple (S, D, ℓ) the value

$$\Psi_{\text{comp}}(S, D, \ell) \;=\; \begin{cases} \Phi_{\text{comp}}(S, D, \ell) & \text{if } \ell \leq O(\sqrt{k}); \\ \max\limits_{\pi \in \Pi(S,D,\ell)} \sum\limits_{(S', D', \ell') \in \pi} \Psi_{\text{comp}}(S', D', \ell'), & \text{otherwise.} \end{cases}$$

Standard induction on ℓ using Lemma 4.1 implies the following property.

▶ **Lemma 4.2.** *For each valid tuple (S, D, ℓ) we have*

$$\Phi_{\text{comp}}(S, D, \ell) \;\leq\; \Psi_{\text{comp}}(S, D, \ell) \;\leq\; \Phi_{\text{free}}(S, D, \ell).$$

Since we know that $\Phi_{\text{free}}(\emptyset, \sigma, k) = \Phi_{\text{comp}}(\emptyset, \sigma, k)$, Lemma 4.2 implies that $\Psi_{\text{comp}}(\emptyset, \sigma, k) = \Phi_{\text{free}}(\emptyset, \sigma, k)$. Hence, it suffices to compute $\Psi_{\text{comp}}(\emptyset, \sigma, k)$ using its recursive definition. In the remainder, we bound the running time of this algorithm.

▶ **Theorem 4.3.** *In 3 dimensions,* Volume Selection *can be solved in time $n^{O(\sqrt{k})}$.*

Proof Sketch. We compute $\Psi_{\text{comp}}(\emptyset, \sigma, k)$ using its recursive definition. The base cases, where $\ell = O(\sqrt{k})$, can be solved in $n^{O(\ell)} = n^{O(\sqrt{k})}$ time using simple enumeration of all size-ℓ subsets.

Starting with $(S_1, D_1, \ell_1) = (\emptyset, \sigma, k)$, consider a sequence of valid tuples (S_1, D_1, ℓ_1), (S_2, D_2, ℓ_2), ... such that, for $i \geq 2$, the tuple (S_i, D_i, ℓ_i) appears in some valid partition of $(S_{i-1}, D_{i-1}, \ell_{i-1})$. By the properties of valid partitions, we have $\ell_i \leq 2\ell_{i-1}/3$ and $|S_{i-1}| \leq |S_i| \leq |S_{i-1}| + O(\sqrt{|S_i| + \ell_{i-1}})$. It follows that the sequence $\ell_1, \ell_2, \ldots$ decreases geometrically, from which one can deduce that $|S_i| = O(\sqrt{k})$ for all i. This means that there are $n^{O(\sqrt{k})}$ valid tuples (S, D, ℓ) that appear in the recursive calls. The same bound can be shown for the number of valid partitions in each step. ◀

We only described an algorithm that computes $\text{VolSel}(P, k)$, i.e., the maximal volume realized by any size-k subset of P. It is easy to augment the algorithm with appropriate bookkeeping to also compute an actual optimal subset.

5 Efficient Polynomial-time Approximation Scheme

In this section we design an approximation algorithm for Volume Selection.

▶ **Theorem 5.1.** *Given a point set P of size n in $\mathbb{R}^d_{>0}$, $0 \leq k \leq n$, and $0 < \varepsilon \leq 1/2$, we can compute a $(1 \pm \varepsilon)$-approximation of $\text{VolSel}(P, k)$ in time $O(n \cdot \varepsilon^{-d}(\log n + k + 2^{O(\varepsilon^{-2} \log 1/\varepsilon)^d}))$. We can also compute a set $S \subseteq P$ of size at most k such that $\mu(S)$ is a $(1 - \varepsilon)$-approximation of $\text{VolSel}(P, k)$ in the same time.*

The approach is based on the shifting technique of Hochbaum and Maass [21]. However, there are some non-standard aspects in our application. It is impossible to break the problem into independent subproblems because all the anchored boxes intersect around the origin. We instead break the input into subproblems that are *almost* independent. To achieve this, we use an exponential grid, instead of the usual regular grid with equal-size cells. Alternatively, this could be interpreted as using a regular grid in a log-log plot of the input points.

Throughout this section we need two numbers $\lambda, \tau \approx d/\varepsilon$. Specifically, we define τ as the smallest integer larger than d/ε, and λ as the smallest power of $(1-\varepsilon)^{-1/d}$ larger than d/ε. We consider a partitioning of the positive quadrant $\mathbb{R}^d_{>0}$ into ***regions*** of the form

$$R(\bar{x}) := \prod_{i=1}^{d} [\lambda^{x_i}, \lambda^{x_i+1}) \quad \text{for} \quad \bar{x} = (x_1, \ldots, x_d) \in \mathbb{Z}^d.$$

On top of this partitioning we consider a grid, where each grid cell contains $(\tau-1)^d$ regions and the grid boundaries are thick, i.e., two grid cells do not touch but have a region in between. More precisely, for any offset $\bar{\ell} = (\ell_1, \ldots, \ell_d) \in \mathbb{Z}^d$, we define the grid ***cells***

$$C_{\bar{\ell}}(\bar{y}) := \prod_{i=1}^{d} [\lambda^{\tau \cdot y_i + \ell_i + 1}, \lambda^{\tau(y_i+1)+\ell_i}) \quad \text{for} \quad \bar{y} = (y_1, \ldots, y_d) \in \mathbb{Z}^d.$$

Note that each grid cell indeed consists of $(\tau-1)^d$ regions, and the space not contained in any grid cell (i.e., the grid boundaries) consists of all regions $R(\bar{x})$ with $x_i \equiv \ell_i \pmod{\tau}$ for *some* $1 \le i \le d$.

5.1 Description of the algorithm

Our approximation algorithm works as follows.

(1) Iterate over all grid offsets $\bar{\ell} \in [\tau]^d$. This is the key step of the shifting technique [21].

(2) For any choice of the offset $\bar{\ell}$, remove all points not contained in any grid cell, i.e., remove points contained in the thick grid boundaries. Call the remaining points $P' \subseteq P$.

(3) The grid cells now induce a partitioning of P' into sets $P'_1, \ldots, P'_m$, where each P'_i is the intersection of P' with a grid cell C_i (with $C_i = C_{\bar{\ell}}(\bar{y}^{(i)})$ for some $\bar{y}^{(i)} \in \mathbb{Z}^d$). Note that these grid cell subproblems $P'_1, \ldots, P'_m$ are not independent, since any two boxes have a common intersection near the origin, no matter how different their coordinates are. However, as shown below treating $P'_1, \ldots, P'_m$ as independent subproblems still yields an approximation.

(4) We discretize by rounding down all coordinates of all points in $P'_1, \ldots, P'_m$ to powers of[3] $(1-\varepsilon)^{1/d}$. We can remove duplicate points that are rounded to the same coordinates. This yields sets $\tilde{P}_1, \ldots, \tilde{P}_m$. Note that within each grid cell in any dimension the largest and smallest coordinate differ by a factor of at most $\lambda^{\tau-1}$. Hence, there are at most $\log_{(1-\varepsilon)^{-1/d}}(\lambda^{\tau-1}) = O(\varepsilon^{-2} \log 1/\varepsilon)$ different rounded coordinates in each dimension, and thus the total number of points in each $\tilde{P}_i$ is $O(\varepsilon^{-2} \log 1/\varepsilon)^d$.

(5) Since there are only few points in each $\tilde{P}_i$, we can precompute all Volume Selection solutions on each set $\tilde{P}_i$, i.e., for any $1 \le i \le m$ and any $0 \le k' \le |\tilde{P}_i|$ we precompute VolSel$(\tilde{P}_i, k')$. We do so by exhaustively enumerating all $2^{|\tilde{P}_i|}$ subsets S of $\tilde{P}_i$, and for each one computing $\mu(S)$ by inclusion-exclusion in time $O(2^{|S|})$ (see, e.g., [32, 33]). This runs in total time $O(m \cdot 2^{O(\varepsilon^{-2} \log 1/\varepsilon)^d}) = O(n \cdot 2^{O(\varepsilon^{-2} \log 1/\varepsilon)^d})$.

[3] Here we use that λ is a power of $(1-\varepsilon)^{-1/d}$, to ensure that rounded points are contained in the same cells as their originals.

(6) It remains to split the at most k points that we want to choose over the subproblems $\tilde{P}_1, \dots, \tilde{P}_m$. As we treat these subproblems independently, we compute

$$V(\bar{\ell}) := \max_{k_1+\dots+k_m \le k} \sum_{i=1}^{m} \textsc{VolSel}(\tilde{P}_i, k_i).$$

Note that if the subproblems would be independent, then this expression would yield the exact result. We argue below that the subproblems are sufficiently close to being independent that this expression yields a $(1-\varepsilon)$-approximation of $\textsc{VolSel}(\bigcup_{i=1}^{m} \tilde{P}_i, k)$. Observe that the expression $V(\bar{\ell})$ can be computed efficiently by dynamic programming, where we compute for each i and k' the following value:

$$T[i,k'] = \max_{k_1+\dots+k_i \le k'} \sum_{i'=1}^{i} \textsc{VolSel}(\tilde{P}_{i'}, k_{i'}).$$

The following rule computes this table:

$$T[i,k'] = \max_{0 \le \kappa \le \min\{k', |\tilde{P}_i|\}} \big(\textsc{VolSel}(\tilde{P}_i, \kappa) + T[i-1, k'-\kappa]\big).$$

(7) Finally, we optimize over the offset $\bar{\ell}$ by returning the maximal $V(\bar{\ell})$.

In pseudocode, this yields the following procedure:

(1) Iterate over all offsets $\bar{\ell} = (\ell_1, \dots, \ell_d) \in [\tau]^d$:
- **(2)** $P' := P$. Delete any p from P' that is not contained in any grid cell $C_{\bar{\ell}}(\bar{y})$.
- **(3)** Partition P' into $P'_1, \dots, P'_m$, where $P'_i = P' \cap C_i$ for some grid cell C_i.
- **(4)** Round down all coordinates to powers of $(1-\varepsilon)^{1/d}$ and remove duplicates, obtaining $\tilde{P}_1, \dots, \tilde{P}_m$.
- **(5)** Compute $H[i,k'] := \textsc{VolSel}(\tilde{P}_i, k')$ for all $1 \le i \le m$, $0 \le k' \le |\tilde{P}_i|$.
- **(6)** Compute $V(\bar{\ell}) := \max_{k_1+\dots+k_m \le k} \sum_{i=1}^{m} \textsc{VolSel}(\tilde{P}_i, k_i)$ by dynamic programming.

(7) Return $\max_{\bar{\ell}} V(\bar{\ell})$.

5.2 Running Time

Step (1) yields a factor $\tau^d = O(\frac{1}{\varepsilon})^d$ in the running time. Since we can compute for each point in constant time the grid cell it is contained in, step (2) runs in time $O(n)$. For the partitioning in step (3), we use a dictionary data structure storing all $\bar{y} \in \mathbb{Z}^d$ with nonempty $P' \cap C_{\bar{\ell}}(\bar{y})$. Then we can assign any point $p \in P'$ to the other points in its cell by one lookup in the dictionary, in time $O(\log n)$. Thus, step (3) can be performed in time $O(n \log n)$. Step (4) immediately works in the same running time. For step (5) we already argued above that it can be performed in time $O\big(n 2^{O(\varepsilon^{-2} \log 1/\varepsilon)^d}\big)$. Finally, step (6) can be implemented in time $O(\sum_{i=1}^{m} |\tilde{P}_i| \cdot k) = O(nk)$. The total running time is thus $O\big(n \cdot \varepsilon^{-d}\big(\log n + k + 2^{O(\varepsilon^{-2} \log 1/\varepsilon)^d}\big)\big)$.

5.3 Correctness

Combining the following lemmas we show that the above algorithm indeed computes a $(1 \pm O(\varepsilon))$-approximation of $\textsc{VolSel}(P)$.

▶ **Lemma 5.2** (Removing grid boundaries). *Let P be a point set and let $0 \le k \le |P|$. Remove all points contained in grid boundaries with offset $\bar{\ell}$ to obtain the point set $P_{\bar{\ell}} := P \cap \bigcup_{\bar{y} \in \mathbb{Z}^d} C_{\bar{\ell}}(\bar{y})$. Then for* all *$\bar{\ell} \in \mathbb{Z}^d$ we have $\textsc{VolSel}(P_{\bar{\ell}}, k) \le \textsc{VolSel}(P, k)$, and for* some *$\bar{\ell} \in \mathbb{Z}^d$ we have $\textsc{VolSel}(P_{\bar{\ell}}, k) \ge (1-\varepsilon)\textsc{VolSel}(P, k)$.*

Proof Sketch. Since we only remove points, the first inequality is immediate. For the second inequality we use a probabilistic argument. Consider an optimal solution, i.e., a set $S \subseteq P$ of size at most k with $\mu(S) = \textsc{VolSel}(P,k)$. Let $S_{\bar{\ell}} := S \cap P_{\bar{\ell}}$. For a uniformly random offset $\bar{\ell} \in [\tau]^d$, the probability that a fixed point $p \in S$ does *not* survive, i.e., we have $p \notin S_{\bar{\ell}}$ is at most $d/\tau \leq \varepsilon$. Hence, p survives with probability at least $1-\varepsilon$.

Now for each point $q \in \mathcal{U}(S)$ identify a point $s(q) \in S$ dominating q. Since $s(q)$ survives in $S_{\bar{\ell}}$ with probability at least $1-\varepsilon$, the point q is dominated by $S_{\bar{\ell}}$ with probability at least $1-\varepsilon$. By integrating over all $q \in \mathcal{U}(S)$ we thus obtain an expected volume of

$$\mathbb{E}_{\bar{\ell}}[\mu(S_{\bar{\ell}})] = \int_{\mathcal{U}(S)} \Pr[q \text{ is dominated by } S_{\bar{\ell}}]dq \geq \int_{\mathcal{U}(S)} (1-\varepsilon)dq = (1-\varepsilon)\mu(S).$$

It follows that for some $\bar{\ell}$ we have $\mu(S_{\bar{\ell}}) \geq \mathbb{E}[\mu(S_{\bar{\ell}})] \geq (1-\varepsilon)\mu(S)$. For this $\bar{\ell}$ we have $\textsc{VolSel}(P_{\bar{\ell}},k) \geq (1-\varepsilon)\textsc{VolSel}(P,k)$. ◀

▶ **Lemma 5.3** (Rounding down coordinates). *Let P be a point set, and let $\tilde{P}$ be the same point set after rounding down all coordinates to powers of $(1-\varepsilon)^{-1/d}$. Then for any k*

$$(1-\varepsilon)\textsc{VolSel}(P,k) \leq \textsc{VolSel}(\tilde{P},k) \leq \textsc{VolSel}(P,k).$$

In the proof of the next lemma it becomes important that we have used the thick grid boundaries, with a separating region, when defining the grid cells.

▶ **Lemma 5.4** (Treating subproblems as independent I). *For any offset $\bar{\ell}$, let $S_1,\ldots,S_m$ be point sets contained in different grid cells with respect to offset $\bar{\ell}$. Then we have*

$$(1-\varepsilon)\sum_{i=1}^{m}\mu(S_i) \leq \mu\Big(\bigcup_{i=1}^{m} S_i\Big) \leq \sum_{i=1}^{m}\mu(S_i).$$

Proof Sketch. The second inequality is the union bound applied to $\mathcal{U}(S_1),\ldots,\mathcal{U}(S_m)$.

For the first inequality, we can decompose $\bigcup_{i=1}^{m}\mathcal{U}(S_i)$ to get

$$\mu\Big(\bigcup_{i=1}^{m} S_i\Big) \;=\; \textsc{vol}\left(\bigcup_{i=1}^{m}\mathcal{U}(S_i)\right) \;=\; \sum_{i=1}^{m}\left(\mu(S_i) - \textsc{vol}\Big(\mathcal{U}(S_i)\cap\bigcup_{j<i}\mathcal{U}(S_j)\Big)\right). \tag{1}$$

Now let $C_{\bar{\ell}}(\bar{y}^{(i)})$ be the grid cell containing P_i for $1 \leq i \leq m$, where $\bar{y}^{(i)} = (y_1^{(i)},\ldots,y_d^{(i)}) \in \mathbb{Z}^d$. We may assume that these cells are ordered in non-decreasing order of $y_1^{(i)}+\ldots+y_d^{(i)}$. Observe that in this ordering, for any $j<i$ we have $y_t^{(j)} < y_t^{(i)}$ for *some* $1 \leq t \leq d$. Recall that $C_{\bar{\ell}}(\bar{y}) = \prod_{t=1}^{d}[\lambda^{\tau\cdot y_t+\ell_t+1}, \lambda^{\tau(y_t+1)+\ell_t})$. It follows that each point in $\bigcup_{j<i}\mathcal{U}(S_j)$ has t-th coordinate at most $\delta_t := \lambda^{\tau\cdot y_t+\ell_t}$ for *some* $1 \leq t \leq d$. Setting $D_t := \{(z_1,\ldots,z_d) \in \mathbb{R}^d_{\geq 0} \mid z_t \leq \delta_t\}$, we thus have $\bigcup_{j<i}\mathcal{U}(S_j) \subseteq \bigcup_{t=1}^{d} D_t$, which yields

$$\textsc{vol}\Big(\mathcal{U}(S_i)\cap\bigcup_{j<i}\mathcal{U}(S_j)\Big) \leq \textsc{vol}\Big(\mathcal{U}(S_i)\cap\bigcup_{t=1}^{d} D_t\Big) \leq \sum_{t=1}^{d}\textsc{vol}\big(\mathcal{U}(S_i)\cap D_t\big). \tag{2}$$

Let A be the $(d-1)$-dimensional volume of the intersection of $\mathcal{U}(S_i)$ with the plane $x_t = 0$. Since all points in S_i have t-th coordinate at least $\lambda^{\tau\cdot y_t+\ell_t+1} = \lambda\cdot\delta_t$, we have $\mu(S_i) \geq A\cdot\lambda\cdot\delta_t$. Moreover, $\mathcal{U}(S_i)\cap D_t$ has d-dimensional volume $A\cdot\delta_t$. Together, this yields $\textsc{vol}(\mathcal{U}(S_i)\cap D_t) \leq \mu(S_i)/\lambda$. With (1) and (2), and using that $\lambda \geq d/\varepsilon$, we thus obtain

$$\mu\Big(\bigcup_{i=1}^{m} S_i\Big) \geq \sum_{i=1}^{m}\big(\mu(S_i) - d\cdot\mu(S_i)/\lambda\big) \geq (1-\varepsilon)\sum_{i=1}^{m}\mu(S_i).$$ ◀

Leveraging the above lemma to VolSel yields the following.

▶ **Lemma 5.5** (Treating subproblems as independent II). *For any offset $\bar{\ell}$, let $P_1, \dots, P_m$ be point sets contained in different grid cells, and $k \geq 0$. Then we have*

$$(1-\varepsilon) \cdot \max_{k_1+\dots+k_m \leq k} \sum_{i=1}^{m} \textsc{VolSel}(P_i, k_i) \leq \textsc{VolSel}(P, k) \leq \max_{k_1+\dots+k_m \leq k} \sum_{i=1}^{m} \textsc{VolSel}(P_i, k_i).$$

Note that the above lemmas indeed prove that the algorithm returns a $(1 \pm O(\varepsilon))$-approximation to the value $\textsc{VolSel}(P, k)$. In step (2) we delete the points containing the the grid boundaries, which yields an approximation for some choice of the offset $\bar{\ell}$ by Lemma 5.2. As we iterate over all possible choices for $\bar{\ell}$ and maximize over the resulting volume, we obtain an approximation. In step (4) we round down coordinates, which yields an approximation by Lemma 5.3. Finally, in step (6) we solve the problem $\max_{k_1+\dots+k_m \leq k} \sum_{i=1}^{m} \textsc{VolSel}(\tilde{P}_i, k_i)$, which yields an approximation to $\textsc{VolSel}(\bigcup_{i=1}^{m} \tilde{P}_i, k)$ by Lemma 5.5. All other steps do not change the point set or the considered problem.

5.4 Computing an Output Set

The above algorithm, as described, only gives an approximation for the value $\textsc{VolSel}(P, k)$. However, by tracing the dynamic programming table we can reconstruct a subset S of P of size at most k yielding a $(1 - O(\varepsilon))$-approximation of the optimal volume $\textsc{VolSel}(P, k)$.

Note that we do not compute the exact volume $\mu(S)$ of the output set S. Instead, the value $V(\bar{\ell})$ only is a $(1 + O(\varepsilon))$-approximation of $\mu(S)$. To explain this effect, recall that exactly computing $\mu(T)$ for any given set T takes time $n^{\Theta(d)}$ (under the Exponential Time Hypothesis). As our running time is $O(n^2)$ for any constant d, ε, we cannot expect to compute $\mu(S)$ exactly.

6 Conclusions

We considered the volume selection problem, where we are given n points in $\mathbb{R}^d_{>0}$ and want to select k of them that maximize the volume of the union of the spanned anchored boxes. We show: (1) Volume selection is NP-hard in dimension $d = 2$ (previously this was only known when d is part of the input). (2) In 3 dimensions, we design an $n^{O(\sqrt{k})}$ algorithm (the previously best was $\Omega\big(\binom{n}{k}\big)$). (3) We design an efficient polynomial time approximation scheme for any constant dimension d (previously only a $(1 - 1/e)$-approximation was known).

We leave open to improve our NP-hardness result to a matching lower bound under the Exponential Time Hypothesis, e.g., to show that in $d = 3$ any algorithm takes time $n^{\Omega(\sqrt{k})}$ and in any constant dimension $d \geq 4$ any algorithm takes time $n^{\Omega(k)}$. Alternatively, there could be a faster algorithm, e.g., in time $n^{O(k^{1-1/d})}$. Finally, we leave open to figure out the optimal dependence on n, k, d, ε of a $(1 - \varepsilon)$-approximation algorithm.

Moving away from the applications, one could also study volume selection on general axis-aligned boxes in $\mathbb{R}^d$, i.e., not necessarily anchored boxes. This problem General Volume Selection is an optimization variant of Klee's measure problem and thus might be theoretically motivated. However, General Volume Selection is probably much harder than the restriction to anchored boxes, by analogies to the problem of computing an independent set of boxes, which is not known to have a PTAS [1]. In particular, General Volume Selection is NP-hard already in 2 dimensions, which follows from NP-hardness of computing an independent set in a family of congruent squares in the plane [18, 22].

Acknowledgements. This work was initiated during the Fixed-Parameter Computational Geometry Workshop at the Lorentz Center, 2016. We are grateful to the other participants of the workshop and the Lorentz Center for their support. We are especially grateful to Günter Rote for several discussions and related work.

References

1 A. Adamaszek and A. Wiese. Approximation schemes for maximum weight independent set of rectangles. In *Proc. of the 54th IEEE Symp. on Found. of Comp. Science (FOCS)*, pages 400–409. IEEE, 2013.

2 A. Auger, J. Bader, D. Brockhoff, and E. Zitzler. Investigating and exploiting the bias of the weighted hypervolume to articulate user preferences. In *Proc. of the 11th Conf. on Genetic and Evolutionary Computation*, pages 563–570. ACM, 2009.

3 A. Auger, J. Bader, D. Brockhoff, and E. Zitzler. Hypervolume-based multiobjective optimization: Theoretical foundations and practical implications. *Theoretical Comp. Science*, 425:75–103, 2012.

4 J. Bader. *Hypervolume-based search for multiobjective optimization: theory and methods.* PhD thesis, ETH Zurich, Zurich, Switzerland, 1993.

5 F. Barahona. On the computational complexity of Ising spin glass models. *J. of Physics A: Mathematical and General*, 15(10):3241, 1982.

6 N. Beume, C. M. Fonseca, M. López-Ibáñez, L. Paquete, and J. Vahrenhold. On the complexity of computing the hypervolume indicator. *IEEE Trans. on Evolutionary Computation*, 13(5):1075–1082, 2009.

7 N. Beume, B. Naujoks, and M. Emmerich. SMS-EMOA: Multiobjective selection based on dominated hypervolume. *European J. of Operational Research*, 181(3):1653–1669, 2007.

8 K. Bringmann. Bringing order to special cases of Klee's measure problem. In *Int. Symp. on Mathematical Foundations of Comp. Science*, pages 207–218. Springer, 2013.

9 K. Bringmann and T. Friedrich. Approximating the volume of unions and intersections of high-dimensional geometric objects. *Computational Geometry*, 43(6):601–610, 2010.

10 K. Bringmann and T. Friedrich. An efficient algorithm for computing hypervolume contributions. *Evolutionary Computation*, 18(3):383–402, 2010.

11 K. Bringmann and T. Friedrich. Approximating the least hypervolume contributor: NP-hard in general, but fast in practice. *Theoretical Comp. Science*, 425:104–116, 2012.

12 K. Bringmann, T. Friedrich, and P. Klitzke. Generic postprocessing via subset selection for hypervolume and epsilon-indicator. In *Int. Conf. on Parallel Problem Solving from Nature*, pages 518–527. Springer, 2014.

13 K. Bringmann, T. Friedrich, and P. Klitzke. Two-dimensional subset selection for hypervolume and epsilon-indicator. In *Proc. of the 2014 Conf. on Genetic and Evolutionary Comput.*, pages 589–596. ACM, 2014.

14 T. M. Chan. A (slightly) faster algorithm for Klee's measure problem. *Computational Geometry*, 43(3):243–250, 2010.

15 T. M. Chan. Klee's measure problem made easy. In *Proc. of the 54th IEEE Symp. on Found. of Comp. Science (FOCS)*, pages 410–419. IEEE, 2013.

16 J. Chen, X. Huang, I. A. Kanj, and G. Xia. Linear FPT reductions and computational lower bounds. In *Proc. of the 36th ACM Symp. on Theory of Computing (STOC)*, pages 212–221. ACM, 2004.

17 M. Emmerich, A. H. Deutz, and I. Yevseyeva. A Bayesian approach to portfolio selection in multicriteria group decision making. *Procedia Comp. Science*, 64:993–1000, 2015.

18 R. J. Fowler, M. S. Paterson, and S. L. Tanimoto. Optimal packing and covering in the plane are NP-complete. *Information Processing Lett.*, 12(3):133–137, 1981.

19 M. R. Garey and D. S. Johnson. The rectilinear Steiner tree problem in NP complete. *SIAM J. of Applied Math.*, 32:826–834, 1977.

20 A. P. Guerreiro, C. M. Fonseca, and L. Paquete. Greedy hypervolume subset selection in low dimensions. *Evolutionary Computation*, 24(3):521–544, 2016.

21 D. S. Hochbaum and W. Maass. Approximation schemes for covering and packing problems in image processing and VLSI. *J. ACM*, 32(1):130–136, 1985.

22 H. Imai and T. Asano. Finding the connected components and a maximum clique of an intersection graph of rectangles in the plane. *J. of Algorithms*, 4(4):310–323, 1983.

23 J. D. Knowles, D. W. Corne, and M. Fleischer. Bounded archiving using the Lebesgue measure. In *Proc. of the 2003 Congress on Evolutionary Computation (CEC)*, volume 4, pages 2490–2497. IEEE, 2003.

24 T. Kuhn, C. M. Fonseca, L. Paquete, S. Ruzika, M. M. Duarte, and J. R. Figueira. Hypervolume subset selection in two dimensions: Formulations and algorithms. *Evolutionary Computation*, 2015.

25 G. L. Miller. Finding small simple cycle separators for 2-connected planar graphs. *J. Comput. Syst. Sci.*, 32(3):265–279, 1986.

26 J. S. B. Mitchell and M. Sharir. New results on shortest paths in three dimensions. In *Proc. of the 20th ACM Symp. on Computational Geometry*, pages 124–133, 2004.

27 G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions – I. *Mathematical Programming*, 14(1):265–294, 1978.

28 G. Rote, K. Buchin, K. Bringmann, S. Cabello, and M. Emmerich. Selecting k points that maximize the convex hull volume (extended abstract). In *JCDCG3 2016; The 19th Japan Conf. on Discrete and Computational Geometry, Graphs, and Games*, pages 58–60, 9 2016. `http://www.jcdcgg.u-tokai.ac.jp/JCDCG3_abstracts.pdf`.

29 J. A. Storer. On minimal-node-cost planar embeddings. *Networks*, 14(2):181–212, 1984.

30 R. Tamassia and I. G. Tollis. Planar grid embedding in linear time. *IEEE Trans. on Circuits and Systems*, 36(9):1230–1234, 1989.

31 T. Ulrich and L. Thiele. Bounding the effectiveness of hypervolume-based (μ+λ)-archiving algorithms. In *Learning and Intelligent Optimization*, pages 235–249. Springer, 2012.

32 L. While, P. Hingston, L. Barone, and S. Huband. A faster algorithm for calculating hypervolume. *IEEE Trans. on Evolutionary Computation*, 10(1):29–38, 2006.

33 J. Wu and S. Azarm. Metrics for quality assessment of a multiobjective design optimization solution set. *J. of Mechanical Design*, 123(1):18–25, 2001.

34 E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. Da Fonseca. Performance assessment of multiobjective optimizers: an analysis and review. *IEEE Trans. on Evolutionary Computation*, 7(2):117–132, 2003.

Declutter and Resample: Towards Parameter Free Denoising*†

Mickaël Buchet[1], Tamal K. Dey[2], Jiayuan Wang[3], and Yusu Wang[4]

1 **Advanced Institute for Materials Research, Tohoku University, Sendai, Japan**
mickael.buchet@m4x.org
2 **Department of Computer Science and Engineering, The Ohio State University, Columbus, OH, USA**
tamaldey@cse.ohio-state.edu
3 **Department of Computer Science and Engineering, The Ohio State University, Columbus, OH, USA**
wang.6195@buckeyemail.osu.edu
4 **Department of Computer Science and Engineering, The Ohio State University, Columbus, OH, USA**
yusu@cse.ohio-state.edu

Abstract

In many data analysis applications the following scenario is commonplace: we are given a point set that is supposed to sample a hidden ground truth K in a metric space, but it got corrupted with noise so that some of the data points lie far away from K creating outliers also termed as *ambient noise*. One of the main goals of denoising algorithms is to eliminate such noise so that the curated data lie within a bounded Hausdorff distance of K. Popular denoising approaches such as deconvolution and thresholding often require the user to set several parameters and/or to choose an appropriate noise model while guaranteeing only asymptotic convergence. Our goal is to lighten this burden as much as possible while ensuring theoretical guarantees in all cases.

Specifically, first, we propose a simple denoising algorithm that requires only a single parameter but provides a theoretical guarantee on the quality of the output on general input points. We argue that this single parameter cannot be avoided. We next present a simple algorithm that avoids even this parameter by paying for it with a slight strengthening of the sampling condition on the input points which is not unrealistic. We also provide some preliminary empirical evidence that our algorithms are effective in practice.

1998 ACM Subject Classification I.3.5 Computational Geometry and Object Modeling

Keywords and phrases denoising, parameter free, k-distance,compact sets

Digital Object Identifier 10.4230/LIPIcs.SoCG.2017.23

1 Introduction

Real life data are almost always corrupted by noise. Of course, when we talk about noise, we implicitly assume that the data sample a hidden space called the *ground truth* with respect to which we measure the extent and type of noise. Some data can lie far away from the ground truth leading to ambient noise. Clearly, the data density needs to be higher near the ground truth if signal has to prevail over noise. Therefore, a worthwhile goal of a denoising

* A full version of the paper is available at https://arxiv.org/abs/1511.05479.
† This work is in part supported by National Science Foundation via grants CCF-1618247, CCF-1526513, CCF-1318595 and IIS-1550757.

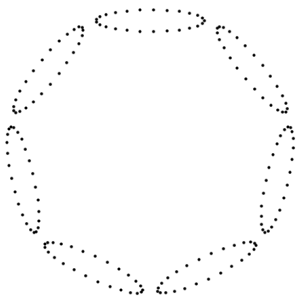


Figure 1 Scale ambiguity.

algorithm is to curate the data, eliminating the ambient noise while retaining most of the subset that lies within a bounded distance from the ground truth.

In this paper we are interested removing "outlier"-type of noise from input data. Numerous algorithms have been developed for this problem in many different application fields; see e.g [16, 21]. There are two popular families of denoising/outlier detection approaches: Deconvolution and Thresholding. Deconvolution methods rely on the knowledge of a generative noise model for the data. For example, the algorithm may assume that the input data has been sampled according to a probability measure obtained by convolving a distribution such as a Gaussian [18] with a measure whose support is the ground truth. Alternatively, it may assume that the data is generated according to a probability measure with a small Wasserstein distance to a measure supported by the ground truth [7]. The denoising algorithm attempts to cancel the noise by deconvolving the data with the assumed model.

A deconvolution algorithm requires the knowledge of the generative model and at least a bound on the parameter(s) involved, such as the standard deviation of the Gaussian convolution or the Wasserstein distance. Therefore, it requires at least one parameter as well as the knowledge of the noise type. The results obtained in this setting are often asymptotic, that is, theoretical guarantees hold in the limit when the number of points tends to infinity.

The method of thresholding relies on a density estimation procedure [20] by which it estimates the density of the input locally. The data is cleaned, either by removing points where density is lower than a threshold [14], or moving them from such areas toward higher densities using gradient-like methods such as mean-shift [11, 19]. It has been recently used for uncovering geometric information such as one dimensional features [15]. In [5], the *distance to a measure* [10] that can also be seen as a density estimator [2] has been exploited for thresholding. Other than selecting a threshold, these methods require the choice of a density estimator. This estimation requires at least one additional parameter, either to define a kernel, or a mass to define the distance to a measure. In the case of a gradient based movement of the points, the nature of the movement also has to be defined by fixing the length of a step and by determining the terminating condition of the algorithm.

New work. In above classical methods, the user is burdened with making several choices such as fixing an appropriate noise model, selecting a threshold and/or other parameters. Our main goal is to lighten this burden as much as possible. First, we show that denoising with a single parameter is possible and this parameter is in some sense unavoidable unless a stronger sampling condition on the input points is assumed. This leads to our main algorithm that is completely free of any parameter when the input satisfies a stronger sampling condition which is not unrealistic.

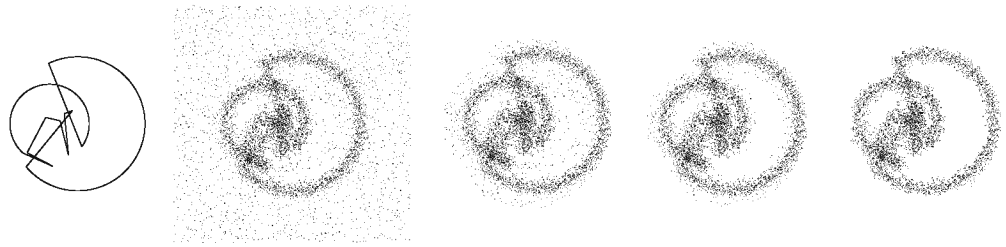


Figure 2 From left to right: the ground truth, the noisy input samples ($\sim$ 7000 points around the ground truth and 2000 ambient noise points), two intermediate steps of our iterative parameter-free denoising algorithm and the final output.

Our first algorithm *Declutter algorithm* uses a single parameter (presented in Section 3) and assumes a very general sampling condition which is not stricter than those for the classical noise models mentioned previously because it holds with high probability for those models as well. Additionally, our sampling condition also allows ambient noise and locally adaptive samplings. Interestingly, we note that our Declutter algorithm is in fact a variant of the approach proposed in [8] to construct the so-called *ε-density net*. Indeed, as we point out in Appendix D of the full version [6], the procedure of [8] can also be directly used for denoising purpose and one can obtain an analog of Theorems 9 and 13 in this paper for the resulting density net.

Use of a parameter in the denoising process is unavoidable in some sense, unless there are other assumptions about the hidden space. This is illustrated by the example in Figure 1. Does the sample here represent a set of small loops or one big circle? The answer depends on the scale at which we examine the data; see the full version [6] for the results of applying our denoising algorithms on this data set. The choice of a parameter may represent this choice of the scale [3, 13]. To remove this parameter, one needs other conditions for either the hidden space itself or for the sample, say by assuming that the data has some uniformity. Aiming to keep the sampling restrictions as minimal as possible, we show that it is sufficient to assume the homogeneity in data *only on or close to* the ground truth for our second algorithm which requires no input parameter.

Specifically, the parameter-free algorithm presented in Section 4 relies on an iteration that intertwines our decluttering algorithm with a novel resampling procedure. Assuming that the sample is sufficiently dense and somewhat uniform near the ground truth at scales beyond a particular scale s, our algorithm selects a subset of the input point set that is close to the ground truth without requiring any input from the user. The output maintains the quality at scale s even though the algorithm has no explicit knowledge of this parameter. See Figure 2 for an example.

All missing details from this extended abstract can be found in the full version [6]. In addition, in Appendix C of the full version [6], we show how the denoised data set can be used for homology inference. In Appendix E of the full version [6] , we provide various preliminary experimental results of our denoising algorithms.

▶ Remark. Very recently, Jiang and Kpotufe proposed a consistent algorithm for estimating the so-called modal-sets with also only one parameter [17]. The problem setup and goals are very different: In their work, they assume that input points are sampled from a density field that is locally maximal and constant on a compact domain. The goal is to show that as the number of samples n tends to infinity, such domains (referred to as modal-sets in their paper) can be recovered, and the recovered set converges to the true modal-sets under the Hausdorff distance. We also note that our Declutter algorithm allows adaptive sampling as well.

2 Preliminaries

We assume that the input is a set of points P sampled around a hidden compact set K, the ground truth, in a metric space $(\mathbb{X}, d_{\mathbb{X}})$. For simplicity, in what follows the reader can assume $\mathbb{X} = \mathbb{R}^d$ with $P, K \subset \mathbb{X} = \mathbb{R}^d$, and the metric $d_{\mathbb{X}}$ of $\mathbb{X}$ is simply the Euclidean distance. Our goal is to process P into another point set Q guaranteed to be Hausdorff close to K and hence to be a better sample of the hidden space K for further applications. By Hausdorff close, we mean that the (standard) *Hausdorff distance* $\delta_H(Q, K)$ between Q and K, defined as the infimum of δ such that $\forall p \in Q, d_{\mathbb{X}}(p, K) \leq \delta$ and $\forall x \in K, d_{\mathbb{X}}(x, Q) \leq \delta$, is bounded. Note that ambient noise/outliers can incur a very large Hausdorff distance.

The quality of the output point set Q obviously depends on the "quality" of input points P, which we formalize via the language of *sampling conditions*. We wish to produce good quality output for inputs satisfying much weaker sampling conditions than a bounded Hausdorff distance. Our sampling condition is based on the sampling condition introduced and studied in [4, 5]; see Chapter 6 of [4] for discussions on the relation of their sampling condition with some of the common noise models such as Gaussian. Below, we first introduce a basic sampling condition deduced from the one in [4, 5], and then introduce its extensions incorporating adaptivity and uniformity.

Basic sampling condition. Our sampling condition is built upon the concept of k-distance, which is a specific instance of a broader concept called *distance to a measure* introduced in [10]. The k-distance $d_{P,k}(x)$ is simply the root mean of square distance from x to its k-nearest neighbors in P. The averaging makes it robust to outliers. One can view $d_{P,k}(x)$ as capturing the inverse of the density of points in P around x [2]. As we show in Appendix D [6], this specific form of k-distance is not essential – Indeed, several of its variants can replace its role in the definition of sampling conditions below, and our Declutter algorithm will achieve similar denoising guarantees.

▶ **Definition 1** ([10]). Given a point $x \in \mathbb{X}$, let $p_i(x) \in P$ denote the i-th nearest neighbor of x in P. The *k-distance* to a point set $P \subseteq \mathbb{X}$ is $d_{P,k}(x) = \sqrt{\frac{1}{k}\sum_{i=1}^{k} d_{\mathbb{X}}(x, p_i(x))^2}$.

▶ **Claim 2** ([10]). *$d_{P,k}(\cdot)$ is 1-Lipschitz, i.e. $|d_{P,k}(x) - d_{P,k}(y)| \leq d_{\mathbb{X}}(x, y)$ for $\forall (x, y) \in \mathbb{X} \times \mathbb{X}$.*

All our sampling conditions are dependent on the choice of k in the k-distance, which we reflect by writing ϵ_k instead of ϵ in the sampling conditions below. The following definition is related to the sampling condition proposed in [5].

▶ **Definition 3.** Given a compact set $K \subseteq \mathbb{X}$ and a parameter k, a point set P is an *ϵ_k-noisy sample* of K if

1. $\forall x \in K,\ d_{P,k}(x) \leq \epsilon_k$
2. $\forall x \in \mathbb{X},\ d_{\mathbb{X}}(x, K) \leq d_{P,k}(x) + \epsilon_k$

Condition 1 in Definition 3 means that the density of P on the compact set K is bounded from below, that is, K is well-sampled by P. Note, we only require P to be a dense enough sample of K – there is no uniformity requirement in the sampling here.

Condition 2 implies that a point with low k-distance, i.e. lying in high density region, has to be close to K. Intuitively, P can contain outliers which can form small clusters but their density can not be significant compared to the density of points near the compact set K.

Note that the choice of ϵ_k always exists for a bounded point set P, no matter what value of k is – For example, one can set ϵ_k to be the diameter of point set P. However, the smallest

possible choice of ϵ_k to make P an ϵ_k-noisy sample of K depends on the value of k. We thus use ϵ_k in the sampling condition to reflect this dependency.

In Section 4, we develop a parameter-free denoising algorithm. As Figure 1 illustrates, it is necessary to have a mechanism to remove potential ambiguity about the ground truth. We do so by using a stronger sampling condition to enforce some degree of uniformity:

▶ **Definition 4.** Given a compact set $K \subseteq \mathbb{X}$ and a parameter k, a point set P is a uniform (ϵ_k, c)-noisy sample of K if P is an ϵ_k-noisy sample of K (i.e, conditions of Def. 3 hold) and
3. $\forall p \in P,\ d_{P,k}(p) \geq \frac{\epsilon_k}{c}$.

It is important to note that the lower bound in Condition 3 enforces that the sampling needs to be homogeneous – i.e, $d_{P,k}(x)$ is bounded both from above and from below by some constant factor of ϵ_k – *only for points on and around* the ground truth K. This is because condition 1 in Def. 3 is only for points from K, and condition 1 together with the 1-Lipschitz property of $d_{P,k}$ (Claim 2) leads to an upper bound of $O(\epsilon_k)$ for $d_{P,k}(y)$ only for points y within $O(\epsilon_k)$ distance to K. There is no such upper bound on $d_{P,k}$ for noisy points far away from K and *thus no homogeneity/uniformity requirements for them.*

Adaptive sampling conditions. The sampling conditions given above are global, meaning that they do not respect the "feature" of the ground truth. We now introduce an adaptive version of the sampling conditions with respect to a feature size function.

▶ **Definition 5.** Given a compact set $K \subseteq \mathbb{X}$, a feature size function $f : K \to \mathbb{R}^+ \cup \{0\}$ is a 1-Lipschitz non-negative real function on K.

Several feature sizes exist in the literature of manifold reconstruction and topology inference, including the *local feature size* [1], *local weak feature size*, *μ-local weak feature size* [9] or *lean set feature size* [12]. All of these functions describe how complicated a compact set is locally, and therefore indicate how dense a sample should be locally so that information can be inferred faithfully. Any of these functions can be used as a feature size function to define the adaptive sampling below. Let $\bar{p}$ denote any one of the nearest points of p in K. Observe that, in general, a point p can have multiple such nearest points.

▶ **Definition 6.** Given a compact set $K \subseteq \mathbb{X}$, a feature size function f of K, and a parameter k, a point set P is a *uniform (ϵ_k, c)-adaptive noisy sample of K* if
1. $\forall x \in K,\ d_{P,k}(x) \leq \epsilon_k f(x)$,
2. $\forall y \in \mathbb{X},\ d_{\mathbb{X}}(y, K) \leq d_{P,k}(y) + \epsilon_k f(\bar{y})$,
3. $\forall p \in P,\ d_{P,k}(p) \geq \frac{\epsilon_k}{c} f(\bar{p})$.
We say that P is *an ϵ_k-adaptive noisy sample of K* if only conditions 1 and 2 above hold.

We require that the feature size is *positive everywhere* as otherwise, the sampling condition may require infinite samples in some cases. We also note that the requirement of the feature size function being 1-Lipschitz is only needed to provide the theoretical guarantee for our second parameter-free algorithm.

3 Decluttering

We now present a simple yet effective denoising algorithm which takes as input a set of points P and a parameter k, and outputs a set of points $Q \subseteq P$ with the following guarantees: If P is an ϵ_k-noisy sample of a hidden compact set $K \subseteq \mathbb{X}$, then the output Q lies close to K in the *Hausdorff distance* (i.e, within a small tubular neighborhood of K and outliers are all

Algorithm 1: Declutter(P,k).

Data: Point set P, parameter k
Result: Denoised point set Q

```
begin
    sort P such that d_{P,k}(p_1) ≤ ··· ≤ d_{P,k}(p_{|P|}).
    Q_0 ⟵ ∅
    for i ⟵ 1 to |P| do
        if Q_{i−1} ∩ B(p_i, 2d_{P,k}(p_i)) = ∅ then
            Q_i = Q_{i−1} ∪ {p_i}
        end
        else  Q_i = Q_{i−1}
    end
    Q ⟵ Q_n
end
```

eliminated). The theoretical guarantee holds for both the non-adaptive and the adaptive cases, as stated in Theorems 9 and 13.

As the k-distance behaves like the inverse of density, points with a low k-distance are expected to lie close to the ground truth K. A possible approach is to fix a threshold α and only keep the points with a k-distance less than α. This thresholding solution requires an additional parameter α. Furthermore, very importantly, such a thresholding approach does not easily work for adaptive samples, where the density in an area with large feature size can be lower than the density of noise close to an area with small feature size.

Our algorithm Declutter(P,k), presented in **Algorithm 1**, works around these problems by considering the points in the order of increasing values of their k-distances and using a pruning step: Given a point p_i, if there exists a point q deemed better in its vicinity, i.e., q has smaller k-distance and has been previously selected ($q \in Q_{i-1}$), then p_i is not necessary to describe the ground truth and we throw it away. Conversely, if no point close to p_i has already been selected, then p_i is meaningful and we keep it. The notion of "closeness" or "vicinity" is defined using the k-distance, so k is the only parameter. In particular, the "vicinity" of a point p_i is defined as the metric ball $B(p_i, 2d_{P,k}(p_i))$; observe that this radius is different for different points, and the radius of the ball is larger for outliers. Intuitively, the radius $2d_{P,k}(p_i)$ of the "vicinity" around p_i can be viewed as the length we have to go over to reach the hidden domain with certainty. So, bad points have a larger "vicinity". We remark that this process is related to the construction of the "density net" introduced in [8], which we discuss more in Appendix D [6].

See Figure 3 on the right for an artificial example, where the black points are input points, and red crosses are in the current output Q_{i-1}. Now, at the ith iteration, suppose we are processing the point p_i (the green point). Since within the vicinity of p_i there is already a good point p, we consider p_i to be not useful, and remove it. Intuitively, for an outlier p_i, it has a large k-distance and hence a large vicinity. As we show later, our ϵ_k-noisy sampling condition ensures that this vicinity of p_i reaches the hidden compact set which the input points presumably sample. Since points around the hidden compact set should have higher density, there should be a good point already chosen in Q_{i-1}. Finally, it is also important to note that, contrary to many common sparsification procedures, our Declutter algorithm removes a noisy point because it has a good point within its vicinity, and *not because* it is within the vicinity of a good point. For example, in Figure 3, the red points such as p have small vicinity, and p_i is not in the vicinity of any of the red point.

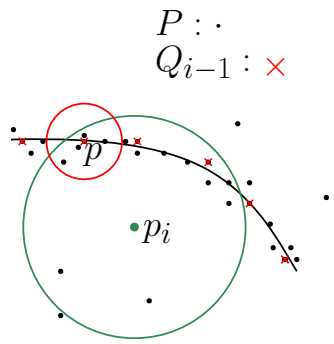


Figure 3 Declutter.

In what follows, we will make this intuition more concrete. We first consider the simpler non-adaptive case where P is an ϵ_k-noisy sample of K. We establish that Q and the ground truth K are Hausdorff close in the following two lemmas. The first lemma says that the ground truth K is well-sampled (w.r.t. ϵ_k) by the output Q of Declutter.

▶ **Lemma 7.** *Let $Q \subseteq P$ be the output of Declutter(P,k) where P is an ϵ_k-noisy sample of a compact set $K \subseteq \mathbb{X}$. Then, for any $x \in K$, there exists $q \in Q$ such that $d_{\mathbb{X}}(x, q) \leq 5\epsilon_k$.*

Proof. Let $x \in K$. By Condition 1 of Def. 3, we have $d_{P,k}(x) \leq \epsilon_k$. This means that the nearest neighbor p_i of x in P satisfies $d_{\mathbb{X}}(p_i, x) \leq d_{P,k}(x) \leq \epsilon_k$. If $p_i \in Q$, then the claim holds by setting $q = p_i$. If $p_i \notin Q$, there must exist $j < i$ with $p_j \in Q_{i-1}$ such that $d_{\mathbb{X}}(p_i, p_j) \leq 2d_{P,k}(p_i)$. In other words, p_i was removed by our algorithm because $p_j \in Q_{i-1} \cap B(p_i, 2d_{P,k}(p_i))$. Combining triangle inequality with the 1-Lipschitz property of $d_{P,k}$ (Claim 2), we then have

$$d_{\mathbb{X}}(x, p_j) \leq d_{\mathbb{X}}(x, p_i) + d_{\mathbb{X}}(p_i, p_j) \leq d_{\mathbb{X}}(x, p_i) + 2d_{P,k}(p_i) \leq 2d_{P,k}(x) + 3d_{\mathbb{X}}(p_i, x) \leq 5\epsilon_k,$$

which proves the claim. ◀

The next lemma implies that all outliers are removed by our denoising algorithm.

▶ **Lemma 8.** *Let $Q \subseteq P$ be the output of Declutter(P,k) where P is an ϵ_k-noisy sample of a compact set $K \subseteq \mathbb{X}$. Then, for any $q \in Q$, there exists $x \in K$ such that $d_{\mathbb{X}}(q, x) \leq 7\epsilon_k$.*

Proof. Consider any $p_i \in P$ and let $\bar{p}_i$ be one of its nearest points in K. It is sufficient to show that if $d_{\mathbb{X}}(p_i, \bar{p}_i) > 7\epsilon_k$, then $p_i \notin Q$.

Indeed, by Condition 2 of Def. 3, $d_{P,k}(p_i) \geq d_{\mathbb{X}}(p_i, \bar{p}_i) - \epsilon_k > 6\epsilon_k$. By Lemma 7, there exists $q \in Q$ such that $d_{\mathbb{X}}(\bar{p}_i, q) \leq 5\epsilon_k$. Thus,

$$d_{P,k}(q) \leq d_{P,k}(\bar{p}_i) + d_{\mathbb{X}}(\bar{p}_i, q) \leq 6\epsilon_k.$$

Therefore, $d_{P,k}(p_i) > 6\epsilon_k \geq d_{P,k}(q)$ implying that $q \in Q_{i-1}$. Combining triangle inequality and Condition 2 of Def. 3, we have

$$d_{\mathbb{X}}(p_i, q) \leq d_{\mathbb{X}}(p_i, \bar{p}_i) + d_{\mathbb{X}}(\bar{p}_i, q) \leq d_{P,k}(p_i) + \epsilon_k + 5\epsilon_k < 2d_{P,k}(p_i).$$

Therefore, $q \in Q_{i-1} \cap B(p_i, 2d_{P,k}(p_i))$, meaning that $p_i \notin Q$. ◀

▶ **Theorem 9.** *Given a point set P which is an ϵ_k-noisy sample of a compact set $K \subseteq \mathbb{X}$, Algorithm Declutter returns a set $Q \subseteq P$ such that $\delta_H(K, Q) \leq 7\epsilon_k$.*

Interestingly, if the input point set is uniform then the denoised set is also uniform, a fact that turns out to be useful for our parameter-free algorithm later.

▶ **Proposition 10.** *If P is a uniform (ϵ_k, c)-noisy sample of a compact set $K \subseteq \mathbb{X}$, then the distance between any pair of points of Q is at least $2\frac{\epsilon_k}{c}$.*

Proof. Let p and q be in Q with $p \neq q$ and, assume without loss of generality that $d_{P,k}(p) \leq d_{P,k}(q)$. Then, $p \notin B(q, 2d_{P,k}(q))$ and $d_{P,k}(q) \geq \frac{\epsilon_k}{c}$. Therefore, $d_{\mathbb{X}}(p,q) \geq 2\frac{\epsilon_k}{c}$. ◀

Adaptive case

Assume the input is an adaptive sample $P \subseteq \mathbb{X}$ with respect to a feature size function f. The denoised point set Q may also be adaptive. We hence need an adaptive version of the Hausdorff distance denoted $\delta_H^f(Q,K)$ and defined as the infimum of δ such that (i) $\forall p \in Q$, $d_{\mathbb{X}}(p,K) \leq \delta f(\bar{p})$, and (ii) $\forall x \in K, d_{\mathbb{X}}(x,Q) \leq \delta f(x)$, where $\bar{p}$ is a nearest point of p in K. Similar to the non-adaptive case, we establish that P and output Q are Hausdorff close via Lemmas 11 and 12 whose proofs are same as those for Lemmas 7 and 8 respectively, but using an adaptive distance w.r.t. the feature size function. Note that the algorithm does not need to know what the feature size function f is, hence only one parameter (k) remains.

▶ **Lemma 11.** *Let $Q \subseteq P$ be the output of Declutter(P,k) where P is an ϵ_k-adaptive noisy sample of a compact set $K \subseteq \mathbb{X}$. Then, $\forall x \in K, \exists q \in Q,\ d_{\mathbb{X}}(x,q) \leq 5\epsilon_k f(x)$.*

▶ **Lemma 12.** *Let $Q \subseteq P$ be the output of Declutter(P,k) where P is an ϵ_k-adaptive noisy sample of a compact set $K \subseteq \mathbb{X}$. Then, for $\forall q \in Q,\ d_{\mathbb{X}}(q,\bar{q}) \leq 7\epsilon_k f(\bar{q})$.*

▶ **Theorem 13.** *Given an ϵ_k-adaptive noisy sample P of a compact set $K \subseteq \mathbb{X}$ with feature size f, Algorithm Declutter returns a sample $Q \subseteq P$ of K where $\delta_H^f(Q,K) \leq 7\epsilon_k$.*

Again, if the input set is uniform, the output remains uniform as stated below.

▶ **Proposition 14.** *Given an input point set P which is an uniform (ϵ_k, c)-adaptive noisy sample of a compact set $K \subseteq \mathbb{X}$, the output $Q \subseteq P$ of Declutter satisfies*

$$\forall (q_i, q_j) \in Q,\ i \neq j \implies d_{\mathbb{X}}(q_i, q_j) \geq 2\frac{\epsilon_k}{c} f(\bar{q}_i)\,.$$

Proof. Let q_i and q_j be two points of Q with $i < j$. Then q_i is not in the ball of center q_j and radius $2d_{P,k}(q_j)$. Hence $d_{\mathbb{X}}(q_i,q_j) \geq 2d_{P,k}(q_j) \geq 2\frac{\epsilon_k}{c} f(\bar{q}_j)$. Since $i < j$, it also follows that $d_{\mathbb{X}}(q_i,q_j) \geq 2d_{P,k}(q_j) \geq 2d_{P,k}(q_i) \geq 2\frac{\epsilon_k}{c} f(\bar{q}_i)$. ◀

The algorithm Declutter removes outliers from the input point set P. As a result, we obtain a point set which lies close to the ground truth with respect to the Hausdorff distance. Such point sets can be used for inference about the ground truth with further processing. For example, in topological data analysis, our result can be used to perform topology inference from noisy input points in the non-adaptive setting; see Appendix C [6] for more details.

An example of the output of algorithm Declutter is given in Figure 4(a)–(d). More examples (including for adaptive inputs) can be found in the full version [6].

Extensions

It turns out that there are many choices that can be used for the k-distance $d_{P,k}(x)$ instead of the one introduced in Definition 1. Indeed, the goal of k-distance intuitively is to provide a more robust distance estimate – Specifically, assume P is a noisy sample of a hidden domain

(a) (b) (c) (d) (e)

Figure 4 (a)–(d) show results of the Algorithm Declutter: (a) the ground truth, (b) the noisy input with 15K points with 1000 ambient noisy points, (c) the output of Algorithm Declutter when $k = 9$, (d) the output of Algorithm Declutter when $k = 30$. In (e), we show the output of Algorithm ParfreeDeclutter. As shown in Appendix E [6], algorithm ParfreeDeclutter can remove ambient noise for much sparser input samples with more noisy points.

$K \subset \mathbb{X}$. With the presence of noisy points far away from K, the distance $d_{\mathbb{X}}(x, P)$ no longer serves as a good approximation of $d_{\mathbb{X}}(x, K)$, the distance from x to the hidden domain K. We thus need a more robust distance estimate. The k-distance $d_{P,k}(x)$ introduced in Definition 1 is one such choice, and there are many other valid choices. As we show in Appendix D [6], we only need the choice of $d_{P,k}(x)$ to be 1-Lipschitz, and is less sensitive than $d_{\mathbb{X}}(x, P)$ (that is, $d_{\mathbb{X}}(x, P) \leq d_{P,k}(x)$). We can then define the sampling condition (as in Definitions 3 and 4) using a different choice of $d_{P,k}(x)$, and Theorems 9 and 13 still hold. For example, we could replace k-distance by $d_{P,k}(x) = \frac{1}{k}\sum_{i=1}^{k} d(x, p_i(x))$ where $p_i(x)$ is the ith nearest neighbor of x in P; that is, $d_{P,k}(x)$ is the average distance to the k nearest neighbors of x in P. Alternatively, we can replace k-distance by $d_{P,k}(x) = d(x, p_k(x))$, the distance from x to its k-th nearest neighbor in P (which was used in [8] to construct the ε-density net). Declutter algorithm works for all these choices with the same denoising guarantees.

One can in fact further relax the conditions on $d_{P,k}(x)$ or even on the input metric space $(\mathbb{X}, d_{\mathbb{X}})$ such that the triangle inequality for $d_{\mathbb{X}}$ only approximately holds. The corresponding guarantees of our Declutter algorithm are provided in Appendix D of the full version [6].

4 Parameter-free decluttering

The algorithm Declutter is not entirely satisfactory. First, we need to fix the parameter k a priori. Second, while providing a Hausdorff distance guarantee, this procedure also "sparsifies" input points. Specifically, the empty-ball test also induces some degree of sparsification, as for any point q kept in Q, the ball $B(q, 2d_{P,k}(q))$ does not contain any other output points in Q. While this sparsification property is desirable for some applications, it removes too many points in some cases – See Figure 4 for an example, where the output density is dominated by ϵ_k and does not preserve the dense sampling provided by the input around the hidden compact set K. In particular, for $k = 9$, it does not completely remove ambient noise, while, for $k = 30$, the output is too sparse.

In this section, we address both of the above concerns by a novel iterative re-sampling procedure as described in Algorithm ParfreeDeclutter(P). Roughly speaking, we start with $k = |P|$ and gradually decrease it by halving each time. At iteration i, let P_i denote the set of points so far kept by the algorithm; i is initialized to be $\lfloor \log_2(|P|) \rfloor$ and is gradually decreased. We perform the denoising algorithm Declutter$(P_i, k = 2^i)$ given in the previous section to first denoise P_i and obtain a denoised output set Q. This set can be too sparse.

Algorithm 2: ParfreeDeclutter(P).

Data: Point set P
Result: Denoised point set P_0
begin
  Set $i_* = \lfloor \log_2(|P|) \rfloor$, and $P_{i_*} \longleftarrow P$
  for $i \longleftarrow i_*$ *to* 1 **do**
    $Q \longleftarrow$ Declutter(P_i,2^i)
    $P_{i-1} \longleftarrow \cup_{q \in Q} B(q, (10 + 2\sqrt{2}) d_{P_i,2^i}(q)) \cap P_i$
  end
end

We enrich it by re-introducing some points from P_i, obtaining a denser sampling $P_{i-1} \subseteq P_i$ of the ground truth. We call this a *re-sampling* process. This re-sampling step may bring some outliers back into the current set. However, it turns out that a repeated cycle of decluttering and resampling with decreasing values of k removes these outliers progressively. See Figure 2 and also more examples in the full version [6]. The entire process remains free of any user supplied parameter. In the end, we show that for an input that satisfies a uniform sampling condition, we can obtain an output set which is both dense and Hausdorff close to the hidden compact set, without the need to know the parameters of the input sampling conditions.

In order to formulate the exact statement of Theorem 15, we need to introduce a more relaxed sampling condition. We relax the notion of uniform (ϵ_k, c)-noisy sample by removing condition 2. We call it a *weak uniform* (ϵ_k, c)*-noisy sample*. Recall that condition 2 was the one forbidding the noise to be too dense. So essentially, a weak uniform (ϵ_k, c)-noisy sample only concerns points on and around the ground truth, with no conditions on outliers.

▶ **Theorem 15.** *Given a point set P and i_0 such that for all $i > i_0$, P is a weak uniform $(\epsilon_{2^i}, 2)$-noisy sample of K and is also a uniform $(\epsilon_{2^{i_0}}, 2)$-noisy sample of K, Algorithm ParfreeDeclutter returns a point set $P_0 \subseteq P$ such that $d_H(P_0, K) \leq (87 + 16\sqrt{2})\epsilon_{2^{i_0}}$.*

We elaborate a little on the sampling conditions. On one hand, as illustrated by Figure 1, the uniformity on input points is somewhat necessary in order to obtain a parameter-free algorithm. So requiring a uniform $(\epsilon_{2^{i_0}}, 2)$-noisy sample of K is reasonable. Now it would have been ideal if the theorem only required that P is a uniform $(\epsilon_{2^{i_0}}, 2)$-noisy sample of K for some $k_0 = 2^{i_0}$. However, to make sure that this uniformity is not destroyed during our iterative declutter-resample process before we reach $i = i_0$, we also need to assume that, *around the compact set*, the sampling is uniform for any $k = 2^i$ with $i > i_0$ (i.e, before we reach $i = i_0$). The specific statement for this guarantee is given in Lemma 17. However, while the uniformity for points *around the compact set* is required for any $i > i_0$, the condition that noisy points cannot be arbitrarily dense is *only* required for one parameter, $k = 2^{i_0}$.

The constant for the ball radius in the resampling step is taken as $10 + 2\sqrt{2}$ which we call the resampling constant C. Our theoretical guarantees hold with this resampling constant though a value of 4 works well in practice. The algorithm reduces more noise with decreasing C. On the flip side, the risk of removing points causing loss of true signal also increases with decreasing C. Section 5 and Appendix E [6] provide several results for Algorithm ParfreeDeclutter. We also point out that while our theoretical guarantee is for non-adaptive case, in practice, the algorithm works well on adaptive sampling as well.

Proof for Theorem 15

Aside from the technical Lemma 16 on the k-distance, the proof is divided into three steps. First, Lemma 17 shows that applying the loop of the algorithm once with parameter $2k$ does not alter the existing sampling conditions for $k' \leq k$. This implies that the $\epsilon_{2^{i_0}}$-noisy sample condition on P will also hold for P_{i_0}. Then Lemma 18 guarantees that the step going from P_{i_0} to P_{i_0-1} will remove all outliers. Combined with Theorem 9, which guarantees that P_{i_0-1} samples well K, it guarantees that the Hausdorff distance between P_{i_0-1} and K is bounded. However, we do not know i_0 and we have no means to stop the algorithm at this point. Fortunately, we can prove Lemma 19 which guarantees that the remaining iterations will not remove too many points and break the theoretical guarantees – that is, no harm is done in the subsequent iterations even after $i < i_0$. Putting all three together leads to our main result Theorem 15.

▶ **Lemma 16.** *Given a point set P, $x \in \mathbb{X}$ and $0 \leq i \leq k$, the distance to the i-th nearest neighbor of x in P satisfies, $d_{\mathbb{X}}(x, p_i) \leq \sqrt{\frac{k}{k-i+1}} d_{P,k}(x)$.*

Proof. The claim is proved by the following derivation.

$$\frac{k-i+1}{k} d_{\mathbb{X}}(x,p_i)^2 \leq \frac{1}{k}\sum_{j=i}^{k} d_{\mathbb{X}}(x,p_j)^2 \leq \frac{1}{k}\sum_{j=1}^{k} d_{\mathbb{X}}(x,p_j)^2 = d_{P,k}(x)^2. \qquad \blacktriangleleft$$

▶ **Lemma 17.** *Let P be a weak uniform $(\epsilon_{2k}, 2)$-noisy sample of K. For any $k' \leq k$ such that P is a (weak) uniform $(\epsilon_{k'}, c)$-noisy sample of K for some c, applying one step of the algorithm, with parameter $2k$ and resampling constant $C = 10 + 2\sqrt{2}$ gives a point set $P' \subseteq P$ which is a (weak) uniform $(\epsilon_{k'}, c)$-noisy sample of K.*

Proof. We show that if P is a uniform $(\epsilon_{k'}, c)$-noisy sample of K, then P' will also be a uniform $(\epsilon_{k'}, c)$-noisy sample of K. The similar version for weak uniformity follows from the same argument.

First, it is easy to see that as $P' \subset P$, the second and third sampling conditions of Def. 4 hold for P' as well. What remains is to show that Condition 1 also holds.

Take an arbitrary point $x \in K$. We know that $d_{P,2k}(x) \leq \epsilon_{2k}$ as P is a weak uniform $(\epsilon_{2k}, 2)$-noisy sample of K. Hence there exists $p \in P$ such that $d_{\mathbb{X}}(p, x) \leq d_{P,2k}(x) \leq \epsilon_{2k}$ and $d_{P,2k}(p) \leq 2\epsilon_{2k}$. Writing Q the result of the decluttering step, $\exists q \in Q$ such that $d_{\mathbb{X}}(p, q) \leq 2d_{P,2k}(p) \leq 4\epsilon_{2k}$. Moreover, $d_{P,2k}(q) \geq \frac{\epsilon_{2k}}{2}$ due to the uniformity condition for P.

Using Lemma 16, for $k' \leq k$, the k' nearest neighbors of x, which are chosen from P, $NN_{k'}(x)$ satisfies:

$$NN_{k'}(x) \subset B(x, \sqrt{2}\epsilon_{2k}) \subset B(p, (1+\sqrt{2})\epsilon_{2k}) \subset B(q, (5+\sqrt{2})\epsilon_{2k}) \subset B(q, (10+2\sqrt{2})d_{P,2k}(q))$$

Hence $NN_{k'}(x) \subset P'$ and $d_{P',k'}(x) = d_{P,k'}(x) \leq \epsilon_k$. This proves the lemma. $\blacktriangleleft$

▶ **Lemma 18.** *Let P be a uniform $(\epsilon_k, 2)$-noisy sample of K. One iteration of decluttering and resampling with parameter k and resampling constant $C = 10 + 2\sqrt{2}$ provides a set $P' \subseteq P$ such that $\delta_H(P', K) \leq 8C\epsilon_k + 7\epsilon_k$.*

Proof. Let Q denote the output after the decluttering step. Using Theorem 9 we know that $\delta_H(Q, K) \leq 7\epsilon_k$. Note that $Q \subset P'$. Thus, we only need to show that for any $p \in P'$, $d_{\mathbb{X}}(p, K) \leq 8C\epsilon_k + 7\epsilon_k$. Indeed, by the way the algorithm removes points, for any $p \in P'$, there exists $q \in Q$ such that $p \in B(q, Cd_{P,k}(q))$. It then follows that

$$d_{\mathbb{X}}(p, K) \leq Cd_{P,k}(q) + d_{\mathbb{X}}(q, K) \leq C(\epsilon_k + d_{\mathbb{X}}(q, K)) + 7\epsilon_k \leq 8C\epsilon_k + 7\epsilon_k. \qquad \blacktriangleleft$$

▶ **Lemma 19.** *Given a point $y \in P_i$, there exists $p \in P_0$ such that $d_{\mathbb{X}}(y,p) \leq \kappa d_{P_i,2^i}(y)$, where $\kappa = \frac{18+17\sqrt{2}}{4}$.*

Proof. We show this lemma by induction on i. First for $i = 0$ the claim holds trivially. Assuming that the result holds for all $j < i$ and taking $y \in P_i$, we distinguish three cases.

Case 1: $y \in P_{i-1}$ and $d_{P_{i-1},2^{i-1}}(y) \leq d_{P_i,2^i}(y)$. Applying the recurrence hypothesis for $j = i-1$ gives the result immediately.

Case 2: $y \notin P_{i-1}$. It means that y has been removed by decluttering and not been put back by resampling. These together imply that there exists $q \in Q_i \subseteq P_{i-1}$ such that $d_{\mathbb{X}}(y,q) \leq 2d_{P_i,2^i}(y)$ and $d_{\mathbb{X}}(y,q) > Cd_{P_i,2^i}(q)$ with $C = 10 + 2\sqrt{2}$. From the proof of Lemma 17, we know that the 2^{i-1} nearest neighbors of q in P_i are resampled and included in P_{i-1}. Therefore, $d_{P_{i-1},2^{i-1}}(q) = d_{P_i,2^{i-1}}(q) \leq d_{P_i,2^i}(q)$. Moreover, since $q \in P_{i-1}$, the inductive hypothesis implies that there exists $p \in P_0$ such that $d_{\mathbb{X}}(p,q) \leq \kappa d_{P_{i-1},2^{i-1}}(q) \leq \kappa d_{P_i,2^i}(q)$. Putting everything together, we get that there exists $p \in P_0$ such that

$$\begin{aligned} d_{\mathbb{X}}(p,y) &\leq d_{\mathbb{X}}(p,q) + d_{\mathbb{X}}(q,y) \\ &\leq \kappa d_{P_i,2^i}(q) + 2d_{P_i,2^i}(y) \\ &\leq \left(\frac{\kappa}{5+\sqrt{2}} + 2\right) d_{P_i,2^i}(y) \\ &\leq \kappa d_{P_i,2^i}(y). \end{aligned}$$

The derivation above also uses the relation that $d_{P_i,2^i}(q) < \frac{1}{C} d_{\mathbb{X}}(y,q) \leq \frac{2}{C} d_{P_i,2^i}(y)$.

Case 3: $y \in P_{i-1}$ and $d_{P_{i-1},2^{i-1}}(y) > d_{P_i,2^i}(y)$. The second part implies that at least one of the 2^{i-1} nearest neighbors of y in P_i does not belong to P_{i-1}. Let z be such a point. Note that $d_{\mathbb{X}}(y,z) \leq \sqrt{2} d_{P_i,2^i}(y)$ by Lemma 16. For point z, we can apply the second case and therefore, there exists $p \in P_0$ such that

$$\begin{aligned} d_{\mathbb{X}}(p,y) &\leq d_{\mathbb{X}}(p,z) + d_{\mathbb{X}}(z,y) \\ &\leq \left(\frac{\kappa}{5+\sqrt{2}} + 2\right) d_{P_i,2^i}(z) + \sqrt{2} d_{P_i,2^i}(y) \\ &\leq \left(\frac{\kappa}{5+\sqrt{2}} + 2\right) \left(d_{P_i,2^i}(y) + d_{\mathbb{X}}(z,y)\right) + \sqrt{2} d_{P_i,2^i}(y) \\ &\leq \left(\left(\frac{\kappa}{5+\sqrt{2}} + 2\right)(1+\sqrt{2}) + \sqrt{2}\right) d_{P_i,2^i}(y) \leq \kappa d_{P_i,2^i}(y) \end{aligned}$$

◀

Putting everything together. A repeated application of Lemma 17 (with weak uniformity) guarantees that P_{i_0+1} is a weak uniform $(\epsilon_{2^{i_0+1}}, 2)$-noisy sample of K. One more application (with uniformity) provides that P_{i_0} is a uniform $(\epsilon_{2^{i_0}}, 2)$-noisy sample of K. Thus, Lemma 18 implies that $d_H(P_{i_0-1}, K) \leq (87 + 16\sqrt{2})\epsilon_{2^{i_0}}$. Notice that $P_0 \subset P_{i_0-1}$ and thus for any $p \in P_0$, $d_{\mathbb{X}}(p,K) \leq (87+16\sqrt{2})\epsilon_{2^{i_0}}$.

To show the other direction, consider any point $x \in K$. Since P_{i_0} is a uniform $(\epsilon_{2^{i_0}}, 2)$-noisy sample of K, there exists $y \in P_{i_0}$ such that $d_{\mathbb{X}}(x,y) \leq \epsilon_{2^{i_0}}$ and $d_{P_{i_0},2^{i_0}}(y) \leq 2\epsilon_{2^{i_0}}$. Applying Lemma 19, there exists $p \in P_0$ such that $d_{\mathbb{X}}(y,p) \leq \frac{18+17\sqrt{2}}{2}\epsilon_{2^{i_0}}$. Hence $d_{\mathbb{X}}(x,p) \leq \left(\frac{18+17\sqrt{2}}{2} + 1\right)\epsilon_{2^{i_0}} \leq (87+16\sqrt{2})\epsilon_{2^{i_0}}$. The theorem then follows.

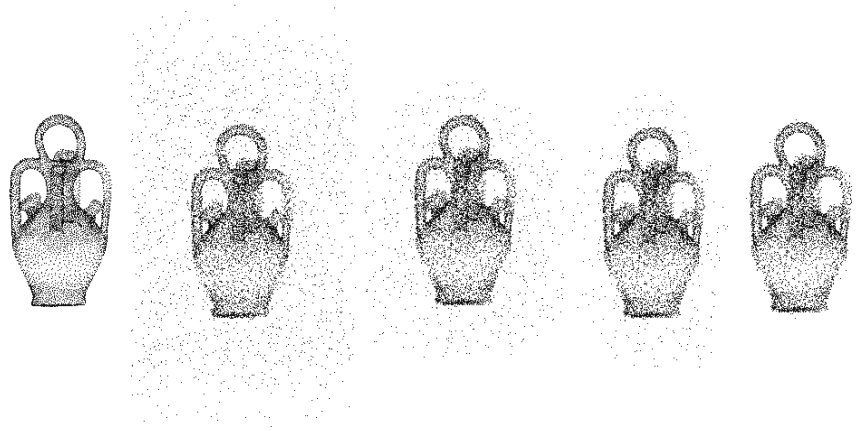


Figure 5 Experiment on a two dimensional manifold in three dimensions. From left to right, the ground truth, the noisy adaptively sampled input, output of two intermediate steps of the algorithm, and the final result.

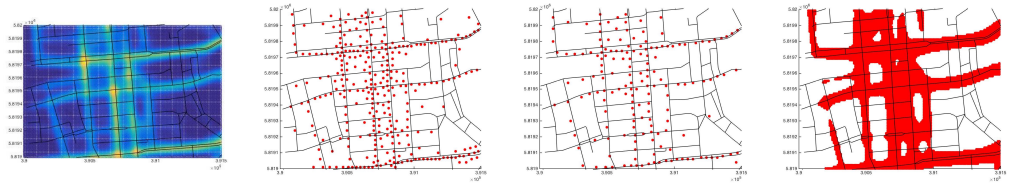
5 Preliminary experimental results

We now present some preliminary experimental results for the two denoising algorithms developed in this paper. See Appendix E of the full version [6] for more results.

In Figure 5, we show different stages of the ParfreeDeclutter algorithm on an input with *adaptively* sampled points. Even though for the parameter-free algorithm, theoretical guarantees are only provided for uniform samples, we note that it performs well on this adaptive case as well.

A second example is given in Figure 6. Here, the input data is obtained from a set of noisy GPS trajectories in the city of Berlin. In particular, given a set of trajectories (each modeled as polygonal curves), we first convert it to a density field by KDE (kernel density estimation). We then take the input as the set of grid points in 2D where every point is associated with a mass (density). Figure 6(a) shows the heat-map of the density field where light color indicates high density and blue indicates low density. In (b) and (c), we show the output of our Declutter algorithm (the ParfreeDeclutter algorithm does not provide good results as the input is highly non-uniform) for $k = 40$ and $k = 75$ respectively. In (d), we show the set of 40% points with the highest density values. The sampling of the road network is highly non-uniform. In particular, in the middle portion, even points off the roads have very high density due to noisy input trajectories. Hence a simple thresholding cannot remove these points and the output in (d) fills the space between roads in the middle portion; however more aggressive thresholding will cause loss of important roads. Our Declutter algorithm can capture the main road structures without collapsing nearby roads in the middle portion though it also sparsifies the data.

In another experiment, we apply the denoising algorithm as a pre-processing for high-dimensional data classification. Here we use MNIST data sets, which is a database of handwritten digits from '0' to '9'. Table 1 shows the experiment on digit 1 and digit 7. We take a random collection of 1352 images of digit '1' and 1279 images of digit '7' correctly labeled as a training set, and take 10816 images of digit 1 and digit 7 as a testing set. Each of the image is 28×28 pixels and thus can be viewed as a vector in $\mathbb{R}^{784}$. We use the L_1 metric to measure distance between such image-vectors. We use a linear SVM to classify the 10816 testing images. The classification error rate for the testing set is 0.6564% shown in the second row of Table 1.



(a) (b) (c) (d)

Figure 6 (a) The heat-map of a density field generated from GPS traces. There are around 15k (weighted) grid points serving as an input point set. The output of Algorithm Declutter when (b) $k = 40$ and (c) $k = 75$, (d) thresholding of 40% points with the highest density.

Table 1 Results of denoising on digit 1 and digit 7 from the MNIST.

1						Error(%)
2	Original	# Digit 1 1352		# Digit 7 1279		0.6564
3	Swap. Noise	# Mislabelled 1 270		# Mislabelled 7 266		4.0957
4		Digit 1		Digit 7		
5		# Removed	# True Noise	# Removed	# True Noise	
6	L1 Denoising	314	264	17	1	2.4500
7	Back. Noise	# Noisy 1 250		# Noisy 7 250		1.1464
8		Digit 1		Digit 7		
9		# Removed	# True Noise	# Removed	# True Noise	
10	L1 Denoising	294	250	277	250	0.7488

Next, we artificially add two types of noises to input data: the *swapping-noise* and the *background-noise*. The swapping-noise means that we randomly mislabel some images of '1' as '7', and some images of '7' as '1'. As shown in the third row of Table 1, the classification error increases to about 4.096% after such mislabeling in the training set.

Next, we apply our ParfreeDeclutter algorithm to this training set with added swapping-noise (to the set of images with label '1' and the set with label '7' separately) to first clean up the training set. As we can see in Row-6 of Table 1, we removed most images with a mislabeled '1' (which means the image is '7' but it is labeled as '1'). A discussion on why mislabeled '7's are not removed is given in the full version [6]. We then use the denoised dataset as the new training set, and improved the classification error to 2.45%.

The second type of noise is the *background noise*, where we replace the black backgrounds of a random subset of images in the training set (250 '1's and 250 '7's) with some other grey-scaled images. Under such noise, the classification error increases to 1.146%. Again, we perform our ParfreeDeclutter algorithm to denoise the training sets, and use the denoised data sets as the new training set. The classification error is then improved to 0.7488%. More results on the MNIST data sets are reported in the full version [6].

6 Discussions

Parameter selection is a notorious problem for many algorithms in practice. Our high level goal is to understand the roles of parameters in algorithms for denoising, how to reduce their use and what theoretical guarantees do they entail. While this paper presented some results

towards this direction, many interesting questions ensue. For example, how can we further relax our sampling conditions, making them allow more general inputs, and how to connect them with other classical noise models?

We also note that while the output of ParfreeDeclutter is guaranteed to be close to the ground truth w.r.t. the Hausdorff distance, this Hausdorff distance itself is not estimated. Estimating this distance appears to be difficult. We could estimate it if we knew the correct scale, i.e. i_0, to remove the ambiguity. Interestingly, even with the uniformity condition, it is not clear how to estimate this distance in a parameter free manner.

We do not provide guarantees for the parameter-free algorithm in an adaptive setting though the algorithm behaved well empirically for the adaptive case too. A partial result is presented in Appendix B of the full version [6], but the need for a small ϵ_k in the conditions defeat the attempts to obtain a complete result.

The problem of parameter-free denoising under more general sampling conditions remains open. It may be possible to obtain results by replacing uniformity with other assumptions, for example topological assumptions: say, if the ground truth is a simply connected manifold without boundaries, can this help to denoise and eventually reconstruct the manifold?

Acknowledgments. We thank Ken Clarkson for pointing out the result in [8].

References

1 N. Amenta and M. Bern. Surface reconstruction by voronoi filtering. *Discr. Comput. Geom.*, 22:481–504, 1999.

2 G. Biau et al. A weighted k-nearest neighbor density estimate for geometric inference. *Electronic Journal of Statistics*, 5:204–237, 2011.

3 J.-D. Boissonnat, L. J. Guibas, and S. Y. Oudot. Manifold reconstruction in arbitrary dimensions using witness complexes. *Discr. Comput. Geom.*, 42(1):37–70, 2009.

4 M. Buchet. *Topological inference from measures.* PhD thesis, Paris 11, 2014.

5 M. Buchet, F. Chazal, T. K. Dey, F. Fan, S. Y. Oudot, and Y. Wang. Topological analysis of scalar fields with outliers. In *Proc. 31st Sympos. Comput. Geom.*, pages 827–841, 2015.

6 M. Buchet, T. K. Dey, J. Wang, and Y. Wang. Declutter and resample: Towards parameter free denoising. *arXiv version of this paper: arXiv 1511.05479*, 2015.

7 C. Caillerie, F. Chazal, J. Dedecker, and B. Michel. Deconvolution for the wasserstein metric and geometric inference. In *Geom. Sci. Info.*, pages 561–568. 2013.

8 T.-H. H. Chan, M. Dinitz, and A. Gupta. Spanners with slack. In *Euro. Sympos. Algo.*, pages 196–207, 2006.

9 F. Chazal, D. Cohen-Steiner, and A. Lieutier. A sampling theory for compact sets in Euclidean space. *Discr. Comput. Geom.*, 41(3):461–479, 2009.

10 F. Chazal, D. Cohen-Steiner, and Q. Mérigot. Geometric inference for probability measures. *Found. Comput. Math.*, 11(6):733–751, 2011.

11 D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Trans. Pattern Anal. Machine Intelligence*, 24(5):603–619, 2002.

12 T. K. Dey, Z. Dong, and Y. Wang. Parameter-free topology inference and sparsification for data on manifolds. In *Proc. ACM-SIAM Sympos. Discr. Algo.*, pages 2733–2747, 2017.

13 T. K. Dey, J. Giesen, S. Goswami, and W. Zhao. Shape dimension and approximation from samples. *Discr. Comput. Geom.*, 29:419–434, 2003.

14 D. L. Donoho. De-noising by soft-thresholding. *IEEE Trans. Info. Theory.*, 41(3):613–627, 1995.

15 C. R. Genovese et al. On the path density of a gradient field. *The Annal. Statistics*, 37(6A):3236–3271, 2009.

16 V. J. Hodge and J. Austin. A survey of outlier detection methodologies. *Artificial Intelligence Review*, 22(2):85–126, 2004.

17 H. Jiang and S. Kpotufe. Modal-set estimation with an application to clustering. *arXiv preprint arXiv:1606.04166*, 2016.

18 A. Meister. *Deconvolution problems in nonparametric statistics*. Lecture Notes in Statistics. Springer, 2009.

19 U. Ozertem and D. Erdogmus. Locally defined principal curves and surfaces. *J. Machine Learning Research*, 12:1249–1286, 2011.

20 B. W. Silverman. *Density estimation for statistics and data analysis*, volume 26. CRC press, 1986.

21 J. Zhang. Advancements of outlier detection: A survey. *EAI Endorsed Trans. Scalable Information Systems*, 1(1):e2, 2013.

Ham Sandwich is Equivalent to Borsuk-Ulam

Karthik C. S.[*1] **and Arpan Saha**[†2]

1 **Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Israel**
karthik.srikanta@weizmann.ac.il

2 **Department of Mathematics, University of Hamburg, Germany**
arpan.saha@studium.uni-hamburg.de

Abstract

The Borsuk-Ulam theorem is a fundamental result in algebraic topology, with applications to various areas of Mathematics. A classical application of the Borsuk-Ulam theorem is the Ham Sandwich theorem: The volumes of any n compact sets in $\mathbb{R}^n$ can always be simultaneously bisected by an $(n-1)$-dimensional hyperplane.

In this paper, we demonstrate the equivalence between the Borsuk-Ulam theorem and the Ham Sandwich theorem. The main technical result we show towards establishing the equivalence is the following: For every odd polynomial restricted to the hypersphere $f : S^n \to \mathbb{R}$, there exists a compact set $A \subseteq \mathbb{R}^{n+1}$, such that for every $x \in S^n$ we have $f(x) = \mathrm{vol}(A \cap H^+) - \mathrm{vol}(A \cap H^-)$, where H is the oriented hyperplane containing the origin with $\vec{x}$ as the normal. A noteworthy aspect of the proof of the above result is the use of hyperspherical harmonics.

Finally, using the above result we prove that there exist constants $n_0, \varepsilon_0 > 0$ such that for every $n \geq n_0$ and $\varepsilon \leq \varepsilon_0/\sqrt{48n}$, any query algorithm to find an ε-bisecting $(n-1)$-dimensional hyperplane of n compact sets in $[-n^{4.51}, n^{4.51}]^n$, even with success probability $2^{-\Omega(n)}$, requires $2^{\Omega(n)}$ queries.

1998 ACM Subject Classification F.2.2 Computations on discrete structures, Geometrical problems and computations

Keywords and phrases Ham Sandwich theorem, Borsuk-Ulam theorem, Query Complexity, Hyperspherical Harmonics

Digital Object Identifier 10.4230/LIPIcs.SoCG.2017.24

1 Introduction

The Borsuk-Ulam theorem states that every continuous function from an n-sphere into Euclidean n-space maps some pair of antipodal points to the same point [6]. This result has countless applications in Mathematics [21]. In particular it implies the Brouwer's Fixed Point Theorem [7, 16] which is the basis of several important results in Economics [5], for example Nash's theorem [23]. Soon after the Borsuk-Ulam theorem was established, the Ham Sandwich theorem was proven using it [28, 29]. The Ham Sandwich theorem states that the volumes of any n compact sets in $\mathbb{R}^n$ can always be simultaneously bisected by an $(n-1)$-dimensional hyperplane. However, as far as we know, there is no result in previous literature establishing the equivalence of the Borsuk-Ulam theorem and the Ham Sandwich theorem.

* This work was partially supported by Irit Dinur's ISF-UGC 1399/14 grant.
† This work was partially supported by the Research Training Group 1670.

From a computational perspective, the computation of Brouwer fixed points has been studied extensively in various models of computations such as the Time/Computational complexity model [24, 12, 10, 9, 26, 27], the Query complexity model [18, 8, 11, 3, 27], and the Communication model [25, 4]. From these results one may obtain a reasonable understanding of the problem of computing equally valued antipodal points in a Borsuk-Ulam function by utilizing the constructive reduction from the Brouwer fixed point computation problem [30, 32]. However, computational aspects of the Ham Sandwich theorem have been poorly understood. In particular, no hardness result or non-trivial lower bounds in *any* model of computation are known in literature for the Ham Sandwich problem.

In this paper, we prove that the Borsuk-Ulam theorem and the Ham Sandwich theorem are indeed equivalent! Moreover, we use this equivalence to prove a query complexity lower bound on the Ham Sandwich problem.

1.1 Our Results

Our main result is a reduction from the Borsuk-Ulam theorem to the Ham Sandwich theorem. A key result in establishing the above reduction is that of establishing the equivalence between the two theorems for polynomial functions:

▶ **Proposition 1.** *For every polynomial $f : S^n \to \mathbb{R}^n$ restricted to the hypersphere, there exist n compact sets $A_1, \ldots, A_n \subseteq \mathbb{R}^{n+1}$, such that for every $x \in S^n$ and $i \in [n]$, we have the following:*

$$f_i(x) - f_i(-x) = vol(A_i \cap H^+) - vol(A_i \cap H^-),$$

where $f_i(x)$ is the projection of $f(x)$ to the i^{th} coordinate, and H is the oriented hyperplane containing the origin with $\vec{x}$ as the normal.

After establishing the above result, we use the Stone-Weierstrass theorem to note that any continuous function can be arbitrarily well approximated by polynomial functions, and prove the Borsuk-Ulam theorem for all continuous functions.

Next, we consider the Ham Sandwich problem in the query model: the input to the problem is n compact sets $A_1, \ldots, A_n \subseteq \left[-n^k, n^k\right]^n$, for some constant $k > 0$, and each query is an oriented hyperplane H and the answer is $\text{vol}(A_i \cap H^+) - \text{vol}(A_i \cap H^-)$, for all $i \in [n]$. The goal is to find a $(n-1)-$dimensional hyperplane H such that each set is ε-bisected by H, i.e., for all $i \in [n]$, we have $|\text{vol}(A_i \cap H^+) - \text{vol}(A_i \cap H^-)| \leq \varepsilon$. We show the following lower bound for the Ham Sandwich problem:

▶ **Theorem 2.** *There exist constants $n_0, \varepsilon_0 > 0$ such that for any $n \geq n_0$, $\varepsilon \leq \varepsilon_0/\sqrt{48n}$, $p = 2^{-\Omega(n)}$, and $k \geq 4.51$ the following holds: any query algorithm to find an ε-bisecting $(n-1)$-dimensional hyperplane of n compact sets in $\left[-n^k, n^k\right]^n$, even with success probability p, requires $2^{\Omega(n)}$ queries.*

By assuming a notion of Lipschitz continuity, we can show that the number of queries needed to compute an $\varepsilon-$bisecting hyperplane is $2^{O(n \log n)}$ by querying translations of hyperplanes over $\left[-n^k, n^k\right]^n$ whose normals form an $O(\varepsilon)$-net over S^n. Thus, the above lower bound is tight up to logarithmic multiplicative factor in the exponent. Furthermore, we remark here that Theorem 2 is the first nontrivial lower bound for the Ham Sandwich problem in *any* model of computation.

1.2 Our Techniques and Proof Overview

We provide below a proof-sketch of the reduction from the Borsuk-Ulam theorem to the Ham Sandwich theorem. The basic idea is to find for a given continuous odd function on S^n taking values in $\mathbb{R}$, a compact measurable set in $\mathbb{R}^{n+1}$, such that the given function is the difference of volumes of the set on the positive and negative side of an oriented hyperplane through the origin. This makes sense as the oriented hyperplanes through the origin are parametrised by S^n on which its positive unit normal takes values, so we actually get a continuous odd function on S^n. Then an oriented hyperplane bisects the set if and only if the given odd function vanishes at the point on S^n corresponding to the positive unit normal of the hyperplane. In particular, if we have have an odd continuous function from S^n to $\mathbb{R}^n$, we can make the above argument for every component. Then an oriented hyperplane bisects the sets if and only if the given odd function taking values in $\mathbb{R}^n$ vanishes at the point on S^n corresponding to the positive unit normal of the hyperplane.

A compact measurable set may be constructed by starting with a solid $(n+1)$-dimensional ball of unit radius centred at the origin and then radially scaling it by a continuous function on S^n taking values in $\mathbb{R}$ that is positive everywhere. Then the volume contained in a solid angle sector would be given by integrating an expression proportional to the $(n+1)$-th power of the scaling function over the region on S^n corresponding to the solid angle sector. Thus the difference of volumes on either side of a hyperplane is related to the $(n+1)$-th power of the scaling function by a linear integral transform. It turns out that this linear map becomes diagonal in the basis of hyperspherical harmonics and in order to invert this integral transform we work in this basis.

Since the basis is infinite-dimensional, there may be issues with convergence. We tackle this by first constructing the inverse transform for functions that are restrictions of polynomial functions on $\mathbb{R}^{n+1}$ to S^n (see Proposition 1), since in these cases, only finitely many elements in the basis suffice. This means that the reduction of Borsuk-Ulam theorem to the Ham Sandwich theorem holds for all functions that are restrictions of polynomial functions on $\mathbb{R}^{n+1}$ to S^n. And then we use the Stone-Weierstrass theorem, which states that any continuous function on $[-1,1]^{n+1}$ may be uniformly approximated using polynomial functions, to extend the reduction to all continuous functions on S^n.

In order to prove Theorem 2, we start from the randomized query complexity lower bound recently obtained by Rubinstein [27] (building on the works of Hirsch et al. [18] and Babichenko [3]) for the computation of *approximate* fixed points in a Brouwer function in the Euclidean norm. We then show that computing approximately equal-valued antipodal points in the query model is as hard as computing approximate fixed points in a Brouwer function by using Su's constructive proof of the Brouwer fixed point theorem from the Borsuk-Ulam theorem [30]. Finally, we use multivariate Bernstein polynomials to approximate the Borsuk-Ulam function and construct an input of the Ham Sandwich problem from Proposition 1 to obtain the randomized query complexity lower bound for the Ham-Sandwich problem.

1.3 Related Works

Papadimitriou considered the Ham Sandwich problem in the computational complexity model: given $2n^2$ points in general position in $\mathbb{R}^n$, separated into n groups with $2n$ points each, find a hyperplane which divides all groups in half. Papadimitriou showed that this search problem is in the complexity class **PPA** [24, 1]. However, no hardness result is known for this problem. On the other hand, there are many algorithms proposed in literature to solve this problem [13, 20, 33]. In particular, the best known algorithm for finding a

hyperplane simultaneously bisecting n point-sets $A_1, \ldots, A_n$ in $\mathbb{R}^d$, is $O\left(|A|^{d-1}\right)$ [20], where $A = \bigcup_{i \in [n]} A_i$.

A variant of the Ham-Sandwich problem was considered by Knauer et al.: Given $d+1$ point-sets in $\mathbb{R}^d$, is there a hyperplane which simultaneously bisects all the point-sets? They showed that this decision problem is **NP**-hard and **W**[1]-hard (with respect to d) [19], even when one of the point-sets is just a single point. However, it is not easy to construct a meaningful decision version of the Ham Sandwich problem because of its totality.

1.4 Organization of the Paper

This paper is organized as follows. In Section 2, we introduce the notations used in the rest of the paper, provide some key results about hyperspherical harmonics, and formally describe the query model of computation. In Section 3, we provide the complete reduction from the Borsuk-Ulam theorem to the Ham Sandwich theorem. In Section 4, we prove the randomized query complexity lower bound for the Ham Sandwich problem. Finally, in Section 5, we conclude by highlighting some open directions for future research.

2 Preliminaries

We formally state the two theorems of interest to this paper.

▶ **Theorem 3** (Borsuk-Ulam Theorem, [6]). *Let S^n denote the set of all points on the unit n-dimensional sphere. If $n \geq 0$ then for any continuous mapping $f : S^n \to \mathbb{R}^n$ there is a point $x \in S^n$ for which $f(x) = f(-x)$.*

▶ **Theorem 4** (Ham Sandwich Theorem, [28, 29]). *Given n compact sets in $\mathbb{R}^n$ there is a $(n-1)$-dimensional hyperplane which bisects each set into two sets of equal measure.*

Below, we list some notations and standard definitions that are used through out the paper.

2.1 Notations

The L^p norm of a vector $x \in \mathbb{R}^n$ is defined in the standard way as follows:

$$\|x\|_p = \left(\sum_{i \in [n]} |x_i|^p \right)^{1/p}.$$

Moreover, we define $S^n = \{x \in \mathbb{R}^{n+1} \mid \|x\|_2 = 1\}$, $S^n_\infty = \{x \in \mathbb{R}^{n+1} \mid \|x\|_\infty = 1\}$, and $B^n = \{x \in \mathbb{R}^n \mid \|x\|_2 \leq 1\}$.

A hyperplane in $\mathbb{R}^{n+1}$ is the set of solutions of an equation of the form

$$a_0 + \sum_{i=1}^{n+1} a_i x_i = 0.$$

The unit normals of the hyperplane are the vectors $\pm(a_1, a_2, \ldots, a_{n+1})$. A choice of one of the two possible unit normals is said to be an orientation on the hyperplane which is referred to as being oriented and the chosen unit normal as the positive unit normal (the other one is said to be the negative unit normal).

The volume of a compact set, assumed to be measurable, is simply its measure.

2.2 Hyperspherical Harmonics

We gather together some definitions and results we need regarding hyperspherical harmonics.

▶ **Definition 5.** A polynomial $H_\ell(x_1, x_2, \ldots, x_{n+1})$ is homogeneous of degree ℓ in the $n+1$ variables $x_1, x_2, ..., x_{n+1}$ provided $H_\ell(tx_1, tx_2, ..., tx_{n+1}) = t^\ell H_\ell(x_1, x_2, ..., x_{n+1})$. The Laplace operator in $\mathbb{R}^{n+1}$ is given by $\Delta_{n+1} := \sum_{i=1}^{n+1} \frac{\partial^2}{\partial x_i^2}$. $H_\ell(x_1, x_2, \ldots, x_{n+1})$ is called harmonic if $\Delta_{n+1} H_\ell(x_1, x_2, \ldots, x_{n+1}) = 0$. A hyperspherical harmonic of degree ℓ, denoted $Y_\ell^{(n+1)}(\xi)$, is a harmonic homogeneous polynomial of degree ℓ in $n+1$ variables restricted to S^n.

▶ **Claim 6.** *The dimension of the vector space of hyperspherical harmonics of degree ℓ on S^n is $M(n, \ell)$ where,*

$$M(n,\ell) = \begin{cases} 1 & \text{if } \ell = 0, \\ \frac{2\ell+n-1}{\ell}\binom{\ell+n-2}{\ell-1} & \text{if } \ell > 0. \end{cases}$$

Proof. See Theorem 4.4 in [14]. ◀

▶ **Definition 7.** Let V_{n+1} be the set of all $n+1$-variate polynomials over $\mathbb{R}$ restricted to S^n.

▶ **Claim 8.** *V_{n+1} is an inner product space, with addition and scaling defined for any two polynomials $f, g : S^n \to \mathbb{R}$ which are restricted to the n-sphere as follows:*

$$(f+g)(x) = f(x) + g(x),$$
$$\forall \alpha \in \mathbb{R},\ (\alpha \cdot f)(x) = \alpha \cdot f(x),$$
$$\langle f, g \rangle = \int_{x \in S^n} f(x) \cdot g(x)\ dx.$$

Proof. A linear combination of two polynomials is another polynomial. Furthermore, from the definition of the inner product, it is clear that $\langle f, g \rangle$ is symmetric under interchange of f and g and bilinear. To prove nondegenerateness, we shall show that $\langle f, f \rangle > 0$ whenever f is not identically zero. Assume that f is not identically zero. Then there must be point $x' \in S^n$ such that $f(x) \neq 0$. Because f is continuous there must be an open neighbourhood around this point such that f^2 is positive at all points in the neighbourhood. The integral of f^2 over this neighbourhood is therefore positive and since the integral of $f^2 \geq 0$ over the rest of the sphere has to be at least zero, it follows $\langle f, f \rangle > 0$. This completes the proof. ◀

▶ **Definition 9.** For $n \geq 2$ and each degree ℓ, the set $\left\{ Y_{\ell,m}^{(n+1)} \middle| m \in [M(n,\ell)] \right\}$ is a fixed orthonormal basis for the vector space of hyperspherical harmonics of degree ℓ on S^n.

▶ **Claim 10.** *For every $n \geq 2$, and $d \in \mathbb{Z}_{\geq 0}$, the set $\left\{ Y_{\ell,m}^{(n+1)} \middle| \ell \in \mathbb{Z}_{\geq 0}, \ell \leq d, m \in [M(n,\ell)] \right\}$ is an orthonormal set spanning all $f \in V_{n+1}$ of total degree d.*

Proof. Since every polynomial can be written as a finite sum of homogeneous polynomials of various degrees, it suffices to prove the above for the case where f is the restriction of a homogeneous polynomial $\tilde{f}$ of degree d to S^n. By Theorem 2.18 in [2], we note that there is a unique decomposition as follows,

$$\tilde{f}(x_1, \ldots, x_{n+1}) = \sum_{i=0}^{\lfloor d/2 \rfloor} H_{d-2i}(x_1, \ldots, x_{n+1}) \left(\sum_{j=1}^{n+1} x_j^2 \right)^i,$$

where H_ℓ is a harmonic homogeneous polynomial of degree ℓ. Restricting to S^n gives the following:

$$f = \sum_{i=0}^{\lfloor d/2 \rfloor} H_{d-2i}|_{S^n}.$$

The restriction $H_\ell|_{S^n}$ is a hyperspherical harmonic of degree ℓ and so is a (finite) linear combination of the functions $Y_{\ell,m}^{(n+1)}$ where m varies over $[M(n,\ell)]$. It follows that f is a (finite) linear combination of hyperspherical harmonics of degree at most d. Finally, orthonormality follows from Definition 9 above and Theorem 4.6 in [14]. ◀

▶ **Lemma 11.** *For every $n \geq 2$, and for any odd function f in V_{n+1}, let it be written as follows:*

$$f = \sum_{\ell \in \mathbb{Z}_{\geq 0}} \sum_{m=1}^{M(n,\ell)} \alpha_{\ell,m} \cdot Y_{\ell,m}^{(n+1)}.$$

Then, for every even integer ℓ, we have that $\alpha_{\ell,m} = 0$.

Proof. Since f is assumed to be odd, we have that $f(x) + f(-x) = 0$ for all $x \in S^n$. Since the sum in the hyperspherical harmonic decomposition of f is finite, we may rearrange the terms to have

$$\begin{aligned} 0 = &\sum_{\ell \in \mathbb{Z}_{\geq 0}^{\text{even}}} \sum_{m=1}^{M(n,\ell)} \alpha_{\ell,m} \cdot (Y_{\ell,m}^{(n+1)}(x) + Y_{\ell,m}^{(n+1)}(-x)) \\ &+ \sum_{\ell \in \mathbb{Z}_{\geq 0}^{\text{odd}}} \sum_{m=1}^{M(n,\ell)} \alpha_{\ell,m} \cdot (Y_{\ell,m}^{(n+1)}(x) + Y_{\ell,m}^{(n+1)}(-x)) \\ = 2 &\sum_{\ell \in \mathbb{Z}_{\geq 0}^{\text{even}}} \sum_{m=1}^{M(n,\ell)} \alpha_{\ell,m} \cdot Y_{\ell,m}^{(n+1)}(x). \end{aligned}$$

Now, for any $\ell' \in \mathbb{Z}_{\geq 0}^{\text{even}}$, we may multiply the above equation by $Y_{\ell',m}^{(n+1)}(x)$ on both sides and integrate over S^n so that, by virtue of Claim 10 we have $0 = 2\alpha_{\ell',m}$. Since $\ell' \in \mathbb{Z}_{\geq 0}^{\text{even}}$ was arbitrary , the result to be proved follows. ◀

▶ **Definition 12.** Let the sign function sgn on the interval $[-1,1]$ be defined as follows.

$$\forall \xi \in [-1,1],\ \mathsf{sgn}(\xi) = \begin{cases} -1 & \text{if } \xi < 0, \\ 0 & \text{if } \xi = 0, \\ 1 & \text{if } \xi > 0. \end{cases}$$

▶ **Definition 13.** For every $\ell \in \mathbb{Z}_{\geq 0}$, $n \geq 2$, and $\xi \in [-1,1]$, $P_\ell^{(n+1)}(\xi)$ is the ℓ^{th}-Gegenbauer polynomial in $n+1$ dimensions defined as follows:

$$P_\ell^{(n+1)}(\xi) = \frac{(-1)^\ell}{2^\ell \cdot \prod_{i=0}^{\ell-1} (\ell + (n-2)/2 - i)} (1-\xi^2)^{(2-n)/2} \left(\frac{\mathrm{d}}{\mathrm{d}\xi}\right)^\ell (1-\xi^2)^{\ell+(n-2)/2}.$$

▶ **Theorem 14** (Funk-Hecke theorem, [15, 17]). *Let $x \in S^n$, $f : [-1,1] \to \mathbb{R}$ a bounded measurable function, and $Y_\ell^{(n+1)}$ a hyperspherical harmonic polynomial of degree ℓ. Then,*

$$\int_{y \in S^n} f(\langle x, y \rangle) Y_\ell^{(n+1)}(y)\ \mathrm{d}y = s_{n-1} Y_\ell^{(n+1)}(x) \cdot \int_{-1}^{1} f(t) P_\ell^{(n+1)}(t)(1-t)^{n/2-1}\ \mathrm{d}t,$$

where s_{n-1} is the volume of the $(n-1)$-sphere, i.e., S^{n-1}.

Proof. See Theorem 4.24 in [14]. ◀

▶ **Lemma 15.** *Let $x \in S^n$, $f : [-1,1] \to \mathbb{R}$ a bounded measurable function, and $Y_\ell^{(n+1)}$ a hyperspherical harmonic polynomial of odd degree ℓ. Then,*

$$Y_\ell^{(n+1)}(x) = \frac{n}{2s_{n-1}} \cdot \frac{\prod_{i=1}^{(\ell-1)/2} (\ell - 2i + n + 1)}{\prod_{i=1}^{(\ell-1)/2} (\ell - 2i)} \int_{y \in S^n} \mathsf{sgn}(\langle x, y \rangle) \cdot Y_\ell^{(n+1)}(y)\ \mathrm{d}y.$$

Proof. Plugging in the sign function in Theorem 14, gives us:

$$\int_{y \in S^n} \mathsf{sgn}(\langle x, y \rangle) Y_\ell^{(n+1)}(y)\ \mathrm{d}y = s_{n-1} Y_\ell^{(n+1)}(x) \cdot \int_{-1}^{1} \mathsf{sgn}(t) P_\ell^{(n+1)}(t)(1-t)^{n/2-1}\ \mathrm{d}t.$$

So it remains to evaluate the below when ℓ is odd:

$$\int_{-1}^{1} \mathsf{sgn}(t) P_\ell^{(n+1)}(t)(1-t)^{n/2-1}\ \mathrm{d}t.$$

We plug in Definition 13 into the above

$$\int_{-1}^{1} \frac{\mathsf{sgn}(t)(-1)^\ell}{2^\ell \cdot \prod_{i=0}^{\ell-1} (\ell + (n-2)/2 - i)} \left(\frac{\mathrm{d}}{\mathrm{d}t}\right)^\ell (1-t^2)^{\ell + (n-2)/2} \mathrm{d}t.$$

When ℓ is odd, the function under the integral is even, so we have:

$$\begin{aligned}
&\int_{-1}^{1} \frac{\mathsf{sgn}(t)(-1)^\ell}{2^\ell \cdot \prod_{i=0}^{\ell-1} (\ell + (n-2)/2 - i)} \left(\frac{\mathrm{d}}{\mathrm{d}t}\right)^\ell (1-t^2)^{\ell + (n-2)/2} \mathrm{d}t \\
&= 2 \int_{0}^{1} \frac{\mathsf{sgn}(t)(-1)^\ell}{2^\ell \cdot \prod_{i=0}^{\ell-1} (\ell + (n-2)/2 - i)} \left(\frac{\mathrm{d}}{\mathrm{d}t}\right)^\ell (1-t^2)^{\ell + (n-2)/2} \mathrm{d}t \\
&= \int_{0}^{1} \frac{(-1)^\ell}{2^{\ell-1} \cdot \prod_{i=0}^{\ell-1} (\ell + (n-2)/2 - i)} \left(\frac{\mathrm{d}}{\mathrm{d}t}\right)^\ell (1-t^2)^{\ell + (n-2)/2} \mathrm{d}t.
\end{aligned}$$

The term under the integral is a total derivative, so the integral may be simplified to

$$\left[\frac{(-1)^\ell}{2^{\ell-1} \cdot \prod_{i=0}^{\ell-1} (\ell + (n-2)/2 - i)} \left(\frac{\mathrm{d}}{\mathrm{d}t}\right)^{\ell-1} (1-t^2)^{\ell + (n-2)/2} \right]_{t=0}^{t=1}.$$

Note that $\ell + (n-2)/2 = (\ell - 1) + (n + 2 - 2)/2$, so we may again use Definition 13 to write the above as

$$\left[-\frac{1}{n/2} \cdot (1-t^2)^{n/2} \cdot P_{\ell-1}^{(n+3)}(t) \right]_{t=0}^{t=1}.$$

When $t = 1$, the expression inside the square brackets vanishes. So all we are left with is $(2/n) \cdot P_{\ell-1}^{(n+3)}(0)$. The recurrence relation from Proposition 4.21 in [14] tells us that for all $\ell \geq 1$ we have

$$(\ell - 1 + n) \cdot P_{\ell-1}^{(n+3)}(0) + (\ell - 2) \cdot P_{\ell-3}^{(n+3)}(0) = 0.$$

This, along with the observation that $P_0^{(n+3)}(0) = 1$ may be used to determine $P_{\ell-1}^{(n+3)}(0)$ to be

$$P_{\ell-1}^{(n+3)}(0) = \frac{\prod_{i=1}^{(\ell-1)/2} (\ell - 2i)}{\prod_{i=1}^{(\ell-1)/2} (\ell - 2i + n + 1)}.$$

This completes the proof. ◀

2.3 Query Model

In this paper, we refer to the query model as described in [3]: every problem is specified by the allowed possible inputs, the desired outputs, and the queries which are specified types of questions that can be asked and by the answers that are provided. A query algorithm, is a procedure that asks queries in an adaptive manner and generates an output for every input. For this paper, a highly relevant remark is that there is no computational constraints on the way the query algorithm generates the next query or the output, given the previous answers.

For randomized query algorithms, errors are allowed in the output. To be precise, we require that for all inputs, the answer is correct only with probability $p < 1$. The randomized query complexity of a problem is the minimal number t such that given an input there exists a randomized query algorithm that makes at most t queries and outputs the correct answer with probability p. We denote the randomized query complexity of a problem Π by $\mathbf{QC}_p(\Pi)$. As noted by Babichenko [3] this measure of randomized query complexity is closely related to another measure: the expected number of queries for outputting the correct answer with probability p. Therefore, any lower bounds on $\mathbf{QC}_p(\Pi)$ can be easily translated to lower bounds on the expected number of queries.

3 Equivalence of Ham Sandwich and Borsuk-Ulam Theorems

In this section, we give the reduction from the Borsuk-Ulam theorem to the Ham Sandwich theorem. First, we show the reduction for polynomials restricted to the hypersphere.

▶ **Proposition 1.** *For every polynomial $f : S^n \to \mathbb{R}^n$ restricted to the hypersphere, there exist n compact sets $A_1, \ldots, A_n \subseteq \mathbb{R}^{n+1}$, such that for every $x \in S^n$ and $i \in [n]$, we have the following:*

$$f_i(x) - f_i(-x) = vol(A_i \cap H^+) - vol(A_i \cap H^-),$$

where $f_i(x)$ is the projection of $f(x)$ to the i^{th} coordinate, and H is the oriented hyperplane containing the origin with $\vec{x}$ as the normal.

Proof. We consider n projection functions of f: $f_1, \ldots, f_n : S^n \to \mathbb{R}$. Let d_i be the total degree of f_i. For every $i \in [n]$ we define $g_i(x) = f_i(x) - f_i(-x)$. Note that the g_is are odd functions. We define below n new functions $h_1, \ldots, h_n : S^n \to \mathbb{R}$ from the g_is. For every $i \in [n]$, from h_i, we construct A_i as follows:

$$A_i = \left\{ k \cdot (h_i(x_1, ..., x_{n+1}) \cdot x_1, \ldots, h_i(x_1, ..., x_{n+1}) \cdot x_{n+1}) \,\middle|\, (x_1, ..., x_{n+1}) \in S^n, 0 \leq k \leq 1 \right\}.$$

Note that we can define the volume of A_i as follows:

$$\mathrm{vol}(A_i) = \int_{y \in S^n} (h_i(y))^{n+1}/(n+1)\ \mathrm{d}y. \tag{1}$$

We will now define the h_is. We fix $i \in [n]$. From Claim 10, we have that g_i can be written as a linear combination of the hyperspherical harmonics.

$$g_i = \sum_{\substack{\ell \leq d_i, \\ \ell \in \mathbb{Z}^{\mathrm{odd}}_{\geq 0}.}} \sum_{m=1}^{M(n,\ell)} \alpha_{\ell,m} \cdot Y^{(n+1)}_{\ell,m}\,. \tag{2}$$

Note that in the above decomposition of g_i into hyperspherical harmonics, we have that only the odd spherical harmonics appear in the support (from Lemma 11). Next, we define a couple of constants (depending on ℓ and m). For every $\ell \in \mathbb{Z}^{\mathrm{odd}}_{\geq 0}$, where $\ell \leq d_i$, we have,

$$\gamma_\ell = \frac{n}{2s_{n-1}} \cdot \frac{\prod_{i=1}^{(\ell-1)/2} (\ell - 2i + n + 1)}{\prod_{i=1}^{(\ell-1)/2} (\ell - 2i)},$$

$$\beta_{\ell,m} = \alpha_{\ell,m} \cdot \gamma_\ell \cdot (n+1), \tag{3}$$

where s_n is the volume of the n-sphere S^n. Let $\Gamma_i : S^n \to \mathbb{R}$ be a function defined as follows:

$$\Gamma_i = \sum_{\substack{\ell \leq d_i, \\ \ell \in \mathbb{Z}^{\mathrm{odd}}_{\geq 0}.}} \sum_{m=1}^{M(n,\ell)} \beta_{\ell,m} \cdot Y^{(n+1)}_{\ell,m}\,. \tag{4}$$

Note that Γ_i is well defined because f is a polynomial function, which implies g_i is a polynomial function. We have the following bound on Γ_i:

▶ **Claim 16.** *Let* $\psi_i = \max_{x \in S^n} |\Gamma_i(x)|$. *Then,*

$$\psi_i < (n+1)^{(n+7)/2} \cdot (d_i+1)^{3/2} \cdot \left(1 + \frac{d_i}{n}\right)^n \cdot \max_{x \in S^n} |g_i(x)|.$$

Finally, we define h_i as follows:

$$h_i = \sqrt[n+1]{(\Gamma_i + \psi_i + 1)}, \tag{5}$$

This completes the construction of the n compact sets $A_1, \ldots, A_n$. Fix $i \in [n]$. Let H be some n-dimensional (oriented) hyperplane containing the origin and let x_H be the unit normal of H.

$$\mathrm{vol}(A_i \cap H^+) - \mathrm{vol}(A_i \cap H^-) = \frac{1}{(n+1)} \cdot \int_{y \in S^n} \mathsf{sgn}(\langle x_H, y \rangle) \cdot (h_i(y))^{n+1}\ \mathrm{d}y \quad \text{(From (1))}$$

$$= \frac{1}{(n+1)} \cdot \int_{y \in S^n} \mathsf{sgn}(\langle x_H, y \rangle) \cdot (\Gamma_i(y) + \psi_i + 1) \, \mathrm{d}y \qquad \text{(From (5))}$$

$$= \frac{1}{(n+1)} \cdot \int_{y \in S^n} \mathsf{sgn}(\langle x_H, y \rangle) \cdot \Gamma_i(y) \, \mathrm{d}y$$

$$= \frac{1}{(n+1)} \cdot \int_{y \in S^n} \mathsf{sgn}(\langle x_H, y \rangle) \cdot \left(\sum_{\substack{\ell \leq d_i, \\ \ell \in \mathbb{Z}^{\text{odd}}_{\geq 0}.}} \sum_{m=1}^{M(n,\ell)} \beta_{\ell,m} \cdot Y^{(n+1)}_{\ell,m}(y) \right) \mathrm{d}y \qquad \text{(From (4))}$$

$$= \sum_{\substack{\ell \leq d_i, \\ \ell \in \mathbb{Z}^{\text{odd}}_{\geq 0}.}} \sum_{m=1}^{M(n,\ell)} \beta_{\ell,m} \cdot \frac{1}{(n+1)} \cdot \int_{y \in S^n} \mathsf{sgn}(\langle x_H, y \rangle) \cdot Y^{(n+1)}_{\ell,m}(y) \, \mathrm{d}y$$

$$= \sum_{\substack{\ell \leq d_i, \\ \ell \in \mathbb{Z}^{\text{odd}}_{\geq 0}.}} \sum_{m=1}^{M(n,\ell)} \beta_{\ell,m}/\gamma_\ell \cdot \frac{1}{(n+1)} \cdot Y^{(n+1)}_{\ell,m}(x_H) \qquad \text{(From Lemma 15)}$$

$$= \sum_{\substack{\ell \leq d_i, \\ \ell \in \mathbb{Z}^{\text{odd}}_{\geq 0}.}} \sum_{m=1}^{M(n,\ell)} \alpha_{\ell,m} \cdot Y^{(n+1)}_{\ell,m}(x_H) \qquad \text{(From (3))}$$

$$= g_i(x_H) = f_i(x_H) - f_i(-x_H) \qquad \text{(From (2))}$$

This completes the proof. ◀

Below we provide the complete reduction from the Borsuk-Ulam theorem to the Ham Sandwich theorem.

▶ **Theorem 17** (Theorem 3 restated for $n \geq 3$)**.** *For every $n \geq 3$, if $f : S^n \to \mathbb{R}^n$ is continuous then there exists an $x \in S^n$ such that, $f(-x) = f(x)$.*

Proof. Given a continuous function $f_i : S^n \to \mathbb{R}$ we may use the Tietze Extension Theorem to extend it to a continuous function $\tilde{f_i}$ on $[-1,1]^{n+1}$ and then use the Stone-Weierstrass theorem to note that for any real $\varepsilon > 0$ we may find an $(n+1)$-variate polynomial function p_i such that $|\tilde{f_i}(x) - p_i(x)| < \varepsilon$ for all $x := (x_1, \ldots, x_{n+1}) \in [-1,1]^{n+1}$. In particular $|f(x) - p(x)| < \varepsilon$ for all $x \in S^n$.

By Proposition 1, we know that there exist n compact sets $A_1, \ldots, A_n \subseteq \mathbb{R}^{n+1}$, such that for every $x \in S^n$ and $i \in [n]$, we have $p_i(x) - p_i(-x) = \mathrm{vol}(A_i \cap H^+) - \mathrm{vol}(A_i \cap H^-)$ where x is the unit normal of the oriented hyperplane H. We introduce another compact set A_{n+1} which is a closed ball centred at the origin, so that any hyperplane bisecting its volume has to necessarily pass through the origin.

By the Ham Sandwich Theorem, we know that there is an oriented hyperplane H' such that $\mathrm{vol}(A_i \cap H'^+) = \mathrm{vol}(A_i \cap H'^-)$ for all $i \in [n+1]$, which is to say, there is an oriented hyperplane H' through the origin such that $\mathrm{vol}(A_i \cap H'^+) = \mathrm{vol}(A_i \cap H'^-)$ for all $i \in [n]$. But this means that $p_i(x') - p_i(-x') = 0$ for all $i \in [n]$ (where x' is the unit normal of H'), and so $|f_i(x') - f_i(-x')| < 2\varepsilon$ for all $i \in [n]$. The map $x \mapsto |f_i(x) - f_i(-x)|$ where $x \in S^n$ is continuous and defined on a compact domain. So it must attain a minimum value somewhere, which is nonnegative. But we have already shown that $|f_i(x) - f_i(-x)| < 2\varepsilon$ for all $\varepsilon > 0$ and $i \in [n]$. It follows that the minimum value attained by this map is 0 (simultaneously for all $i \in [n]$). Let it be attained at $x'' \in S^n$. Then $f(x'') = f(-x'')$. This completes the proof. ◀

4 Query Complexity Lower Bounds

In this section, we show query complexity lower bounds on the Ham Sandwich problem, by using the connection established through Proposition 1.

4.1 Borsuk-Ulam problem in Query Model

The query complexity of computing an approximate fixed-point of a Brouwer function in the *max norm* was studied by Hirsch et al. [18] in the deterministic setting. Recently, Babichenko [3] extended their lower bounds to the randomized setting. Rubinstein [27], furthered this direction to the case of fixed point computation in the *Euclidean norm*. Before stating the result of Rubinstein, we formally define the approximate fixed point problem in the query model as follows:

$\mathsf{AFP}^{\mathbf{Q}}(n, \lambda, \varepsilon)$ Problem:
Input: λ-Lipschitz function $f : [-1, 1]^n \to [-1, 1]^n$.
Output: $x \in [-1, 1]^n$ such that $\|f(x) - x\|_2^2 \leq \varepsilon \cdot n$.
Queries: Each query is a point $x \in [-1, 1]^n$ and the answer is $f(x)$.

We have the following lower bound on $\mathbf{QC}_p(\mathsf{AFP}^{\mathbf{Q}}(n, \lambda, \varepsilon))$.

▶ **Theorem 18** (Rubinstein [27]). *There exist constants $\varepsilon_0, \lambda_0, n_0 > 0$ such that for any $n \geq n_0$, $\varepsilon \leq \varepsilon_0$, and $\lambda \geq \lambda_0$, and for $p = 2^{-\Omega(n)}$ the following holds:*

$$\mathbf{QC}_p(\mathsf{AFP}^{\mathbf{Q}}(n, \lambda, \varepsilon)) = 2^{\Omega(n)}.$$

Next, we define the approximate equally valued antipodal point problem in the query model as follows:

$\mathsf{AAP}^{\mathbf{Q}}(n, \lambda, \varepsilon)$ Problem:
Input: λ-Lipschitz function $f : \sqrt{n+1} \cdot S^n \to \sqrt{n} \cdot B^n$.
Output: $x \in B^n$ such that $\|f(x) - f(-x)\|_2^2 \leq \varepsilon \cdot n$.
Queries: Each query is a point $x \in \sqrt{n+1} \cdot S^n$ and the answer is $f(x)$.

We have the following lower bound on $\mathbf{QC}_p(\mathsf{AAP}^{\mathbf{Q}}(n, \lambda, \varepsilon))$.

▶ **Theorem 19.** *There exist constants $\varepsilon_0, n_0 > 0$ such that for any $n \geq n_0$, $\varepsilon \leq \varepsilon_0/12n$, and $\lambda \geq 5\sqrt{n}$, and for $p = 2^{-\Omega(n)}$ the following holds:*

$$\mathbf{QC}_p(\mathsf{AAP}^{\mathbf{Q}}(n, \lambda, \varepsilon)) = 2^{\Omega(n)}.$$

Proof. We show that $\mathbf{QC}_p(\mathsf{AFP}^{\mathbf{Q}}(n, \lambda, \varepsilon)) \leq 2 \cdot \mathbf{QC}_p\left(\mathsf{AAP}^{\mathbf{Q}}\left(n, 5\sqrt{n}, \varepsilon^2/12n\right)\right)$ by using the construction of Su [30]. We start from a λ-Lipschitz continuous function $f : [-1, 1]^n \to [-1, 1]^n$ which is the input of $\mathsf{AFP}^{\mathbf{Q}}$ and have the following reduction to $\mathsf{AAP}^{\mathbf{Q}}$.

Adopting Su's Construction. Below, we describe the function $g_{\mathsf{Su}} : S_\infty^n \to [-3, 3]^n$, constructed by Su to build an instance of Borsuk-Ulam by starting from an instance of Brouwer. Let P be the projection function on to the first n coordinates. We define g_{Su} as follows:

$$g_{\mathsf{Su}}(x_1, \ldots, x_{n+1}) = \begin{cases} P(x) - f(P(x)) & \text{if } x_{n+1} = 1, \\ P(x) + f(P(-x)) & \text{if } x_{n+1} = -1, \\ P(x) + \frac{g_{\mathsf{Su}}(P(x),1) + g_{\mathsf{Su}}(P(x),-1)}{2} & \text{if } x_{n+1} = 0, \\ x_{n+1} \cdot g_{\mathsf{Su}}(P(x), 1) + (1 - x_{n+1}) \cdot g_{\mathsf{Su}}(P(x), 0) & \text{if } 0 \leq x_{n+1} \leq 1, \\ -x_{n+1} \cdot g_{\mathsf{Su}}(P(x), -1) + (1 + x_{n+1}) \cdot g_{\mathsf{Su}}(P(x), 0) & \text{if -1} \leq x_{n+1} \leq 0. \end{cases}$$

Using the above function, we can construct $g : \sqrt{n+1} \cdot S^n \to [-1,1]^n$ from g_{Su} as follows:

$$\forall x \in \sqrt{n+1} \cdot S^n,\ g(x) = \frac{1}{3} \cdot g_{\mathsf{Su}}\left(\frac{x}{\|x\|_\infty}\right).$$

First, we observe that g is an odd function:

▶ **Claim 20.** *For every $x \in \sqrt{n+1} \cdot S^n$, we have $g(x) = -g(-x)$.*

Next, we compute the Lipschitz constant of g below.

▶ **Claim 21.** *g is $5\sqrt{n}$-Lipschitz continuous.*

Furthermore, we note that we can obtain approximate fixed points of f from approximate equally valued antipodal points of g in a natural way as follows.

▶ **Claim 22.** *Fix $x \in \sqrt{n+1} \cdot S^n$. If $\|g(x) - g(-x)\|_2^2 \leq (\varepsilon^2/12n) \cdot n$ then,*

$$\left\| f\left(P\left(\frac{x}{\|x\|_\infty}\right)\right) - P\left(\frac{x}{\|x\|_\infty}\right)\right\|_2^2 \leq \varepsilon \cdot n.$$

Finally, the proof follows by noting that in order to compute g at a point, we need to query f in at most two points. ◀

Note that there is an easy deterministic query algorithm for $\mathsf{AAP}^{\mathbf{Q}}(n,\lambda,\varepsilon)$ which solves it with $\left(1+\frac{4\lambda}{\sqrt{\varepsilon}}\right)^{n+1}$ queries by building an $\frac{\sqrt{\varepsilon}}{2\lambda}$-net (Lemma 5.2 in [31]). In other words we have that $\mathbf{QC}_p\left(\mathsf{AAP}^{\mathbf{Q}}(n,\lambda,\varepsilon)\right) \leq 2^{O(n\log n)}$. Thus, the above lower bound is tight up to logarithmic multiplicative factor in the exponent.

Finally, we define the problem of interest for this section below.

4.2 Ham Sandwich Problem in Query Model

The approximate bisecting hyperplane problem in the query model is defined as follows:

$\mathsf{ABH}^{\mathbf{Q}}(n,k,\varepsilon)$ Problem:
Input: n compact sets $A_1,\ldots,A_n \subseteq \left[-n^k,n^k\right]^n$.
Output: $(n-1)$-dimensional hyperplane H such that $\forall i \in [n], |\mathrm{vol}(A_i \cap H^+) - \mathrm{vol}(A_i \cap H^-)| \leq \varepsilon$.
Queries: Each query is an oriented hyperplane H and the answer is $\mathrm{vol}(A_i \cap H^+) - \mathrm{vol}(A_i \cap H^-)$, for every $i \in [n]$.

We have the following lower bound on $\mathbf{QC}_p(\mathsf{ABH}^{\mathbf{Q}}(n,k,\varepsilon))$.

▶ **Theorem 2.** *There exist constants $n_0, \varepsilon_0 > 0$ such that for any $n \geq n_0$, $\varepsilon \leq \varepsilon_0/\sqrt{48n}$, $p = 2^{-\Omega(n)}$, and $k \geq 4.51$ the following holds: any query algorithm to find an ε-bisecting $(n-1)$-dimensional hyperplane of n compact sets in $\left[-n^k,n^k\right]^n$, even with success probability p, requires $2^{\Omega(n)}$ queries.*

Proof. We show $\mathbf{QC}_p\left(\mathsf{AAP}^{\mathbf{Q}}\left(n,5\sqrt{n},\varepsilon^2/12n\right)\right) \leq 2 \cdot \mathbf{QC}_p\left(\mathsf{ABH}^{\mathbf{Q}}\left(n+1,4.51,\varepsilon/\sqrt{48n}\right)\right)$ by using Proposition 1. We start from a $5\sqrt{n}$-Lipschitz continuous function $f : \sqrt{n+1} \cdot S^n \to \sqrt{n} \cdot B^n$ which is the input of $\mathsf{AAP}^{\mathbf{Q}}$ and have the following preprocessing step.

Preprocessing Step. Fix $i \in [n]$. Let $f_i : \sqrt{n+1} \cdot S^n \to [-\sqrt{n}, \sqrt{n}]$ be the i-th component of f which is $5\sqrt{n}$-Lipschitz continuous. We define f'_i as follows: $f'_i(x) = f_i(\sqrt{n+1} \cdot x)$. Note that f'_i is a function from S^n to $\sqrt{n} \cdot B^n$ and is $(\sqrt{n+1} \cdot 5\sqrt{n})$-Lipschitz continuous. Now by the Tietze extension theorem f'_i may be extended to a continuous function $\tilde{f}'_i$ on the cube $[-1,1]^{n+1} \supset S^n$ without increasing the Lipschitz constant [22]. Then from the Stone-Weierstrass theorem we have that for any $\varepsilon' > 0$ there is a polynomial function $\tilde{p}_i : [-1,1]^{n+1} \to \mathbb{R}$ such that $|\tilde{f}'_i(x) - \tilde{p}_i(x)| \leq \varepsilon'$ for all $x \in [-1,1]^{n+1}$. Let p_i be the restriction of $\tilde{p}_i$ to S^n. So, we have a polynomial function $p_i : S^n \to [-\sqrt{n} - \varepsilon/4\sqrt{12n}, \sqrt{n} + \varepsilon/4\sqrt{12n}]$ such that for all $x \in S^n$, we have $|f'_i(x) - p_i(x)| \leq \varepsilon/4\sqrt{12n}$ by setting $\varepsilon' = \varepsilon/4\sqrt{12n}$. Furthermore, we have that the degree of p_i is $O(n^5)$ (using multivariate Bernstein polynomials).

Adopting Proposition 1. We have from Proposition 1, that there exist n compact sets $A'_1, \ldots, A'_n \subseteq \mathbb{R}^{n+1}$, such that for every $x \in S^n$ and $i \in [n]$, $p_i(x) - p_i(-x) = \text{vol}(A'_i \cap H^+) - \text{vol}(A'_i \cap H^-)$, where H is the oriented hyperplane containing the origin with $\vec{x}$ as the normal. Fix $i \in [n]$. From the construction in proof of Proposition 1, we have that $A_i \subseteq [-h^\star_i, h^\star_i]^{n+1}$, where $h^\star_i = \max_{x \in S^n} |h_i(x)|$. We have the following upper bound on $h^\star_i$ from Claim 16:

$$\begin{aligned} h^\star_i &= \max_{x \in S^n} |h_i(x)| = \max_{x \in S^n} \left| \sqrt[n+1]{\Gamma_i(x) + \psi_i + 1} \right| \leq \sqrt[n+1]{2\psi_i + 1} \\ &= O\left(\sqrt[n+1]{\psi_i} \right) = O\left(\sqrt[n+1]{(n)^{(n+1)/2}} \cdot \sqrt[n+1]{n^{4n}} \right) \\ &= O\left(\sqrt{n} \cdot \sqrt[n+1]{n^{4n}} \right) = O\left(n^{4.5} \right). \end{aligned}$$

Next, we know that $|f'_i(x) - f'_i(-x)| \leq |p_i(x) - p_i(-x)| + \varepsilon/2\sqrt{12n}$. Thus, we have:

$$|f'_i(x) - f'_i(-x)| \leq \left| \text{vol}(A_i \cap H^+) - \text{vol}(A_i \cap H^-) \right| + \varepsilon/2\sqrt{12n}.$$

Therefore, if we are given some hyperplane H such that for every $i \in [n]$, we have $|\text{vol}(A_i \cap H^+) - \text{vol}(A_i \cap H^-)| \leq \varepsilon/2\sqrt{12n}$ then, we would obtain $x \in S^n$ such that $|f'_i(x) - f'_i(-x)| \leq \varepsilon/\sqrt{12n}$. This implies that $\|f(\sqrt{n+1} \cdot x) - f(-\sqrt{n+1} \cdot x)\|_2^2 \leq \varepsilon^2/12$. Finally, we complete the proof by noting that to answer $\text{vol}(A_i \cap H^+) - \text{vol}(A_i \cap H^-)$ for an oriented hyperplane H, we need to query f in at most two points. ◀

We note here that one can construct an easier (to solve) problem than $\mathsf{ABH}^{\mathsf{Q}}$, namely the Euclidean$-\mathsf{ABH}^{\mathsf{Q}}$ (or $\mathsf{ABH}^{\mathsf{Q}}_{\mathsf{E}}$ for short), where we need to find an $(n-1)$-dimensional hyperplane H such that $\left(\mathbb{E}_{i \in [n]} \left[(\text{vol}(A_i \cap H^+) - \text{vol}(A_i \cap H^-))^2 \right] \right)^{1/2} \leq \varepsilon$, and still obtain the same lower bounds as in Theorem 2, i.e., $\mathbf{QC}_p \left(\mathsf{ABH}^{\mathsf{Q}}_{\mathsf{E}} \left(n+1, 4.51, \varepsilon/\sqrt{48n} \right) \right) = 2^{\Omega(n)}$ by starting from $\mathbf{QC}_p \left(\mathsf{AAP}^{\mathsf{Q}} \left(n, 5\sqrt{n}, \varepsilon^2/12n \right) \right)$.

Finally, we remark that one could obtain lower bounds for the case of fixed dimension, i.e., when the compacts objects to be bisected are in a fixed dimension, by using the lower bounds of Chen and Teng [11] for the fixed point computation in Brouwer functions of fixed dimension.

5 Discussion and Conclusion

In this paper, we established the equivalence between the Borsuk-Ulam theorem and the Ham Sandwich theorem. Further, we used this equivalence to prove a lower bound on the Ham Sandwich problem in the query model.

It would be interesting to extend our lower bounds for the Ham Sandwich problem in the query model where the queries are to a membership oracle. Finally, showing that the Ham Sandwich problem introduced by Papadimitriou [24] is **PPA**-complete, remains an interesting and challenging open problem.

Acknowledgements. We would like to thank Irit Dinur for discussions which helped us to simplify the proof of Proposition 1. We would like to thank Inbal Livni Navon, and the anonymous reviewers of SoCG'17 for helping us improve the presentation of the paper.

References

1 James Aisenberg, Maria Luisa Bonet, and Sam Buss. 2-D Tucker is PPA complete. *Electronic Colloquium on Computational Complexity (ECCC)*, 22:163, 2015. URL: http://eccc.hpi-web.de/report/2015/163.

2 Kendall Atkinson and Weimin Han. *Spherical Harmonics and Approximations on the Unit Sphere: An Introduction*. Springer-Verlag Berlin Heidelberg, 2012. doi:10.1007/978-3-642-25983-8.

3 Yakov Babichenko. Query complexity of approximate nash equilibria. *J. ACM*, 63(4):36, 2016. doi:10.1145/2908734.

4 Yakov Babichenko and Aviad Rubinstein. Communication complexity of approximate nash equilibria. *CoRR*, abs/1608.06580, 2016. URL: http://arxiv.org/abs/1608.06580.

5 Kim Border. *Fixed Point Theorems with Applications to Economics and Game Theory*. Cambridge University Press, 1989. doi:10.1137/1028074.

6 Karol Borsuk. Drei sätze über die n-dimensionale euklidische sphäre. *Fundamental Mathematics*, 20:177–190, 1933.

7 L. E. J. Brouwer. Über Abbildung von Mannigfaltigkeiten. *Mathematische Annalen*, 71:97–115, 1912. URL: http://eudml.org/doc/158520.

8 Xi Chen and Xiaotie Deng. Matching algorithmic bounds for finding a brouwer fixed point. *J. ACM*, 55(3), 2008. doi:10.1145/1379759.1379761.

9 Xi Chen and Xiaotie Deng. On the complexity of 2D discrete fixed point problem. *Theor. Comput. Sci.*, 410(44):4448–4456, 2009. doi:10.1016/j.tcs.2009.07.052.

10 Xi Chen, Xiaotie Deng, and Shang-Hua Teng. Settling the complexity of computing two-player nash equilibria. *J. ACM*, 56(3), 2009. doi:10.1145/1516512.1516516.

11 Xi Chen and Shang-Hua Teng. Paths beyond local search: A tight bound for randomized fixed-point computation. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2007), October 20-23, 2007, Providence, RI, USA, Proceedings*, pages 124–134, 2007. doi:10.1109/FOCS.2007.53.

12 Constantinos Daskalakis, Paul W. Goldberg, and Christos H. Papadimitriou. The complexity of computing a nash equilibrium. *SIAM J. Comput.*, 39(1):195–259, 2009. doi:10.1137/070699652.

13 Herbert Edelsbrunner and Roman Waupotitsch. Computing a Ham-Sandwich Cut in Two Dimensions. *J. Symb. Comput.*, 2(2):171–178, 1986. doi:10.1016/S0747-7171(86)80020-7.

14 Costas Efthimiou and Christopher Frye. *Spherical harmonics in p dimensions*. World Scientific, Singapore, 2014. URL: https://cds.cern.ch/record/1953578.

15 P. Funk. Beiträge zur Theorie der Kugelfunktionen. *Mathematische Annalen*, 77:136–152, 1916. URL: http://eudml.org/doc/158720.

16 Jacques Hadamard. Note sur quelques applications de l'indice de kronecker. *Jules Tannery: Introduction à la théorie des fonctions d'une variable*, 2:437–477, 1910.

17 E. Hecke. Über orthogonal-invariante Integralgleichungen. *Mathematische Annalen*, 78:398–404, 1917. URL: http://eudml.org/doc/158775.

18 Michael D. Hirsch, Christos H. Papadimitriou, and Stephen A. Vavasis. Exponential lower bounds for finding brouwer fix points. *J. Complexity*, 5(4):379–416, 1989. doi:10.1016/0885-064X(89)90017-4.

19 Christian Knauer, Hans Raj Tiwary, and Daniel Werner. On the computational complexity of Ham-Sandwich cuts, Helly sets, and related problems. In *28th International Symposium on Theoretical Aspects of Computer Science, STACS 2011, March 10-12, 2011, Dortmund, Germany*, pages 649–660, 2011. doi:10.4230/LIPIcs.STACS.2011.649.

20 Chi-Yuan Lo, Jirí Matousek, and William L. Steiger. Algorithms for Ham-Sandwich Cuts. *Discrete & Computational Geometry*, 11:433–452, 1994. doi:10.1007/BF02574017.

21 Jirí Matousek. *Using the Borsuk-Ulam Theorem*. Springer-Verlag Berlin Heidelberg, 2003. doi:10.1007/978-3-540-76649-0.

22 E. J. McShane. Extension of range of functions. *Bulletin of the American Mathematical Society*, 40(12):837–843, 1934.

23 John Nash. Non-cooperative games. *Annals of Mathematics*, 54(2):286–295, 1951. URL: http://www.jstor.org/stable/1969529.

24 Christos H. Papadimitriou. On the complexity of the parity argument and other inefficient proofs of existence. *J. Comput. Syst. Sci.*, 48(3):498–532, 1994. doi:10.1016/S0022-0000(05)80063-7.

25 Tim Roughgarden and Omri Weinstein. On the communication complexity of approximate fixed points. *Electronic Colloquium on Computational Complexity (ECCC)*, 23:55, 2016. URL: http://eccc.hpi-web.de/report/2016/055.

26 Aviad Rubinstein. Inapproximability of Nash Equilibrium. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 409–418, 2015. doi:10.1145/2746539.2746578.

27 Aviad Rubinstein. Settling the complexity of computing approximate two-player Nash equilibria. *CoRR*, abs/1606.04550, 2016. URL: http://arxiv.org/abs/1606.04550.

28 Hugo Steinhaus. A note on the ham sandwich theorem. *Mathesis Polska*, 9:26–28, 1938.

29 A. H. Stone and J. W. Tukey. Generalized "sandwich" theorems. *Duke Mathematical Journal*, 9(2):356–359, 1942.

30 Francis Edward Su. Borsuk-Ulam implies Brouwer: a direct construction. *The American mathematical monthly*, 104(9):855–859, 1997.

31 Roman Vershynin. *Introduction to the non-asymptotic analysis of random matrices*, page 210–268. Cambridge University Press, May 2012. doi:10.1017/CBO9780511794308.006.

32 A. Yu. Volovikov. Brouwer, kakutani, and borsuk – ulam theorems. *Mathematical Notes*, 79(3):433–435, 2006. doi:10.1007/s11006-006-0048-0.

33 Fuxiang Yu. On the complexity of the pancake problem. *Math. Log. Q.*, 53(4-5):532–546, 2007. doi:10.1002/malq.200710016.

Local Equivalence and Intrinsic Metrics Between Reeb Graphs*

Mathieu Carrière[1] **and Steve Oudot**[2]

1 **DataShape, Inria Saclay, France**
`mathieu.cariere@inria.fr`
2 **DataShape, Inria Saclay, France**
`steve.oudot@inria.fr`

Abstract

As graphical summaries for topological spaces and maps, Reeb graphs are common objects in the computer graphics or topological data analysis literature. Defining good metrics between these objects has become an important question for applications, where it matters to quantify the extent by which two given Reeb graphs differ. Recent contributions emphasize this aspect, proposing novel distances such as *functional distortion* or *interleaving* that are provably more discriminative than the so-called *bottleneck distance*, being true metrics whereas the latter is only a pseudo-metric. Their main drawback compared to the bottleneck distance is to be comparatively hard (if at all possible) to evaluate. Here we take the opposite view on the problem and show that the bottleneck distance is in fact good enough *locally*, in the sense that it is able to discriminate a Reeb graph from any other Reeb graph in a small enough neighborhood, as efficiently as the other metrics do. This suggests considering the *intrinsic metrics* induced by these distances, which turn out to be all *globally* equivalent. This novel viewpoint on the study of Reeb graphs has a potential impact on applications, where one may not only be interested in discriminating between data but also in interpolating between them.

1998 ACM Subject Classification F.2.2 [Nonnumerical Algorithms and Problems:]Geometrical Problems and Computations

Keywords and phrases Reeb Graphs, Extended Persistence, Induced Metrics, Topological Data Analysis

Digital Object Identifier 10.4230/LIPIcs.SoCG.2017.25

1 Introduction

In the context of shape analysis, the Reeb graph [26] provides a meaningful summary of a topological space and a real-valued function defined on that space. Intuitively, it continuously collapses the connected components of the level sets of the function into single points, thus tracking the values of the functions at which the connected components merge or split. Reeb graphs have been widely used in computer graphics and visualization – see [7] for a survey, and their discrete versions, including the so-called *Mappers* [27], have become emblematic tools of topological data analysis due to their success in applications [2, 3, 20, 23].

Finding relevant dissimilarity measures for comparing Reeb graphs has become an important question in the recent years. The quality of a dissimilarity measure is usually

* This work was partially supported by ERC Grant Agreement No. 339025 GUDHI (Algorithmic Foundations of Geometry Understanding in Higher Dimensions) and was carried out while the second author was visiting the ICERM at Brown University.

assessed through three criteria: its ability to satisfy the axioms of a metric, its discriminative power, and its computational efficiency. The most natural choice to begin with is to use the *Gromov-Hausdorff distance* d_{GH} [10] for Reeb graphs seen as metric spaces. The main drawback of this distance is to quickly become intractable to compute in practice, even for graphs that are metric trees [1]. Among recent contributions, the *functional distortion distance* d_{FD} [4] and the *interleaving distance* d_{I} [15] share the same advantages and drawbacks as d_{GH}, in particular they enjoy good stability and discriminativity properties but they lack efficient algorithms for their computation, moreover they can be difficult to interpret. By contrast, the *bottleneck distance* d_{B} comes with a signature for Reeb graphs, called the *extended persistence diagram* [14], which acts as a stable bag-of-feature descriptor. Furthermore, d_{B} can be computed efficiently in practice. Its main drawback though is to be only a pseudo-metric, so distinct graphs can have the same signature and therefore be deemed equal in d_{B}.

Another desired property for dissimilarity measures is to be *intrinsic*, i.e. realized as the lengths of shortest continuous paths in the space of Reeb graphs [10]. This is particularly useful when one actually needs to interpolate between data, and not just discriminate between them, which happens in applications such as image or 3-d shape morphing, skeletonization, and matching [18, 21, 22, 28]. At this time, it is unclear whether the metrics proposed so far for Reeb graphs are intrinsic or not. Using intrinsic metrics would not only open the door to the use of Reeb graphs in the aforementioned applications, but it would also provide a better understanding of the intrinsic structure of the space of Reeb graphs, and give a deeper meaning to the distance values.

Our contributions. In the first part of the paper we show that the bottleneck distance can discriminate a Reeb graph from any other Reeb graph in a small enough neighborhood, as efficiently as the other metrics do, even though it is only a pseudo-metric globally. More precisely, we show that, given any constant $K \in (0, 1/22]$, in a sufficiently small neighborhood of a given Reeb graph R_f in the functional distortion distance (that is: for any Reeb graph R_g such that $d_{\mathrm{FD}}(\mathrm{R}_f, \mathrm{R}_g) < c(f, K)$, where $c(f, K) > 0$ is a positive constant depending only on f and K), one has:

$$K d_{\mathrm{FD}}(\mathrm{R}_f, \mathrm{R}_g) \leq d_{\mathrm{B}}(\mathrm{R}_f, \mathrm{R}_g) \leq 2 d_{\mathrm{FD}}(\mathrm{R}_f, \mathrm{R}_g). \tag{1}$$

The second inequality is already known [4], and it asserts that the bottleneck distance between Reeb graphs is stable. The first inequality is new, and it asserts that the bottleneck distance is discriminative locally, in fact just as discriminative as the other distances mentioned above. Equation (1) can be viewed as a local equivalence between metrics although not in the usual sense: firstly, all comparisons are anchored to a fixed Reeb graph R_f, and secondly, the constants K and 2 are absolute.

The second part of the paper advocates the study of intrinsic metrics on the space of Reeb graphs, for the reasons mentioned above. As a first step, we propose to study the intrinsic metrics $\hat{d}_{\mathrm{GH}}$, $\hat{d}_{\mathrm{FD}}$, $\hat{d}_{\mathrm{I}}$ and $\hat{d}_{\mathrm{B}}$ induced respectively by d_{GH}, d_{FD}, d_{I} and d_{B}. While the first three are obviously globally equivalent because their originating metrics are, our second contribution is to show that the last one is also globally equivalent to the other three.

The paper concludes with a discussion and some directions for the study of the space of Reeb graphs as an intrinsic metric space.

Related work. Interpolation between Reeb graphs is also the underlying idea of the *edit distance* recently proposed by Di Fabio and Landi [16]. The problem with this distance, in its current form at least, is that it restricts the interpolation to pairs of graphs lying in the same

homeomorphism class. By contrast, our class of admissible paths is defined with respect to the topology induced by the functional distortion distance, as such it allows interpolating between distinct homeomorphism classes.

Interpolation between Reeb graphs is also related to the study of inverse problems in topological data analysis. To our knowledge, the only result in this vein shows the differentiability of the operator sending point clouds to the persistence diagram of their distance function [17]. Our first contribution (1) sheds light on the operator's local injectivity properties over the class of Reeb graphs.

2 Background

Throughout the paper we work with singular homology with coefficients in the field $\mathbb{Z}_2$, which we omit in our notations for simplicity. In the following, "connected" stands for "path-connected", and "cc" stands for "connected component(s)". Given a map $f : X \to \mathbb{R}$ and an interval $I \subseteq \mathbb{R}$, we write X_f^I as a shorthand for the preimage $f^{-1}(I)$, and we omit the subscript when the map is obvious from the context.

2.1 Morse-Type Functions

▶ **Definition 1.** A continuous real-valued function f on a topological space X is *of Morse type* if:

- **(i)** there is a finite set $\mathrm{Crit}(f) = \{a_1 < ... < a_n\} \subset \mathbb{R}$, called the set of *critical values*, such that over every open interval $(a_0 = -\infty, a_1), ..., (a_i, a_{i+1}), ..., (a_n, a_{n+1} = +\infty)$ there is a compact and locally connected space Y_i and a homeomorphism $\mu_i : Y_i \times (a_i, a_{i+1}) \to X^{(a_i,a_{i+1})}$ such that $\forall i = 0, ..., n, f|_{X^{(a_i,a_{i+1})}} = \pi_2 \circ \mu_i^{-1}$, where π_2 is the projection onto the second factor;
- **(ii)** $\forall i = 1, ..., n-1, \mu_i$ extends to a continuous function $\bar{\mu}_i : Y_i \times [a_i, a_{i+1}] \to X^{[a_i,a_{i+1}]}$; similarly, μ_0 extends to $\bar{\mu}_0 : Y_0 \times (-\infty, a_1] \to X^{(-\infty,a_1]}$ and μ_n extends to $\bar{\mu}_n : Y_n \times [a_n, +\infty) \to X^{[a_n,+\infty)}$;
- **(iii)** Each levelset $f^{-1}(t)$ has a finitely-generated homology.

Let us point out that a Morse function is also of Morse type, and that its critical values remain critical in the definition above. Note that some of its regular values may be termed critical as well in this terminology, with no effect on the analysis.

2.2 Extended Persistence

Let f be a real-valued function on a topological space X. The family $\{X^{(-\infty,\alpha]}\}_{\alpha\in\mathbb{R}}$ of sublevel sets of f defines a *filtration*, that is, it is nested w.r.t. inclusion: $X^{(-\infty,\alpha]} \subseteq X^{(-\infty,\beta]}$ for all $\alpha \leq \beta \in \mathbb{R}$. The family $\{X^{[\alpha,+\infty)}\}_{\alpha\in\mathbb{R}}$ of superlevel sets of f is also nested but in the opposite direction: $X^{[\alpha,+\infty)} \supseteq X^{[\beta,+\infty)}$ for all $\alpha \leq \beta \in \mathbb{R}$. We can turn it into a filtration by reversing the order on the real line. Specifically, let $\mathbb{R}^{\mathrm{op}} = \{\tilde{x} \mid x \in \mathbb{R}\}$, ordered by $\tilde{x} \leq \tilde{y} \Leftrightarrow x \geq y$. We index the family of superlevel sets by $\mathbb{R}^{\mathrm{op}}$, so now we have a filtration: $\{X^{[\tilde{\alpha},+\infty)}\}_{\tilde{\alpha}\in\mathbb{R}^{\mathrm{op}}}$, with $X^{[\tilde{\alpha},+\infty)} \subseteq X^{[\tilde{\beta},+\infty)}$ for all $\tilde{\alpha} \leq \tilde{\beta} \in \mathbb{R}^{\mathrm{op}}$.

Extended persistence connects the two filtrations at infinity as follows. First, replace each superlevel set $X^{[\tilde{\alpha},+\infty)}$ by the pair of spaces $(X, X^{[\tilde{\alpha},+\infty)})$ in the second filtration. This maintains the filtration property since we have $(X, X^{[\tilde{\alpha},+\infty)}) \subseteq (X, X^{[\tilde{\beta},+\infty)})$ for all $\tilde{\alpha} \leq \tilde{\beta} \in \mathbb{R}^{\mathrm{op}}$. Then, let $\mathbb{R}_{\mathrm{Ext}} = \mathbb{R} \cup \{+\infty\} \cup \mathbb{R}^{\mathrm{op}}$, where the order is completed by $\alpha < +\infty < \tilde{\beta}$ for all $\alpha \in \mathbb{R}$ and $\tilde{\beta} \in \mathbb{R}^{\mathrm{op}}$. This poset is isomorphic to $(\mathbb{R}, \leq)$. Finally, define the *extended*

filtration of f over $\mathbb{R}_{\mathrm{Ext}}$ by:

$$F_\alpha = X^{(-\infty,\alpha]} \text{ for } \alpha \in \mathbb{R},\ F_{+\infty} = X \equiv (X, \emptyset) \text{ and } F_{\tilde{\alpha}} = (X, X^{[\tilde{\alpha},+\infty)}) \text{ for } \tilde{\alpha} \in \mathbb{R}^{\mathrm{op}},$$

where we have identified the space X with the pair of spaces $(X, \emptyset)$ at infinity. The subfamily $\{F_\alpha\}_{\alpha\in\mathbb{R}}$ is the *ordinary* part of the filtration, while $\{F_{\tilde{\alpha}}\}_{\tilde{\alpha}\in\mathbb{R}^{\mathrm{op}}}$ is the *relative* part.

Applying the homology functor H_* to this filtration gives the so-called *extended persistence module* $\mathbb{V}$ of f, which is a sequence of vector spaces connected by linear maps induced by the inclusions in the extended filtration. For functions of Morse type, the extended persistence module can be decomposed as a finite direct sum of half-open *interval modules* – see e.g. [13]: $\mathbb{V} \simeq \bigoplus_{k=1}^{n} \mathbb{I}[b_k, d_k)$, where each summand $\mathbb{I}[b_k, d_k)$ is made of copies of the field of coefficients at every index $\alpha \in [b_k, d_k)$, and of copies of the zero space elsewhere, the maps between copies of the field being identities. Each summand represents the lifespan of a *homological feature* (cc, hole, void, etc.) within the filtration. More precisely, the *birth time* b_k and *death time* d_k of the feature are given by the endpoints of the interval. Then, a convenient way to represent the structure of the module is to plot each interval in the decomposition as a point in the extended plane, whose coordinates are given by the endpoints. Such a plot is called the *extended persistence diagram* of f, denoted $\mathrm{Dg}(f)$. The distinction between ordinary and relative parts of the filtration allows us to classify the points in $\mathrm{Dg}(f)$ as follows:

- $p = (x, y)$ is called an *ordinary* point if $x, y \in \mathbb{R}$;
- $p = (x, y)$ is called a *relative* point if $x, y \in \mathbb{R}^{\mathrm{op}}$;
- $p = (x, y)$ is called an *extended* point if $x \in \mathbb{R}, y \in \mathbb{R}^{\mathrm{op}}$;

Note that ordinary points lie strictly above the diagonal $\Delta = \{(x, x) \mid x \in \mathbb{R}\}$ and relative points lie strictly below Δ, while extended points can be located anywhere, including on Δ (e.g. when a cc lies inside a single critical level, see Section 2.3). It is common to partition $\mathrm{Dg}(f)$ according to this classification: $\mathrm{Dg}(f) = \mathrm{Ord}(f) \sqcup \mathrm{Rel}(f) \sqcup \mathrm{Ext}^+(f) \sqcup \mathrm{Ext}^-(f)$, where by convention $\mathrm{Ext}^+(f)$ includes the extended points located on the diagonal Δ.

Stability. An important property of extended persistence diagrams is to be stable in the so-called *bottleneck distance* d_b^∞. Given two persistence diagrams D, D', a *partial matching* between D and D' is a subset Γ of $D \times D'$ where for every $p \in D$ there is at most one $p' \in D'$ such that $(p, p') \in \Gamma$, and conversely, for every $p' \in D'$ there is at most one $p \in D$ such that $(p, p') \in \Gamma$. Furthermore, Γ must match points of the same type (ordinary, relative, extended) and of the same homological dimension only. The *cost* of Γ is: $\mathrm{cost}(\Gamma) = \max\{\max_{p\in D} \delta_D(p),\ \max_{p'\in D'} \delta_{D'}(p')\}$, where $\delta_D(p) = \|p - p'\|_\infty$ if p is matched to some $p' \in D'$ and $\delta_D(p) = d_\infty(p, \Delta)$ if p is unmatched – same for $\delta_{D'}(p')$.

▶ **Definition 2.** The *bottleneck distance* between two persistence diagrams D and D' is $d_{\mathrm{B}}(D, D') = \inf_\Gamma \mathrm{cost}(\Gamma)$, where Γ ranges over all partial matchings between D and D'.

▶ **Theorem 3** (Stability [14])**.** *For any Morse-type functions $f, g : X \to \mathbb{R}$,*

$$d_{\mathrm{B}}(\mathrm{Dg}(f), \mathrm{Dg}(g)) \leq \|f - g\|_\infty. \tag{2}$$

2.3 Reeb Graphs

▶ **Definition 4.** Given a topological space X and a continuous function $f : X \to \mathbb{R}$, we define the equivalence relation $\sim_f$ between points of X by $x \sim_f y$ if and only if $f(x) = f(y)$ and x, y belong to the same cc of $f^{-1}(f(x)) = f^{-1}(f(y))$. The *Reeb graph* $\mathrm{R}_f(X)$ is the quotient space $X/\sim_f$. As f is constant on equivalence classes, there is a well-defined induced map $\tilde{f} : \mathrm{R}_f(X) \to \mathbb{R}$.

Connection to extended persistence. If f is a function of Morse type, then the pair (X, f) is an $\mathbb{R}$-*constructible space* in the sense of [15]. This ensures that the Reeb graph is a multigraph, whose nodes are in one-to-one correspondence with the cc of the critical level sets of f. In that case, there is a nice interpretation of $\mathrm{Dg}(\tilde{f})$ in terms of the structure of $\mathrm{R}_f(X)$. We refer the reader to [4, 14] and the references therein for a full description as well as formal definitions and statements. Orienting the Reeb graph vertically so $\tilde{f}$ is the height function, we can see each cc of the graph as a trunk with multiple branches (some oriented upwards, others oriented downwards) and holes. Then, one has the following correspondences, where the *vertical span* of a feature is the span of its image by $\tilde{f}$:

- The vertical spans of the trunks are given by the points in $\mathrm{Ext}_0^+(\tilde{f})$;
- The vertical spans of the downward branches are given by the points in $\mathrm{Ord}_0(\tilde{f})$;
- The vertical spans of the upward branches are given by the points in $\mathrm{Rel}_1(\tilde{f})$;
- The vertical spans of the holes are given by the points in $\mathrm{Ext}_1^-(\tilde{f})$.

The rest of the diagram of $\tilde{f}$ is empty. These correspondences provide a dictionary to read off the structure of the Reeb graph from the persistence diagram of the quotient map $\tilde{f}$. Note that it is a bag-of-features type of descriptor, taking an inventory of all the features together with their vertical spans, but leaving aside the actual layout of the features. As a consequence, it is an incomplete descriptor: two Reeb graphs with the same persistence diagram may not be isomorphic. See the two Reeb graphs in Figure 1 for instance.

Notation. Throughout the paper, we consider Reeb graphs coming from Morse-type functions, equipped with their induced maps. We denote by Reeb the space of such graphs. In the following, we have $\mathrm{R}_f, \mathrm{R}_g \in \mathsf{Reeb}$, with induced maps $f : \mathrm{R}_f \to \mathbb{R}$ with critical values $\{a_1, ..., a_n\}$, and $g : \mathrm{R}_g \to \mathbb{R}$ with critical values $\{b_1, ..., b_m\}$. Note that we write f, g instead of $\tilde{f}, \tilde{g}$ for convenience. We also assume without loss of generality (w.l.o.g.) that R_f and R_g are connected. If they are not connected, then our analysis can be applied component-wise.

2.4 Distances for Reeb graphs

▶ **Definition 5.** The *bottleneck distance* between R_f and R_g is:

$$d_{\mathrm{B}}(\mathrm{R}_f, \mathrm{R}_g) := d_{\mathrm{B}}(\mathrm{Dg}(f), \mathrm{Dg}(g)). \tag{3}$$

▶ **Definition 6.** The *functional distortion distance* between R_f and R_g is:

$$d_{\mathrm{FD}}(\mathrm{R}_f, \mathrm{R}_g) := \inf_{\phi,\psi} \max \left\{ \frac{1}{2} D(\phi, \psi),\ \|f - g \circ \phi\|_\infty,\ \|f \circ \psi - g\|_\infty \right\}, \tag{4}$$

where:

- $\phi : \mathrm{R}_f \to \mathrm{R}_g$ and $\psi : \mathrm{R}_g \to \mathrm{R}_f$ are continuous maps,
- $D(\phi, \psi) = \sup \{|d_f(x, x') - d_g(y, y')| \text{ such that } (x, y), (x', y') \in C(\phi, \psi)\}$, where:
  - $C(\phi, \psi) = \{(x, \phi(x)) \mid x \in \mathrm{R}_f\} \cup \{(\psi(y), y) \mid y \in \mathrm{R}_g\}$,
  - $d_f(x, x') = \min_{\pi : x \to x'} \left\{ \max_{t \in [0,1]} f \circ \pi(t) - \min_{t \in [0,1]} f \circ \pi(t) \right\}$, where $\pi : [0, 1] \to \mathrm{R}_f$ is a continuous path from x to x' in R_f ($\pi(0) = x$ and $\pi(1) = x'$),
  - $d_g(y, y') = \min_{\pi : y \to y'} \left\{ \max_{t \in [0,1]} g \circ \pi(t) - \min_{t \in [0,1]} g \circ \pi(t) \right\}$, where $\pi : [0, 1] \to \mathrm{R}_g$ is a continuous path from y to y' in R_g ($\pi(0) = y$ and $\pi(1) = y'$).

Bauer et al. [4] related these distances as follows:

▶ **Theorem 7.** *The following inequality holds:* $d_{\mathrm{B}}(\mathrm{R}_f, \mathrm{R}_g) \leq 3\, d_{\mathrm{FD}}(\mathrm{R}_f, \mathrm{R}_g)$.

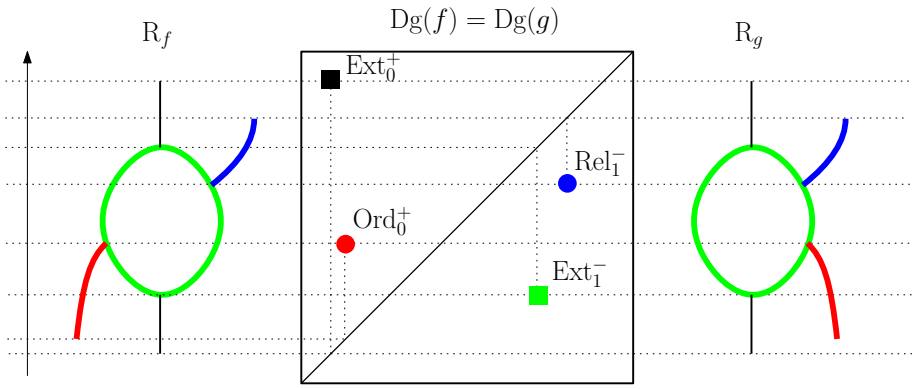


Figure 1 Example of two different Reeb graphs R_f and R_g that have the same extended persistence diagram $\mathrm{Dg}(f) = \mathrm{Dg}(g)$. These graphs are at bottleneck distance 0 from each other, while their functional distortion distance is positive.

This result can be improved using the end of Section 3.4 of [8], then noting that level set diagrams and extended diagrams are essentially the same [11], and finally Lemma 9 of [6]:

▶ **Theorem 8.** *The following inequality holds:* $d_{\mathrm{B}}(\mathrm{R}_f, \mathrm{R}_g) \leq 2\, d_{\mathrm{FD}}(\mathrm{R}_f, \mathrm{R}_g)$.

We emphasize that, even though Theorem 8 allows us to improve on the constants of our main result – see Theorem 9, the reduction from $3\, d_{\mathrm{FD}}(\mathrm{R}_f, \mathrm{R}_g)$ in Theorem 7 to $2\, d_{\mathrm{FD}}(\mathrm{R}_f, \mathrm{R}_g)$ in Theorem 8 is not fundamental for our analysis and proofs.

Since the bottleneck distance is only a pseudo-metric – see Figure 1, the inequality given by Theorem 8 cannot be turned into an equivalence result. However, for any pair of Reeb graphs $\mathrm{R}_f, \mathrm{R}_g$ that have the same extended persistence diagram $\mathrm{Dg}(f) = \mathrm{Dg}(g)$, and that are at positive functional distortion distance from each other, every continuous path in d_{FD} from R_f to R_g will perturb the points of $\mathrm{Dg}(f)$ and eventually drive them back to their initial position, suggesting first that d_{B} is locally equivalent to d_{FD} – see Theorem 9 in Section 3, but also that, even though $d_{\mathrm{B}}(\mathrm{R}_f, \mathrm{R}_g) = 0$, the intrinsic metric $\hat{d}_{\mathrm{B}}(\mathrm{R}_f, \mathrm{R}_g)$ induced by d_{B} is positive – see Theorem 17 in Section 4.

3 Local Equivalence

Let $a_f = \min_{1 \leq i \leq n} a_{i+1} - a_i > 0$ and $a_g = \min_{1 \leq j \leq m} b_{j+1} - b_j > 0$. In this section, we show the following local equivalence theorem:

▶ **Theorem 9.** *Let* $K \in (0, 1/22]$. *If* $d_{\mathrm{FD}}(\mathrm{R}_f, \mathrm{R}_g) \leq \max\{a_f, a_g\}/(8(1 + 22K))$, *then:*

$$K d_{\mathrm{FD}}(\mathrm{R}_f, \mathrm{R}_g) \leq d_{\mathrm{B}}(\mathrm{R}_f, \mathrm{R}_g) \leq 2 d_{\mathrm{FD}}(\mathrm{R}_f, \mathrm{R}_g).$$

Note that the notion of locality used here is slightly different from the usual one. On the one hand, the equivalence does not hold for any arbitrary pair of Reeb graphs inside a neighborhood of some fixed Reeb graph, but rather for any pair involving the fixed graph. On the other hand, the constants in the equivalence are independent of the pair of Reeb graphs considered. The upper bound on $d_{\mathrm{B}}(\mathrm{R}_f, \mathrm{R}_g)$ is given by Theorem 8 and always holds. The aim of this section is to prove the lower bound.

Convention: We assume w.l.o.g. that $\max\{a_f, a_g\} = a_f$, and we let $\varepsilon = d_{\mathrm{FD}}(\mathrm{R}_f, \mathrm{R}_g)$.

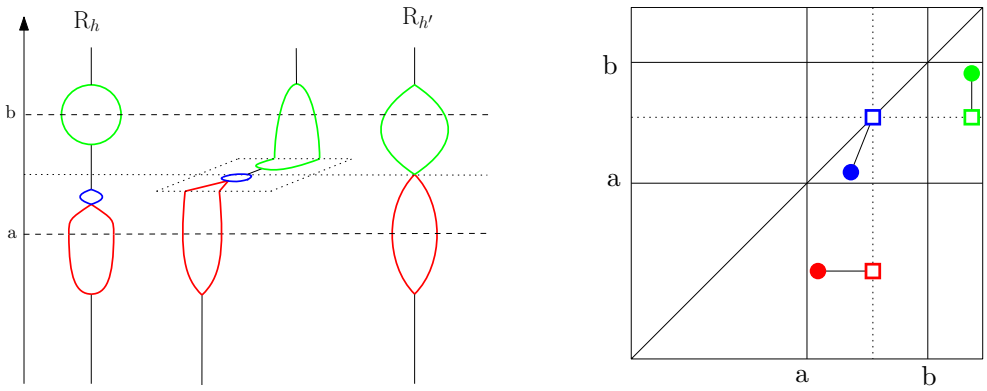


Figure 2 Left: effect of $\text{Merge}_{a,b}$ on a Reeb graph R_h. Right: Effect on its persistence diagram.

3.1 Proof of Theorem 9

Let $K \in (0, 1/22]$. The proof proceeds by contradiction. Assuming $d_{\text{B}}(\text{R}_f, \text{R}_g) < K\varepsilon$, where $\varepsilon = d_{\text{FD}}(\text{R}_f, \text{R}_g) < a_f/(8(1+22K))$, we progressively transform R_g into some other Reeb graph $\text{R}_{g'}$ (Definition 12) that satisfies both $d_{\text{FD}}(\text{R}_g, \text{R}_{g'}) < \varepsilon$ (Proposition 14) and $d_{\text{FD}}(\text{R}_f, \text{R}_{g'}) = 0$ (Proposition 15). The contradiction follows from the triangle inequality.

3.1.1 Graph Transformation

The graph transformation is defined as the composition of the *simplification operator* from [4] and the *Merge operator*[1] from [12]. We refer the reader to these articles for the precise definitions. Below we merely recall their main properties. Given a set $S \subseteq X$ and a scalar $\alpha > 0$, we recall that $S^\alpha = \{x \in X \mid d(x, S) \leq \alpha\}$ denotes the α-offset of S.

▶ **Lemma 10** (Theorem 7.3 and following remark in [5]). *Given $\alpha > 0$, the simplification operator $S_\alpha : \mathsf{Reeb} \to \mathsf{Reeb}$ takes any Reeb graph R_h to $\text{R}_{h'} = S_\alpha(\text{R}_h)$ such that $\text{Dg}(h') \cap \Delta^{\alpha/2} = \emptyset$ and $d_{\text{B}}(\text{R}_h, \text{R}_{h'}) \leq 2\, d_{\text{FD}}(\text{R}_h, \text{R}_{h'}) \leq 4\alpha$.*

▶ **Lemma 11** (Theorem 2.5 and Lemma 4.3 in [12]). *Given $a \leq b$, the merge operator $\text{Merge}_{a,b} : \mathsf{Reeb} \to \mathsf{Reeb}$ takes any Reeb graph R_h to $\text{R}_{h'} = \text{Merge}_{a,b}(\text{R}_h)$ such that $\text{Dg}(h')$ is obtained from $\text{Dg}(h)$ through the following snapping principle (see Figure 2 for an illustration):*

$$(x, y) \in \text{Dg}(h) \mapsto (x', y') \in \text{Dg}(h') \text{ where } x' = \begin{cases} x \text{ if } x \notin [a, b] \\ \frac{a+b}{2} \text{ otherwise} \end{cases} \text{ and similarly for } y'.$$

▶ **Definition 12.** Let R_f be a fixed Reeb graph with critical values $\{a_1, \cdots, a_n\}$. Given $\alpha > 0$, the full transformation $F_\alpha : \mathsf{Reeb} \to \mathsf{Reeb}$ is defined as $F_\alpha = \text{Merge}_{9\alpha} \circ S_{2\alpha}$, where $\text{Merge}_{9\alpha} = \text{Merge}_{a_n - 9\alpha,\, a_n + 9\alpha} \circ \ldots \circ \text{Merge}_{a_1 - 9\alpha,\, a_1 + 9\alpha}$. See Figure 3 for an illustration.

3.1.2 Properties of the transformed graph

Let $\text{R}_f, \text{R}_g \in \mathsf{Reeb}$ such that $d_{\text{B}}(\text{R}_f, \text{R}_g) < K\varepsilon$ where $\varepsilon = d_{\text{FD}}(\text{R}_f, \text{R}_g) < a_f/(8(1+22K))$. Letting $\text{R}_{g'} = F_{K\varepsilon}(\text{R}_g)$, we want to show both that $d_{\text{FD}}(\text{R}_g, \text{R}_{g'}) < 22K\varepsilon < \varepsilon$ and $d_{\text{FD}}(\text{R}_f, \text{R}_{g'}) = 0$, which will lead to a contradiction as mentioned previously.

Let $B_\infty(\cdot, \cdot)$ denote balls in the ℓ_∞-norm.

[1] Strictly speaking, the output of our Merge is the Reeb graph of the output of the Merge from [12].

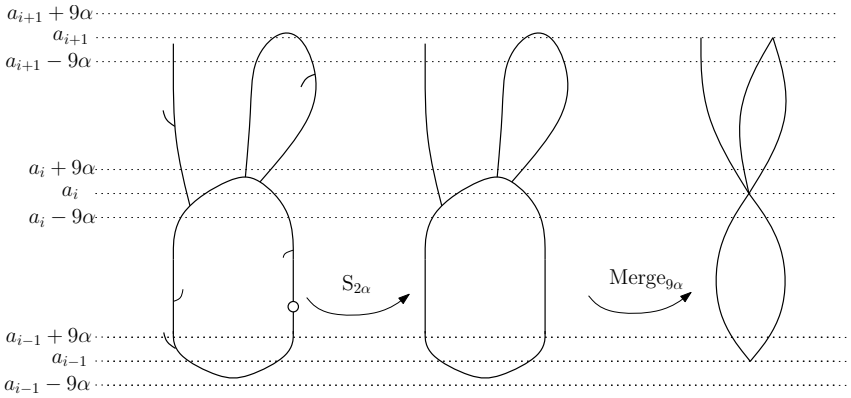


Figure 3 Illustration of F_α.

▶ **Lemma 13.** *Let* $\mathrm{R}_h = S_{2K\varepsilon}(\mathrm{R}_g)$. *Under the above assumptions, one has*

$$\mathrm{Dg}(h) \subseteq \bigcup\nolimits_{\tau\in\mathrm{Dg}(f)} B_\infty(\tau, 9K\varepsilon). \tag{5}$$

Proof. Since $d_{\mathrm{B}}(\mathrm{R}_f, \mathrm{R}_g) < K\varepsilon$, we have $\mathrm{Dg}(g) \subseteq \bigcup_{\tau\in\mathrm{Dg}(f)} B_\infty(\tau, K\varepsilon) \cup \Delta^{K\varepsilon}$. Since $\mathrm{R}_h = S_{2K\varepsilon}(\mathrm{R}_g)$, it follows from Lemma 10 that $d_{\mathrm{B}}(\mathrm{Dg}(h), \mathrm{Dg}(g)) \leq 8K\varepsilon$. Moreover, since every persistence pair in $\mathrm{Dg}(g) \cap \Delta^{K\varepsilon}$ is removed by $S_{2K\varepsilon}$, it results that:
$\mathrm{Dg}(h) \subseteq \bigcup_{\tau\in\mathrm{Dg}(g)\setminus\Delta^{K\varepsilon}} B_\infty(\tau, 8K\varepsilon) \subseteq \bigcup_{\tau\in\mathrm{Dg}(f)} B_\infty(\tau, 9K\varepsilon)$. ◀

Now we bound $d_{\mathrm{FD}}(\mathrm{R}_{g'}, \mathrm{R}_g)$. Recall that, given an arbitrary Reeb graph R_h, with critical values $\mathrm{Crit}(h) = \{c_1, ..., c_p\}$, if C is a cc of $h^{-1}(I)$, where I is an open interval such that $\exists c_i, c_{i+1}$ s.t. $I \subseteq (c_i, c_{i+1})$, then C is a *topological arc*, i.e. homeomorphic to an open interval.

▶ **Proposition 14.** *Under the same assumptions as above, one has* $d_{\mathrm{FD}}(\mathrm{R}_g, \mathrm{R}_{g'}) < 22K\epsilon$.

Proof. Let $\mathrm{R}_h = S_{2K\varepsilon}(\mathrm{R}_g)$. We have $d_{\mathrm{FD}}(\mathrm{R}_{g'}, \mathrm{R}_g) \leq d_{\mathrm{FD}}(\mathrm{R}_{g'}, \mathrm{R}_h) + d_{\mathrm{FD}}(\mathrm{R}_h, \mathrm{R}_g)$ by the triangle inequality. It suffices therefore to bound both $d_{\mathrm{FD}}(\mathrm{R}_{g'}, \mathrm{R}_h)$ and $d_{\mathrm{FD}}(\mathrm{R}_h, \mathrm{R}_g)$. By Lemma 10, we have $d_{\mathrm{FD}}(\mathrm{R}_h, \mathrm{R}_g) < 4K\varepsilon$. Now, recall from (5) that the points of the extended persistence diagram of R_h are included in $\bigcup_{\tau\in\mathrm{Dg}(f)} B_\infty(\tau, 9K\varepsilon)$. Moreover, since $\mathrm{R}_{g'} = \mathrm{Merge}_{9K\varepsilon}(\mathrm{R}_h)$, $\mathrm{R}_{g'}$ and R_h are composed of the same number of arcs in each $[a_i + 9K\varepsilon, a_{i+1} - 9K\varepsilon]$. Hence, we can define explicit continuous maps $\phi : \mathrm{R}_h \to \mathrm{R}_{g'}$ and $\psi : \mathrm{R}_{g'} \to \mathrm{R}_h$ as depicted in Figure 4. More precisely, since R_h and $\mathrm{R}_{g'}$ are composed of the same number of arcs in each $[a_i + 9K\varepsilon, a_{i+1} - 9K\varepsilon]$, we only need to specify ϕ and ψ inside each interval $(a_i - 9K\varepsilon, a_i + 9K\varepsilon)$ and then ensure that the piecewise-defined maps are assembled consistently. Since the critical values of R_h are within distance less that $9K\varepsilon$ of the critical values of f, there exist two levels $a_i - 9K\varepsilon < \alpha_i \leq \beta_i < a_i + 9K\varepsilon$ such that R_h is only composed of arcs in $(a_i - 9K\varepsilon, \alpha_i]$ and $[\beta_i, a_i + 9K\varepsilon)$ for each i (dashed lines in Figure 4). For any cc C of $h^{-1}((a_i - 9K\varepsilon, a_i + 9K\varepsilon))$, the map ϕ sends all points of $C \cap h^{-1}([\alpha_i, \beta_i])$ to the corresponding critical point y_C created by the Merge in $\mathrm{R}_{g'}$, and it extends the arcs of $C \cap h^{-1}((a_i - 9K\varepsilon, \alpha_i])$ (resp. $C \cap h^{-1}([\beta_i, a_i + 9K\varepsilon))$) into arcs of $(g')^{-1}([a_i - 9K\varepsilon, a_i])$ (resp. $(g')^{-1}([a_i, a_i + 9K\varepsilon])$). In return, the map ψ sends the critical point y_C to an arbitrary point of C. Then, since the Merge operation preserves connected components, for each arc A' of $(g')^{-1}((a_i - 9K\varepsilon, a_i + 9K\varepsilon))$ connected to y_C, there is at least one corresponding path A in R_h whose endpoint in $h^{-1}(a_i - 9K\varepsilon)$ or $h^{-1}(a_i + 9K\varepsilon)$

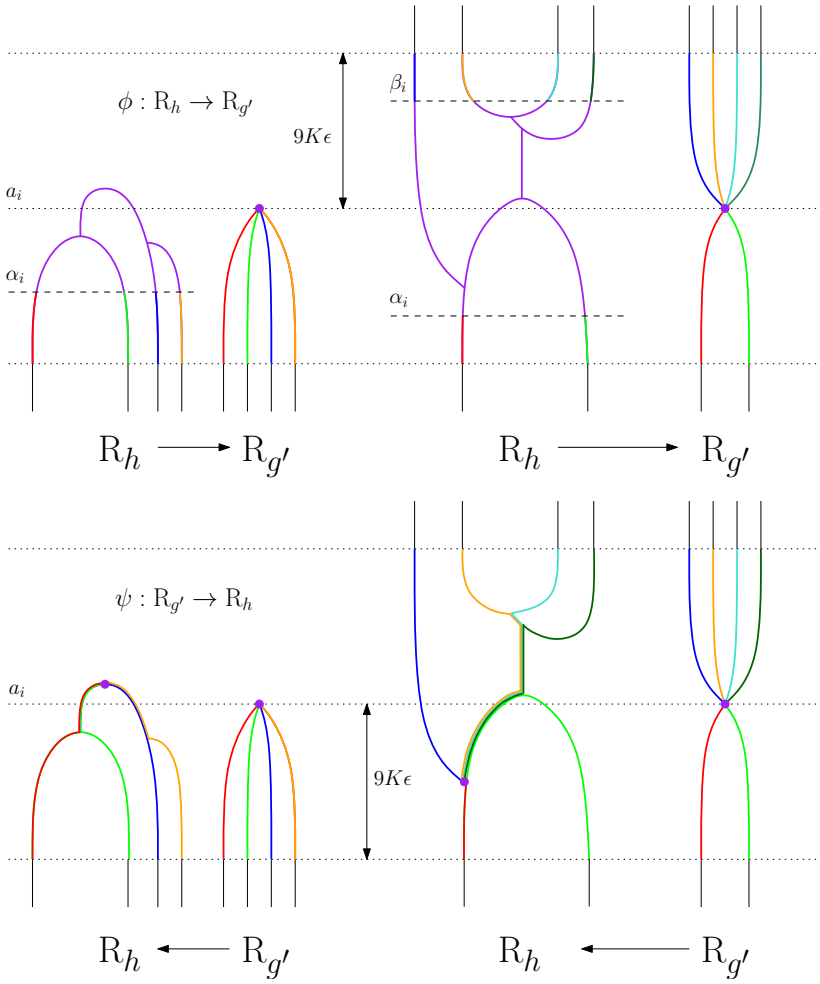


Figure 4 The effects of ϕ and ψ around a specific critical value a_i of f. Segments are matched according to their colors (up to reparameterization).

matches with the one of A' (see the colors in Figure 4). Hence ψ sends A' to A and the piecewise-defined maps are assembled consistently.

Let us bound the three terms in the $\max\{\cdots\}$ in (4) with this choice of maps ϕ, ψ:

- We first bound $\|g' - h \circ \psi\|_\infty$. Let $x \in \mathrm{R}_{g'}$. Either $g'(x) \in \bigcup_{i\in\{1,\dots,n-1\}}[a_i + 9K\varepsilon, a_{i+1} - 9K\varepsilon]$, and in this case we have $g'(x) = h(\psi(x))$ by definition of ψ; or, there is $i_0 \in \{1, ..., n\}$ such that $g'(x) \in (a_{i_0} - 9K\varepsilon, a_{i_0} + 9K\varepsilon)$ and then $h(\psi(x)) \in (a_{i_0} - 9K\varepsilon, a_{i_0} + 9K\varepsilon)$. In both cases $|g'(x) - h \circ \psi(x)| < 18K\varepsilon$. Hence, $\|g' - h \circ \psi\|_\infty < 18K\varepsilon$.
- Since the previous proof is symmetric in h and g', one also has $\|h - g' \circ \phi\|_\infty < 18K\varepsilon$.
- We now bound $D(\phi, \psi)$. Let $(x, \phi(x)), (\psi(y), y) \in C(\phi, \psi)$ (the cases $(x, \phi(x)), (x', \phi(x'))$ and $(\psi(y), y), (\psi(y'), y')$ are similar). Let $\pi_{g'} : [0, 1] \to \mathrm{R}_{g'}$ be a continuous path from $\phi(x)$ to y which achieves $d_{g'}(\phi(x), y)$.
  - Assume $h(x) \in \bigcup_{i\in\{1,\dots,n-1\}}[a_i + 9K\varepsilon, a_{i+1} - 9K\varepsilon]$. Then one has $\psi \circ \phi(x) = x$. Hence, $\pi_h := \psi \circ \pi_{g'}$ is a valid path from x to $\psi(y)$. Moreover, since $\|g' - h \circ \psi\|_\infty < 18K\varepsilon$, it follows that

$$\begin{aligned} &\max\ \mathrm{im}(h \circ \pi_h) < \max\ \mathrm{im}(g' \circ \pi_{g'}) + 18K\varepsilon, \\ &\min\ \mathrm{im}(h \circ \pi_h) > \min\ \mathrm{im}(g' \circ \pi_{g'}) - 18K\varepsilon. \end{aligned} \tag{6}$$

Hence, one has

$$d_h(x,\psi(y)) \leq \max\ \mathrm{im}(h\circ\pi_h) - \min\ \mathrm{im}(h\circ\pi_h) < d_{g'}(\phi(x),y) + 36K\varepsilon,$$
$$-d_h(x,\psi(y)) \geq \min\ \mathrm{im}(h\circ\pi_h) - \max\ \mathrm{im}(h\circ\pi_h) > -d_{g'}(\phi(x),y) - 36K\varepsilon.$$

This shows that $|d_h(x,\psi(y)) - d_{g'}(\phi(x),y)| < 36K\varepsilon$.

- Assume that there is $i_0 \in \{1,...,n\}$ such that $h(x) \in (a_{i_0} - 9K\varepsilon, a_{i_0} + 9K\varepsilon)$. Then, by definition of ϕ,ψ, we have $g'(\phi(x)) \in (a_{i_0} - 9K\varepsilon, a_{i_0} + 9K\varepsilon)$, and, since ϕ and ψ preserve connected components, there is a path $\pi'_h : [0,1] \to \mathrm{R}_h$ from x to $\psi\circ\phi(x)$ within the interval $(a_{i_0} - 9K\varepsilon, a_{i_0} + 9K\varepsilon)$, which itself is included in the interior of the offset $\mathrm{im}(g'\circ\pi_{g'})^{18K\varepsilon}$. Let now π_h be the concatenation of π'_h with $\psi\circ\pi_{g'}$, which goes from x to $\psi(y)$. Since $\|g' - h\circ\psi\| < 18K\varepsilon$, it follows that $\mathrm{im}(h\circ\psi\circ\pi_{g'}) \subseteq \mathrm{int}\ \mathrm{im}(g'\circ\pi_{g'})^{18K\varepsilon}$, and since $\mathrm{im}(h\circ\pi_h) = \mathrm{im}(h\circ\pi'_h) \cup \mathrm{im}(h\circ\psi\circ\pi_{g'})$ by concatenation, one finally has $\mathrm{im}(h\circ\pi_h) \subseteq \mathrm{int}\ \mathrm{im}(g'\circ\pi_{g'})^{18K\varepsilon}$. Hence, the inequalities of (6) hold, implying that $|d_h(x,\psi(y)) - d_{g'}(\phi(x),y)| < 36K\varepsilon$.

Since these inequalities hold for any couples $(x,\phi(x))$ and $(\psi(y),y)$, we deduce that $D(\phi,\psi) \leq 36K\varepsilon$.

Thus, $d_{\mathrm{FD}}(\mathrm{R}_h,\mathrm{R}_g) < 4K\varepsilon$ and $d_{\mathrm{FD}}(\mathrm{R}_h,\mathrm{R}_{g'}) \leq 18K\varepsilon$, so $d_{\mathrm{FD}}(\mathrm{R}_{g'},\mathrm{R}_g) < 22K\varepsilon$ as desired. ◀

Now we show that $\mathrm{R}_{g'}$ is isomorphic to R_f (i.e. it lies at functional distortion distance 0).

▶ **Proposition 15.** *Under the same assumptions as above, one has* $d_{\mathrm{FD}}(\mathrm{R}_f,\mathrm{R}_{g'}) = 0$.

Proof. First, recall from (5) that the points of the extended persistence diagram of R_h are included in $\bigcup_{\tau\in\mathrm{Dg}(f)} B_\infty(\tau, 9K\varepsilon)$. Since $\mathrm{R}_{g'} = \mathrm{Merge}_{9K\varepsilon}(\mathrm{R}_h)$, it follows from Lemma 11 that $\mathrm{Crit}(g') \subseteq \mathrm{Crit}(f)$. Hence, both $\mathrm{R}_{g'}$ and R_f are composed of arcs in each (a_i,a_{i+1}).

Now, we show that, for each i, the number of arcs of $(g')^{-1}((a_i,a_{i+1}))$ and $f^{-1}((a_i,a_{i+1}))$ are the same. By the triangle inequality and Proposition 14, we have:

$$d_{\mathrm{FD}}(\mathrm{R}_f,\mathrm{R}_{g'}) \leq d_{\mathrm{FD}}(\mathrm{R}_f,\mathrm{R}_g) + d_{\mathrm{FD}}(\mathrm{R}_g,\mathrm{R}_{g'}) < (1+22K)\varepsilon. \tag{7}$$

Let $\phi : \mathrm{R}_f \to \mathrm{R}_{g'}$ and $\psi : \mathrm{R}_{g'} \to \mathrm{R}_f$ be optimal continuous maps that achieve $d_{\mathrm{FD}}(\mathrm{R}_f,\mathrm{R}_{g'})$. Let $i \in \{1,...,n-1\}$. Assume that there are more arcs of $f^{-1}((a_i,a_{i+1}))$ than arcs of $(g')^{-1}((a_i,a_{i+1}))$. For every arc A of $f^{-1}((a_i,a_{i+1}))$, let $x_A \in A$ such that $f(x_A) = \bar{a} = \frac{1}{2}(a_i + a_{i+1})$. First, note that $\phi(x_A)$ must belong to an arc of $(g')^{-1}((a_i,a_{i+1}))$. Indeed, since $\|f - g'\circ\phi\|_\infty < (1+22K\varepsilon)$, one has $g'(\phi(x_A)) \in (\bar{a} - (1+22K)\varepsilon, \bar{a} + (1+22K)\varepsilon) \subseteq (a_i,a_{i+1})$. Then, according to the pigeonhole principle, there exist $x_A, x_{A'}$ such that $\phi(x_A)$ and $\phi(x_{A'})$ belong to the same arc of $(g')^{-1}((a_i,a_{i+1}))$.

- Since x_A and $x_{A'}$ do not belong to the same arc, we have $d_f(x_A,x_{A'}) > a_f/2$.
- Now, since $\|f - g'\circ\phi\|_\infty < (1+22K)\varepsilon$ and $\phi(x_A),\phi(x_{A'})$ belong to the same arc of $(g')^{-1}((a_i,a_{i+1}))$, we also have $d_{g'}(\phi(x_A),\phi(x_{A'})) < 2(1+22K)\varepsilon$ (see Figure 5).

Hence, $D(\phi,\psi) \geq |d_f(x_A,x_{A'}) - d_{g'}(\phi(x_A),\phi(x_{A'}))| > a_f/2 - 2(1+22K)\varepsilon$, which is greater than $2(1+22K)\varepsilon$ because $\varepsilon < a_f/(8(1+22K))$. Thus, $d_{\mathrm{FD}}(\mathrm{R}_f,\mathrm{R}_{g'}) > (1+22K)\varepsilon$, which leads to a contradiction with (7). This means that there cannot be more arcs in $f^{-1}((a_i,a_{i+1}))$ than in $(g')^{-1}((a_i,a_{i+1}))$. Since the proof is symmetric in f and g', the numbers of arcs in $(g')^{-1}((a_i,a_{i+1}))$ and in $f^{-1}((a_i,a_{i+1}))$ are actually the same.

Finally, we show that the attaching maps of these arcs are also the same. In this particular graph setting, this is equivalent to showing that corresponding arcs in R_f and $\mathrm{R}_{g'}$ have the same endpoints. Let a_i be a critical value. Let $A^-_{f,i}$ and $A^+_{f,i}$ (resp. $A^-_{g',i}$ and $A^+_{g',i}$) be the sets of arcs in $f^{-1}((a_{i-1},a_i))$ and $f^{-1}((a_i,a_{i+1}))$ (resp. $(g')^{-1}((a_{i-1},a_i))$ and $(g')^{-1}((a_i,a_{i+1}))$).

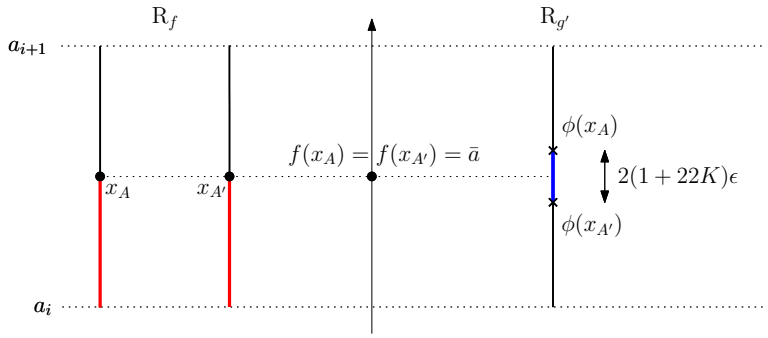


Figure 5 Any path between x_A and $x_{A'}$ must contain the red segments, and the blue segment is a particular path between $\phi(x_A)$ and $\phi(x_{A'})$.

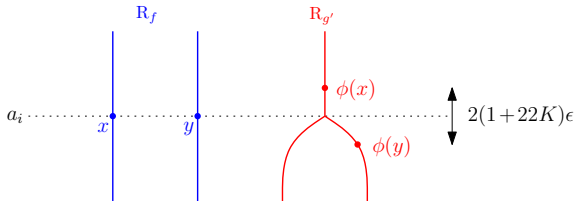


Figure 6 Any path from x to y must go through an entire arc, hence $d_f(x,y) \geq a_f$. On the contrary, there exists a direct path between $\phi(x)$ and $\phi(y)$, hence $d_{g'}(\phi(x),\phi(y)) < 2(1+22K)\varepsilon$.

Morevover, we let ζ^i_f and ξ^i_f (resp. $\zeta^i_{g'}$ and $\xi^i_{g'}$) be the corresponding attaching maps that send arcs to their endpoints in $f^{-1}(a_i)$ (resp. $(g')^{-1}(a_i)$). Let $A, B \in A^-_{f,i}$. We define an equivalence relation $\sim_{f,i}$ between A and B by: $A \sim_{f,i} B$ iff $\zeta^i_f(A) = \zeta^i_f(B)$, i.e. the endpoints of the arcs in the critical slice $f^{-1}(a_i)$ are the same. Similarly, $C, D \in A^+_{f,i}$ are equivalent if and only if $\xi^i_f(C) = \xi^i_f(D)$. One can define $\sim_{g',i}$ in the same way. To show that the attaching maps of R_f and $\mathrm{R}_{g'}$ are the same, we need to find a bijection b between the arcs of R_f and $\mathrm{R}_{g'}$ such that $A \sim_{f,i} B \Leftrightarrow b(A) \sim_{g',i} b(B)$ for each i.

We will now define b then check that it satisfies the condition. Recall from (7) that $d_{\mathrm{FD}}(\mathrm{R}_f, \mathrm{R}_{g'}) < (1+22K)\varepsilon$. Hence there exists a continuous map $\phi : \mathrm{R}_f \to \mathrm{R}_{g'}$ such that $\|f - g' \circ \phi\|_\infty < (1+22K)\varepsilon$. This map induces a bijection b between the arcs of R_f and $\mathrm{R}_{g'}$. Indeed, given an arc $A \in A^-_{f,i}$, let $x \in A$ such that $f(x) = \bar{a} = \frac{1}{2}(a_{i-1} + a_i)$. We define $b(A)$ as the arc of $A^-_{g,i}$ that contains $\phi(x)$. The map b is well-defined since $g' \circ \phi(x) \in [\bar{a} - (1+22K)\varepsilon, \bar{a} + (1+22K)\varepsilon] \subseteq (a_{i-1}, a_i)$, hence $\phi(x)$ must belong to an arc of $(g')^{-1}((a_{i-1}, a_i))$. Let us show that $b(A) \sim_{g',i} b(B) \Rightarrow A \sim_{f,i} B$. Assume there exist $A, B \in A^-_{f,i}$ (the treatment of $A, B \in A^+_{f,i}$ is similar) such that $A \not\sim_{f,i} B$ and $b(A) \sim_{g',i} b(B)$. Let $x = \zeta^i_f(A)$ and $y = \zeta^i_f(B)$. Then we have $d_f(x,y) \geq a_f$ while $d_{g'}(\phi(x),\phi(y)) < 2(1+22K)\varepsilon$ (see Figure 6). Hence $|d_f(x,y) - d_{g'}(\phi(x),\phi(y))| > a_f - 2(1+22K)\varepsilon > 2(1+22K)\varepsilon$, so $d_{\mathrm{FD}}(\mathrm{R}_f, \mathrm{R}_{g'}) > (1+22K)\varepsilon$, which leads to a contradiction with (7). The same argument applies to show that $A \sim_{f,i} B \Rightarrow b(A) \sim_{g',i} b(B)$. ◀

4 Induced Intrinsic Metrics

In this section we leverage the local equivalence given by Theorem 9 to derive a global equivalence between the intrinsic metrics $\hat{d}_{\mathrm{B}}$ and $\hat{d}_{\mathrm{FD}}$ induced by d_{B} and d_{FD}. Note that we already know $\hat{d}_{\mathrm{FD}}$ to be equivalent to $\hat{d}_{\mathrm{GH}}$ and $\hat{d}_{\mathrm{I}}$ since d_{FD} is equivalent to d_{GH} and d_{I}. To

the best of our knowledge, the question whether d_{FD}, d_{I} or d_{GH} is intrinsic on the space of Reeb graphs has not been settled, although d_{GH} itself is known to be intrinsic on the larger space of compact metric spaces – see e.g. [19].

Convention. In the following, whatever the metric $d : \mathsf{Reeb} \times \mathsf{Reeb} \to \mathbb{R}_+$ under consideration, we define the class of *admissible paths* in Reeb to be those maps $\gamma : [0,1] \to \mathsf{Reeb}$ that are continuous in d_{FD}. This makes sense when d is either d_{FD} itself, d_{GH}, or d_{I}, all of which are equivalent to d_{FD} and therefore have the same continuous maps $\gamma : [0,1] \to \mathsf{Reeb}$. In the case $d = d_{\mathrm{B}}$ our convention means restricting the class of admissible paths to a strict subset of the maps $\gamma : [0,1] \to \mathsf{Reeb}$ that are continuous in d (by Theorem 8), which is required by some of our following claims.

▶ **Definition 16.** Let $d : \mathsf{Reeb} \times \mathsf{Reeb} \to \mathbb{R}_+$ be a metric on Reeb. Let $\mathrm{R}_f, \mathrm{R}_g \in \mathsf{Reeb}$, and $\gamma : [0,1] \to \mathsf{Reeb}$ be an admissible path such that $\gamma(0) = \mathrm{R}_f$ and $\gamma(1) = \mathrm{R}_g$. The *length* of γ induced by d is defined as $L_d(\gamma) = \sup_{n,\Sigma} \sum_{i=0}^{n-1} d(\gamma(t_i), \gamma(t_{i+1}))$ where n ranges over $\mathbb{N}$ and Σ ranges over all partitions $0 = t_0 \leq t_1 \leq ... \leq t_n = 1$ of $[0,1]$. The *intrinsic metric induced by* d, denoted $\hat{d}$, is defined by $\hat{d}(\mathrm{R}_f, \mathrm{R}_g) = \inf_\gamma \mathrm{L_d}(\gamma)$ where γ ranges over all admissible paths $\gamma : [0,1] \to \mathsf{Reeb}$ such that $\gamma(0) = \mathrm{R}_f$ and $\gamma(1) = \mathrm{R}_g$.

The following result is, in our view, the starting point for the study of intrinsic metrics over the space of Reeb graphs. It comes as a consequence of the (local or global) equivalences between d_{B} and d_{FD} stated in Theorems 8 and 9. The intuition is that integrating two locally equivalent metrics along the same path using sufficiently small integration steps yields the same total length up to a constant factor, hence the global equivalence between the induced intrinsic metrics[2].

▶ **Theorem 17.** *$\hat{d}_{\mathrm{B}}$ and $\hat{d}_{\mathrm{FD}}$ are globally equivalent. Specifically, for any $\mathrm{R}_f, \mathrm{R}_g \in \mathsf{Reeb}$,*

$$\hat{d}_{\mathrm{FD}}(\mathrm{R}_f, \mathrm{R}_g)/22 \leq \hat{d}_{\mathrm{B}}(\mathrm{R}_f, \mathrm{R}_g) \leq 2\,\hat{d}_{\mathrm{FD}}(\mathrm{R}_f, \mathrm{R}_g). \tag{8}$$

Proof. We first show that $\hat{d}_{\mathrm{B}}(\mathrm{R}_f, \mathrm{R}_g) \leq 2\,\hat{d}_{\mathrm{FD}}(\mathrm{R}_f, \mathrm{R}_g)$. Let γ be an admissible path and let $\Sigma = \{t_0, ..., t_n\}$ be a partition of $[0,1]$. Then, by Theorem 8,

$$\sum_{i=0}^{n-1} d_{\mathrm{FD}}(\gamma(t_i), \gamma(t_{i+1})) \geq \frac{1}{2} \sum_{i=0}^{n-1} d_{\mathrm{B}}(\gamma(t_i), \gamma(t_{i+1})).$$

Since this is true for any partition Σ of any finite size n, it follows that

$$L_{d_{\mathrm{FD}}}(\gamma) \geq \frac{1}{2} L_{d_{\mathrm{B}}}(\gamma) \geq \frac{1}{2} \hat{d}_{\mathrm{B}}(\mathrm{R}_f, \mathrm{R}_g).$$

Again, this inequality holds for any admissible path γ, so $\hat{d}_{\mathrm{B}}(\mathrm{R}_f, \mathrm{R}_g) \leq 2\hat{d}_{\mathrm{FD}}(\mathrm{R}_f, \mathrm{R}_g)$.
We now show that $\hat{d}_{\mathrm{FD}}(\mathrm{R}_f, \mathrm{R}_g)/22 \leq \hat{d}_{\mathrm{B}}(\mathrm{R}_f, \mathrm{R}_g)$. Let γ be an admissible path and $\Sigma = \{t_0, ..., t_n\}$ a partition of $[0,1]$. We claim that there is a refinement of Σ (i.e. a partition $\Sigma' = \{t'_0, ..., t'_m\} \supseteq \Sigma$ for some $m \geq n$) such that $d_{\mathrm{FD}}(\gamma(t'_j), \gamma(t'_{j+1})) < \max\{c_{t'_j}, c_{t'_{j+1}}\}/16$ for all $j \in \{0, ..., m-1\}$, where $c_t > 0$ denotes the minimal distance between consecutive critical values of $\gamma(t)$. Indeed, since γ is continuous in d_{FD}, for any $t \in [0,1]$ there exists $\delta_t > 0$ such that $d_{\mathrm{FD}}(\gamma(t), \gamma(t')) < c_t/16$ for all $t' \in [0,1]$ with $|t - t'| < \delta_t$. Consider the open cover $\{(\max\{0, t - \delta_t/2\}, \min\{1, t + \delta_t/2\})\}_{t \in [0,1]}$ of $[0,1]$. Since $[0,1]$ is compact, there

[2] Provided the induced metrics are defined using the same class of admissible paths, hence our convention.

exists a finite subcover containing all the intervals $(t_i - \delta_{t_i}/2, t_i + \delta_{t_i}/2)$ for $t_i \in \Sigma$. Assume w.l.o.g. that this subcover is minimal (if it is not, then reduce the δ_{t_i} as much as needed). Let then $\Sigma' = \{t'_0, ..., t'_m\} \supseteq \Sigma$ be the partition of $[0,1]$ given by the midpoints of the intervals in this subcover, sorted by increasing order. Since the subcover is minimal, we have $t'_{j+1} - t'_j < (\delta_{t'_j} + \delta_{t'_{j+1}})/2 < \max\{\delta_{t'_j}, \delta_{t'_{j+1}}\}$ hence $d_{\mathrm{FD}}(\gamma(t'_j), \gamma(t'_{j+1})) < \max\{c_{t'_j}, c_{t'_{j+1}}\}/16$ for each $j \in \{0, m-1\}$. It follows that

$$\begin{aligned}\sum_{i=0}^{n-1} d_{\mathrm{FD}}(\gamma(t_i), \gamma(t_{i+1})) &\leq \sum_{j=0}^{m-1} d_{\mathrm{FD}}(\gamma(t'_j), \gamma(t'_{j+1})) \text{ by the triangle inequality since } \Sigma' \supseteq \Sigma \\ &\leq 22 \sum_{j=0}^{m-1} d_{\mathrm{B}}(\gamma(t'_j), \gamma(t'_{j+1})) \text{ by Theorem 9 with } K = 1/22 \\ &\leq 22\, L_{d_{\mathrm{B}}}(\gamma).\end{aligned}$$

Since this is true for any partition Σ of any finite size n, it follows that

$$\hat{d}_{\mathrm{FD}}(\mathrm{R}_f, \mathrm{R}_g) \leq L_{d_{\mathrm{FD}}}(\gamma) \leq 22\, L_{d_{\mathrm{B}}}(\gamma).$$

Again, this inequality is true for any admissible path γ, so $\hat{d}_{\mathrm{FD}}(\mathrm{R}_f, \mathrm{R}_g) \leq 22\, \hat{d}_{\mathrm{B}}(\mathrm{R}_f, \mathrm{R}_g)$. ◀

Theorem 17 implies in particular that $\hat{d}_{\mathrm{B}}$ is a true metric on Reeb graphs, as opposed to d_{B} which is only a pseudo-metric. Moreover, the simplification operator defined in Section 3.1.1 makes it possible to continuously deform any Reeb graph into a trivial segment-shaped graph then into the empty graph. This shows that Reeb is path-connected in d_{FD}. Since the length of such continuous deformations is finite if the Reeb graph is finite, $\hat{d}_{\mathrm{FD}}$ and $\hat{d}_{\mathrm{B}}$ are finite metrics. Finally, the global equivalence of $\hat{d}_{\mathrm{FD}}$ and $\hat{d}_{\mathrm{B}}$ yields the following:

▶ **Corollary 18.** *The metrics $\hat{d}_{\mathrm{FD}}$ and $\hat{d}_{\mathrm{B}}$ induce the same topology on* Reeb*, which is a refinement of the ones induced by d_{FD} or d_{B}.*

▶ Remark. Note that the first inequality in (8) and, consequently, Corollary 18, are wrong if one defines the admissible paths for $\hat{d}_{\mathrm{B}}$ to be the whole class of maps $[0,1] \to$ Reeb that are continuous in d_{B} – hence our convention. For instance, let us consider the two Reeb graphs R_f and R_g of Figure 1 such that $\mathrm{Dg}(f) = \mathrm{Dg}(g)$, and let us define $\gamma : [0,1] \to$ Reeb by $\gamma(t) = \mathrm{R}_f$ if $t \in [0, 1/2)$ and $\gamma(t) = \mathrm{R}_g$ if $t \in [1/2, 1]$. Then γ is continuous in d_{B} while it is not in d_{FD} at $1/2$ since $d_{\mathrm{FD}}(\mathrm{R}_f, \mathrm{R}_g) > 0$. In this case, $\hat{d}_{\mathrm{B}}(\mathrm{R}_f, \mathrm{R}_g) \leq L_{d_{\mathrm{B}}}(\gamma) = 0 < \hat{d}_{\mathrm{FD}}(\mathrm{R}_f, \mathrm{R}_g)$.

5 Discussion

In this article, we proved that the bottleneck distance, even though it is only a pseudo-metric on Reeb graphs, can actually discriminate a Reeb graph from the other Reeb graphs in a small enough neighborhood, as efficiently as the other metrics do. This theoretical result legitimates the use of the bottleneck distance to discriminate between Reeb graphs in applications. It also motivates the study of intrinsic metrics, which can potentially shed new light on the structure of the space of Reeb graphs and open the door to new applications where interpolation plays a key part. This work has raised numerous questions, some of which we plan to investigate in the upcoming months:

- **Can the lower bound be improved?** We believe that $\varepsilon/22$ is not optimal. Specifically, a more careful analysis of the simplification operator should allow us to derive a tighter upper bound than the one in Lemma 10, and improve the current lower bound on d_{B}.

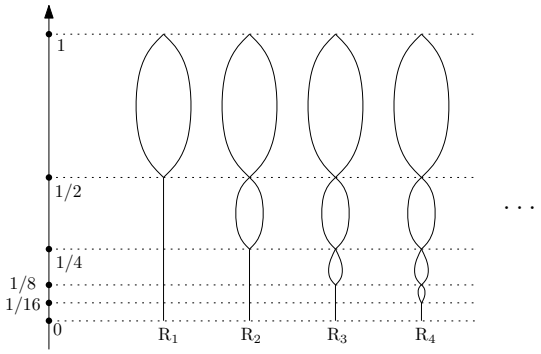


Figure 7 A sequence of Reeb graphs that is Cauchy but that does not converge in Reeb because the number of critical values goes to $+\infty$. Indeed, each R_n has $n+2$ critical values.

- **Do shortest paths exist in Reeb?** The existence of shortest paths achieving $\hat{d}_{\mathrm{B}}$ is an important question since a positive answer would enable us to define and study the *intrinsic curvature* of Reeb. Moreover, characterizing and computing these shortest paths would be useful for interpolating between Reeb graphs. The existence of shortest paths is guaranteed if the space is complete and locally compact. Note that Reeb is not complete, as shown by the counter-example of Figure 7. Hence, we plan to restrict the focus to the subspace of Reeb graphs having at most N features with height at most H, for fixed but arbitrary $N, H > 0$. We believe this subspace is complete and locally compact, like its counterpart in the space of persistence diagrams [9].
- **Is Reeb an Alexandrov space?** Provided shortest paths exist in Reeb (or in some subspace thereof), we plan to determine whether the intrinsic curvature is bounded, either from above or from below. This is interesting because barycenters in metric spaces with bounded curvature enjoy many useful properties [25], and they can be approximated effectively [24].
- **Can the local equivalence be extended to general metric spaces?** We have reasons to believe that our local equivalence result can be used to prove similar results for more general classes of metric spaces than Reeb graphs. If true, this would shed new light on inverse problems in persistence theory.

References

1. P. Agarwal, K. Fox, A. Nath, A. Sidiropoulos, and Y. Wang. Computing the Gromov-Hausdorff Distance for Metric Trees. In *Symp. Algo. Comput.*, 2015.
2. M. Alagappan. From 5 to 13: Redefining the Positions in Basketball. MIT Sloan Sports Analytics Conference, 2012.
3. V. Barra and S. Biasotti. 3D Shape Retrieval and Classification using Multiple Kernel Learning on Extended Reeb graphs. *The Visual Computer*, 30(11):1247–1259, 2014.
4. U. Bauer, X. Ge, and Y. Wang. Measuring Distance between Reeb Graphs. In *Symp. Comput. Geom.*, pages 464–473, 2014.
5. U. Bauer, X. Ge, and Y. Wang. Measuring Distance between Reeb Graphs (v2). *CoRR*, abs/1307.2839v2, 2016.
6. U. Bauer, E. Munch, and Y. Wang. Strong Equivalence of the Interleaving and Functional Distortion Metrics for Reeb Graphs. In *Symp. Comput. Geom.*, 2015.
7. S. Biasotti, D. Giorgi, M. Spagnuolo, and B. Falcidieno. Reeb Graphs for Shape Analysis and Applications. *Theo. Comp. Sci.*, 392(1-3):5–22, 2008.

8 H. Bjerkevik. Stability of Higher Dimensional Interval Decomposable Persistence Modules. *CoRR*, abs/1609.02086, 2016.

9 A. Blumberg, I. Gall, M. Mandell, and M. Pancia. Robust Statistics, Hypothesis Testing, and Confidence Intervals for Persistent Homology on Metric Measure Spaces. *CoRR*, abs/1206.4581, 2012.

10 D. Burago, Y. Burago, and S. Ivanov. *A Course in Metric Geometry*, volume 33 of *Graduate Studies in Mathematics*. AMS, Providence, RI, 2001.

11 G. Carlsson, V. de Silva, and D. Morozov. Zigzag Persistent Homology and Real-valued Functions. In *Symp. Comput. Geom.*, pages 247–256, 2009.

12 M. Carrière and S. Oudot. Structure and Stability of the 1-Dimensional Mapper. In *Symp. Comput. Geom.*, volume 51, pages 1–16, 2016.

13 Frédéric Chazal, Vin de Silva, Marc Glisse, and Steve Oudot. *The Structure and Stability of Persistence Modules*. Springer, 2016.

14 D. Cohen-Steiner, H. Edelsbrunner, and J. Harer. Extending persistence using Poincaré and Lefschetz duality. *Found. Comput. Math.*, 9(1):79–103, 2009.

15 Vin de Silva, Elizabeth Munch, and Amit Patel. Categorified Reeb Graphs. *Discr. Comput. Geom.*, 55:854–906, 2016.

16 B. di Fabio and C. Landi. The Edit Distance for Reeb Graphs of Surfaces. *Discrete Computational Geometry*, 55(2):423–461, 2016.

17 M. Gameiro, Y. Hiraoka, and I. Obayashi. Continuation of Point Clouds via Persistence Diagrams. *Physica D*, 334:118–132, 2016.

18 X. Ge, I. Safa, M. Belkin, and Y. Wang. Data Skeletonization via Reeb Graphs. In *Neural Inf. Proc. Sys.*, pages 837–845, 2011.

19 Alexandr Ivanov, Nadezhda Nikolaeva, and Alexey Tuzhilin. The Gromov-Hausdorff Metric on the Space of Compact Metric Spaces is Strictly Intrinsic. *Mathematical Notes*, 100(6):947–950, 2016.

20 P. Lum, G. Singh, A. Lehman, T. Ishkanov, M. Vejdemo-Johansson, M. Alagappan, J. Carlsson, and G. Carlsson. Extracting insights from the shape of complex data using topology. *Scientific Reports*, 3(1236), 2013.

21 W. Mohamed and A. Ben Hamza. Reeb graph path dissimilarity for 3d object matching and retrieval. *The Visual Computer*, 28(3):305–318, 2012.

22 T. Mukasa, S. Nobuhara, A. Maki, and T. Matsuyama. *Finding Articulated Body in Time-Series Volume Data*, pages 395–404. Springer Berlin Heidelberg, 2006.

23 M. Nicolau, A. Levine, and G. 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 Science*, 108(17):7265–7270, 2011.

24 S. Ohta. Gradient flows on Wasserstein spaces over compact Alexandrov spaces. *American Journal Mathematics*, 131(2):475–516, 2009.

25 S. Ohta. Barycenters in Alexandrov spaces of curvature bounded below. *Advances Geometry*, 12:571–587, 2012.

26 G. Reeb. Sur les points singuliers d'une forme de pfaff complètement intégrable ou d'une fonction numérique. *CR Acad. Sci. Paris*, 222:847–849, 1946.

27 G. Singh, F. Mémoli, and G. Carlsson. Topological Methods for the Analysis of High Dimensional Data Sets and 3D Object Recognition. In *Symp. PB Graphics*, 2007.

28 J. Tierny, J.-P. Vandeborre, and M. Daoudi. Invariant High Level Reeb Graphs of 3D Polygonal Meshes. *Symp. 3D Data Proc. Vis. Trans.*, pages 105–112, 2006.

Applications of Chebyshev Polynomials to Low-Dimensional Computational Geometry

Timothy M. Chan

Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL, USA
tmc@illinois.edu

Abstract

We apply the *polynomial method* – specifically, Chebyshev polynomials – to obtain a number of new results on geometric approximation algorithms in low constant dimensions. For example, we give an algorithm for constructing ε-*kernels* (coresets for approximate width and approximate convex hull) in close to optimal time $O(n + (1/\varepsilon)^{(d-1)/2})$, up to a small near-$(1/\varepsilon)^{3/2}$ factor, for any d-dimensional n-point set. We obtain an improved data structure for Euclidean *approximate nearest neighbor search* with close to $O(n \log n + (1/\varepsilon)^{d/4} n)$ preprocessing time and $O((1/\varepsilon)^{d/4} \log n)$ query time. We obtain improved approximation algorithms for *discrete Voronoi diagrams*, *diameter*, and *bichromatic closest pair* in the L_s-metric for any even integer constant $s \geq 2$. The techniques are general and may have further applications.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases diameter, coresets, approximate nearest neighbor search, the polynomial method, streaming

Digital Object Identifier 10.4230/LIPIcs.SoCG.2017.26

1 Introduction

This paper presents new results on a number of fundamental problems in low-dimensional geometric approximation algorithms. Let P be a set of n points in d-dimensional Euclidean space where d is a constant. Let $\varepsilon > 0$ be a user-specified parameter (not necessarily a constant). As a shorthand, let $E := \lceil 1/\varepsilon \rceil$. Below, the O notation may hide factor that depends on d but not ε. The notation O^* will be used to suppress small factors of the form E^c for some constant c independent of d.

Diameter. We present a new algorithm to compute a $(1+\varepsilon)$-approximation of the *diameter* of P (the farthest pair distance) in $O^*(n + E^{d/2})$ time.

There have been a long series of prior results:

$O^*(E^{d/2} n)$ time	by Agarwal, Matoušek, and Suri [3] ('91);
$O^*(n + E^{2d})$	by Barequet and Har-Peled [14] (SODA'99);
$O^*(n + E^{3d/2})$	by combining the two algorithms [18];
$O^*(n + E^d)$	by Chan [18] (SoCG'00);
$O^*(n + E^{d/2}\sqrt{n})$	by Arya and Chan [9] (SoCG'14).

Our new result is a substantial improvement, and provides a near $E^{d/4}$-factor speedup in the case when n is near $E^{d/2}$, for example.

ε-Kernels. We obtain an algorithm to compute an *ε-kernel* of P with worst-case optimal size $O(E^{(d-1)/2})$ in $O^*(n + E^{d/2})$ expected time; ε-kernels [1] provide coresets for a variety of problems such as diameter, width, minimum enclosing cylinder, minimum bounding box, and convex hull.

This is again a substantial improvement in terms of ε-dependencies over prior results:

$O^*(n + E^{3d/2})$ time	by Agarwal, Har-Peled, and Varadarajan [1] (SODA'01/FOCS'01);
$O^*(n + E^d)$	by Chan [18] (SoCG'00);
$O^*(n + E^{d/2}\sqrt{n})$	by Arya and Chan [9] (SoCG'14).

More importantly, since the size of the ε-kernel may be $\Omega(E^{(d-1)/2})$, our result for this problem is near worst-case optimal, up to an $O^*(1)$ factor (more precisely, an $O(E^{3/2}\log^{O(1)} E)$ factor).

Bichromatic closest pair. Assuming each input point is colored red or blue, we present a new algorithm to compute a $(1+\varepsilon)$-approximation of the *bichromatic closest pair* of P in $O^*(E^{d/4}n)$ expected time.

This improves a series of prior results:

$O^*(E^d n \log n)$ time	by Arya et al. [8] (SODA'04);
$O^*(E^{d/2} n \log n)$	by Chan [22] (SoCG'97);
$O^*(E^{d/3} n)$	by Arya and Chan [9] (SoCG'14).

Approximate nearest neighbors. More generally, we can preprocess P in $O(n \log n) + O^*(E^{d/4}n)$ expected time so that $(1+\varepsilon)$-factor *approximate nearest neighbor* queries can be answered in $O^*(E^{d/4} \log n)$ query time.

This improves prior results with:

$O(n \log n)$ preprocessing time &	$O^*(E^d \log n)$ query time	by Arya et al. [8] (SODA'04);
$O^*(E^{d/2} n \log n)$	$O^*(E^{d/2} \log n)$	by Chan [22] (SoCG'97);
$O(n \log n) + O^*(E^{d/3} n)$	$O^*(E^{d/3} \log n)$	by Arya and Chan [9] (SoCG'14).

There were more previous results by Arya et al. [7, 6, 13] giving space/query-time tradeoffs. For example, one method already achieved $O^*(E^{d/4}n)$ space and $O^*(E^{d/4} \log n)$ query time, but preprocessing time has large ε-dependencies, making the method unsuitable for bichromatic closest pair, for example.

Streaming diameter. In the insertion-only streaming model, we can maintain a $(1+\varepsilon)$-approximation of the diameter with $O(E^{(d-1)/2})$ space and $O^*(E^{d/4})$ time per insertion of point.

This improves prior results with:

$O(E^{(d-1)/2})$ space &	$O^*(E^{d/2})$ time	(folklore);
$O(E^{(d-1)/2})$	$O^*(E^{d/3})$	by Arya and Chan [9] (SoCG'14).

Streaming ε-kernels. In the insertion-only streaming model, we can maintain an ε-kernel of $O(E^{(d-1)/2})$ size with $O(E^{(d-1)/2})$ space and $O^*(1)$ time per insertion of point.

This improves prior results with:

$O(E^{(d-1)/2})$ space &	$O^*(E^d \log^d n)$ time	by Agarwal, Har-Peled, and Varadarajan [1] ('01);
$O^*(E^d)$	$O(1)$	by Chan [19] (SoCG'04);
$O(E^{(d-1)/2})$	$O^*(E^{d/2})$	by Zarrabi-Zadeh [31] (ESA'08);
$O(E^{(d-1)/2})$	$O^*(E^{d/4})$	by Arya and Chan [9] (SoCG'14).

Since our result has $O^*(1)$ time, it is near optimal (up to an $O(E^{3/2}\log^{O(1)} E)$ factor).

Discrete upper envelopes and discrete Voronoi diagrams. Most of the above results are obtained by solving the following key subproblem of independent interest, called *discrete upper envelope* (introduced in [19]): the problem is to find extreme points along various uniformly spaced directions, or more precisely, find the point $p_\xi \in P$ that maximizes $p_\xi \cdot \xi$ for each $\xi \in \Xi \times \{1\}$, where Ξ is the set of all points in a uniform grid of side length δ over $[0,1]^{d-1}$. To explain the name, note that after dualization the problem corresponds to evaluating the upper envelope of a set of hyperplanes (i.e., pointwise maximum of $(d-1)$-variate linear functions) at the vertical lines through all grid points in Ξ.

In the related $(d-1)$-dimensional *discrete Voronoi diagram* problem [19] (also called the "Euclidean distance transform" [17, 27]), we want to find the nearest neighbor p'_ξ in P to each grid point $\xi \in \Xi \times \{0\}$.

In both problems, we allow approximation with an additive error of $O(\varepsilon)$, given that $P \subset [0,1]^d$. We present an algorithm with $O^*(n + E^{d/2} + F^d)$ time where $F := \lceil 1/\delta \rceil$. Since the output size is $\Theta^*(F^d)$, our algorithm is near-optimal up to $O^*(1)$ factors if $F \geq \sqrt{E}$ (indeed, in applications, the main case of interest is when $F = \sqrt{E}$). This improves prior results (assuming $F \leq E$) with:

$O^*(F^d n)$ time	(trivial);
$O^*(n + E^d)$	by Chan [19] (SoCG'04);
$O^*(\min_{k=0}^{d} F^{d-k}(n + E^k))$	by Arya and Chan [9] (SoCG'14).

The last bound is $O^*(n + E^{d/2}\sqrt{n})$ in the case $F = \sqrt{E}$. Interestingly, these prior algorithms actually solve the discrete upper envelope problem *exactly* after an initial rounding of P to a uniform grid of side length ε (in other words, error solely comes from rounding). Our new approach will make more powerful use of approximation.

Significance. For specific small constants d, our improvements may not be dramatic when factors hidden in the O^* notation are taken into account. On the other hand, for large constants d, the time bound may become impractically large as E grows (not to mention that hidden constant factors, of the form $d^{O(d)}$, may become an issue). For example, for diameter with $d = 15$ and $n = E^7$, the old bound [19] was $O(E^{13} \log E)$, the most recent previous bound [9] was $O(E^{9.5} \log E)$, and the new bound is $O(E^8 \log^{O(1)} E)$. For bichromatic closest pair with $d = 25$, the old bound [22] was $O(E^{12} n \log n)$, the most recent previous bound [9] was $O(E^{7.5} n \log E)$, and the new bound is $O(E^{5.75} n \log^{O(1)} E)$. Also, for a problem such as diameter, there are existing alternatives that do not necessarily have good worst-case running time but performs much better on "realistic input" [24].

However, we believe that more significant than the results are the techniques. Surprisingly, we bring in algebraic techniques – namely, the *polynomial method* – to tackle computational geometry problems that have traditionally been solved using geometric techniques only. Specifically, our algorithms use *Chebyshev polynomials*.

The polynomial method and Chebyshev polynomials have found applications before in theoretical computer science, numerical analysis, and other areas. Two recent lines of research are particularly relevant:

- Andoni and Nguyen [5] (SODA'12) applied the polynomial method to obtain *dynamic streaming* algorithms for the width problem. This was followed up by Chan [20] (SoCG'16) for the ε-kernel problem. Chebyshev polynomials were not used. In regards to traditional (or insertion-only streaming) algorithms, these ideas seem to give poorer results than what is already known.

- Valiant [28] (FOCS'12) applied Chebyshev polynomials to obtain faster algorithms for approximate bichromatic closest pair and offline approximate nearest neighbor search *in high dimensions*. This was later improved by Alman, Chan, and Williams [4] (FOCS'16). These ideas do not seem directly useful for low-dimensional nearest neighbor search, as the target time bounds are vastly different in low vs. high dimensions.

Interestingly, our work will combine ideas from these two research threads, along with existing geometric techniques on low-dimensional ε-kernels and approximate nearest neighbor search. The connection may be simple in hindsight (none of our ideas are original in isolation), but honestly the author did not anticipate that these threads could come together so neatly!

Although we draw on algebraic techniques, our algorithms for discrete upper envelopes and diameter are easy to understand and do not require advanced background. Our first algorithm does not even need fast Fourier transform or fast matrix multiplication, just simple arithmetic on roughly $\sqrt{E}$-bit-long numbers (our time bounds already account for the bit complexity of such operations). Section 2 giving a self-contained description of the first algorithm is about two pages long. Our second algorithm for diameter, which uses fast Fourier transform, as described in Section 4, is even shorter.

Note. After completing a preliminary draft of this paper, the author has learned that Arya, de Fonseca, and Mount (personal communication, late Nov. 2016) have independently obtained similar results [12]. In fact, their time bounds are a little better in the hidden $O^*(1)$ factors (for example, for diameter, we obtain $O((n\sqrt{E} + E^{(d+1)/2}) \log^{O(1)} E)$ time, whereas they obtain $O(n \log E + E^{(d-1)/2+\delta})$ time for an arbitrarily small constant $\delta > 0$). The fact that the techniques are completely different makes the independent discovery all the more exciting. Arya et al.'s techniques build on a long series of their earlier work involving *Macbeath regions* [6, 10, 11, 13], and the analysis in these previous papers appears complicated. In contrast, our algorithms make minimal use of geometry, and are more general in some sense. For instance, our second algorithm for diameter works in the L_s metric for any integer constant $s \geq 2$ (or other similarly behaved distance functions) with the same running time, whereas the approach by Arya et al. does not appear to generalize because of its reliance on a certain lifting transformation. The polynomial method is very powerful, and we anticipate more applications will follow.

2 First Algorithm

Our first algorithm solves a generalization of the discrete upper envelope problem:

▶ **Problem 1** (Generalized Discrete Upper Envelope). *Let d be a constant and let $\psi_1, \ldots, \psi_{d-1}$ be bivariate $O(1)$-degree polynomials with integer coefficients. Given a set P of n points in $\mathbb{Z}^d$, we want to compute*

$$f(x_1, \ldots, x_{d-1}) := \max_{(a_1,\ldots,a_d)\in P} (\psi_1(a_1, x_1) + \cdots + \psi_{d-1}(a_{d-1}, x_{d-1}) + a_d) \tag{1}$$

for all[1] $(x_1, \ldots, x_{d-1}) \in [F]^{d-1}$*, while allowing additive error $O(\varepsilon U)$, where U is a given upper bound on $|\psi_1(a_1, x_1) + \cdots + \psi_{d-1}(a_{d-1}, x_{d-1}) + a_d|$ over all $(a_1, \ldots, a_d) \in P$ and $(x_1, \ldots, x_{d-1}) \in [F]^{d-1}$.*

[1] $[m]$ denotes the integer set $\{0, 1, \ldots, m-1\}$.

For example, for the discrete upper envelope problem as defined in the Introduction, we can round the given point set $P \subset [0,1]^d$ to a uniform grid of side length ε and then rescale so that $\Xi = [F]^{d-1}$ and $P \subset [E]^{d-1} \times [EF]$. We then get an instance of Problem 1 with $\psi_i(a_i, x_i) = a_i x_i$. Here, $U = O(EF)$, and $n \le (EF)^{O(1)}$ after removing duplicates.

For the $(d-1)$-dimensional discrete Voronoi diagram problem as defined in the Introduction, approximating the distance with additive error $O(\varepsilon)$ is equivalent to approximating the squared distance with additive error $O(\varepsilon)$. We can round and rescale so that $\Xi = [F]^{d-1}$ and $P \subset [E]^d$. We then get an instance of Problem 1 with $\psi_i(a_i, x_i) = (\frac{E}{F}x_i - a_i)^2$ (assuming that F divides E). Here, $U = O(E^2)$, and $n \le E^{O(1)}$ after removing duplicates. We can also take $\psi_i(a_i, x_i) = (\frac{E}{F}x_i - a_i)^s$ for the analogous problem under the L_s metric for any even integer constant s.

We now solve Problem 1 using the polynomial method. We start with basic properties about Chebyshev polynomials (e.g., see [28, 4] for quick proofs):

▶ **Lemma 2.** *Let*

$$T_q(x) := \sum_{i=0}^{\lfloor q/2 \rfloor} \binom{q}{2i} (x^2-1)^i x^{q-2i}$$

be the degree-q Chebyshev polynomial (of the first kind).

(i) *If* $|x| \le 1$, *then* $|T_q(x)| \le 1$.
(ii) *If* $x > 1$, *then* $T_q(x) > 1$.
(iii) *If* $x \ge 1 + \varepsilon$, *then* $T_q(x) > \frac{1}{2}e^{q\sqrt{\varepsilon}}$.

Set $q := \left\lceil \sqrt{E}\ln(4n) \right\rceil$ and $D := U^q$. Let $T(x) := D \cdot T_q(1 + \frac{x}{U})$, which is a polynomial with integer coefficients. Our main idea is to work with the following function instead of f:

$$\widetilde{f}(x_1, \ldots, x_{d-1}, t) := \sum_{(a_1,\ldots,a_d)\in P} T(\psi_1(a_1, x_1) + \cdots + \psi_{d-1}(a_{d-1}, x_{d-1}) + a_d - t) \qquad (2)$$

where $x_1, \ldots, x_{d-1} \in [F]$ and $|t| \in [U]$. The function $\widetilde{f}$ is "nicer" since it is a sum (instead of a max) of polynomials, and is thus itself a polynomial. The following observations explain the relationship between the two functions:

- **Case 1**: $f(x_1, \ldots, x_{d-1}) \le t$. Then $-2U \le \psi_1(a_1, x_1) + \cdots + \psi_{d-1}(a_{d-1}, x_{d-1}) + a_d - t \le 0$ for all $(a_1, \ldots, a_d) \in P$. By Lemma 2(i), all n terms in the sum (2) are at most D, and so $\widetilde{f}(x_1, \ldots, x_{d-1}, t) \le Dn$.

- **Case 2**: $f(x_1, \ldots, x_{d-1}) \ge t + \varepsilon U$. Then $\psi_1(a_1, x_1) + \cdots + \psi_{d-1}(a_{d-1}, x_{d-1}) + a_d - t \ge \varepsilon U$ for at least one $(a_1, \ldots, a_d) \in P$. By Lemma 2(iii), at least one term in the sum (2) exceeds $D \cdot \frac{1}{2}e^{q\sqrt{\varepsilon}}$. By Lemma 2(i,ii), all other terms are at least $-D$. So, $\widetilde{f}(x_1, \ldots, x_{d-1}, t) \ge D(\frac{1}{2}e^{q\sqrt{\varepsilon}} - (n-1)) > Dn$ by our choice of q.

Thus, we can approximately compare $f(x_1, \ldots, x_{d-1})$ against t with additive error εU, by evaluating $\widetilde{f}(x_1, \ldots, x_{d-1}, t)$. So, we can approximately compute $f(x_1, \ldots, x_{d-1})$ with additive error $O(\varepsilon U)$ by binary search, using $O(\log E)$ evaluations of $\widetilde{f}$. The total number of evaluations over all $(x_1, \ldots, x_{d-1}) \in [F]^{d-1}$ is $O(F^{d-1} \log E)$.

To evaluate $\widetilde{f}$, one could expand the expression into monomials, as $\widetilde{f}$ is a low-degree multivariate polynomial, but this approach seems too costly. Instead, we use the Chinese remainder theorem. In the stated domain, $\widetilde{f}$ is upper-bounded by $M := DnT_q(3) \le Dn2^{O(q)} \le 2^{O(\sqrt{E}\log^{O(1)}(nEU))}$. Let $\mathcal{P}$ be a set of primes whose product exceeds M; by known bounds, we can choose such a set so that $|\mathcal{P}| = O(\log M / \log\log M)$ and each prime in

$\mathcal{P}$ is at most $O(\log M)$. We describe how to evaluate $\widetilde{f}(x_1, \ldots, x_{d-1}, t) \bmod p$ for each $p \in \mathcal{P}$. Afterwards, we can reconstruct each value $\widetilde{f}(x_1, \ldots, x_{d-1}, t)$ by the Chinese remainder theorem, which takes at most $O(\log^2 M)$ time by elementary methods (for example, by repeated application of Euclid's algorithm, although faster, more sophisticated methods are possible). The total time of this step is $O(F^{d-1} \log E \log^2 M) = O(F^{d-1} E \log^{O(1)}(nEU))$.

Fix a prime $p \in \mathcal{P}$. For each $a_1, \ldots, a_d \in [p]$, let $w_{a_1,\ldots,a_d}$ be the number of points $(a'_1, \ldots, a'_d) \in P$ such that $a'_1 \equiv a_1, \ \ldots, \ a'_d \equiv a_d \pmod p$; we can precompute all these counts by a linear scan over P, using $O(n)$ arithmetic operations. Now,

$$\widetilde{f}(x_1, \ldots, x_{d-1}, t) \equiv \sum_{a_1,\ldots,a_d \in [p]} w_{a_1,\ldots,a_d} T(\psi_1(a_1, x_1) + \cdots + \psi_{d-1}(a_{d-1}, x_{d-1}) + a_d - t) \pmod p.$$

To generate all $\widetilde{f} \bmod p$ values, we use dynamic programming. For each $i \in \{1, \ldots, d\}$ and each $a_1, \ldots, a_{i-1}, x_i, \ldots, x_{d-1}, t \in [p]$, define

$$g^{(i)}_{a_1,\ldots,a_{i-1}}(x_i, \ldots, x_{d-1}, t) := \sum_{a_i,\ldots,a_d \in [p]} w_{a_1,\ldots,a_d} T(\psi_i(a_i, x_i) + \cdots + \psi_{d-1}(a_{d-1}, x_{d-1}) + a_d - t) \pmod p.$$

Then $g^{(1)}$ gives us $\widetilde{f} \bmod p$.

For the base case, we can compute $g^{(d)}$ using the formula

$$g^{(d)}_{a_1,\ldots,a_{d-1}}(t) \equiv \sum_{a_d \in [p]} w_{a_1,\ldots,a_d} T(a_d - t) \pmod p \tag{3}$$

for all $a_1, \ldots, a_{d-1}, t \in [p]$. For $i = d-1, \ldots, 1$, we can compute $g^{(i)}$ using the recursive formula

$$g^{(i)}_{a_1,\ldots,a_{i-1}}(x_i, \ldots, x_{d-1}, t) \equiv \sum_{a_i \in [p]} g^{(i+1)}_{a_1,\ldots,a_i}(x_{i+1}, \ldots, x_{d-1}, t - \psi_i(a_i, x_i)) \pmod p \tag{4}$$

for all $a_1, \ldots, a_{i-1}, x_i, \ldots, x_{d-1}, t \in [p]$.

The resulting dynamic program requires $O(p^d)$ table entries, each computed using $O(p)$ arithmetic operations by (3) and (4). This assumes that we have precomputed $T(x)$ for all $x \in [p]$ (which straightforwardly requires $O(pq)$ arithmetic operations). All arithmetic operations are done modulo p, each costing at most $O(\log^2 p)$ by elementary methods. Thus, the running time of the dynamic program is $O(p^{d+1} \log^2 p) = O(\log^{d+1} M \log^2 \log M) = O(E^{(d+1)/2} \log^{O(1)}(nEU))$.

Including the cost of computing the counts, the running time is $O((n + E^{(d+1)/2}) \log^{O(1)}(nEU))$ for each fixed prime p. The total over all $O(\log M / \log \log M) = O(\sqrt{E} \log^{O(1)}(nEU))$ primes $p \in \mathcal{P}$ becomes $O((n\sqrt{E} + E^{d/2+1}) \log^{O(1)}(nEU))$.

▶ **Theorem 3.** *Problem 1 can be solved in $O((n\sqrt{E} + E^{d/2+1} + F^{d-1}E) \log^{O(1)}(nEU))$ time, where $E = \lceil 1/\varepsilon \rceil$.*

In the case $F = \sqrt{E}$, the bound is $O((n\sqrt{E} + E^{d/2+1}) \log^{O(1)}(nEU))$, which nearly matches the lower bound $\Omega(n + E^{(d-1)/2})$ up to a factor of about $E^{3/2}$.

Note that the above method actually solves a data structure version of Problem 1: after preprocessing in $O((n\sqrt{E} + E^{d/2+1}) \log^{O(1)}(nEU))$ time, we can approximate $f(x_1, \ldots, x_{d-1})$ for any query point $(x_1, \ldots, x_{d-1}) \in [F]^{d-1}$ in $O(E \log^{O(1)}(nEU))$ time. This data structure problem is similar to the approximate polytope membership problem studied by Arya et al. [6, 10, 11, 13, 12].

Appendix B describes one small improvement of the $E^{d/2+1}$ term to a bound approaching $E^{(d+1)/2}$ as d gets large. This improvement requires fast rectangular matrix multiplication, however.

3 Applications

We now sketch how our new algorithm for discrete upper envelopes and discrete Voronoi diagrams automatically leads to better algorithms for various problems, by combining with existing techniques from computational geometry.

Diameter. Given a set P of n points in d dimensions, we consider the problem of computing a $(1 + O(\varepsilon))$-factor approximation of the diameter, i.e, the distance of the farthest pair of points.

We can adopt the following standard algorithm, described in [19] (see also [3, 18]): First compute a constant-factor approximation in $O(n)$ time (e.g., by picking any arbitrary point of P and taking the farthest neighbor distance from that point). By translation and scaling, we may assume that the diameter is $\Theta(1)$ and $P \subset [0, 1]^d$. Let Ξ be the set of all grid points over $\partial[-1, 1]^d$ with side length $\delta := \sqrt{\varepsilon}$. For each $\xi \in \Xi$, find a point $p_\xi \in P$ that maximizes $p_\xi \cdot \xi$ and a point $q_\xi \in P$ that maximizes $-q_\xi \cdot \xi$, while allowing additive error $O(\varepsilon)$. Return the maximum of $p_\xi \cdot \xi - q_\xi \cdot \xi$ over all $\xi \in \Xi$. See [19] for the correctness proof.

Observe that computing all the p_ξ's and q_ξ's corresponds to $O(1)$ instances of the discrete upper envelope problem (one per facet of $\partial[-1, 1]^d$). By applying Theorem 3 with $F = \sqrt{E}$, $U = O(EF)$, and $n \le E^{O(1)}$, we immediately obtain:

▶ **Corollary 4.** *Given n points in constant dimension d, we can compute a $(1+\varepsilon)$-approximation of the diameter in $O((n\sqrt{E} + E^{d/2+1}) \log^{O(1)} E)$ time, where $E = \lceil 1/\varepsilon \rceil$.*

ε-kernels. Given a set P of n points in d dimensions, an *ε-kernel* is, roughly speaking, a subset $Q \subset P$ whose width approximates the width of P to within a factor of $1 + \varepsilon$ along every direction simultaneously. Alternatively, it can be viewed as a "coreset" for approximate convex hulls. The concept was introduced by Agarwal, Har-Peled, and Varadarajan [1] and has a plethora of applications; see [1, 2] for the precise definition and background.

Previous work [19, 30] suggested the following algorithm which computes an ε-kernel of worst-case optimal size $O((1/\varepsilon)^{(d-1)/2})$: First find an affine transformation that makes P "fat" and lie in $[-1, 1]^d$; this is known to be doable in $O(n)$ time. Let Ξ be the set of all grid points over $\partial[-2, 2]^d$ with side length $\sqrt{\varepsilon}$. For each $\xi \in \Xi$, find a nearest neighbor $p_\xi \in P$ to ξ, while allowing additive error $O(\varepsilon)$. Return the subset $\{p_\xi : \xi \in \Xi\}$. See [19] for the correctness proof.

Observe that computing all the p_ξ's reduces to $O(1)$ instances of the $(d - 1)$-dimensional discrete Voronoi diagram problem. (Technically, we need *witness finding*; see Appendix A for a solution, requiring Las Vegas randomization.) By Theorem 3, we immediately obtain:

▶ **Corollary 5.** *Given n points in constant dimension d, we can compute an ε-kernel of size $O(E^{(d-1)/2})$ in $O((n\sqrt{E} + E^{d/2+1}) \log^{O(1)} E)$ expected time, where $E = \lceil 1/\varepsilon \rceil$.*

Bichromatic closest pair. Given a set P of n red points and Q of n blue points in d dimensions, we next examine the problem of finding a $(1 + O(\varepsilon))$-factor approximation of the closest red-blue pair.

We first consider the "well-separated" case, where by translation, rotation, and scaling, we can make $P \subset [-1, 1]^{d-1} \times [-2, -1]$ and $Q \subset [-1, 1]^{d-1} \times [1, 2]$. Arya and Chan [9] suggested the following algorithm to solve this case: Let Ξ be the set of all grid points over $[-1, 1]^{d-1} \times \{0\}$ with side length $\delta := \sqrt{\varepsilon}$. For each $\xi \in \Xi$, find a nearest neighbor $p_\xi \in P$ to ξ and a nearest neighbor $q_\xi \in Q$ to ξ, while allowing additive error $O(\varepsilon)$. Return the closest pair (p_ξ, q_ξ) over all $\xi \in \Xi$. See [9] for the correctness proof.

Observe that computing all the p_ξ's reduces to $O(1)$ instances of the $(d-1)$-dimensional discrete Voronoi diagram problem. (Technically, we again need witness finding.) By Theorem 3, the running time is $O((n\sqrt{E} + E^{d/2+1}) \log^{O(1)} E)$. An alternative upper bound is $O(n^2)$, by brute-force search. The smaller of the two bounds is always at most $O(nE^{d/4+1/2} \log^{O(1)} E)$.

As observed by Arya and Chan [9], a simple grid approach can reduce the general problem to a number of well-separated instances whose input sizes sum to $O(n)$. Thus, the total time is at most $O(nE^{d/4+1/2} \log^{O(1)} E)$.

▶ **Corollary 6.** *Given n red and blue points in constant dimension d, we can compute a $(1+\varepsilon)$-approximate bichromatic closest pair in $O(nE^{d/4+1/2} \log^{O(1)} E)$ expected time, where $E = \lceil 1/\varepsilon \rceil$.*

Approximate nearest neighbor search. The result for bichromatic closest pair can be extended to (offline or online) approximate nearest neighbor search, by following Arya and Chan [9]. The techniques are more involved, requiring balanced box decomposition trees and ideas from earlier papers of Arya, da Fonseca, Malamatos, and Mount [7, 6]. We omit the details, but by reexamining [9] closely and incorporating our new time bound for discrete Voronoi diagrams, we obtain:

▶ **Corollary 7.** *We can preprocess n points in a constant dimension d in $O(n \log n) + O^*(nE^{d/4})$ expected time so that we can find a $(1+\varepsilon)$-approximate nearest neighbor to any query point in $O^*(E^{d/4} \log n)$ time, where $E = \lceil 1/\varepsilon \rceil$.*

Streaming diameter and ε-kernels. The same paper [9] also described an application to insertion-only streaming algorithms for approximating the diameter. Their solution requires first designing a data structure for approximate farthest neighbor queries using techniques similar to [7, 6], and then combining with Bentley and Saxe's logarithmic method (or "merge-and-reduce") [15]. We omit the details, but by examining [9] closely and incorporating our new time bound for discrete Voronoi diagrams, we obtain:

▶ **Corollary 8.** *Given a stream of n points in constant dimension d, we can maintain a $(1+\varepsilon)$-approximation of the diameter using $O(E^{(d-1)/2})$ space and supporting insertions in $O^*(E^{d/4})$ expected time, where $E = \lceil 1/\varepsilon \rceil$.*

The paper [9] also studied the insertion-only streaming algorithms for ε-kernels. Here, the solution is easier. We first consider the special case where the point set P is promised to be fat and lie in $[0,1]^d$ at all times. If we insist on $O(E^{(d-1)/2})$ space, we can handle insertions lazily until a block of $E^{(d-1)/2}$ points is read. Then following Section 3, we can recompute all the p_ξ's and q_ξ's by running our discrete upper envelope algorithm on $E^{(d-1)/2}$ points, taking $O(E^{d/2+1} \log^{O(1)} E)$ time. The amortized insertion time is $O(E^{d/2+1} \log^{O(1)} E)/E^{(d-1)/2} = O(E^{3/2} \log^{O(1)} E)$. (Deamortization is straightforward.)

Building on an earlier streaming algorithm in [19], Zarrabi-Zadeh [31] has given a reduction of the general problem to the above special case that does not increase the processing time or space in the insertion-only streaming model. As a result, we obtain:

▶ **Corollary 9.** *Given a stream of n points in constant dimension d, we can maintain an ε-kernel using $O(E^{(d-1)/2})$ space and supporting insertions in $O(E^{3/2} \log^{O(1)} E)$ expected time, where $E = \lceil 1/\varepsilon \rceil$.*

4 Second Algorithm

We now present an alternative algorithm for the diameter problem, which is also based on Chebyshev polynomials, but bypasses dynamic programming, instead using fast Fourier transform. It is slightly faster (the $E^{d/2+1}$ term in the time bound is reduced to $E^{(d+1)/2}$). It is also more direct, without going through discrete upper envelopes. The algorithm can also be applied to the bichromatic closest pair problem. An advantage is that it can be generalized to the L_s metric for any even integer constant s (although the algorithm for discrete Voronoi diagrams in Section 2 works also for L_s, the reductions from diameter and bichromatic closest pair in Section 3 rely on properties of Euclidean space).

▶ **Problem 10** (Generalized Diameter). *Let d be a constant and φ be a d-variate $O(1)$-degree polynomial with integer coefficients. Given two sets P and Q of n points in $\mathbb{Z}^d$, we want to compute*

$$Z := \max_{(a_1,\ldots,a_d)\in P,\ (b_1,\ldots,b_d)\in Q} \varphi(a_1-b_1,\ldots,a_d-b_d)$$

while allowing additive error $O(\varepsilon U)$, where U is a known upper bound on $|\varphi(a_1-b_1,\ldots,a_d-b_d)|$ over all $(a_1,\ldots,a_d)\in P,\ (b_1,\ldots,b_d)\in Q$.

For example, for diameter in the L_s metric for an even integer constant s, we can first compute a constant-factor approximation in $O(n)$ time. By translation, scaling, and rounding, we may assume that the diameter is $\Theta(E)$ and $P\subset[E]^d$. Approximating the diameter with additive error $O(\varepsilon E)$ is equivalent to approximate the s-th power of the diameter with additive error $O(\varepsilon E^s)$. We then get an instance of Problem 10 with $\varphi(x_1,\ldots,x_d)=x_1^s+\cdots+x_d^s$. Here, $U=O(E^s)$, and $n\le E^{O(1)}$ after removing duplicates.

We now solve Problem 10 using the polynomial method. It suffices to solve the decision problem, of deciding whether the maximum is at least a given value t (with additive error $O(\varepsilon U)$), since the original problem can then be solved by binary search with $O(\log E)$ calls to the decision algorithm.

Reset $q:=\left\lceil\sqrt{E}\ln(4n^2)\right\rceil$ and let D and the degree-q polynomial T be as in Section 2. Define

$$\widetilde{Z} := \sum_{(a_1,\ldots,a_d)\in P,\ (b_1,\ldots,b_d)\in Q} T(\varphi(a_1-b_1,\ldots,a_d-b_d)-t).$$

By a similar analysis as in Section 2, we have:

- **Case 1**: $Z\le t$. Then $\widetilde{Z}\le Dn^2$.
- **Case 2:** $Z\ge t+\varepsilon U$. Then $\widetilde{Z}>D(\frac{1}{2}e^{q\sqrt{\varepsilon}}-(n^2-1))>Dn^2$ by our choice of q.

It suffices to compute $\widetilde{Z}$. At first $\widetilde{Z}$ appears expensive to compute, since we are summing n^2 polynomials. We follow the approach in Section 2 and use the Chinese remainder theorem. Define the set $\mathcal{P}$ of primes as before, with $M:=Dn^2T_q(3)\le 2^{O(\sqrt{E}\log^{O(1)}(nEU))}$. We describe how to compute $\widetilde{Z}$ mod p for each $p\in\mathcal{P}$. Afterwards, we can reconstruct $\widetilde{Z}$ as before in at most $O(\log^2 M)=O(E\log^{O(1)}(nEU))$ time.

Fix a prime $p\in\mathcal{P}$. As before, for each $a_1,\ldots,a_d\in[p]$, let $w_{a_1,\ldots,a_d}$ be the number of points $(a'_1,\ldots,a'_d)\in P$ such that $a'_1\equiv a_1,\ \ldots,\ a'_d\equiv a_d \pmod p$. Similarly, for each $b_1,\ldots,b_d\in[p]$, let $v_{b_1,\ldots,b_d}$ be the number of points $(b'_1,\ldots,b'_d)\in Q$ such that $b'_1\equiv b_1,\ \ldots,\ b'_d\equiv b_d \pmod p$. We can precompute all these counts by a linear scan over P and Q, using $O(n)$ arithmetic

operations. Then

$$\widetilde{Z} \equiv \sum_{a_1,\ldots,a_d,b_1,\ldots,b_d\in[p]} w_{a_1,\ldots,a_d} v_{b_1,\ldots,b_d} T(\varphi(a_1-b_1,\ldots,a_d-b_d)-t) \pmod p \quad (5)$$

$$\equiv \sum_{c_1,\ldots,c_d\in[p]} u_{c_1,\ldots,c_d} T(\varphi(c_1,\ldots,c_d)-t) \pmod p, \quad (6)$$

where

$$u_{c_1,\ldots,c_d} := \sum_{a_1,\ldots,a_d\in[p]} w_{a_1,\ldots,a_d} v_{(a_1-c_1) \bmod p,\ldots,(a_d-c_d) \bmod p} \pmod p.$$

The key is to recognize this expression as a d-dimensional convolution (with wraparound indices modulo p). This can be converted to standard 1-dimensional convolution as follows: Initialize arrays $A[0,\ldots,(2p)^d]$ and $B[0,\ldots,(2p)^d]$ to 0. For each $a_1,\ldots,a_d \in [p]$, set $A[a_1(2p)^{d-1} + a_2(2p)^{d-2} + \cdots + a_d] = w_{a_1,\ldots,a_d}$. For each $b_1,\ldots,b_d \in [p]$, set $B[(p-b_1)(2p)^{d-1}+(p-b_2)(2p)^{d-2}+\cdots+(p-b_d)] = v_{b_1,\ldots,b_d}$. Compute the convolution $C[0,\ldots,(2p)^d]$ where $C[i] := \sum_{k=0}^{i} A[k]B[i-k] \pmod p$. Then for each $c_1,\ldots,c_d \in [p]$, set $u_{c_1,\ldots,c_d} = \sum_{j_1,\ldots,j_d\in\{0,1\}} C[(c_1+j_1p)(2p)^{d-1} + (c_2+j_dp)(2p)^{d-2} + \cdots + (c_d+j_dp)] \pmod p$.

By fast Fourier transform, we can compute all $u_{c_1,\ldots,c_d}$ values using $O(p^d \log p)$ arithmetic operations. Afterwards, we can compute $\widetilde{Z} \bmod p$ by (6) using $O(p^d)$ arithmetic operations. This assumes that we have precomputed $T(x)$ for all $x \in [p]$ (which straightforwardly requires $O(pq)$ arithmetic operations). All arithmetic operations are done modulo p, each costing at most $O(\log^2 p)$ time. The running time is thus $O(p^d \log^3 p) = O(\log^d M \log^3 \log M) = O(E^{d/2} \log^{O(1)}(nEU))$.

Including the cost of computing the counts, the running time is $O((n+E^{d/2}) \log^{O(1)}(nEU))$ for each fixed prime p. The total over all $O(\log M/\log\log M) = O(\sqrt{E}\log^{O(1)}(nEU))$ primes $p \in \mathcal{P}$ is $O((n\sqrt{E} + E^{(d+1)/2}) \log^{O(1)}(nEU))$.

▶ **Theorem 11.** *Problem 10 can be solved in $O((n\sqrt{E}+E^{(d+1)/2}) \log^{O(1)}(nEU))$ time, where $E = \lceil 1/\varepsilon \rceil$.*

5 Applications

***L_s*-diameter.** The algorithm in Section 4 can immediately be applied to approximate the diameter in the L_s metric for any even integer constant s. For the case of odd s, we can use standard range-tree divide-and-conquer to reduce to bichromatic instances (P, Q) such that P and Q are separated along all d axis directions, in which case the preceding algorithm can be applied. The divide-and-conquer increases the running time by a factor of $O(\log^d n)$, which is $O(\log^d E)$ since $n \le E^{O(1)}$ (after initial rounding and removal of duplicates).

▶ **Corollary 12.** *Given n points in constant dimension d and any integer constant $s \ge 2$, we can compute a $(1+\varepsilon)$-approximation of the L_s-diameter in $O((n\sqrt{E} + E^{(d+1)/2}) \log^{O(1)} E)$ time, where $E = \lceil 1/\varepsilon \rceil$.*

Bichromatic *L_s*-closest pair. For bichromatic closest pair in the L_s metric for any even integer constant s, it suffices to solve the well-separated case, as noted in Section 3, where $P \subset B_P$ and $Q \subset B_Q$ for two unit hypercubes B_P and B_Q of distance $\Theta(1)$ apart. (Note that we can no longer rotate.) Approximating the closest pair distance with additive error $O(\varepsilon)$ is equivalent to approximating the s-th power of the closest pair distance with additive error $O(\varepsilon)$. We can round and rescale so that $P, Q \subset [E]^d$. We then get an instance of

Problem 10 with $\varphi(x_1, \ldots, x_d) = -(x_1^s + \cdots + x_d^s)$. Here, $U = O(E^s)$, and $n \leq E^{O(1)}$ after removing duplicates. The rest of the analysis is as in Section 3. The case of odd s can again be handled by incorporating range-tree divide-and-conquer.

▶ **Corollary 13.** *Given n red and blue points in constant dimension d and any integer constant $s \geq 2$, we can compute a $(1+\varepsilon)$-approximate bichromatic L_s-closest pair in $O(nE^{(d+1)/4} \log^{O(1)} E)$ time, where $E = \lceil 1/\varepsilon \rceil$.*

6 Final Remarks

We now reveal the origins of the ideas behind our first algorithm in Section 2.

- The application of the polynomial method to approximately find extreme points along arbitrary directions was first proposed by Andoni and Nguyen [5], specifically for the *dynamic streaming* model. This line of work was continued in [20] for the ε-kernel problem; in fact, the idea of applying the Chinese remainder theorem and keeping the counts $w_{a_1,\ldots,a_d}$ (which are easy to maintain in the dynamic streaming setting) is taken from [20]. However, it has not been realized before that the approach could give better algorithms in the standard nonstreaming setting. Also, these previous algorithms [5, 20] constructed polynomials by summing q-th powers rather than degree-q Chebyshev polynomials, which caused a larger degree bound on q (of the order E instead of $\sqrt{E}$) and thus larger ε-dependencies in time and space complexity.
- The theoretical computer science literature contains a number of earlier applications of Chebyshev polynomials. The closest to our work are perhaps the papers by Valiant [28] and Alman, Chan, and Williams [4] on approximate closest pair and offline nearest neighbor search in *high dimensions*. The latter paper also played with sums of Chebyshev polynomials, but the algorithms were put together quite differently. For example, they dealt primarily with polynomials with Boolean variables, they needed to expand polynomials into monomials, and they relied on fast matrix multiplication rather than dynamic programming.
- Related is another polynomial-method-based algorithm for #SAT by Chan and Williams [21]. There, a multivariate polynomial is evaluated over all points in $\{0,1\}^m$ in near 2^m time, without fast matrix multiplication. This subproblem in Boolean space reduces to computing a *Möbius* or *zeta transform*, for which a standard dynamic programming algorithm by Yates can be invoked [29, 16]. Our dynamic programming algorithm, to evaluate a polynomial over all points in the space $[E]^d$, is not entirely "original" and can be viewed as a variant of Yates' algorithm.

Our second algorithm in Section 4 which exploits fast Fourier transform seems more original, although the idea is simple in hindsight.

The main advantage of the polynomial method is its generality. For example, the approach in our second algorithm might potentially be applicable to *kinetic* variants of the diameter decision problem where points are moving according to $O(1)$-degree polynomial functions in time.

We can consider a still more general version of the diameter problem than Problem 10, where φ can be any $(2d)$-variate polynomial with integer coefficients and we seek $Z := \max_{(a_1,\ldots,a_d)\in P,\ (b_1,\ldots,b_d)\in Q} \varphi(a_1, \ldots, a_d, b_1, \ldots, b_d)$. Fast Fourier transform does not seem applicable here, and we have to adapt (5):

$$\widetilde{Z} \equiv \sum_{a_1,\ldots,a_d,b_1,\ldots,b_d \in [p]} w_{a_1,\ldots,a_d} v_{b_1,\ldots,b_d} T(\varphi(a_1, \ldots, a_d, b_1, \ldots, b_d) - t) \pmod p,$$

which can be evaluated using $O(p^{2d})$ arithmetic operations by brute force (instead of $O(p^d \log p)$). This yields a slower (but still new) running time of $O((n\sqrt{E}+E^{d+(1/2)}) \log^{O(1)} E) = O^*(n + E^d)$.

To close, we mention two specific open problems:

- Can we approximate the width of a point set in $O^*(n + E^{d/2})$ time? The issue is that knowing an ε-kernel, we still need to compute the width of the kernel efficiently.
- Can we approximate the diameter in $O^*(n + E^{\alpha d})$ time for some absolute constant $\alpha < 1/2$?

References

1. Pankaj K. Agarwal, Sariel Har-Peled, and Kasturi R. Varadarajan. Approximating extent measures of points. *J. ACM*, 51(4):606–635, 2004. Preliminary version in SODA'01 and FOCS'01.
2. Pankaj K. Agarwal, Sariel Har-Peled, and Kasturi R. Varadarajan. Geometric approximation via coresets. In Emo Welzl, editor, *Current Trends in Combinatorial and Computational Geometry*, pages 1–30. Cambridge University Press, 2005.
3. Pankaj K. Agarwal, Jirí Matoušek, and Subhash Suri. Farthest neighbors, maximum spanning trees and related problems in higher dimensions. *Comput. Geom. Theory Appl.*, 1:189–201, 1991. doi:10.1016/0925-7721(92)90001-9.
4. Josh Alman, Timothy M. Chan, and Ryan Williams. Polynomial representation of threshold functions and algorithmic applications. In *Proc. 57th IEEE Symp. Found. Comput. Sci. (FOCS)*, pages 467–476, 2016.
5. Alexandr Andoni and Huy L. Nguyen. Width of points in the streaming model. In *Proc. 23rd ACM–SIAM Symp. Discrete Algorithms (SODA)*, pages 447–452, 2012. *ACM Trans. Algorithms*, to appear.
6. S. Arya, G. D. da Fonseca, and D. M. Mount. Approximate polytope membership queries. In *Proc. 43rd ACM Symp. Theory Comput. (STOC)*, pages 579–586, 2011. *SIAM J. Comput.*, to appear. URL: http://www.uniriotec.br/~fonseca/polytope_conf.pdf, doi:10.1145/1993636.1993713.
7. S. Arya, T. Malamatos, and D. M. Mount. Space-time tradeoffs for approximate nearest neighbor searching. *J. ACM*, 57:1–54, 2009. Preliminary version in SODA'02 and STOC'02. doi:10.1145/1613676.1613677.
8. S. Arya, D. M. Mount, N. Netanyahu, R. Silverman, and A. Y. Wu. An optimal algorithm for approximate nearest neighbor searching in fixed dimensions. *J. ACM*, 45:891–923, 1998. Preliminary version in SODA'94.
9. Sunil Arya and Timothy M. Chan. Better ε-dependencies for offline approximate nearest neighbor search, Euclidean minimum spanning trees, and ε-kernels. In *Proc. 30th Symp. Comput. Geom. (SoCG)*, pages 416–425, 2014. doi:10.1145/2582112.2582161.
10. Sunil Arya, Guilherme Dias da Fonseca, and David M. Mount. Optimal area-sensitive bounds for polytope approximation. In *Proc. 28th Symp. Comput. Geom. (SoCG)*, pages 363–372, 2012. doi:10.1145/2261250.2261305.
11. Sunil Arya, Guilherme Dias da Fonseca, and David M. Mount. On the combinatorial complexity of approximating polytopes. In *Proc. 32nd Symp. Comput. Geom. (SoCG)*, pages 11:1–11:15, 2016. doi:10.4230/LIPIcs.SoCG.2016.11.
12. Sunil Arya, Guilherme Dias da Fonseca, and David M. Mount. Near-optimal ε-kernel construction and related problems. In *Proc. 33rd Symp. Comput. Geom. (SoCG)*, 2017.
13. Sunil Arya, Guilherme Dias da Fonseca, and David M. Mount. Optimal approximate polytope membership. In *Proc. 28th ACM–SIAM Symp. Discrete Algorithms (SODA)*, pages 270–288, 2017.

14 Gill Barequet and Sariel Har-Peled. Efficiently approximating the minimum-volume bounding box of a point set in three dimensions. *J. Algorithms*, 38(1):91–109, 2001. Preliminary version in SODA'99. doi:10.1006/jagm.2000.1127.

15 J. L. Bentley and J. B. Saxe. Decomposable searching problems I: Static-to-dynamic transformation. *J. Algorithms*, 1(4):301–358, 1980.

16 Andreas Björklund, Thore Husfeldt, and Mikko Koivisto. Set partitioning via inclusion-exclusion. *SIAM J. Comput.*, 39(2):546–563, 2009. doi:10.1137/070683933.

17 H. Breu, J. Gil, D. Kirkpatrick, and M. Werman. Linear time Euclidean distance transform algorithms. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 17:529–533, 1995.

18 Timothy M. Chan. Approximating the diameter, width, smallest enclosing cylinder, and minimum-width annulus. *Int. J. Comput. Geom.Appl.*, 12(1-2):67–85, 2002. Preliminary version in SoCG'00. doi:10.1142/S0218195902000748.

19 Timothy M. Chan. Faster core-set constructions and data-stream algorithms in fixed dimensions. *Comput. Geom. Theory Appl.*, 35(1-2):20–35, 2006. Preliminary version in SoCG'04. doi:10.1016/j.comgeo.2005.10.002.

20 Timothy M. Chan. Dynamic streaming algorithms for ε-kernels. In *Proc. 32nd Symp. Comput. Geom. (SoCG)*, pages 27:1–27:11, 2016. doi:10.4230/LIPIcs.SoCG.2016.27.

21 Timothy M. Chan and Ryan Williams. Deterministic APSP, orthogonal vectors, and more: Quickly derandomizing Razborov–Smolensky. In *Proc. 27th ACM–SIAM Symp. Discrete Algorithms (SODA)*, pages 1246–1255, 2016.

22 T. M. Chan. Approximate nearest neighbor queries revisited. *Discrete Comput. Geom.*, 20:359–373, 1998. Preliminary version in SoCG'97.

23 Don Coppersmith. Rapid multiplication of rectangular matrices. *SIAM J. Comput.*, 11(3):467–471, 1982. doi:10.1137/0211037.

24 Sariel Har-Peled. A practical approach for computing the diameter of a point set. In *Proc. 17th Symp. Comput. Geom. (SoCG)*, pages 177–186, 2001. doi:10.1145/378583.378662.

25 Xiaohan Huang and Victor Y. Pan. Fast rectangular matrix multiplication and applications. *J. Complexity*, 14(2):257–299, 1998. doi:10.1006/jcom.1998.0476.

26 R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

27 Otfried Schwarzkopf. Parallel computation of distance transforms. *Algorithmica*, 6(5):685–697, 1991. doi:10.1007/BF01759067.

28 Gregory Valiant. Finding correlations in subquadratic time, with applications to learning parities and the closest pair problem. *J. ACM*, 62(2):13, 2015. Preliminary version in FOCS'12.

29 F. Yates. The design and analysis of factorial experiments. *Technical Communication No. 35, Commonwealth Bureau of Soil Science, Harpenden, UK*, 1937.

30 H. Yu, P. K. Agarwal, R. Poreddy, and K. R. Varadarajan. Practical methods for shape fitting and kinetic data structures using coresets. *Algorithmica*, 52(3):378–402, 2008. Preliminary version in SoCG'04.

31 Hamid Zarrabi-Zadeh. An almost space-optimal streaming algorithm for coresets in fixed dimensions. *Algorithmica*, 60(1):46–59, 2011. Preliminary version in ESA'08. doi:10.1007/s00453-010-9392-2.

A Finding Witnesses

One technical issue not addressed in Section 2 is how to find a *witness* point $(a_1, \ldots, a_d) \in P$ that approximately attains the maximum in (1), for every $(x_1, \ldots, x_{d-1}) \in [F]^{d-1}$. This is needed in some of the applications from Section 3. One standard approach to find such witnesses is via binary search, using a binary tree of subsets of P, but this seems to hurt the $E^{d/2}$ term in the running time. We adopt another standard approach, using *random*

sampling to isolate witnesses [26]. In the approximate setting, the details are trickier, but have been worked out in the previous paper [20]. Although that paper dealt with q-th powers instead of degree-q Chebyshev polynomials, the same ideas can be applied, as we now explain. (In fact, the details get a little simpler when we are not working in the streaming model.)

We assume that $P \subset [U]^d$ (which is true in all our applications). First let $\ell(a_1, \ldots, a_d) = a_1 U^{d-1} + a_2 U^{d-2} + \cdots + a_d + 1$ denote the *label* of a point $(a_1, \ldots, a_d) \in P$.

Let $k := \lceil \log n \rceil$. For each $j \in [k]$, draw a random sample $R_j \subset P$ where each point is chosen with probability $1/2^j$.

Reset $q := \left\lceil \sqrt{kE} \ln(10nU^{2d}) \right\rceil$. Define the polynomial functions

$$\widetilde{f}_j(x_1, \ldots, x_{d-1}, t) := \sum_{(a_1,\ldots,a_d)\in R_j} T(\psi_1(a_1, x_1) + \cdots + \psi_{d-1}(a_{d-1}, x_{d-1}) + a_d - t)$$

$$f_j^*(x_1, \ldots, x_{d-1}, t) := \sum_{(a_1,\ldots,a_d)\in R_j} \ell(a_1, \ldots, a_d) T(\psi_1(a_1, x_1) + \cdots + \psi_{d-1}(a_{d-1}, x_{d-1}) + a_d - t)$$

where $x_1, \ldots, x_{d-1} \in [F]$ and $|t| \in [U]$. We can evaluate $\widetilde{f}_j$ and f_j^* by the same approach as in Section 2 (after resetting $M := U^d \cdot DnT_q(3)$). The running time remains the same up to polylogarithmic factors, since the number of choices for j is $k = O(\log n)$, and the degree q increases only by a polylogarithmic factor.

Suppose we want to find a witness for a given $(x_1, \ldots, x_{d-1}) \in [F]^{d-1}$. We first find an approximation t to the maximum, with $t \le f(x_1, \ldots, x_{d-1}) \le t + \lceil \varepsilon U \rceil$, by the method in Section 2. Intuitively, if the number of witnesses is near 2^j, then with good probability exactly one witness is in R_j and the ratio $\frac{f_j^*(x_1,\ldots,x_{d-1},t)}{\widetilde{f}_j(x_1,\ldots,x_{d-1},t)}$ rounded to the nearest integer should give us the label to a witness, because the sums in the numerator and denominator are both dominated by a single term which corresponds to the witness. More care is needed in the approximate setting, however.

Let $\Delta := \lceil \lceil \varepsilon U \rceil / k \rceil$. More precisely, we claim that with probability $\Omega(1)$, the label of a witness can be found among the following ratios after rounding:

$$\frac{f_j^*(x_1, \ldots, x_{d-1}, t - i\Delta)}{\widetilde{f}_j(x_1, \ldots, x_{d-1}, t - i\Delta)} \qquad (i, j \in [k]).$$

To prove the claim, let $P_i = \{(a_1, \ldots, a_d) \in P : \psi_1(a_1, x_1) + \cdots + \psi_{d-1}(a_{d-1}, x_{d-1}) + a_d \ge t - i\Delta\}$. Any point in P_i for any $i \le k$ may be used as a witness with additive error $O(k\Delta) = O(\varepsilon U)$. Since $|P_0| \ge 1$, there exists $i \le k$ such that $|P_i| \le 2|P_{i-1}|$ (for otherwise, $|P_k| > 2^k \ge n$, a contradiction). Suppose $2^j \le |P_{i-1}| \le 2^{j+1}$. Then $|P_i| \le 2^{j+2}$. Let $\mathcal{E}$ be the event that exactly one point of P_{i-1} is chosen to be in R_j and no point of $P_i \setminus P_{i-1}$ is chosen to be in R_j. Then $\Pr(\mathcal{E}) \ge |P_{i-1}| \frac{1}{2^j}(1 - \frac{1}{2^j})^{|P_i|-1} \ge (1 - \frac{1}{2^j})^{2^{j+2}} \ge \Omega(1)$. Suppose that $\mathcal{E}$ is true. Let $(a_1, \ldots, a_d) \in P$ be the unique point of P_{i-1} that is chosen to be in R_j. Let $\ell = \ell(a_1, \ldots, a_d)$ and $T = T(\psi_1(a_1, x_1) + \cdots + \psi_{d-1}(a_{d-1}, x_{d-1}) + a_d - (t - i\Delta))$. Since $\psi_1(a_1, x_1) + \cdots + \psi_{d-1}(a_{d-1}, x_{d-1}) + a_d - (t - i\Delta) \ge \Delta$, by Lemma 2, $T \ge D \cdot \frac{1}{2} e^{\sqrt{\varepsilon/kq}} \ge 5DnU^{2d} \ge 5nU^d D\ell$; in other words, $nU^d D \le T/(5\ell)$. Thus,

$$\begin{aligned} \frac{f_j^*(x_1, \ldots, x_{d-1}, t - i\Delta)}{\widetilde{f}_j(x_1, \ldots, x_{d-1}, t - i\Delta)} &\in \left[\frac{\ell T - (n-1)U^d D}{T + (n-1)D}, \frac{\ell T + (n-1)U^d D}{T - (n-1)D}\right] \\ &\subset \left[\frac{\ell - 1/(5\ell)}{1 + 1/(5\ell)}, \frac{\ell + 1/(5\ell)}{1 - 1/(5\ell)}\right] \subset \left(\ell - \frac{1}{2}, \ell + \frac{1}{2}\right), \end{aligned}$$

as desired.

Since each division costs at most $O(\log^2 M)$, each witness can be found in $O(\log^2 M \log^2 n)$ time with success probability $\Omega(1)$. The total time for all $(x_1, \ldots, x_{d-1}) \in [F]^{d-1}$ is $O(F^{d-1} \log^2 M \log^2 n) = O(F^{d-1} E \log^{O(1)}(nEU))$. We can find all witnesses correctly by repeating the algorithm an expected $O(\log(F^{d-1}))$ number of times (since verifying a given witness is easy). Thus, the total time of the entire randomized (Las Vegas) algorithm remains the same in expectation, up to polylogarithmic factors.

B Small Improvement

In this appendix, we note a small speedup to the algorithm in Section 2 by exploiting fast Fourier transform and fast matrix multiplication. This improvement is mainly of theoretical interest.

For the base case of the dynamic program, observe that when $a_1, \ldots, a_{d-1}$ are fixed, equation (3) can be rewritten as a convolution of two p-dimensional vectors: letting $A[a_d] := w_{a_1,\ldots,a_d}$ and $B[x] = T(-x)$, we have $g^{(d)}_{a_1,\ldots,a_{d-1}}(t) \equiv \sum_{a_d \in [p]} A[a_d] B[t - a_d]$. By fast Fourier transform, the $O(p^{d-1})$ convolutions require $O(p^{d-1} \cdot p \log p)$ arithmetic operations.

For the main dynamic program, observe that equation (4) can be rewritten as a product of a $p^{d-2} \times p^2$ matrix and a $p^2 \times p^2$ matrix: letting

$$\begin{aligned} C[(a_1, \ldots, a_{i-1}, x_{i+1}, \ldots, x_{d-1}), (x_i, t)] &:= g^{(i)}_{a_1,\ldots,a_{i-1}}(x_i, \ldots, x_{d-1}, t) \\ A[(a_1, \ldots, a_{i-1}, x_{i+1}, \ldots, x_{d-1}), (a_i, z)] &:= g^{(i+1)}_{a_1,\ldots,a_i}(x_{i+1}, \ldots, x_{d-1}, z) \\ B[(a_i, z), (x_i, t)] &:= \begin{cases} 1 & \text{if } z \equiv t - \psi_i(a_i, x_i) \pmod p \\ 0 & \text{else,} \end{cases} \end{aligned}$$

we have $C[\xi, \eta] \equiv \sum_\zeta A[\xi, \zeta] B[\zeta, \eta]$. The computation requires $O(p^{\omega(d-2,2,2)})$ arithmetic operations, where $\omega(\alpha, \beta, \gamma)$ denotes the matrix multiplication exponent for multiplying an $n^\alpha \times n^\beta$ and an $n^\beta \times n^\gamma$ matrix. All arithmetic operations are done modulo p. The running time of the dynamic program is then $O(p^{\omega(d-2,2,2)} \log^2 p) = O(E^{\omega(d/2-1,1,1)} \log^{O(1)} E)$. The total over all $O(\sqrt{E} \log^{O(1)} E)$ primes $p \in \mathcal{P}$ gives $O(E^{\omega(d/2-1,1,1)+1/2} \log^{O(1)} E)$.

▶ **Theorem 14.** *Problem 1 can be solved in* $O((n\sqrt{E} + E^{\omega(d/2-1,1,1)+1/2} + F^{d-1}E) \log^{O(1)} E)$ *time, where* $E = \lceil 1/\varepsilon \rceil$.

Note that as d grows, $\omega(d/2-1, 1, 1) - d/2$ approaches 0, by known results on rectangular matrix multiplication [23, 25].

Orthogonal Range Searching in Moderate Dimensions: k-d Trees and Range Trees Strike Back*†

Timothy M. Chan

Dept. of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL, USA
tmc@illinois.edu

Abstract

We revisit the *orthogonal range searching* problem and the *exact* ℓ_∞ *nearest neighbor searching* problem for a static set of n points when the dimension d is moderately large. We give the first data structure with near linear space that achieves truly sublinear query time when the dimension is any constant multiple of $\log n$. Specifically, the preprocessing time and space are $O(n^{1+\delta})$ for any constant $\delta > 0$, and the expected query time is $n^{1-1/O(c \log c)}$ for $d = c \log n$. The data structure is simple and is based on a new "augmented, randomized, lopsided" variant of k-d trees. It matches (in fact, slightly improves) the performance of previous combinatorial algorithms that work only in the case of offline queries [Impagliazzo, Lovett, Paturi, and Schneider (2014) and Chan (SODA'15)]. It leads to slightly faster combinatorial algorithms for *all-pairs shortest paths* in general real-weighted graphs and *rectangular Boolean matrix multiplication*.

In the offline case, we show that the problem can be reduced to the *Boolean orthogonal vectors* problem and thus admits an $n^{2-1/O(\log c)}$-time non-combinatorial algorithm [Abboud, Williams, and Yu (SODA'15)]. This reduction is also simple and is based on range trees.

Finally, we use a similar approach to obtain a small improvement to Indyk's data structure [FOCS'98] for *approximate* ℓ_∞ nearest neighbor search when $d = c \log n$.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases computational geometry, data structures, range searching, nearest neighbor searching

Digital Object Identifier 10.4230/LIPIcs.SoCG.2017.27

1 Introduction

In this paper, we revisit some classical problems in computational geometry:

- In *orthogonal range searching*, we want to preprocess n data points in $\mathbb{R}^d$ so that we can detect if there is a data point inside any query axis-aligned box, or report or count all such points.
- In *dominance range searching*, we are interested in the special case when the query box is d-sided, of the form $(-\infty, q_1] \times \cdots \times (-\infty, q_d]$; in other words, we want to detect if there is a data point $(p_1, \ldots, p_d)$ that is dominated by a query point $(q_1, \ldots, q_d)$, in the sense that $p_j \le q_j$ for all $j \in \{1, \ldots, d\}$, or report or count all such points.
- In ℓ_∞ *nearest neighbor searching*, we want to preprocess n data points in $\mathbb{R}^d$ so that we can find the nearest neighbor to the given query point under the ℓ_∞ metric.

* A full version of the paper is available at http://tmc.web.engr.illinois.edu/high_ors3_17.pdf.
† This work was done while the author was at the Cheriton School of Computer Science, University of Waterloo.

All three problems are related. Orthogonal range searching in d dimensions reduces to dominance range searching in $2d$ dimensions.[1] Furthermore, ignoring logarithmic factors, ℓ_∞ nearest neighbor searching reduces to its decision problem (deciding whether the ℓ_∞ nearest neighbor distance to a given query point is at most a given radius) by parametric search or randomized search [7], and the decision problem clearly reduces to orthogonal range searching.

The standard *k-d tree* [22] has $O(dn \log n)$ preprocessing time and $O(dn)$ space, but the worst-case query time is $O(dn^{1-1/d})$. The standard *range tree* [22] requires $O(n \log^d n)$ preprocessing time and space and $O(\log^d n)$ query time, excluding an $O(K)$ term for the reporting version of the problem with output size K. Much work in computational geometry has been devoted to small improvements of a few logarithmic factors. For example, the current best result for orthogonal range reporting has $O(n \log^{d-3+\varepsilon} n)$ space and $O(\log^{d-3} n / \log^{d-4} \log n + K)$ time [12]; there are also other small improvements for various offline versions of the problems [12, 13, 2].

In this paper, we are concerned with the setting *when the dimension is nonconstant*. Traditional approaches from computational geometry tend to suffer from exponential dependencies in d (the so-called "curse of dimensionality"). For example, the $O(dn^{1-1/d})$ or $O(\log^d n)$ query time bound for range trees or k-d trees is sublinear only when $d \ll \log n / \log \log n$. By a more careful analysis [10], one can show that range trees still have sublinear query time when $d \ll \alpha_0 \log n$ for a sufficiently small constant α_0. The case when the dimension is close to logarithmic in n is interesting in view of known dimensionality reduction techniques [16] (although such techniques technically are not applicable to exact problems and, even with approximation, do not work well for ℓ_∞). The case of polylogarithmic dimensions is also useful in certain *non-geometric* applications such as all-pairs shortest paths (as we explain later). From a theoretical perspective, it is important to understand when the time complexity transitions from sublinear to superlinear.

Previous offline results. We first consider the *offline* version of the problems where we want to answer a batch of n queries all given in advance. In high dimensions, it is possible to do better than $O(dn^2)$-time brute-force search, by a method of Matoušek [21] using fast (rectangular) matrix multiplication [20]; for example, we can get $n^{2+o(1)}$ time for $d \ll n^{0.15}$. However, this approach inherently cannot give *sub*quadratic bounds.

In 2014, a surprising discovery was made by Impagliazzo et al. [17]: range-tree-like divide-and-conquer can still work well even when the dimension goes a bit above logarithmic. Their algorithm can answer n offline dominance range queries (and thus orthogonal range queries and ℓ_∞ nearest neighbor queries) in total time $n^{2-1/O(c^{15} \log c)}$ (ignoring an $O(K)$ term for reporting) in dimension $d = c \log n$ for any possibly nonconstant c ranging from 1 to about $\log^{1/15} n$ (ignoring $\log \log n$ factors). Shortly after, by a more careful analysis of the same algorithm, Chan [8] refined the time bound to $n^{2-1/O(c \log^2 c)}$, which is subquadratic for c up to about $\log n$, i.e., dimension up to about $\log^2 n$.

At SODA'15, Abboud, Williams, and Yu [1] obtained an even better time bound for dominance range detection in the *Boolean* special case, where all coordinate values are 0's and 1's (in this case, the problem is better known as the *Boolean orthogonal vectors* problem[2]).

---

[1] $(p_1, \ldots, p_d)$ is inside the box $[a_1, b_1] \times \cdots \times [a_d, b_d]$ iff $(-p_1, p_1, \ldots, -p_d, p_d)$ is dominated by $(-a_1, b_1, \ldots, -a_d, b_d)$ in $\mathbb{R}^{2d}$.

[2] Two vectors $(p_1, \ldots, p_d), (q_1, \ldots, q_d) \in \{0,1\}^d$ are orthogonal iff $\sum_{i=1}^d p_i q_i = 0$ iff $(p_1, \ldots, p_d)$ is dominated by $(1 - q_1, \ldots, 1 - q_d)$ (recalling that our definition of dominance uses non-strict inequality).

The total time for n offline Boolean dominance range detection queries is $n^{2-1/O(\log c)}$. The bound $n^{2-1/O(\log c)}$ is a natural barrier, since a faster offline Boolean dominance algorithm would imply an algorithm for CNF-SAT with n variables and cn clauses that would beat the currently known $2^{n(1-1/O(\log c))}$ time bound [1]; and an $O(n^{2-\delta})$-time algorithm for any $c = \omega(1)$ would break the strong exponential-time hypothesis (SETH) [24]. Abboud et al.'s algorithm was based on the *polynomial method* pioneered by Williams [23] (see [4, 3] for other geometric applications). The algorithm was originally randomized but was subsequently derandomized by Chan and Williams [9] in SODA'16 (who also extended the result from detection to counting).

Abboud et al.'s approach has two main drawbacks, besides being applicable to the Boolean case only: 1. it is not "combinatorial" and relies on fast rectangular matrix multiplication, making the approach less likely to be practical, and 2. it only works in the offline setting.

Impagliazzo et al.'s range-tree method [17] is also inherently restricted to the offline setting – in their method, the choice of dividing hyerplanes crucially requires knowledge of all query points in advance. All this raises an intriguing open question: are there nontrivial results for online queries in $d = c \log n$ dimensions?

New online result. In Section 2, we resolve this question by presenting a randomized data structure with $O(n^{1+\delta})$ preprocessing time and space that can answer online dominance range queries (and thus orthogonal range queries and ℓ_∞ nearest neighbor queries) in $n^{1-1/O(c \log^2 c)}$ expected time for any $d = c \log n \ll \log^2 n / \log\log n$ and for any constant $\delta > 0$. (We assume an oblivious adversary, i.e., that query points are independent of the random choices made by the preprocessing algorithm.) The total time for n queries is $n^{2-1/O(c \log^2 c)}$, matching the offline bound from Impagliazzo et al. [17] and Chan [8]. The method is purely combinatorial, i.e., does not rely on fast matrix multiplication.

More remarkable than the result perhaps is the simplicity of the solution: it is just a variant of k-d trees! More specifically, the dividing hyperplane is chosen in a "lopsided" manner, along a randomly chosen coordinate axis; each node is augmented with secondary structures for some lower-dimensional projections of the data points. The result is surprising, considering the longstanding popularity of k-d trees among practitioners. Our contribution lies in recognizing, and proving, that they can have good theoretical worst-case performance. (Simple algorithms with nonobvious analyses are arguably the best kind.)

In Appendix A.1, we also describe a small improvement of the query time to $n^{1-1/O(c \log c)}$. This involves an interesting application of so-called *covering designs* (from combinatorics), not often seen in computational geometry.

Applications. By combining with previous techniques [10, 8], our method leads to new results for two classical, non-geometric problems: *all-pairs shortest paths (APSP)* and *Boolean matrix multiplication (BMM)*.

- We obtain a new combinatorial algorithm for solving the APSP problem for arbitrary real-weighted graphs with n vertices (or equivalently the *(min,+) matrix multiplication* problem for two $n \times n$ real-valued matrices) in $O((n^3/\log^3 n)\,\mathrm{poly}(\log\log n))$ time; see Appendix A.2. This is about a logarithmic factor faster than the best previous combinatorial algorithm [11, 15, 8], not relying on fast matrix multiplication à la Strassen. It also extends Chan's combinatorial algorithm for Boolean matrix multiplication from SODA'15 [8], which has a similar running time (although for Boolean matrix multiplication, Yu [26] has recently obtained a further logarithmic-factor improvement).
  This extension is intriguing, as (min,+) matrix multiplication over the reals appears tougher than other problems such as standard matrix multiplication over $\mathbb{F}_2$, for which the

well-known "four Russians" time bound of $O(n^3/\log^2 n)$ [6] has still not been improved for combinatorial algorithms.

- We obtain a new combinatorial algorithm to multiply an $n \times \log^2 n$ and a $\log^2 n \times n$ Boolean matrix in $O((n^2/\log n)\operatorname{poly}(\log\log n))$ time, which is almost optimal in the standard word RAM model since the output requires $\Omega(n^2/\log n)$ words; see the full paper. The previous combinatorial algorithm by Chan [8] can multiply an $n \times \log^3 n$ and a $\log^3 n \times n$ Boolean matrix in $O(n^2 \operatorname{poly}(\log\log n))$ time. The new result implies the old, but not vice versa.

New offline result. Returning to the offline dominance or orthogonal range searching problem, Abboud, Williams, and Yu's non-combinatorial algorithm [1] has a better $n^{2-1/O(\log c)}$ time bound but is only for the Boolean case, leading to researchers to ask whether the same result holds for the more general problem for real input. In one section of Chan and Williams' paper [9], such a result was obtained but only for $d \approx 2^{\Theta(\sqrt{\log n})}$.

In Section 3, we resolve this question by giving a black-box reduction from the real case to the Boolean case, in particular, yielding $n^{2-1/O(\log c)}$ time for any $d = c\log n \ll 2^{\Theta(\sqrt{\log n})}$.

This equivalence between general dominance searching and the Boolean orthogonal vectors problem is noteworthy, since the Boolean orthogonal vectors problem has been used recently as a basis for many conditional hardness results in P.

As one immediate application, we can now solve the *integer linear programming* problem on n variables and cn constraints in $2^{(1-1/O(\log c))n}$ time, improving Impagliazzo et al.'s $2^{(1-1/\operatorname{poly}(c))n}$ algorithm [17].

Our new reduction is simple, this time, using a range-tree-like recursion.

Approximate ℓ_∞ nearest neighbor searching. So far, our discussion has been focused on exact algorithms. We now turn to ℓ_∞ nearest neighbor searching in the *approximate* setting. By known reductions (ignoring polylogarithmic factors) [16], it suffices to consider the fixed-radius decision problem: deciding whether the nearest neighbor distance is approximately less than a fixed value. Indyk [18] provided the best data structure for the problem, achieving $O(\log_\rho \log d)$ approximation factor, $O(dn^\rho \log n)$ preprocessing time, $O(dn^\rho)$ space, and $O(d\log n)$ query time for any ρ ranging from 1 to $\log d$. The data structure is actually based on traditional-style geometric divide-and-conquer. Andoni, Croitoru, and Pătraşcu [5] proved a nearly matching lower bound.

In Section 4, we improve the approximation factor of Indyk's data structure to $O(\log_\rho \log c)$ for dimension $d = c\log n$, for any ρ ranging from $1+\delta$ to $\log c$ (as an unintended byproduct, we also improve Indyk's query time to $O(d)$). The improvement in the approximation factor is noticeable when the dimension is close to logarithmic. It does not contradict Andoni et al.'s lower bound [5], since their proof assumed $d \gg \log^{1+\Omega(1)} n$.

For example, by setting $\rho \approx \log c$, we get $O(1)$ approximation factor, $n^{O(\log c)}$ preprocessing time/space, and $O(d)$ query time. By dividing into $n^{1-\alpha}$ groups of size n^α, we can lower the preprocessing time/space to $n^{1-\alpha}\cdot(n^\alpha)^{O(\log(c/\alpha))}$ while increasing the query time to $O(dn^{1-\alpha})$. Setting $\alpha \approx 1/\log c$, we can thus answer n (online) queries with $O(1)$ approximation factor in $n^{2-1/O(\log c)}$ total time, which curiously matches our earlier result for exact ℓ_∞ nearest neighbor search but by a purely combinatorial algorithm.

In the full paper, we also provide an alternative data structure with linear space but a larger $O(c^{(1-\rho)/\rho^2})$ approximation factor, and $O(dn^{\rho+\delta})$ query time for any $\rho \in (\delta, 1-\delta)$.

The idea is to modify Indyk's method to incorporate, once again, a range-tree-like recursion.

2 Online Dominance Range Searching

In this section, we study data structures for online orthogonal range searching in the reporting version (counting or detection can be dealt with similarly), using only combinatorial techniques without fast matrix multiplication. By doubling the dimension (footnote 1), it suffices to consider the dominance case.

Our data structure is an augmented, randomized lopsided variant of the k-d tree, where each node contains secondary structures for various lower-dimensional projections of the input.

Data structure. Let $\delta \in (0,1)$ and $c \in [\delta C_0, (\delta/C_0)\log N/\log^2 \log N]$ be user-specified parameters, for a sufficiently large constant C_0, where N is a fixed upper bound on the size of the input point set. Let $b \geq 2$ and $\alpha \in (0, 1/2)$ be parameters to be chosen later.

Given a set P of $n \leq N$ data points in $d \leq c \log N$ dimensions, our data structure is simple and is constructed as follows:

0. If $n \leq 1/\alpha$ or $d = 0$, then just store the given points.
1. Otherwise, let $\mathcal{J}$ be the collection of all subsets of $\{1, \ldots, d\}$ of size $\lfloor d/b \rfloor$. Then $|\mathcal{J}| = \binom{d}{\lfloor d/b \rfloor} = b^{O(d/b)}$. For each $J \in \mathcal{J}$, recursively[3] construct a data structure for the projection P_J of P that keeps only the coordinate positions in J.
2. Pick a random $i^* \in \{1, \ldots, d\}$. Let $\mu(i^*)$ be the $\lceil (1-\alpha)n \rceil$-th smallest i^*-th coordinate value in P; let $p(i^*)$ be the corresponding point in P. Store n, i^*, and $p(i^*)$. Recursively construct data structures for
   - the subset P_L of all points in P with i^*-th coordinate less than $\mu(i^*)$, and
   - the subset P_R of all points in P with i^*-th coordinate greater than $\mu(i^*)$.

Analysis. The preprocessing time and space satisfy the recurrence

$$T_d(n) \leq T_d(\lfloor \alpha n \rfloor) + T_d(\lfloor (1-\alpha)n \rfloor) + b^{O(d/b)} T_{\lfloor d/b \rfloor}(n) + O(n),$$

with $T_d(n) = O(n)$ for the base case $n \leq 1/\alpha$ or $d = 0$. This solves to

$$\begin{aligned} T_d(N) &\leq b^{O(d/b + d/b^2 + \cdots)} N (\log_{1/(1-\alpha)} N)^{O(\log_b d)} \\ &= b^{O(d/b)} N ((1/\alpha)\log N)^{O(\log_b d)} \\ &= N^{1+O((c/b)\log b)} 2^{O(\log((1/\alpha)\log N)\log_b d)} \leq N^{1+O(\delta)} 2^{O(\log^2((1/\alpha)\log N))} \end{aligned}$$

by setting $b := (c/\delta)\log(c/\delta)$.

Query algorithm. Given the preprocessed set P and a query point $q = (q_1, \ldots, q_d)$, our query algorithm proceeds as follows.

0. If $n \leq 1/\alpha$ or $d = 0$, then answer the query directly by brute-force search.
1. Otherwise, let $J_q = \{i \in \{1, \ldots, d\} : q_i \neq \infty\}$. If $|J_q| \leq d/b$, then recursively answer the query for P_{J_q} and the projection of q with respect to J_q.
2. Else,
   - if $q_{i^*} \leq \mu(i^*)$, then recursively answer the query for P_L and q;
   - if $q_{i^*} > \mu(i^*)$, then recursively answer the query for P_R and q, and recursively answer the query for P_L and $q' = (q_1, \ldots, q_{i^*-1}, \infty, q_{i^*+1}, \ldots, q_d)$;
   - in addition, if q dominates $p(i^*)$, then output $p(i^*)$.

[3] There are other options beside recursion here; for example, we could just use a range tree for P_J.

Analysis. We assume that the query point q is independent of the random choices made during the preprocessing of P. Let $L_q = \{i \in \{1, \ldots, d\} : \mu(i) < q_i \neq \infty\}$. Let $j = |J_q|$ and $\ell = |L_q|$.

Suppose that $j > d/b$. The probability that we make a recursive call for P_R is equal to $\Pr[(i^* \in L_q) \vee (i^* \notin J_q)] = \ell/d + (1 - j/d)$. We always make a recursive call for P_L, either for q or a point q' with $j - 1$ non-∞ values; the probability of the latter is equal to $\Pr[i^* \in L_q] = \ell/d$.

Hence, the expected number of leaves in the recursion satisfies the following recurrence:

$$Q_{d,j}(n) \leq \begin{cases} Q_{\lfloor d/b \rfloor, j}(n) & \text{if } j \leq d/b \\ \max_{\ell \leq j} \left[\left(\frac{\ell}{d} + 1 - \frac{j}{d}\right) Q_{d,j}(\lfloor \alpha n \rfloor) + \left(\frac{\ell}{d}\right) Q_{d,j-1}(\lfloor (1-\alpha) n \rfloor) \right. & \\ \qquad \left. + \left(1 - \frac{\ell}{d}\right) Q_{d,j}(\lfloor (1-\alpha) n \rfloor) \right] & \text{if } j > d/b, \end{cases} \tag{1}$$

with $Q_{d,j}(n) = 1$ for the base case $n \leq 1/\alpha$ or $d = 0$.

This recurrence looks complicated. Following [8], one way to solve it is by "guessing". We guess that

$$Q_{d,j}(n) \leq (1+\gamma)^j n^{1-\varepsilon}$$

for some choice of parameters $\gamma, \varepsilon \in (0, 1/2)$ to be specified later. We verify the guess by induction.

The base case $n \leq 1/\alpha$ or $d = 0$ is trivial. Assume that the guess is true for lexicographically smaller tuples (d, j, n). For $j \leq d/b$, the induction trivially goes through. So assume $j > d/b$. Let ℓ be the index that attains the maximum in (1). Then

$$\begin{aligned} Q_{d,j}(n) &\leq \left(\frac{\ell}{d} + 1 - \frac{j}{d}\right)(1+\gamma)^j (\alpha n)^{1-\varepsilon} + \left(\frac{\ell}{d}\right)(1+\gamma)^{j-1}((1-\alpha)n)^{1-\varepsilon} + \\ &\quad \left(1 - \frac{\ell}{d}\right)(1+\gamma)^j ((1-\alpha)n)^{1-\varepsilon} \\ &= \left[\left(\frac{\ell}{d} + 1 - \frac{j}{d}\right)\alpha^{1-\varepsilon} + \left(\frac{\ell}{d} \cdot \frac{1}{1+\gamma} + 1 - \frac{\ell}{d}\right)(1-\alpha)^{1-\varepsilon}\right](1+\gamma)^j n^{1-\varepsilon} \\ &\leq \left[\left(1 - \frac{j-\ell}{d}\right)\alpha^{1-\varepsilon} + \left(1 - \frac{\gamma\ell}{2d}\right)(1-\alpha)^{1-\varepsilon}\right](1+\gamma)^j n^{1-\varepsilon} \\ &\leq (1+\gamma)^j n^{1-\varepsilon}. \end{aligned}$$

For the last inequality, we need to upper-bound the following expression by 1:

$$\left(1 - \frac{j-\ell}{d}\right)\alpha^{1-\varepsilon} + \left(1 - \frac{\gamma\ell}{2d}\right)(1-\alpha)^{1-\varepsilon}. \tag{2}$$

- **Case I:** $j - \ell > d/(2b)$. Then (2) is at most

$$\begin{aligned} \left(1 - \frac{1}{2b}\right)\alpha^{1-\varepsilon} + (1-\alpha)^{1-\varepsilon} &\leq \left(1 - \frac{1}{2b}\right)\alpha e^{\varepsilon \ln(1/\alpha)} + 1 - (1-\varepsilon)\alpha \\ &\leq \left(1 - \frac{1}{2b}\right)\alpha(1 + 2\varepsilon \log(1/\alpha)) + 1 - (1-\varepsilon)\alpha \\ &\leq 1 - \frac{\alpha}{2b} + 3\alpha\varepsilon \log(1/\alpha), \end{aligned}$$

which is indeed at most 1 by setting $\varepsilon := 1/(6b \log(1/\alpha))$.

- **Case II**: $\ell > d/(2b)$. Then (2) is at most
$$\begin{aligned}\alpha^{1-\varepsilon} + 1 - \frac{\gamma}{4b} &\le \alpha e^{\varepsilon \ln(1/\alpha)} + 1 - \frac{\gamma}{4b} \\ &\le \alpha(1 + 2\varepsilon \log(1/\alpha)) + 1 - \frac{\gamma}{4b} \\ &\le 2\alpha + 1 - \frac{\gamma}{4b},\end{aligned}$$
which is indeed at most 1 by setting $\gamma := 8b\alpha$.

  We can set $\alpha := 1/b^4$, for example. Then $\gamma = O(1/b^3)$. We conclude that

$$Q_d(N) \le (1+\gamma)^d N^{1-\varepsilon} \le e^{\gamma d} N^{1-\varepsilon} \le N^{1-\varepsilon+O(c\gamma)} \le N^{1-1/O(b\log b)}.$$

Now, $Q_d(N)$ only counts the number of leaves in the recursion. The recursion has depth $O(\log_{1/(1-\alpha)} N + \log d)$. Each internal node of the recursion has cost $O(d)$, and each leaf has cost $O(d/\alpha)$, excluding the cost of outputting points (which occurs during the base case $d = 0$). Thus, the actual expected query time can be bounded by $Q_d(N)(bd \log N)^{O(1)}$, which is $N^{1-1/O(b\log b)}$ for $b \ll \log N/\log^2 \log N$. As $b = (c/\delta)\log(c/\delta)$, the bound is $N^{1-1/O((c/\delta)\log^2(c/\delta))}$.

Slight improvement of one $\log(c/\delta)$ factor in the exponent is possible, by an interesting application of *covering designs*. The details are explained in Appendix A.1. Thus:

▶ **Theorem 1.** *Let $\delta > 0$ be any fixed constant and $c \in [C_1, (1/C_1)\log N/\log^2 \log N]$ for a sufficiently large constant C_1. Given N points in $d = c \log N$ dimensions, we can construct a data structure in $O(N^{1+\delta})$ preprocessing time and space, so that for any query point, we can answer a dominance range reporting query in $N^{1-1/O(c\log c)} + O(K)$ expected time where K is the number of reported points. For dominance range counting, we get the same time bound but without the K term.*

We mention one application to online (min,+) matrix-vector multiplication. The corollary below follows immediately from a simple reduction [10] to d instances of d-dimensional dominance range reporting with disjoint output.[4]

▶ **Corollary 2.** *Let $\delta > 0$ be any fixed constant and $d = (1/C_1)\log^2 N/\log^2 \log N$ for a sufficiently large constant C_1. We can preprocess an $N \times d$ real-valued matrix A in $O(N^{1+\delta})$ time, so that given a query real-valued d-dimensional vector x, we can compute the (min,+)-product of A and x in $O(N)$ expected time.*

Applying the above corollary N/d times yields:

▶ **Corollary 3.** *Let $\delta > 0$ be any fixed constant. We can preprocess an $N \times N$ real-valued matrix A in $O(N^{2+\delta})$ time, so that given a query N-dimensional real-valued vector x, we can compute the (min,+)-product of A and x in $O((N^2/\log^2 N)\log^2 \log N)$ expected time.*

A similar result was obtained by Williams [25] for online *Boolean* matrix-vector multiplication. Recently Larsen and Williams [19] have found a faster algorithm, in the Boolean case, but it is not combinatorial, requires amortization, and does not deal with the rectangular matrix case in Corollary 2.

In Appendix A.2, we further show how to reduce the $O(K)$ term in Theorem 1 by about a logarithmic factor in the offline case, by modifying the algorithm to incorporate

[4] For any $j_0 \in \{1, \ldots, d\}$, the key observation is that $\min_{j=1}^d (a_{ij} + x_j) = a_{ij_0} + x_{j_0}$ iff $(a_{ij_0} - a_{i1}, \ldots, a_{ij_0} - a_{id})$ is dominated by $(x_1 - x_{j_0}, \ldots, x_d - x_{j_0})$ in $\mathbb{R}^d$.

bit-packing tricks. This has applications to speeding up combinatorial algorithms for (min,+) matrix-matrix multiplication and all-pairs shortest paths.

In the full paper, we note that the method can be simplified in the Boolean case – the data structure becomes just an augmented, randomized variant of the *trie*. This has an application to combinatorial algorithms for Boolean matrix multiplication.

3 Offline Dominance Range Searching

In this section, we study the offline orthogonal range searching problem in the counting version (which includes the detection version), allowing the use of fast matrix multiplication. By doubling the dimension (footnote 1), it suffices to consider the dominance case: given n data/query points in $\mathbb{R}^d$, we want to count the number of data points dominated by each query point. We describe a black-box reduction of the real case to the Boolean case.

We use a recursion similar to a degree-s range tree (which bears some resemblance to a low-dimensional algorithm from [13]).

Algorithm. Let $\delta \in (0,1)$ and s be parameters to be set later. Let $[s]$ denote $\{0,1,\ldots,s-1\}$.

Given a set P of $n \le N$ data/query points in $\mathbb{R}^j \times [s]^{d-j}$, with $d \le c \log N$, our algorithm is simple and proceeds as follows:

0. If $j = 0$, then all points are in $[s]^d$ and we solve the problem directly by mapping each point $(p_1,\ldots,p_d)$ to a binary string $1^{p_1}0^{s-p_1}\cdots 1^{p_d}0^{s-p_d} \in \{0,1\}^{ds}$ and running a known Boolean offline dominance algorithm in ds dimensions.

1. Otherwise, for each $i \in [s]$, recursively solve the problem for the subset P_i of all points in P with ranks from $i(n/s)+1$ to $(i+1)(n/s)$ in the j-th coordinate.

2. "Round" the j-th coordinate values of all data points in P_i to $i+1$ and all query points in P_i to i, and recursively solve the problem for P after rounding (which now lies in $\mathbb{R}^{j-1} \times [s]^{d-j+1}$); add the results to the existing counts of all the query points.

Analysis. Suppose that the Boolean problem for n points in $d \le c \log n$ dimensions can be solved in $d^C n^{2-f(c)}$ time for some absolute constant $C \ge 1$ and some function $f(c) \in [0, 1/4]$. The following recurrence bounds the total cost of the leaves of the recursion in our algorithm (assuming that n is a power of s, for simplicity):

$$T_{d,j}(n) = s\,T_{d,j}(n/s) + T_{d,j-1}(n).$$

For the base cases, $T_{d,j}(1) = 1$; and if $n > \sqrt{N}$, then $T_{d,0}(n) \le (ds)^C n^{2-f(2cs)}$ (since the Boolean subproblems have dimension $ds \le cs \log N \le 2cs \log n$). On the other hand, if $n \le \sqrt{N}$, we can use brute force to get $T_{d,0}(n) \le dn^2 \le dn^{3/2}N^{1/4}$. In any case, $T_{d,0}(n) \le (ds)^C n^{3/2} N^{1/2-f(2cs)} = An^{3/2}$ where we let $A := (ds)^C N^{1/2-f(2cs)}$.

One way[5] to solve this recurrence is again by "guessing". We guess that

$$T_{d,j}(n) \le (1+\gamma)^j An^{3/2}$$

for some choice of parameter $\gamma \in (0,1)$ to be determined later. We verify the guess by induction.

[5] Since this particular recurrence is simple enough, an alternative, more direct way is to expand $T_{d,d}(N)$ into a sum $\sum_{i\ge 0}\binom{d+i}{i}s^i T_{d,0}(N/s^i) \le \sum_{i\ge 0} O(\frac{d+i}{i\sqrt{s}})^i \cdot AN^{3/2}$, and observe that the maximum term occurs when i is near $d/\sqrt{s}\ldots$

The base cases are trivial. Assume that the guess is true for lexicographically smaller (j, n). Then

$$\begin{aligned} T_{d,j}(n) &\leq (1+\gamma)^j As(n/s)^{3/2} + (1+\gamma)^{j-1} An^{3/2} \\ &= \left[\frac{1}{\sqrt{s}} + \frac{1}{1+\gamma}\right](1+\gamma)^j An^{3/2} \leq (1+\gamma)^j An^{3/2}, \end{aligned}$$

provided that

$$\frac{1}{\sqrt{s}} + \frac{1}{1+\gamma} \leq 1,$$

which is true by setting $\gamma := 2/\sqrt{s}$.

We can set $s := c^4$, for example. Then $\gamma = O(1/c^2)$. We conclude that

$$\begin{aligned} T_{d,d}(N) &\leq (1+\gamma)^d AN^{3/2} \leq e^{\gamma d}(ds)^{O(1)} N^{2-f(2cs)} \\ &\leq (ds)^{O(1)} N^{2-f(2cs)+O(\gamma c)} \\ &= d^{O(1)} N^{2-f(2c^5)+O(1/c)}. \end{aligned}$$

Now, $T_{d,d}(N)$ excludes the cost at internal nodes of the recursion. Since the recursion has depth at most $\log_s N + d$, the actual running time can be bounded by $T_{d,d}(n)(d \log N)^{O(1)}$.

Abboud, Williams, and Yu's algorithm [1] for the Boolean case, as derandomized by Chan and Williams [9], achieves $f(c) = 1/O(\log c)$, yielding an overall time bound of $N^{2-1/O(\log c)}(d \log N)^{O(1)}$, which is $N^{2-1/O(\log c)}$ for $\log c \ll \sqrt{\log N}$.

▶ **Theorem 4.** *Let $c \in [1, 2^{(1/C_1)\sqrt{\log N}}]$ for a sufficiently large constant C_1. Given N points in $d = c \log N$ dimensions, we can answer N offline dominance range counting queries in $N^{2-1/O(\log c)}$ time.*

We remark that if the Boolean problem could be solved in truly subquadratic time $d^{O(1)}N^{2-\varepsilon}$, then the above analysis (with $s := (c \log N)^2$, say) would imply that the general problem could be solved in truly subquadratic time with the *same* ε, up to $(d \log N)^{O(1)}$ factors.

4 Approximate ℓ_∞ Nearest Neighbor Searching

In this section, we study (online, combinatorial) data structures for t-approximate ℓ_∞ nearest neighbor search. By known reductions [16, 18], it suffices to solve the fixed-radius approximate decision problem, say, for radius $r = 1/2$: given a query point q, we want to find a data point of distance at most distance $t/2$ from q, under the promise that the nearest neighbor distance is at most $1/2$.

Our solution closely follows Indyk's divide-and-conquer method [18], with a simple modification that incorporates a range-tree-like recursion.

Data structure. Let $\delta \in (0, 1)$, $\rho > 1$, and $c \geq 4$ be user-specified parameters. Let s and k be parameters to be chosen later.

Given a set P of $n \leq N$ data points in $d \leq c \log N$ dimensions, our data structure is constructed as follows:

0. If $n \leq s$ or $d = 0$, then just store the points in P.

Otherwise, compute and store the median first coordinate μ in P. Let $P_{>i}$ (resp. $P_{<i}$) denote the subset of all points in P with first coordinate greater than (resp. less than) $\mu + i$. Let $\alpha_i := |P_{>i}|/n$ and $\beta_i := |P_{<-i}|/n$. Note that the α_i's and β_i's are decreasing sequences with $\alpha_0 = \beta_0 = 1/2$.

1. If $\alpha_k > 1/s$ and $\alpha_{i+1} > \alpha_i^\rho$ for some $i \in \{0, 1, \ldots, k-1\}$, then set $type = (1, i)$ and recursively construct a data structure for $P_{>i}$ and for $P_{<i+1}$.
2. Else if $\beta_k > 1/s$ and $\beta_{i+1} > \beta_i^\rho$ for some $i \in \{0, 1, \ldots, k-1\}$, then set $type = (2, i)$ and recursively construct a data structure for $P_{<-i}$ and for $P_{>-(i+1)}$.
3. Else if $\alpha_k, \beta_k \leq 1/s$, then set $type = 3$ and recursively construct a data structure for
   - the set $P_{>k} \cup P_{<-k}$ and
   - the $(d-1)$-dimensional projection of $P - (P_{>k+1} \cup P_{<-(k+1)})$ that drops the first coordinate (this recursion in $d-1$ dimensions is where our algorithm differs from Indyk's).

We set $k := \lceil \log_\rho \log s \rceil$. Then one of the tests in steps 1–3 must be true. To see this, suppose that $\alpha_k > 1/s$ (the scenario $\beta_k > 1/s$ is symmetric), and suppose that i does not exist in step 1. Then $\alpha_k \leq (1/2)^{\rho^k} \leq 1/s$, a contradiction.

Analysis. The space usage is proportional to the number of points stored at the leaves in the recursion, which satisfies the following recurrence (by using the top expression with $(\alpha, \alpha') = (\alpha_i, \alpha_{i+1})$ for step 1 or $(\alpha, \alpha') = (\beta_i, \beta_{i+1})$ for step 2, or the bottom expression for step 3):

$$S_d(n) \leq \max \begin{cases} \max\limits_{\alpha,\alpha':\ \alpha' > \alpha^\rho,\ 1/s < \alpha' \leq \alpha \leq 1/2} [S_d(\alpha n) + S_d((1-\alpha')n)] \\ S_d(2n/s) + S_{d-1}(n), \end{cases} \tag{3}$$

with $S_d(n) = n$ for the base case $n \leq s$ or $d = 0$.

We guess that

$$S_d(n) \leq (1+\gamma)^d n^\rho$$

for some choice of parameter $\gamma \in (0, 1)$. We verify the guess by induction.

The base case is trivial. Assume that the guess is true for lexicographically smaller (d, n).

- **Case I**: the maximum in (3) is attained by the top expression and by α, α'. Then
$$\begin{aligned} S_d(n) &\leq (1+\gamma)^d [(\alpha n)^\rho + ((1-\alpha')n)^\rho] \\ &\leq [\alpha^\rho + 1 - \alpha'] (1+\gamma)^d n^\rho \\ &\leq (1+\gamma)^d n^\rho. \end{aligned}$$
- **Case II**: the maximum in (3) is attained by the bottom expression. Then
$$\begin{aligned} S_d(n) &\leq (1+\gamma)^d (2n/s)^\rho + (1+\gamma)^{d-1} n^\rho \\ &\leq \left[\left(\frac{2}{s}\right)^\rho + \frac{1}{1+\gamma} \right] (1+\gamma)^d n^\rho \\ &\leq (1+\gamma)^d n^\rho \end{aligned}$$
by setting $s := 2(2/\gamma)^{1/\rho}$.

Set $\gamma := \delta/c$. Then $s = O((c/\delta)^{1/\rho})$ and $k = \log_\rho \log(c/\delta) + O(1)$. We conclude that

$$S_d(N) \ \leq \ e^{\gamma d} N^\rho \ \leq \ N^{\rho + O(\gamma c)} \ = \ N^{\rho + O(\delta)}.$$

For the preprocessing time, observe that the depth of the recursion is $h := O(\log_{s/(s-1)} N + d)$ (since at each recursive step, the size of the subsets drops by a factor of $1 - 1/s$ or the dimension decreases by 1). Now, $h = O(s \log N + d) \leq O((c/\delta) \log N + d) = O((c/\delta) \log N)$. Hence, the preprocessing time can be bounded by $O(S_d(N)h) = O((c/\delta) N^{\rho+\delta} \log N)$.

Query algorithm. Given the preprocessed set P and a query point $q = (q_1, \ldots, q_d)$, our query algorithm proceeds as follows:

0. If $n \le s$ or $d = 0$, then answer the query directly by brute-force search.
1. If $type = (1, i)$: if $q_1 > i + 1/2$, then recursively answer the query in $P_{>i}$, else recursively answer the query in $P_{<i+1}$.
2. If $type = (2, i)$: proceed symmetrically.
3. If $type = 3$:
- if $q_1 > k + 1/2$ or $q_1 < -(k + 1/2)$, then recursively answer the query in $P_{>k} \cup P_{<-k}$;
- else recursively answer the query in $P - (P_{>k+1} \cup P_{<-(k+1)})$, after dropping the first coordinate of q.

Note that in the last subcase of step 3, any returned point has distance at most $2k + 3/2$ from q in terms of the first coordinate. By induction, the approximation factor t is at most $4k + 3 = O(\log_\rho \log(c/\delta))$.

Analysis. The query time is clearly bounded by the depth h, which is $O((c/\delta) \log N)$.

▶ **Theorem 5.** *Let $\delta > 0$ be any fixed constant. Let $\rho > 1$ and $c \ge \Omega(1)$. Given N points in $d = c \log N$ dimensions, we can construct a data structure in $O(dN^{\rho+\delta})$ time and $O(dN + N^{\rho+\delta})$ space, so that we can handle the fixed-radius decision version of approximate ℓ_∞ nearest neighbor queries in $O(d)$ time with approximation factor $O(\log_\rho \log c)$.*

References

1 Amir Abboud, Ryan Williams, and Huacheng Yu. More applications of the polynomial method to algorithm design. In *Proc. 26th ACM–SIAM Symp. Discrete Algorithms (SODA)*, pages 218–230, 2015.
2 Peyman Afshani, Timothy M. Chan, and Konstantinos Tsakalidis. Deterministic rectangle enclosure and offline dominance reporting on the RAM. In *Proc. 41st Int'l Colloq. Automata, Languages, and Programming (ICALP), Part I*, pages 77–88, 2014.
3 Josh Alman, Timothy M. Chan, and Ryan Williams. Polynomial representation of threshold functions with applications. In *Proc. 57th IEEE Symp. Found. Comput. Sci. (FOCS)*, pages 467–476, 2016.
4 Josh Alman and Ryan Williams. Probabilistic polynomials and Hamming nearest neighbors. In *Proc. 56th IEEE Symp. Found. Comput. Sci. (FOCS)*, pages 136–150, 2015.
5 Alexandr Andoni, Dorian Croitoru, and Mihai M. Pătraşcu. Hardness of nearest neighbor under L_∞. In *Proc. 49th IEEE Symp. Found. Comput. Sci. (FOCS)*, pages 424–433, 2008.
6 V. Z. Arlazarov, E. A. Dinic, M. A. Kronrod, and I. A. Faradzhev. On economical construction of the transitive closure of a directed graph. *Soviet Mathematics Doklady*, 11:1209–1210, 1970.
7 Timothy M. Chan. Geometric applications of a randomized optimization technique. *Discrete Comput. Geom.*, 22(4):547–567, 1999.
8 Timothy M. Chan. Speeding up the Four Russians algorithm by about one more logarithmic factor. In *Proc. 26th ACM–SIAM Symp. Discrete Algorithms (SODA)*, pages 212–217, 2015.
9 Timothy M. Chan and Ryan Williams. Deterministic APSP, orthogonal vectors, and more: Quickly derandomizing Razborov–Smolensky. In *Proc. 27th ACM–SIAM Symp. Discrete Algorithms (SODA)*, pages 1246–1255, 2016.
10 T. M. Chan. All-pairs shortest paths with real weights in $O(n^3/\log n)$ time. *Algorithmica*, 50:236–243, 2008.

11 T. M. Chan. More algorithms for all-pairs shortest paths in weighted graphs. *SIAM J. Comput.*, 39:2075–2089, 2010.

12 T. M. Chan, K. G. Larsen, and M. Pătraşcu. Orthogonal range searching on the RAM, revisited. In *Proc. 27th ACM Symp. Comput. Geom. (SoCG)*, pages 1–10, 2011.

13 T. M. Chan and M. Pătraşcu. Counting inversions, offline orthogonal range counting, and related problems. In *Proc. 21st ACM–SIAM Symp. Discrete Algorithms (SODA)*, pages 161–173, 2010.

14 Daniel M. Gordon, Oren Patashnik, Greg Kuperberg, and Joel Spencer. Asymptotically optimal covering designs. *J. Combinatorial Theory, Series A*, 75(2):270–280, 1996.

15 Y. Han and T. Takaoka. An $O(n^3 \log\log n/\log^2 n)$ time algorithm for all pairs shortest paths. In *Proc. 13th Scand. Symp. and Workshops on Algorithm Theory (SWAT)*, pages 131–141, 2012.

16 Sariel Har-Peled, Piotr Indyk, and Rajeev Motwani. Approximate nearest neighbor: Towards removing the curse of dimensionality. *Theory Comput.*, 8(1):321–350, 2012.

17 R. Impagliazzo, S. Lovett, R. Paturi, and S. Schneider. 0-1 integer linear programming with a linear number of constraints, 2014.

18 Piotr Indyk. On approximate nearest neighbors under l_∞ norm. *J. Comput. Sys. Sci.*, 63(4):627–638, 2001.

19 Kasper Green Larsen and Ryan Williams. Faster online matrix-vector multiplication. In *Proc. 28th ACM–SIAM Symp. Discrete Algorithms (SODA)*, pages 2182–2189, 2017.

20 François Le Gall. Faster algorithms for rectangular matrix multiplication. In *Proc. 53rd IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 514–523, 2012.

21 Jirí Matoušek. Computing dominances in E^n. *Inform. Process. Lett.*, 38(5):277–278, 1991.

22 F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction*. Springer–Verlag, 1985.

23 R. Williams. Faster all-pairs shortest paths via circuit complexity. In *Proc. 46th ACM Symp. Theory Comput. (STOC)*, pages 664–673, 2014.

24 Ryan Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theor. Comput. Sci.*, 348(2-3):357–365, 2005.

25 Ryan Williams. Matrix-vector multiplication in sub-quadratic time (some preprocessing required). In *Proc. 18th ACM–SIAM Symp. Discrete Algorithms (SODA)*, pages 995–1001, 2007. URL: `http://dl.acm.org/citation.cfm?id=1283383.1283490`.

26 Huacheng Yu. An improved combinatorial algorithm for Boolean matrix multiplication. In *Proc. 42nd Int'l Colloq. Automata, Languages, and Programming (ICALP), Part I*, pages 1094–1105, 2015.

A Online Dominance Range Searching (Continued)

A.1 Slightly Improved Version

We now describe a small improvement to the data structure in Section 2. The idea is to replace $\mathcal{J}$ with a collection of slightly larger subsets, but with fewer subsets, so that any set J_q of size $t := \lfloor d/b \rfloor$ is covered by some subset in $J \in \mathcal{J}$. Such a collection is called a *covering design* (e.g., see [14]), which can be constructed easily by random sampling, as explained in see part (i) of the lemma below. In our application, we also need a good time bound for finding such a $J \in \mathcal{J}$ for a given query set J_q; this is addressed in part (ii) of the lemma. (Proofs are deferred to the full paper.)

▶ **Lemma 6** (Covering designs). *Given numbers $v \geq k \geq t$ and N, and given a size-v ground set V,*

(i) *we can construct a collection $\mathcal{J}$ of at most $\left(\binom{v}{t}/\binom{k}{t}\right) \ln N$ size-k subsets of V in $O(v|\mathcal{J}|)$ time, so that given any query size-t subset $J_q \subset V$, we can find a subset $J \in \mathcal{J}$ containing J_q in $O(v|\mathcal{J}|)$ time with success probability at least $1 - 1/N$;*

(ii) *alternatively, with a larger collection $\mathcal{J}$ of at most $\left(\binom{v}{t}/\binom{k}{t}\right)^2 \ln^2(vN)$ subsets, we can reduce the query time to $O(v^3 \log^2(vN))$.*

We now modify the data structure in Section 2 as follows. In step 1, we change $\mathcal{J}$ to a collection of size-$\lfloor d/2 \rfloor$ subsets of $\{1, \ldots, d\}$ obtained from Lemma 6(ii) with $(v, k, t) = (d, \lfloor d/2 \rfloor, \lfloor d/b \rfloor)$. Then $|\mathcal{J}| \leq \left(\binom{d}{\lfloor d/b \rfloor}/\binom{\lfloor d/2 \rfloor}{\lfloor d/b \rfloor}\right)^2 \ln^2(dN) \leq 2^{O(d/b)} \log^2 N$. The recurrence for the preprocessing time and space then improves to

$$T_d(n) \;\leq\; T_d(\lfloor \alpha n \rfloor) + T_d(\lfloor (1-\alpha)n \rfloor) + (2^{O(d/b)} \log^2 N) T_{\lfloor d/b \rfloor}(n) + O(n),$$

which solves to $T_d(N) \leq 2^{O(d/b + d/b^2 + \cdots)} N (\log_{1/(1-\alpha)} N)^{O(\log_b d)} \leq N^{1+O(\delta)} 2^{O(\log^2((1/\alpha)\log N))}$, this time by setting $b := c/\delta$ (instead of $b := (c/\delta)\log(c/\delta)$).

In the query algorithm, we modify step 1 by finding a set $J \in \mathcal{J}$ containing J_q by Lemma 6(ii) and recursively querying P_J (instead of P_{J_q}). If no such J exists, we can afford to switch to brute-force search, since this happens with probability less than $1/N$. The analysis of the recurrence for $Q_d(N)$ remains the same. Each internal node of the recursion now has cost $O(d^3 \log^2 N)$ by Lemma 6(ii); the extra factor will not affect the final bound. The overall query time is still $N^{1-1/O(b \log b)}$, which is now $N^{1-1/O((c/\delta)\log(c/\delta))}$.

A.2 Offline Packed-Output Version, with Application to APSP

In this subsection, we discuss how to refine the algorithm in Section 2, so that the output can be reported in roughly $O(K/\log n)$ time instead of $O(K)$ in the offline setting. The approach is to combine the algorithm with bit-packing tricks.

We assume a w-bit word RAM model which allows for certain exotic word operations. In the case of $w := \delta_0 \log N$ for a sufficiently small constant $\delta_0 > 0$, exotic operations can be simulated in constant time by table lookup; the precomputation of the tables requires only $N^{O(\delta_0)}$ time.

We begin with techniques to represent and manipulate sparse sets of integers in the word RAM model. Let z be a parameter to be set later. In what follows, an interval $[a, b)$ refers to the integer set $\{a, a+1, \ldots, b-1\}$. A *block* refers to an interval of the form $[kz, (k+1)z)$. Given a set S of integers over an interval I of length n, we define its *compressed representation* to be a doubly linked list of *mini-sets*, where for each of the $O(\lceil n/z \rceil)$ blocks B intersecting I (in sorted order), we store the *mini-set* $\{j \bmod z : j \in S \cap B\}$, which consists of small $(\log z)$-bit numbers and can be packed in $O((|S \cap B|/w)\log z + 1)$ words. The total number of words in the compressed representation is $O((|S|/w)\log z + n/z + 1)$. Proofs of the following facts can be found in the full paper.

▶ **Lemma 7** (Bit-packing tricks).

(i) *Given compressed representations of two sets S_1 and S_2 over two disjoint intervals, we can compute the compressed representation of $S_1 \cup S_2$ in $O(1)$ time.*

(ii) *Given compressed representations of $S_0, \ldots, S_{m-1} \subset [0, n)$, we can compute the compressed representations of $T_0, \ldots, T_{n-1} \subset [0, m)$ with $T_j = \{i : j \in S_i\}$ (called the* transposition *of $S_0, \ldots, S_{m-1}$), in $O((K/w)\log^2 z + mn/z + m + n + z)$ time, where $K = \sum_{i=0}^{m-1} |S_i|$.*

(iii) *Given compressed representations of $S_0, \ldots, S_{m-1} \subset [0, n)$ and a bijective function $\pi : [0, n) \to [0, n)$ which is evaluable in constant time, we can compute compressed representations of $\pi(S_1), \ldots, \pi(S_m)$ in $O((K/w) \log^2 z + mn/z + m + n + z)$ time, where $K = \sum_{i=0}^{m-1} |S_i|$.*

▶ **Theorem 8.** *Assume $z \leq N^{o(1)}$. Let $\delta > 0$ be any fixed constant and $c \in [C_1, (1/C_1) \log N / \log^2 \log N]$ for a sufficiently large constant C_1. Given a set P of N points in $d = c \log N$ dimensions, we can construct a data structure in $O(N^{1+\delta})$ preprocessing time and space, so that we can answer N offline dominance range reporting queries (with a compressed output representation) in $N^{2-1/O(c \log c)} + O(((K/w) \log^2 z + N^2/z) \log d)$ time where K is the total number of reported points over the N queries.*

Proof. We adapt the preprocessing and query algorithm in Section 2, with the improvement from Appendix A.1. A *numbering* of a set S of n elements refers to a bijection from S to n consecutive integers. For each point set P generated by the preprocessing algorithm, we define a numbering ϕ_P of P simply by recursively "concatenating" the numberings ϕ_{P_L} and ϕ_{P_R} and appending $p(i^*)$. The output to each query for P will be a compressed representation of the subset of dominated points after applying ϕ_P.

In step 2 of the query algorithm, we can union the output for P_L and for P_R in $O(1)$ time by Lemma 7(i). In step 1 of the query algorithm, we need additional work since the output is with respect to a different numbering ϕ_{P_J}, for some set $J \in \mathcal{J}$. For each $J \in \mathcal{J}$, we can change the compressed representation to follow the numbering ϕ_P by invoking Lemma 7(iii), after collecting all query points $Q(P_J)$ that are passed to P_J (since queries are offline). To account for the cost of this invocation to Lemma 7(iii), we charge (a) $(1/w) \log^2 z$ units to each output feature, (b) $1/z$ units to each point pair in $P_J \times Q(P_J)$, (c) 1 unit to each point in P_J, and (d) 1 unit to each point in $Q(P_J)$, and (e) z units to the point set P_J itself.

Each output feature or each point pair is charged $O(\log d)$ times, since d decreases to $\lfloor d/2 \rfloor$ with each charge. Thus, the total cost for (a) and (b) is $O((K/w) \log^2 z \log d + (N^2/z) \log d)$. The total cost of (c) is $N^{1+o(1)}$ by the analysis of our original preprocessing algorithm; similarly, the total cost of (e) is $zN^{1+o(1)}$. The total cost of (d) is $N^{2-1/O(c \log c)}$ by the analysis of our original query algorithm.

We can make the final compressed representations to be with respect to any user-specified numbering of P, by one last invocation to Lemma 7(iii). The algorithm can be derandomized, as noted in the full paper. ◀

One may wonder whether the previous range-tree-like offline algorithm by Impagliazzo et al. [17, 8] could also be adapted; the problem there is that d is only decremented rather than halved, which makes the cost of re-numbering too large.

The main application is to (min,+) matrix multiplication and all-pairs shortest paths (APSP). The corollary below follows immediately from a simple reduction [10] (see footnote 4) to d instances of d-dimensional offline dominance range reporting where the total output size K is $O(n^2)$. Here, we set $w := \delta_0 \log N$ and $z := \text{poly}(\log N)$.

▶ **Corollary 9.** *Let $d = (1/C_1) \log^2 N / \log^2 \log N$ for a sufficiently large constant C_1. Given an $N \times d$ and a $d \times N$ real-valued matrix, we can compute their (min,+)-product (with a compressed output representation) in $O((N^2/\log N) \log^3 \log N)$ expected time.*

The corollary below follows from applying Corollary 9 q/d times, in conjunction with a subroutine by Chan [11, Corollary 2.5]. (The result improves [11, Corollary 2.6].)

▶ **Corollary 10.** *Let $q = \log^3 N / \log^5 \log N$. Given an $N \times q$ and a $q \times N$ real-valued matrix, we can compute their (min,+)-product in $O(N^2)$ time.*

Applying Corollary 10 N/q times (and using a standard reduction from APSP to (min,+)-multiplication), we obtain:

▶ **Corollary 11.** *Given two $N \times N$ real-valued matrices, we can compute their (min,+)-product by a combinatorial algorithm in $O((N^3/\log^3 N)\log^5 \log N)$ time. Consequently, we obtain a combinatorial algorithm for APSP for arbitrary N-vertex real-weighted graphs with the same time bound.*

Note that Williams' algorithm [23] is faster (achieving $N^3/2^{\Omega(\sqrt{\log N})}$ time), but is non-combinatorial and gives a worse time bound ($O(N^2 \log^{O(1)} N)$) for the rectangular matrix case in Corollary 10.

Dynamic Orthogonal Range Searching on the RAM, Revisited*

Timothy M. Chan[1] and Konstantinos Tsakalidis[2]

1 **Dept. of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL, USA**
tmc@illinois.edu

2 **Cheriton School of Computer Science, University of Waterloo, Waterloo, Canada**
ktsakali@uwaterloo.ca

Abstract

We study a longstanding problem in computational geometry: 2-d dynamic orthogonal range reporting. We present a new data structure achieving $O\left(\frac{\log n}{\log\log n}+k\right)$ optimal query time and $O\left(\log^{2/3+o(1)} n\right)$ update time (amortized) in the word RAM model, where n is the number of data points and k is the output size. This is the first improvement in over 10 years of Mortensen's previous result [*SIAM J. Comput.*, 2006], which has $O\left(\log^{7/8+\varepsilon} n\right)$ update time for an arbitrarily small constant ε.

In the case of 3-sided queries, our update time reduces to $O\left(\log^{1/2+\varepsilon} n\right)$, improving Wilkinson's previous bound [ESA 2014] of $O\left(\log^{2/3+\varepsilon} n\right)$.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases dynamic data structures, range searching, computational geometry

Digital Object Identifier 10.4230/LIPIcs.SoCG.2017.28

1 Introduction

Orthogonal range searching is one of the most well-studied and fundamental problems in computational geometry: the goal is to design a data structure to store a set of n points so that we can quickly report all points inside a query axis-aligned rectangle. In the "emptiness" version of the problem, we just want to decide if the rectangle contains any point. (We will not study the counting version of the problem here.)

The static 2-d problem has been extensively investigated [15, 6, 25, 13, 11, 22, 1, 21], with the current best results in the word RAM model given by Chan, Larsen, and Pătraşcu [9].

In this paper, we are interested in the *dynamic* 2-d problem, allowing insertions and deletions of points. A straightforward dynamization of the standard *range tree* [27] supports queries in $O\left(\log^2 n + k\right)$ time and updates in $O\left(\log^2 n\right)$ time, where k denotes the number of reported points (for the emptiness problem, we can take $k = 0$). Mehlhorn and Näher [17] improved the query time to $O\left(\log n \log\log n + k\right)$ and the update time to $O\left(\log n \log\log n\right)$ by *dynamic fractional cascading*.

* This work was done while the first author was at the University of Waterloo, and was partially supported by an NSERC Discovery Grant.

The first data structure to achieve logarithmic query and update (amortized) time was presented by Mortensen [19]. In fact, he obtained *sub*logarithmic bounds in the word RAM model: the query time is $O\left(\frac{\log n}{\log\log n}+k\right)$ and the amortized update time is $O\left(\log^{7/8+\varepsilon} n\right)$ where ε denotes an arbitrarily small positive constant.

On the lower bound side, Alstrup et al. [2] showed that any data structure with t_u update time for 2-d range emptiness requires $\Omega\left(\frac{\log n}{\log(t_u \log n)}\right)$ query time in the cell-probe model. Thus, Mortensen's query bound is optimal for any data structure with polylogarithmic update time. However, it is conceivable that the update time could be improved further while keeping the same query time. Indeed, the $O\left(\log^{7/8+\varepsilon} n\right)$ update bound looks too peculiar to be optimal, one would think.

Let us remark how intriguing this type of "fractional-power-of-log" bound is, which showed up only on a few occasions in the literature. For example, Chan and Pătraşcu [10] gave a dynamic data structure for 1-d rank queries (counting number of elements less than a given value) with $O\left(\frac{\log n}{\log\log n}\right)$ query time and $O\left(\log^{1/2+\varepsilon} n\right)$ update time. Chan and Pătraşcu also obtained more $\sqrt{\log n}$-type results for various offline range counting problems. Another example is Wilkinson's recent paper [24]: he studied a special case of 2-d orthogonal range reporting for *2-sided* and *3-sided* rectangles and obtained a solution with $O\left(\frac{\log n}{\log\log n}+k\right)$ amortized query time, $O\left(\log^{1/2+\varepsilon} n\right)$ update time for the 2-sided case, and $O\left(\log^{2/3+\varepsilon} n\right)$ update time for 3-sided; the latter improves Mortensen's $O\left(\log^{5/6+\varepsilon} n\right)$ update bound for 3-sided [19]. He also showed that in the insertion-only and deletion-only settings, it is possible to get fractional-power-of-log bounds for both the update and the query time. However, he was unable to make progress for general 4-sided rectangles in the insertion-only and deletion-only settings, let alone the fully dynamic setting.

New results. Our main new result is a fully dynamic data structure for 2-d orthogonal range reporting with $O\left(\frac{\log n}{\log\log n}+k\right)$ optimal query time and $O\left(\log^{2/3+o(1)} n\right)$ update time, greatly improving Mortensen's $O\left(\log^{7/8+\varepsilon} n\right)$ bound. In the 3-sided case, we obtain $O\left(\log^{1/2+\varepsilon} n\right)$ update time, improving Wilkinson's $O\left(\log^{2/3+\varepsilon} n\right)$ bound. (See Table 1 for comparison.) Our update bounds seem to reach a natural limit with this type of approach. In particular, it is not unreasonable to conjecture that the near-$\sqrt{\log n}$ update bound for the 3-sided case is close to optimal, considering prior "fractional-power-of-log" upper-bound results in the literature (although there have been no known lower bounds of this type so far).

Like previous methods, our bounds are amortized (this includes query time). Our results are in the word-RAM model, under the standard assumption that the word size w is at least $\log n$ bits (in fact, except for an initial predecessor search during each query/update, we only need operations on $(\log n)$-bit words). Even to researchers uncomfortable with sublogarithmic algorithms on the word RAM, such techniques are still relevant. For example, Mortensen extended his data structure to $d \geq 3$ dimensions and obtained $O\left(\left(\frac{\log n}{\log\log n}\right)^{d-1}+k\right)$ query time and $O\left(\log^{d-9/8+\varepsilon} n\right)$ update time, even in the real-RAM model (where each word can hold an input real number or a $(\log n)$-bit number). Our result automatically leads to improvements in higher dimensions as well.

Table 1 Dynamic planar orthogonal range reporting: previous and new results.

		Update time	Query time
4-sided	Lueker and Willard [27]	$\log^2 n$	$\log^2 n + k$
	Mehlhorn and Näher [17]	$\log n \log\log n$	$\log n \log\log n + k$
	Mortensen [19]	$\log^{7/8+\varepsilon} n$	$\frac{\log n}{\log\log n} + k$
	New	$\log^{2/3} n \log^{O(1)} \log n$	$\frac{\log n}{\log\log n} + k$
3-sided	McCreight [16]	$\log n$	$\log n + k$
	Willard [26]	$\frac{\log n}{\log\log n}$	$\frac{\log n}{\log\log n} + k$
	Mortensen [19]	$\log^{5/6+\varepsilon} n$	$\frac{\log n}{\log\log n} + k$
	Wilkinson [24]	$(\log n \log\log n)^{2/3}$	$\log n + k$
	Wilkinson [24]	$\log^{2/3+\varepsilon} n$	$\frac{\log n}{\log\log n} + k$
	New	$\log^{1/2+\varepsilon} n$	$\frac{\log n}{\log\log n} + k$

Overview of techniques: Micro- and macro-structures. Our solution builds on ideas from Mortensen's paper [19]. His paper was long and not easy to follow, unfortunately; we strive for a clearer organization and a more accessible exposition (which in itself would be a valuable contribution).

The general strategy towards obtaining fractional-power-of-log bounds, in our view, can be broken into two parts: the design of what we will call *micro-structures* and *macro-structures*.

- Micro-structures refer to data structures for handling a small number s of points; by "small", we mean $s = 2^{\log^\alpha n}$ for some fraction $\alpha < 1$ (rather than s being polylogarithmic, as is more usual in other contexts). When s is small, by *rank space reduction* we can make the universe size small, and as a consequence pack multiple points (about $\frac{w}{\log s}$) into a single word. As observed by Chan and Pătraşcu [10] and Wilkinson [24], we can design micro-structures by thinking of each word as a block of multiple points, and borrowing known techniques from the world of *external-memory* algorithms (specifically, *buffer trees* [4]) to achieve *(sub)constant* amortized update time. Alternatively, Mortensen described his micro-structures from scratch, which required a more complicated solution to a certain "pebble game" [19, Section 6].

  One subtle issue is that to simulate rank space reduction dynamically, we need *list labeling* techniques, which, if not carefully implemented, can worsen the exponent in the update bound (as was the case in both Mortensen's and Wilkinson's solutions).
- Macro-structures refer to data structures for large input size n, constructed using micro-structures as black boxes. This part does not involve bit packing, and relies on more traditional geometric divide-and-conquer techniques such as higher-degree range trees, as in Mortensen's and Chan and Pătraşcu's solutions, with degree $2^{\log^\beta n}$ for some fraction $\beta < 1$. Van Emde Boas recursion is also a crucial ingredient in Mortensen's macro-structures.

Our solution will require a number of new ideas in both micro- and macro-structures. On the micro level, we bypass the "pebbling" problem by explicitly invoking external-memory techniques, as in Wilkinson's work [24], but we handle the list labeling issue more carefully, to avoid worsening the update time. On the macro level, we use higher-degree range trees but with a more intricate analysis (involving Harmonic series, interestingly), plus a few bootstrapping steps, in order to achieve the best update and query bounds.

2 Preliminaries

In all our algorithms, we assume that during each query or update, we are given a pointer to the predecessor/successor of the x- and y-values of the given point or rectangle. At the end, we can add the cost of predecessor search to the query and update time (which is no bigger than $O\left(\sqrt{\log n}\right)$ [3] in the word RAM model).

We assume a word RAM model that allows for a constant number of "exotic" operations on w-bit words. By setting $w := \delta \log n$ for a sufficiently small constant δ, these operations can be simulated in constant time by table lookup, after preprocessing the tables in $2^{O(w)} = n^{O(\delta)}$ time.

For simplicity, we concentrate on emptiness queries; all our algorithms can be easily modified for reporting queries, with an additional $O(k)$ term to the query time bounds.

A *3-sided* query deals with a rectangle that is unbounded on the left or right side, by default. A *2-sided* (or *dominance*) query deals with a rectangle that is unbounded on two adjacent sides.

Let $[n]$ denote $\{0, 1, \ldots, n-1\}$.

We now quickly review a few useful tools.

List labeling. *Monotone list labeling* is the problem of assigning *labels* to a dynamic set of totally ordered elements, such that whenever $x < y$, the label of x is less than the label of y. As elements are inserted, we are allowed to change labels. The following result is well known:

▶ **Lemma 1** ([12]). *A monotone labeling for n totally ordered elements with labels in $\left[n^{O(1)}\right]$ can be maintained under insertions by making $O(n \log n)$ label changes.*

Weight-balancing. *Weight-balanced B-trees* [5] are B-tree implementations with a rebalancing scheme that is based on the nodes' *weights*, i.e., subtree sizes, in order to support updates of secondary structures efficiently.

▶ **Lemma 2** ([5], Lemma 4). *In a weight-balanced B-tree of degree s, nodes at height i have weight $\Theta\left(s^i\right)$, and any sequence of n insertions requires at most $O\left(n/s^i\right)$ splits of nodes at height i.*

Colored predecessors. *Colored predecessor searching* is the problem of maintaining a dynamic set of multi-colored, totally ordered elements and searching for the predecessors with a given color.

▶ **Lemma 3** ([19], Theorem 14). *Colored predecessor searches and updates on n colored, totally ordered elements can be supported in $O\left(\log^2 \log n\right)$ time deterministically.*

Van Emde Boas transformation. A crucial ingredient we will use is a general technique of Mortensen [18, 19] that transforms any given data structure for orthogonal range emptiness on small sets of s points, to one for point sets in a *narrow grid* $[s] \times \mathbb{R}$, at the expense of a $\log \log n$ factor increase in cost. We state the result in a slightly more general form:

▶ **Lemma 4** ([19], Theorem 1). *Let X be a set of $O(s)$ values. Given a dynamic data structure for j-sided orthogonal range emptiness ($j \in \{3, 4\}$) on s points in $X \times \mathbb{R}$ with update time $U_j(s)$ and query time $Q_j(s)$, there exists a dynamic data structure for j-sided orthogonal range emptiness on n points in $X \times \mathbb{R}$ with update time $O(U_j(s) \log \log n)$ and query time $O(Q_j(s) \log \log n)$.*

If the given data structure supports updates to X in $U_X(s)$ time and this update procedure depends solely on X (and not the point set), the new data structure can support updates to X in $U_X(s)$ time.

Mortensen's transformation is obtained via a van-Emde-Boas-like recursion [23]: Roughly, we divide the plane into $\sqrt{n}$ horizontal slabs each with $\sqrt{n}$ points; for each slab, we store the topmost and bottommost point at each x-coordinate of X in a data structure for $O(s)$ points, and handle the remaining points recursively. (Note that all these data structures for $O(s)$ points work with a common set X of x-coordinates.)

3 Part 1: Micro-Structures

We first design *micro-structures* for 3- and 4-sided dynamic orthogonal range emptiness when the number of points s is small. This part heavily relies on bit-packing techniques.

3.1 Static universe

We begin with the case of a static universe $\left[s^{O(1)}\right]^2$.

▶ **Lemma 5.** *For s points in the static universe $\left[s^{O(1)}\right]^2$, there exist data structures for dynamic orthogonal range emptiness that support*

(i) *updates in $O\left(\frac{\log^2 s}{w}+1\right)$ amortized time and 3-sided queries in $O(\log s)$ amortized time;*

(ii) *updates in $O\left(\frac{\log^3 s}{w}+1\right)$ amortized time and 4-sided queries in $O\left(\log^2 s\right)$ amortized time.*

Proof. We mimick existing *external-memory* data structures with a block size of $B := \left\lceil \frac{\delta w}{\log s} \right\rceil$ for a sufficiently small constant δ, observing that B points can be packed into a single word.

(i) For the 3-sided case, Wilkinson [24, Lemma 1] has already adapted such an external-memory data structure, namely, a *buffered* version of a binary *priority search tree* due to Kumar and Schwabe [14] (see also Brodal's more recent work [7]), which is similar to the *buffer tree* of Arge [4]. For 3-sided rectangles unbounded to the left/right, the priority search tree is ordered by y, where each node stores $O(B)$ x-values. Wilkinson obtained $O\left(\frac{1}{B}\cdot\log s+1\right) = O\left(\frac{\log^2 s}{w}+1\right)$ amortized update time and $O(\log s)$ amortized query time.

(ii) For the general 4-sided case, we use a *buffered* version of a binary *range tree*. Although we are not aware of prior work explicitly giving such a variant of the range tree, the modifications are straightforward, and we will provide only a rough outline. The range tree is ordered by y. Each node holds a buffer of up to B update requests that have not yet been processed. Each node is also augmented with a 1-d binary *buffer tree* (already described by Arge [4]) for the x-projection of the points. To insert or delete a point, we add the update request to the root's buffer. Whenever a buffer's size of a node exceeds B, we empty the buffer by applying the following procedure: we divide the list of $\Theta(B)$ update requests into two sublists for the two children in $O(1)$ time using an exotic word operation (since B update requests fit in a word); we then pass these sublists to the buffers at the two children, and also pass another copy of the list to the node's 1-d buffer tree. These 1-d updates cost $O\left(\frac{1}{B}\cdot\log s\right)$ each [4], when amortized over $\Omega(B)$ updates. Since each update eventually travels to $O(\log s)$ nodes of the range tree, the amortized update time of the 4-sided structure is $O\left(\frac{1}{B}\log^2 s+1\right) = O\left(\frac{\log^3 s}{w}+1\right)$.

A 4-sided query is answered by following two paths in the range tree in a top-down manner, performing $O(\log s)$ 1-d queries; since each 1-d query takes $O(\log s)$ time, the

overall query time is $O\left(\log^2 s\right)$. However, before we can answer the query, we need to first empty the buffers along the two paths of the range tree. This can be done by applying the procedure in the preceding paragraph at the $O(\log s)$ nodes top-down; this takes $O(\log s)$ time, plus the time needed for $O(B \log s)$ 1-d updates, costing $O\left(\frac{1}{B} \cdot \log s\right)$ each [4]. The final amortized query time is thus $O\left(\log^2 s\right)$. ◀

Notice that the above update time is *constant* when the number of points s is as large as $2^{\sqrt{w}}$ for 3-sided queries or $2^{w^{1/3}}$ for 4-sided.

(It is possible to eliminate one of the logarithmic factors in the query time for the above 4-sided result, by augmenting nodes of the range tree with 3-sided structures. However, this alternative causes difficulty later in the extension to dynamic universes. Besides, the larger query time turns out not to matter for our macro-structures at the end.)

3.2 Dynamic universe

To make the preceding data structure support a dynamic universe, the simplest way is to apply monotone list labeling (Lemma 1), which maps coordinates to $\left[s^{O(1)}\right]^2$. Whenever a label of a point changes, we just delete the point and reinsert a copy with the new coordinates into the data structure. However, since the total number of label changes is $O(s \log s)$ over s insertions, this slows down the amortized update time by a $\log s$ factor and will hurt the final update bound.

Our approach is as follows. We first observe that the list labeling approach works fine for changes to the y-universe. For changes to the x-universe, we switch to a "brute-force" method with large running time, but luckily, since the number of such changes will be relatively small, this turns out to be adequate for our macro-structures at the end. (The brute-force idea can also be found in Mortensen's paper [19], but his macro-structures were less efficient.)

▶ **Lemma 6.** *Both data structures in Lemma 5 can be modified to work for s points in a universe $X \times Y$ with $|X|, |Y| = O(s)$. The update and query time bounds are the same, and we can support*

- **(i)** *updates to Y in $O\left(\log^2 \log s\right)$ amortized time (given a pointer to the predecessor/successor in Y), and*
- **(ii)** *updates to X in $2^{O(w)}$ time, where the update procedure for X depends solely on X (and not the point set).*

Proof. (i) To start, let us assume that $X = \left[s^{O(1)}\right]$ but Y is arbitrary. We divide the sorted list Y into $O(s/A)$ blocks of size $\Theta(A)$ for a parameter A to be set later. It is easy to maintain such a blocking using $O(s/A)$ number of block merges and splits over s updates. (Such a blocking was also used by Wilkinson [24].) We maintain a monotone labeling of the blocks by Lemma 1. In the proof of Lemma 5, we construct the y-ordered priority search tree or range tree using the block labels as the y-values. Each leaf then corresponds to a block. We build a small range tree for each leaf block to support updates and queries for the $O(A)$ points in, say, $O\left(\log^2 A\right)$ time. We can encode a y-value $\eta \in Y$ by a pair consisting of the label of the block containing η, and the rank of η with respect to the block. We will use these encoded values, which still are $O(\log s)$-bit long, in all the buffers. The block labels provide sufficient information to pass the update requests to the leaves and the x-ordered 1-d buffer trees. The ranks inside a block provide sufficient information to handle a query or update at a leaf.

During each block split/merge and each block label change, we need to first empty the buffers along the path to the block before applying the change. This can be done by applying

the procedure from the proof of Lemma 5 at $O(\log s)$ nodes top-down, requiring $O(\log s)$ amortized time. Since the total number of block label changes is $O\left(\frac{s}{A}\log\frac{s}{A}\right)$, the total time for these steps is $O\left(\frac{s}{A}\log\frac{s}{A}\cdot\log s\right) = O(s)$ by setting $A := \log^2 s$. The amortized cost for these steps is thus $O(1)$.

(ii) Now, we remove the $X = \left[s^{O(1)}\right]$ assumption. We assign elements in X to labels in $[O(s)]$ but do not insist on a monotone labeling. Then no label change is necessary! We will use these labels for the x-values in all the buffers. The exotic word operations are simulated by table lookup, but in the precomputation of each table entry, we need to first map the labels to their actual x-values. During each update to X, we now need to recompute all table entries by brute force, taking $2^{O(w)}$ time. ◀

4 Part 2: Macro-Structures

We now present macro-structures for 3- and 4-sided dynamic orthogonal range emptiness when the number of points n is large, by using micro-structures as black boxes. This part does not involve bit packing (and hence is more friendly to computational geometers). The transformation from micro- to macro-structures is based on variants of range trees.

4.1 Range tree transformation I

▶ **Lemma 7.** *Given a family of data structures $\mathcal{D}_j^{(i)}$ $(i \in \{1,\ldots,\log_s n\})$ for dynamic j-sided orthogonal range emptiness $(j \in \{3,4\})$ on s points in $X \times \mathbb{R}$ $(|X| = O(s))$ with update time $U_j^{(i)}(s)$ and query time $Q_j^{(i)}(s)$, where updates to X take $U_X^{(i)}(s)$ time with a procedure that depends solely on X, there exist data structures for dynamic orthogonal range emptiness on n points in the plane with the following amortized update and query time:*

(i) *for the 3-sided case,*

$$U_3'(n) = O\left(\sum_{i=1}^{\log_s n} U_3^{(i)}(s)\log\log n + \sum_{i=1}^{\log_s n}\frac{U_X^{(i)}(s)}{s^{i-1}} + \log_s n\log^2\log n\right)$$

$$Q_3'(n) = O\left(\max_i Q_3^{(i)}(s)\log_s n\log\log n + \log_s n\log^2\log n\right);$$

(ii) *for the 4-sided case,*

$$U_4'(n) = O\left(\sum_{i=1}^{\log_s n}(U_4^{(i)}(s)+U_3^{(i)}(s))\log\log n + \sum_{i=1}^{\log_s n}\frac{U_X^{(i)}(s)}{s^{i-1}} + \log_s n\log^2\log n\right)$$

$$Q_4'(n) = O\left(\max_i Q_4^{(i)}(s)\log\log n + \max_i Q_3^{(i)}(s)\log_s n\log\log n + \log_s n\log^2\log n\right).$$

Proof. We store a range tree ordered by x, implemented as a degree-s weight-balanced B-tree. (Deletions can be handled lazily without changing the weight-balanced tree; we can rebuild periodically when n decreases or increases by a constant factor.) At every internal node v at height i, we store the points in its subtree in a data structure for j-sided orthogonal range emptiness on a *narrow grid* $X_v \times \mathbb{R}$, obtained by applying Lemma 4 to the given structure $\mathcal{D}_j^{(i)}$, where X_v is the set of x-coordinates of the $O(s)$ dividing vertical lines at the node, and the x-coordinate of every point is replaced with the predecessor in X_v. We also store the y-coordinates of these points in a colored predecessor searching structure of Lemma 3, where points in the same child's vertical slab are assigned the same color. And we store the x-coordinates in another colored predecessor searching structure, where X_v is colored black and the rest is colored white.

To insert or delete a point, we update the narrow-grid structures at the nodes along the path in the tree. This takes $O\left(\sum_{i=1}^{\log_s n} U_j^{(i)}(s) \log\log n\right)$ total time. Note that given the y-predecessor/successor of the point at a node, we can obtain the y-predecessor/successor at the child by using the colored predecessor searching structure. We can also determine the x-predecessor in X_v by another colored predecessor search. This takes total time $O\left(\log_s n \log^2 \log n\right)$ along the path.

To keep the tree balanced, we need to handle node splits. For nodes at height i, there are $O\left(n/s^i\right)$ splits by Lemma 2. Each such split requires rebuilding two narrow-grid structures on $O\left(s^i\right)$ points, which can be done naively by $O\left(s^i\right)$ insertions to empty structures. This has $O\left(\sum_{i=1}^{\log_s n} \left(n/s^i\right) \cdot s^i U_j^{(i)}(s) \log\log n\right)$ total cost, i.e., an amortized cost of $O\left(\sum_{i=1}^{\log_s n} U_j^{(i)}(s) \log\log n\right)$. A split of a child of v also requires updating (deleting and reinserting) the points at the child's slab. This has $O\left(\sum_{i=1}^{\log_s n} \left(n/s^{i-1}\right) \cdot s^{i-1} U_j^{(i)}(s) \log\log n\right)$ total cost, i.e., an amortized cost of $O\left(\sum_{i=1}^{\log_s n} U_j^{(i)}(s) \log\log n\right)$. Furthermore, a split of a child of v requires an update to X_v. This has $O\left(\sum_{i=1}^{\log_s n} \left(n/s^{i-1}\right) \cdot U_X^{(i)}(s)\right)$ total cost, i.e., an amortized cost of $O\left(\sum_{i=1}^{\log_s n} \left(1/s^{i-1}\right) \cdot U_X^{(i)}(s)\right)$.

To answer a 3-sided query, we proceed down a path of the tree and perform queries in the narrow-grid structures at nodes along the path. This takes $O\left(\log_s n \cdot \max_i Q_3^{(i)}(s) \log\log n\right)$ total time. As before, given the y-predecessor/successor of the coordinates of the rectangle at a node, we can obtain the y-predecessor/successor at the child by using the colored predecessor searching structure. This takes total time $O\left(\log_s n \log^2 \log n\right)$ along the path.

To answer a 4-sided query, we find the highest node v whose dividing vertical lines cut the query rectangle. We obtain two 3-sided queries at two children of v, which can be answered as above, plus a remaining query that can be answered via the narrow-grid structure at v in $O\left(\max_i Q_4^{(i)}(s) \log\log n\right)$ time. ◀

Combining with our preceding micro-structures, we obtain the following results, achieving the desired update time but slightly suboptimal query time (which we will fix later):

▶ **Theorem 8.** *Given n points in the plane, there exist data structures for dynamic orthogonal range emptiness that support*

- **(i)** *updates in amortized $O\left(\log^{1/2} n \log^{O(1)} \log n\right)$ time and 3-sided queries in amortized $O\left(\log n \log\log n\right)$ time;*
- **(ii)** *updates in amortized $O\left(\log^{2/3} n \log^{O(1)} \log n\right)$ time and 4-sided queries in amortized $O\left(\log n \log\log n\right)$ time.*

Proof. (i) For the 3-sided case, Lemmata 5(i) and 6 give micro-structures with update time $O\left(\frac{\log^2 s}{\overline{w}} + \log^2 \log s\right)$ and query time $O\left(\log s\right)$, while supporting updates to X in $2^{O(\overline{w})}$ time. Observe that we can choose to work with a smaller word size $\overline{w} \leq w$, so long as $\overline{w} = \Omega\left(\log s\right)$. We choose $\overline{w} := \delta i \log s$ for a sufficiently small absolute constant δ and for any given $i \in [2, \log_s n]$. This gives

$$\begin{aligned} U_3^{(i)}(s) &= O\left(\frac{\log s}{i} + \log^2 \log s\right) \\ Q_3^{(i)}(s) &= O\left(\log s\right) \\ U_X^{(i)}(s) &= s^{O(\delta i)}. \end{aligned}$$

For the special case $i = 1$, we use a standard priority search tree, achieving $U_3^{(1)}(s), Q_3^{(1)}(s) = O(\log s)$ and $U_X^{(1)}(s) = 0$. Substituting into Lemma 7, we obtain

$$\begin{aligned} U_3'(n) &= O\left(\sum_{i=1}^{\log_s n} \frac{\log s \log\log n}{i} + \log_s n \log^3 \log n + \sum_{i=2}^{\log_s n} \frac{s^{O(\delta i)}}{s^{i-1}} \log_s n \log^2 \log n\right) \\ &= O\left(\log s \log^2 \log n + \log_s n \log^3 \log n\right), \end{aligned}$$

since the first sum is a Harmonic series and the second sum is a geometric series. (This assumes a sufficiently small constant for δ, as the hidden constant in the exponent $O(\delta i)$ does not depend on δ.) Furthermore,

$$\begin{aligned} Q_3'(n) &= O\left(\log s \log_s n \log\log n + \log_s n \log^2 \log n\right) \\ &= O\left(\log n \log\log n + \log_s n \log^2 \log n\right). \end{aligned}$$

We set $s := 2^{\sqrt{\log n}}$ to get $U_3'(n) = O\left(\log^{1/2} n \log^{O(1)} \log n\right), Q_3'(n) = O(\log n \log\log n)$.

(ii) Similarly, for the 4-sided case, Lemmata 5(ii) and 6 with a smaller word size $\overline{w} := \delta i \log s$ give micro-structures with

$$\begin{aligned} U_4^{(i)}(s) &= O\left(\frac{\log^2 s}{i} + \log^2 \log s\right) \\ Q_4^{(i)}(s) &= O\left(\log^2 s\right) \\ U_X^{(i)}(s) &= s^{O(\delta i)}. \end{aligned}$$

For the special case $i = 1$, we use a standard range tree, achieving $U_4^{(1)}(s), Q_4^{(1)}(s) = O\left(\log^2 s\right)$ and $U_X^{(1)}(s) = 0$. Substituting into Lemma 7, we obtain

$$\begin{aligned} U_4'(n) &= O\left(\sum_{i=1}^{\log_s n} \frac{\log^2 s \log\log n}{i} + \log_s n \log^3 \log n + \sum_{i=2}^{\log_s n} \frac{s^{O(\delta i)}}{s^{i-1}} \log_s n \log^2 \log n\right) \\ &= O\left(\log^2 s \log^2 \log n + \log_s n \log^3 \log n\right) \end{aligned}$$

and

$$\begin{aligned} Q_4'(n) &= O\left(\log^2 s \log\log n + \log s \log_s n \log\log n + \log_s n \log^2 \log n\right) \\ &= O\left(\log^2 s \log\log n + \log n \log\log n + \log_s n \log^2 \log n\right). \end{aligned}$$

We set $s := 2^{\log^{\frac{1}{3}} n}$ to get $U_4'(n) = O\left(\log^{2/3} n \log^{O(1)} \log n\right), Q_4'(n) = O(\log n \log\log n)$. ◀

4.2 Range tree transformation II

We now reduce the query time to optimal by another transformation:

▶ **Lemma 9.** *Given data structures for dynamic j-sided orthogonal range emptiness ($j \in \{2, 3, 4\}$) on n points in the plane with update time $U_j(n)$ and query time $Q_j(n)$, there exist data structures for dynamic j-sided orthogonal range emptiness ($j \in \{3, 4\}$) on n points in the plane with the following amortized update and query time:*

$$\begin{aligned} U_j'(n) &= O\left(U_j(s) \log_s n \log\log n + U_{j-1}(n) \log_s n + \log_s n \log^2 \log n\right) \\ Q_j'(n) &= O\left(Q_j(s) \log\log n + Q_{j-1}(n) + \log_s n \log^2 \log n\right). \end{aligned}$$

Proof. We first switch the x- and y-coordinates of the points. This is fine since the given data structures in the statement of this lemma still exist by symmetry (unlike in Lemma 7).

We modify the range tree in the proof of Lemma 7, where every internal node is augmented with a $(j-1)$-sided structure.

During an insertion or deletion of a point, we update the narrow-grid structures along a path as before, in $O(\log_s n \cdot U_j(s) \log\log n)$ time. We now also need to update the $(j-1)$-sided structures at nodes along the path. This adds $O(U_{j-1}(n) \log_s n)$ to the update time.

During rebalancing, each split of a node at height i now requires rebuilding the $(j-1)$-sided structures, which can be done naively by $O(s^i)$ insertions to an empty structure. This has $O\left(\sum_{i=1}^{\log_s n} (n/s^i) \cdot s^i U_{j-1}(n)\right)$ total cost, i.e., an amortized cost of $O(U_{j-1}(n) \log_s n)$.

To answer a j-sided query, we find the highest node v whose dividing vertical lines cut the query rectangle. We obtain two $(j-1)$-sided queries at two children of v, plus a query in the narrow-grid structure at v. (In the case $j = 3$, recall that the input to a 3-sided query is now a rectangle unbounded from above or below, because of the switching of x and y.) The two $(j-1)$-sided queries can be answered directly using the augmented structures. This takes $O(Q_j(s) \log\log n + Q_{j-1}(n))$ time, plus the cost $O(\log_s n \log^2 \log n)$ to descend along the path to that node. ◀

We obtain our final results by bootstrapping:

▶ **Theorem 10.** *Given n points in the plane, there exist data structures for dynamic orthogonal range emptiness that support*

- **(i)** *updates in amortized $O\left(\log^{1/2+O(\varepsilon)} n\right)$ time and 3-sided queries in amortized $O\left(\frac{\log n}{\log\log n}\right)$ time;*
- **(ii)** *updates in amortized $O\left(\log^{2/3} n \log^{O(1)} \log n\right)$ time and 4-sided queries in amortized $O\left(\frac{\log n}{\log\log n}\right)$ time.*

Proof. (i) Theorem 8(i) achieves

$$\begin{aligned} U_3(s) &= O\left(\log^{1/2} s \log^{O(1)} \log s\right) \\ Q_3(s) &= O(\log s \log\log s). \end{aligned}$$

Wilkinson [24] has given a data structure for 2-sided (dominance) queries with

$$\begin{aligned} U_2(n) &= O\left(\log^{1/2+\varepsilon} n\right) \\ Q_2(n) &= O\left(\frac{\log n}{\log\log n}\right). \end{aligned}$$

Substituting into Lemma 9, we obtain

$$\begin{aligned} U_3'(n) &= O\left(\log^{1/2} s \log_s n \log^{O(1)} \log n \;+\; \log^{1/2+\varepsilon} n \log_s n + \log_s n \log^2 \log n\right) \\ Q_3'(n) &= O\left(\log s \log\log s \log\log n \;+\; \frac{\log n}{\log\log n} \;+\; \log_s n \log^2 \log n\right). \end{aligned}$$

We set $s := 2^{\frac{\log n}{\log^3 \log n}}$ to get $U_3'(n) = O\left(\log^{1/2+O(\varepsilon)} n\right), Q_3'(n) = O\left(\frac{\log n}{\log\log n}\right)$.

(ii) Similarly, Theorem 8(ii) achieves

$$\begin{aligned} U_4(s) &= O\left(\log^{2/3} s \log^{O(1)} \log s\right) \\ Q_4(s) &= O(\log s \log\log s). \end{aligned}$$

Part (i) above gives

$$\begin{aligned} U_3(n) &= O\left(\log^{1/2+O(\varepsilon)} n\right) \\ Q_3(n) &= O\left(\frac{\log n}{\log\log n}\right). \end{aligned}$$

Substituting into Lemma 9, we obtain

$$\begin{aligned} U'_4(n) &= O\left(\log^{2/3} s \log_s n \log^{O(1)} \log n \;+\; \log^{1/2+O(\varepsilon)} n \log_s n + \log_s n \log^2 \log n\right) \\ Q'_4(n) &= O\left(\log s \log\log s \log\log n \;+\; \frac{\log n}{\log\log n} \;+\; \log_s n \log^2 \log n\right). \end{aligned}$$

We set $s := 2^{\frac{\log n}{\log^3 \log n}}$ to get $U'_4(n) = O\left(\log^{2/3} n \log^{O(1)} \log n\right), Q'_4(n) = O\left(\frac{\log n}{\log\log n}\right)$. ◀

5 Future Work

We have not yet mentioned space complexity. We can trivially upper-bound the space of our data structure by n times the update time, i.e., $O\left(n \log^{2/3+o(1)} n\right)$ for the 4-sided case, which is already an improvement over Mortensen's $O\left(n \log^{7/8+\varepsilon} n\right)$ space bound. We are currently working on ways to improve space further to near-linear. (See [20, 21] for the current best data structures with near-linear space.)

We can automatically extend our result to higher constant dimensions $d \geq 3$ by using a standard degree-b range tree, which adds a $b \log_b n$ factor per dimension to the update time and a $\log_b n$ factor per dimension to the query time. With $b = \log^{\varepsilon} n$, this gives $O\left((\log n/\log\log n)^{d-1}\right)$ query time and $O\left(\log^{d-5/3+O(\varepsilon)} n\right)$ update time, improving Mortensen's result. Alternatively, we can directly modify our micro- and macro-structures, which should give a better update time of the form $O\left(\log^{d-2+O(1/d)} n\right)$. We are currently working on obtaining the best precise exponent with this approach. (See [8] for a different tradeoff with query time better by about a logarithmic factor but update time worse by several logarithmic factors.)

References

1. Stephen Alstrup, Gerth Stølting Brodal, and Theis Rauhe. New data structures for orthogonal range searching. In *Proceedings of the 41st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 198–207, 2000. doi:10.1109/SFCS.2000.892088.
2. Stephen Alstrup, Thore Husfeldt, and Theis Rauhe. Marked ancestor problems. In *Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 534–543, Nov 1998. doi:10.1109/SFCS.1998.743504.
3. Arne Andersson and Mikkel Thorup. Dynamic ordered sets with exponential search trees. *Journal of the ACM*, 54(3):13, 2007. doi:10.1145/1236457.1236460.
4. Lars Arge. The buffer tree: A technique for designing batched external data structures. *Algorithmica*, 37(1):1–24, 2003. doi:10.1007/s00453-003-1021-x.
5. Lars Arge and Jeffrey Scott Vitter. Optimal external memory interval management. *SIAM Journal on Computing*, 32(6):1488–1508, 2003. doi:10.1137/S009753970240481X.
6. Jon Louis Bentley. Decomposable searching problems. *Information Processing Letters*, 8(5):244–251, 1979. doi:10.1016/0020-0190(79)90117-0.

7 Gerth Stølting Brodal. External memory three-sided range reporting and top-k queries with sublogarithmic updates. In *Proceedings of the 33rd Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 23:1–23:14, 2016. doi:10.4230/LIPIcs.STACS.2016.23.

8 Timothy M. Chan. Three problems about dynamic convex hulls. *International Journal of Computational Geometry and Applications*, 22(4):341–364, 2012. doi:10.1142/S0218195912600096.

9 Timothy M. Chan, Kasper Green Larsen, and Mihai Pătraşcu. Orthogonal range searching on the RAM, revisited. In *Proceedings of the 27th Annual Symposium on Computational Geometry (SoCG)*, pages 1–10, 2011. doi:10.1145/1998196.1998198.

10 Timothy M. Chan and Mihai Pătraşcu. Counting inversions, offline orthogonal range counting, and related problems. In *Proceedings of the 21st Annual ACM–SIAM Symposium on Discrete Algorithms (SODA)*, pages 161–173, 2010. doi:10.1137/1.9781611973075.15.

11 Bernard Chazelle. A functional approach to data structures and its use in multidimensional searching. *SIAM Journal on Computing*, 17(3):427–462, 1988. doi:10.1137/0217026.

12 Paul F. Dietz. Maintaining order in a linked list. In *Proceedings of the 14th Annual ACM Symposium on Theory of Computing (STOC)*, pages 122–127, 1982. doi:10.1145/800070.802184.

13 Otfied Fries, Kurt Mehlhorn, Stefan Näher, and Athanasios K. Tsakalidis. A $\log \log n$ data structure for three-sided range queries. *Information Processing Letters*, 25(4):269–273, 1987. doi:10.1016/0020-0190(87)90174-8.

14 Vijay Kumar and Eric J. Schwabe. Improved algorithms and data structures for solving graph problems in external memory. In *Proceedings of the 8th Annual IEEE Symposium on Parallel and Distributed Processing*, pages 169–176, 1996. doi:10.1109/SPDP.1996.570330.

15 George S. Lueker. A data structure for orthogonal range queries. In *Proceedings of the 19th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 28–34, 1978. doi:10.1109/SFCS.1978.1.

16 Edward M. McCreight. Priority search trees. *SIAM Journal on Computing*, 14(2):257–276, 1985. doi:10.1137/0214021.

17 Kurt Mehlhorn and Stefan Näher. Dynamic fractional cascading. *Algorithmica*, 5(1):215–241, 1990. doi:10.1007/BF01840386.

18 Christian Worm Mortensen. Fully-dynamic two dimensional orthogonal range and line segment intersection reporting in logarithmic time. In *Proceedings of the 14th Annual ACM–SIAM Symposium on Discrete Algorithms (SODA)*, pages 618–627, 2003. URL: http://dl.acm.org/citation.cfm?id=644108.644210".

19 Christian Worm Mortensen. Fully dynamic orthogonal range reporting on RAM. *SIAM Journal on Computing*, 35(6):1494–1525, 2006. doi:10.1137/S0097539703436722.

20 Yakov Nekrich. Space efficient dynamic orthogonal range reporting. *Algorithmica*, 49(2):94–108, 2007. doi:10.1007/s00453-007-9030-9.

21 Yakov Nekrich. Orthogonal range searching in linear and almost-linear space. *Computational Geometry*, 42(4):342–351, 2009. doi:10.1016/j.comgeo.2008.09.001.

22 Mark H. Overmars. Efficient data structures for range searching on a grid. *Journal of Algorithms*, 9(2):254–275, 1988. doi:10.1016/0196-6774(88)90041-7.

23 Peter van Emde Boas. Preserving order in a forest in less than logarithmic time and linear space. *Information Processing Letters*, 6(3):80–82, 1977. doi:10.1016/0020-0190(77)90031-X.

24 Bryan T. Wilkinson. Amortized bounds for dynamic orthogonal range reporting. In *Proceedings of the 22th Annual European Symposium on Algorithms (ESA)*, pages 842–856, 2014. doi:10.1007/978-3-662-44777-2_69.

25 Dan E. Willard. New data structures for orthogonal range queries. *SIAM Journal on Computing*, 14(1):232–253, 1985. doi:10.1137/0214019.

26 Dan E. Willard. Examining computational geometry, Van Emde Boas trees, and hashing from the perspective of the fusion tree. *SIAM Journal on Computing*, 29(3):1030–1049, 2000. doi:10.1137/S0097539797322425.

27 Dan E. Willard and George S. Lueker. Adding range restriction capability to dynamic data structures. *Journal of the ACM*, 32(3):597–617, 1985. doi:10.1145/3828.3839.

On Bend-Minimized Orthogonal Drawings of Planar 3-Graphs

Yi-Jun Chang[*1] **and Hsu-Chun Yen**[†2]

1 **Department of EECS, University of Michigan, Ann Arbor, MI, USA**
2 **Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan**

Abstract

An *orthogonal drawing* of a graph is a planar drawing where each edge is drawn as a sequence of horizontal and vertical line segments. Finding a bend-minimized orthogonal drawing of a planar graph of maximum degree 4 is NP-hard. The problem becomes tractable for planar graphs of maximum degree 3, and the fastest known algorithm takes $O(n^5 \log n)$ time. Whether a faster algorithm exists has been a long-standing open problem in graph drawing. In this paper we present an algorithm that takes only $\tilde{O}(n^{17/7})$ time, which is a significant improvement over the previous state of the art.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Bend minimization, graph drawing, orthogonal drawing, planar graph

Digital Object Identifier 10.4230/LIPIcs.SoCG.2017.29

1 Introduction

An *orthogonal drawing* of a graph is a planar drawing where each edge is composed of a sequence of horizontal and vertical line segments with no crossings. Orthogonal drawings appear in many applications such as the automation of VLSI circuit layout and the drawing of diagrams in information systems. Variants of orthogonal drawings have been introduced in the literature to cope with different constraints or to improve the readability and aesthetic feel: smoothing the drawing [2, 1], requiring orthogonal convexity [5], accommodating vertices of more than 4 incident edges [16, 8], and restricting directions of vertices [11, 13]. Refer to [12] for a survey on orthogonal drawings.

Bend-minimization is one of the most classical optimization problems on orthogonal drawings. Given a planar (or plane) graph, the problem asks for an orthogonal drawing with the total number of bends minimized. However, the problem is NP-hard for planar 4-graphs [15].[1] To obtain polynomial time algorithms, one has to relax the problem one way or another. For example, it is known that a polynomial time algorithm exists when the first bend on an edge does not incur any cost [3].

Much research effort has been made on bend-minimization for subclasses of planar 4-graphs [9, 7, 14, 18, 17]. The two most *natural* subclasses are planar 3-graphs (reducing

* Yi-Jun Chang was supported in part by NSF grant CCF-1514383.
† Hsu-Chun Yen was supported in part by Ministry of Science and Technology, Taiwan, under grant MOST 103-2221-E-002-154-MY3.

[1] We write k-graphs to denote graphs of maximum degree k. Note that the degree of each vertex cannot exceed 4 in an orthogonal drawing, and hence planar 4-graphs are the most general graph class that can be drawn orthogonally.

the maximum degree by 1) and plane 4-graphs (fixing the planar embedding). For plane 4-graphs, a seminal work by Tamassia [20] demonstrates a reduction from bend-minimization to a computation of a min-cost flow. Following this approach, the runtime has been reduced to $O(n^{7/4}\sqrt{\log n})$ [14], and subsequently to $O(n^{1.5})$ [7].[2]

For planar 3-graphs, the fastest known algorithm for bend-minimization is the $O(n^5 \log n)$ time algorithm designed by Di Battista, Liotta, and Vargiu, which dates back to 1998 [9]. As stated as an open problem in [4], further improving the time complexity is identified as an important issue in the field of graph drawing.

Problem 15: *Let G be a planar graph whose vertices have degree at most three. Is there an algorithm to compute a planar bend-minimum orthogonal drawing of G in $o(n^5 \log n)$ time?*

In this paper, we answer the problem affirmatively by demonstrating an $\tilde{O}(n^{17/7})$ time[3] algorithm. Precisely, our algorithm takes $O(n \cdot T(n))$ time, where $T(n)$ is the time complexity of constructing a bend-minimized orthogonal drawing of a plane 3-graph subject to the constraint that some designated edges have no bend. We will later see that bend-minimization of a plane 3-graph can be reduced to min-cost flow of *constant* capacity, and a recent breakthrough on unit-capacity min-cost flow in sparse graphs [6] implies $T(n) = \tilde{O}(n^{10/7})$.

The main challenge of designing a bend-minimization algorithm for planar 3-graphs is to handle the transition from the *variable* embedding setting to the *fixed* embedding setting. The naïve approach of testing *all* possible planar embeddings is very inefficient, as there can be an exponential number of different planar embeddings. A natural way to approach this problem is to devise a dynamic programming procedure on an SPQR-tree, which is a tree structure capable of storing all possible embeddings of a planar graph using linear space. The approach is briefly sketched as follows. By "contracting" all subgraphs of the planar graph G that can be flipped, a graph G' that has a fixed combinatorial embedding is obtained. An optimal drawing of G' can be computed quickly using a bend-minimization algorithm for plane 3-graphs. The optimal drawings of the contracted subgraphs are computed recursively. Merging these drawings yields a drawing of G. Using the terminology of SPQR-trees, if G is the *pertinent graph* of a node μ in the SPQR-tree, then the graph G' is (a subdivision of) the *skeleton* of μ, and the contracted subgraphs are the pertinent graphs of some descendants of μ. See Fig. 1 for a conceptual example: (1) $G' = H_2$ is resulting from contracting the subgraph H_1 into a vertex v in G. (2) Merging the drawings of H_1 and H_2 yields an orthogonal drawing of G (treating each white dot in the figure as a bend).

The above strategy does not immediately give us a bend-minimization algorithm. Observe that the outer boundary of G needs to have 4 convex corners. Thus we need an additional constraint which requires that the drawing of G' and the drawings computed by recursive calls jointly supply 4 convex corners in the outer boundary. To ensure that the final drawing of G is optimal, one has to compute *multiple* drawings of each contracted subgraph H subject to different *constraints* on the number of convex corners in the outer boundary of G that H can supply, and to examine all possible combinations of these constraints. In [9] the notion of *spirality*, which measures how an orthogonal drawing is "rolled up", is developed to serve as the aforementioned constraints.

In this paper, we utilize some tools developed by Rahman et al. in [19]. They characterized the condition for the existence of a no-bend orthogonal drawing based on the number of

[2] Throughout the paper, we define $n := |V(G)|$ as the number of vertices in the graph. Note that we have $|E(G)| = O(n)$ for planar graphs.

[3] The $\tilde{O}(\cdot)$ notation suppresses any $\operatorname{poly}\log n$ factor.

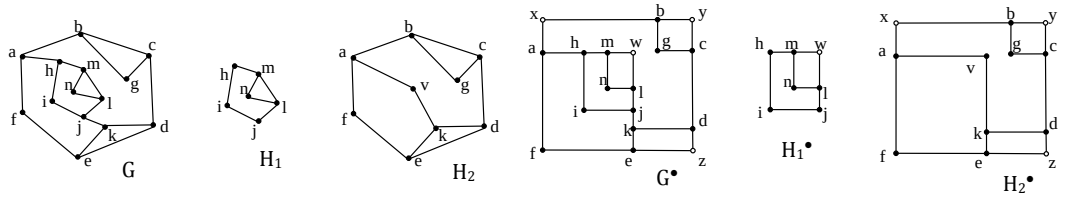


Figure 1 Contracting subgraphs and merging sub-drawings.

2-vertices[4] on some cycles, and they gave a linear time algorithm to construct such a drawing if one exists. Note that bend-minimization can be equated with finding a minimum number of subdivisions and a planar embedding to meet the condition for a no-bend orthogonal drawing to exist.

We first reduce the bend-minimization problem to a constrained version, and then we use the SPQR-tree dynamic programming to solve the constrained version of the bend-minimization problem recursively. In our algorithm, for each contracted subgraph, only a *constant* number of recursive calls is needed, and the merging of the subdrawings can be performed in time *linear* to the number of contracted subgraphs. Our algorithm is presented in a top-down manner. The main algorithm (for biconnected planar 3-graphs) is described in Sec. 3, and the details of some subroutines are left in Sec. 4 and 5. The SPQR-tree implementation is described in Sec. 6. Our algorithm can also be extended to planar 3-graphs that are not biconnected based on block-cutvertex tree. The detail is omitted due to space limit.

It is expected that our approach be applicable to some other variants of orthogonal drawings, such as the orthogonally convex drawing [5], as its no-bend version can be characterized analogously along the line of the work by Rahman at el. [19].

2 Preliminaries

Given a graph $G = (V, E)$, we write $\Delta(G)$ to denote the maximum degree of G. We write $V(G)$ and $E(G)$ to denote the sets of vertices and edges of G, respectively. We call G a d-graph if $\Delta(G) \leq d$. A vertex of degree d is called a d-vertex. A *multi-graph* is a graph where self-loops are disallowed while multi-edges are allowed. Throughout the paper, all graphs under consideration are planar 3-graphs with possibly multi-edges. Unless otherwise stated, all cycles and paths are assumed to be *simple* in the sense that they do not have repeated vertices.

A graph H is a *subgraph* of G if $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$. We write $H \subseteq G$ to denote the subgraph relation. We write $H \subsetneq G$ if $H \subseteq G$ and $H \neq G$. For $H \subsetneq G$, we write $G - H$ to denote the graph resulting from removing from G all vertices in H and the edges incident to these vertices. A graph is *planar* if it can be drawn on a plane without any edge crossing. A planar drawing partitions the plane into several disconnected regions called *faces*. The face corresponding to the region of unbounded size is called the *outer face*. The remaining ones are called *inner faces*. A *facial cycle* is a cycle surrounding a face. All facial cycles are simple in a biconnected plane graph. A *plane graph* is a planar graph with a fixed combinatorial embedding (which specifies a cyclic order of the edges incident to each vertex in the planar embedding) and a designated *outer cycle* C_O that surrounds the outer face.

[4] We write a k-vertex to denote a vertex of degree k.

We call a cycle *inner* if it is not the outer one. For any vertex and edge, we call it *boundary* if it is located in C_O; otherwise, it is *non-boundary*. A cycle $C \neq C_O$ is called *boundary* if it contains some edges in C_O. For a cycle C in a plane graph G, we write $G[C]$ to denote the subgraph of G that contains exactly C and vertices and edges residing in its interior region. Note that $G[C] = G$ iff $C = C_O$; and $G[C] = C$ iff C is an inner facial cycle.

With respect to a cycle C of a plane graph G, an edge $e = \{u, v\} \notin E(G[C])$ is called a *leg* of C if at least one of u and v belongs to $V(C)$. A vertex in $V(C)$ incident to some leg of C is called a *legged-vertex* of C. In a 3-graph, each legged-vertex of C is incident to exactly one leg of C. A cycle C is *k-legged* if C has exactly k legs.

Consider the plane graph G in Fig. 1. The cycle $C_1 = (h, m, l, j, i)$ is a non-boundary 2-legged cycle of which the two legged-vertices are h and j, and the two legs are $\{a, h\}$ and $\{j, k\}$. The facial cycle $C_2 = (d, e, k)$ is a boundary 3-legged cycle in G. A *subdivision* is a process of adding a new 2-vertex w to an edge $e = \{u, v\}$ by replacing e with two edges $\{u, w\}$ and $\{w, v\}$. A *smoothing* is the reverse process of a subdivision which removes a 2-vertex w by replacing two edges $\{u, w\}$ and $\{w, v\}$ with a new edge $\{u, v\}$, assuming $\{u, w\}, \{w, v\} \in E(G)$.

2.1 Theorems of Rahman et al.

Rahman st al. [19] gave a characterization of those biconnected plane 3-graphs that admit orthogonal drawings without bends. Based on the characterization, they presented a linear time algorithm to construct such a drawing, if one exists.

▶ **Theorem 1** ([19]). *A biconnected plane 3-graph G admits a no-bend orthogonal drawing if and only if the following conditions are met:*

1. *The outer cycle C_O contains at least four 2-vertices.*
2. *Each 2-legged cycle contains at least two 2-vertices.*
3. *Each 3-legged cycle contains at least one 2-vertex.*

The conditions in Theorem 1 can be reformulated as a single condition requiring the number of 2-vertices plus the number of legged-vertices to be at least 4 in each cycle. Note that a cycle is drawn as an orthogonal polygon in an orthogonal drawing. Since an orthogonal polygon must have at least four 90° corners, the necessity of the conditions in Theorem 1 follows from the fact that only 2-vertices and legged-vertices can be drawn as 90° corners of a cycle. Corollary 2 is an immediate consequence of Theorem 1.

▶ **Corollary 2.** *A biconnected planar 3-graph G admits an orthogonal drawing using x bends if and only if there is a plane graph $G^\bullet$ which is a subdivision of G with $|V(G^\bullet)| - |V(G)| = x$ meeting the three conditions in Theorem 1.*

To better understand Theorem 1 and Corollary 2, consider the plane graph G in Fig. 1, the non-boundary 2-legged cycle $C_1 = (h, m, l, j, i)$, and the boundary 3-legged cycle $C_2 = (d, e, k)$. As $C_O(G)$ contains only one 2-vertex f, C_1 contains only one 2-vertex i, and $V(C_2)$ contains no 2-vertex, all three conditions in Theorem 1 are violated. The plane graph $G^\bullet$ in Fig. 1, which results from making 3 subdivisions in G, fulfills the three conditions. A no-bend orthogonal drawing of $G^\bullet$ can be seen as an orthogonal drawing of G with 3 bends.

Observe that in the linear time drawing algorithm of Rahman et al. [19], it is possible to choose *any* set of four 2-vertices in C_O as four convex corners in the outer boundary.[5] Hence we have the following theorem.

[5] The outer boundary refers to the orthogonal polygon corresponding to C_O in the drawing. Note that a convex corner in the outer boundary is also a concave corner of the outer face.

▶ **Theorem 3** ([19]). *Let G be a biconnected plane 3-graph that admits a no-bend orthogonal drawing, and let S be a set of at most four 2-vertices in C_O. Then there exists a no-bend orthogonal drawing of G in which all vertices in S are convex corners in the outer boundary.*

2.2 Orthogonal Representations and Min Cost Flow Formulation.

Let G be a biconnected plane 3-graph that admits a no-bend orthogonal drawing. A naïve way of describing a no-bend orthogonal drawing of G is to specify the actual coordinates of all vertices. The concept of an *orthogonal representation* [20] allows us to describe the shape of an orthogonal drawing, expressed in terms of angles around the vertices without reporting any information about the actual coordinates of the vertices.[6] Formally speaking, an orthogonal representation consists of assigning an angle $\theta \in \{90°, 180°, 270°\}$ to each pair (v, F) such that the vertex v belongs to the cycle surrounding face F. This indicates that v is a degree θ corner in face F. It is required that the summation of all angles around a vertex is 360°, and the difference between the number of convex corners and the number of concave corners is 4 in each inner face, and is -4 in the outer face. Given an orthogonal representation meeting the above requirements, in $O(n)$ time a no-bend orthogonal drawing realizing the angle assigned to each corner can be constructed [20].

In view of the above, the task of bend-minimization of a biconnected plane 3-graph reduces to applying a minimum number of subdivisions to make the graph to have an orthogonal representation. This can be formulated as a min-cost flow problem [20]. Each vertex v of degree k supplies $4 - k$ units of flow. Each inner face F consumes $p - 4$ units of flow, where p is the number of vertices surrounding F. The outer face F_O consumes $p + 4$ units of flow, where p is the number of vertices in C_O. For each pair (v, F) such that v belongs to the cycle surrounding F, add an arc (v, F) with capacity 2 and cost 0. Flowing k units of flow from v to F indicates that v is a $90(k + 1)°$ corner in F. For each edge e which borders the two faces F and F', add two arcs (F, F') and (F', F) with capacity 4 and cost 1. Flowing k units of flow from F to F' along the arc associated with e indicates that the edge e is subdivided k times, and these k new 2-vertices are convex corners in F and concave corners in F'. Since Theorem 1 implies that subdividing an edge more than 4 times makes no use, it is fine to set the capacity of these arcs to 4. It is straightforward to verify that any feasible flow corresponds to an orthogonal representation, and the cost of the flow equals the number of 2-vertices introduced by subdivisions. The flow network can be made unit-capacity by emulating an arc of capacity k by k arcs with capacity 1. Since the total number of arcs in the flow network is $m = O(n)$, and since the maximum cost of an arc is $W = 1$, the min-cost flow algorithm of [6] solves the bend-minimization problem of a biconnected plane 3-graph in $\tilde{O}(m^{10/7} \log W) = \tilde{O}(n^{10/7})$ time.

▶ **Theorem 4.** *A bend-minimized orthogonal drawing of a biconnected plane 3-graph can be constructed in $T(n) = \tilde{O}(n^{10/7})$ time.*

We are not aware of any unit-capacity min-cost flow formulation of bend-minimization of plane 4-graphs, so the $O(n^{1.5})$ time algorithm of [7] remains state-of-the-art.

[6] Here we only describe a simplified version of orthogonal representations that works for no-bend drawings of biconnected plane 3-graphs. See [20] for a complete definition that handles bends and general plane graphs.

Algorithm 1: Min-Bend-Draw(G, s).

Input: (1) a biconnected planar 3-graph G that is not a cycle, (2) a 2-vertex $s \in V(G)$
Output: a bend-minimized orthogonal drawing of G subject to the condition that s belongs to the outer boundary

1 Let u and v be the two neighbors of s.
2 **for** *each* $b \in \{1, 2, 3\}$ **do**
3 $\quad \tilde{G} \leftarrow$ the graph resulting from replacing (u, s, v) with a path $P = (u, s_1, \ldots, s_b, v)$.
4 $\quad G^\bullet \leftarrow$ Subdiv-Embed$(\tilde{G}, P)$.
5 $\quad$ Compute a no-bend orthogonal drawing of $G^\bullet$ (which can be seen as an orthogonal drawing of G).
6 Among all drawings computed, return the one that uses the minimum number of bends.

3 The Main Algorithm

In this section we present our main algorithm, which applies $O(n)$ calls to a subroutine that solves a *constrained* version of the bend-minimization problem. Let G be a biconnected planar 3-graph, and P be a u–v path such that $V(P) \setminus \{u, v\}$ contains only 2-vertices. A *P-orthogonal drawing* of G is an orthogonal drawing of G such that the outer cycle contains P as a subpath, and no bend is imposed on P. A P-orthogonal drawing for a biconnected plane 3-graph G, with $P \subseteq C_O(G)$, is defined analogously.

The procedure Subdiv-Embed(G, P), which is described in the next section, computes a plane graph $G^\bullet$ such that any no-bend orthogonal drawing of $G^\bullet$ is a bend-minimized P-orthogonal drawing of G. More precisely, the plane graph $G^\bullet$ is required to meet the following conditions:

- $G^\bullet$ is a subdivision of G, and no subdivision is made on the path P.
- P belongs to the outer cycle of $G^\bullet$.
- $G^\bullet$ admits a no-bend orthogonal drawing.
- Among all plane graphs meeting the above 3 criteria, $G^\bullet$ is chosen such that $|V(G^\bullet)| - |V(G)|$ is minimized.

To describe our main algorithm, we first note the following result.

▶ **Lemma 5.** *For any orthogonal drawing of a planar graph G, at least one of the following is true: (1) the outer boundary contains a 2-vertex $v \in V(G)$; (2) the outer boundary contains an edge $e \in E(G)$ that has at least one bend.*

Proof. If no 2-vertex of G belongs to C_O, there must be at least 4 bends in the outer cycle to serve as 4 convex corners of the orthogonal polygon corresponding to C_O. ◀

Using Subdiv-Embed as a blackbox, Algorithm 1 and Algorithm 2 compute bend-minimized orthogonal drawings subject to the two cases of Lemma 5. We remark that the reason that we do not need to consider the case of $b > 3$ in these two algorithms is due to Lemma 7. Based on these two algorithms, Algorithm 3, which is the main algorithm of the paper, computes a bend-minimized orthogonal drawing of a biconnected planar 3-graph G that is not a cycle. See Fig. 2 for an illustration of Min-Bend-Draw. The subdivisions introduced in the procedure Subdiv-Embed are drawn as white dots.

Let G be a plane graph, and let $H = G[C]$ be a subgraph of G such that C is a 2-legged cycle. Define *flipping* H as the operation that reverses the cyclic order of the edges incident to v in the combinatorial embedding of G, for each $v \in V(H)$. As long as C is 2-legged, the

Algorithm 2: Min-Bend-Draw(G, e).

Input: (1) a biconnected planar 3-graph G that is not a cycle, (2) an edge $e = \{u, v\} \in E(G)$

Output: a bend-minimized orthogonal drawing of G subject to the condition that e has at least one bend and belongs to the outer boundary

1 **for** *each* $b \in \{1, 2, 3\}$ **do**
2 $\quad \tilde{G} \leftarrow$ the graph resulting from replacing (u, v) with a path $P = (u, s_1, \ldots, s_b, v)$.
3 $\quad G^\bullet \leftarrow$ Subdiv-Embed$(\tilde{G}, P)$.
4 $\quad$ Compute a no-bend orthogonal drawing of $G^\bullet$ (which can be seen as an orthogonal drawing of G).
5 Among all drawings computed, return the one that uses the minimum number of bends.

Algorithm 3: Min-Bend-Draw(G).

Input: a biconnected planar 3-graph G that is not a cycle

Output: a bend-minimized orthogonal drawing of G

1 For each $e \in E(G)$, run Min-Bend-Draw(G, e).
2 For each 2-vertex $s \in V(G)$, run Min-Bend-Draw(G, s).
3 Among all drawings computed, return the one that uses the minimum number of bends.

flipping operation preserves the planarity of G. For example, in Fig. 3 the plane graph G_2 is resulting from flipping $H = G[C]$ in the plane graph G_1, where $C = (c, k, l, h, i, e, d)$. To prove the correctness of Min-Bend-Draw(G), we need the following lemmas.

▶ **Lemma 6.** *Let G be a biconnected plane 3-graph admitting a no-bend orthogonal drawing. Let C be a boundary 2-legged cycle that has no boundary 2-vertex. Then flipping $G[C]$ preserves the property of having a no-bend orthogonal drawing.*

▶ **Lemma 7.** *Let G be a planar graph that is not a cycle, and P be a path of four consecutive 2-vertices in G. Then smoothing one 2-vertex in P to make it a path of three consecutive 2-vertices does not increase the minimum number of bends needed to have an orthogonal drawing.*

Due to the space limit, the proof of Lemma 6 and Lemma 7 are omitted (The proof of Lemma 7 utilizes Lemma 6). Refer to Fig. 3 for an illustration:

- The plane graph G_2 is resulting from flipping $H = G[C]$ in the plane graph G_1, where $C = (c, k, l, h, i, e, d)$. The property of having no-bend orthogonal drawing is preserved.
- If we treat G_1 and G_3 as planar graphs, the planar graph G_3 is resulting from smoothing f in G_1; the graph G_3 still has a no-bend orthogonal drawing (with a different embedding than G_1).

▶ **Theorem 8.** *Min-Bend-Draw(G), Min-Bend-Draw(G, e), and Min-Bend-Draw(G, s) are correct.*

Proof. The correctness of Min-Bend-Draw(G, e) and Min-Bend-Draw(G, s) follows from the correctness of Subdiv-Embed. Note that Lemma 7 allows us not to consider the case of $b > 3$.

Lemma 5 ensures that there exists a bend-minimized orthogonal drawing of G such that either (1) the outer boundary contains a 2-vertex $s \in V(G)$, or (2) the outer boundary

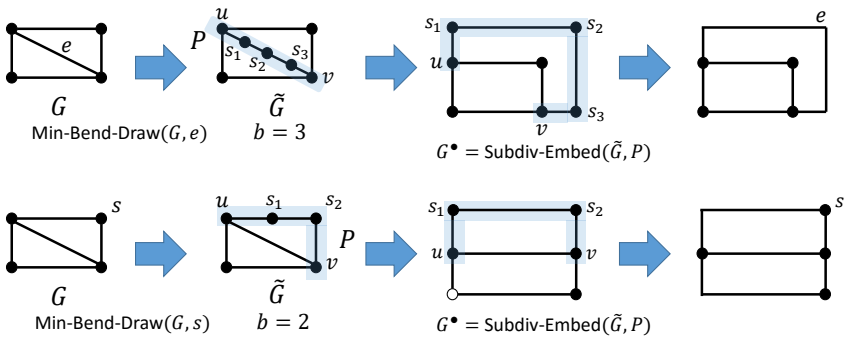


Figure 2 Illustration of Min-Bend-Draw.

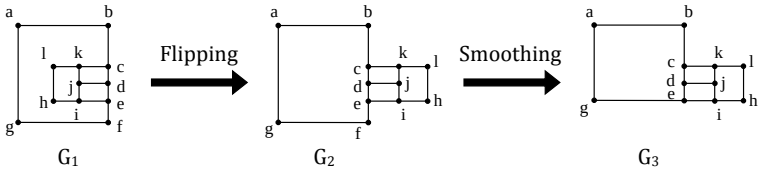


Figure 3 Flipping and smoothing.

contains an edge $e \in E(G)$ whose number of bends is at least 1. Therefore, among all edges $e \in E(G)$ and all 2-vertices $s \in V(G)$, one of Min-Bend-Draw(G, e) or Min-Bend-Draw(G, s) returns a bend-minimized orthogonal drawing. Thus, Min-Bend-Draw(G) is correct. ◀

4 Constrained Orthogonal Drawing

In this section we describe the procedure Subdiv-Embed(G, P) and prove its correctness. Recall that we need Subdiv-Embed(G, P) to return a plane graph $G^\bullet$ such that its no-bend orthogonal drawing is also a bend-minimized P-orthogonal drawing of G.

To better understand how Subdiv-Embed(G, P) works, the reader is encouraged to consult Fig. 4 as our discussion proceeds. Let G be a biconnected planar 3-graph, and P be a u–v path in G such that $V(P) \setminus \{u, v\}$ contains only 2-vertices (see G in the leftmost figure of Fig. 4). With respect to the given G and P, a biconnected subgraph B with $E(P) \subsetneq E(G) \setminus E(B)$ is said to be *essential* if there exists an embedding of G where $P \subseteq C_O$ such that in this particular embedding the unique cycle C with $B = G[C]$ is 2-legged. Note that changing "there exists" to "for all" does not alter the definition. That is, for any essential subgraph B, in *any* embedding of G with $P \subseteq C_O$, the unique cycle C with $B = G[C]$ must be 2-legged. Moreover, all 2-legged cycles C resulting from different embeddings of G with $P \subseteq C_O$ have the *same* two 2-vertices s and t. We define $\{s, t\}$ as the *poles* of B.[7]

With respect to G and P, an essential subgraph B is said to be *maximal* if for all essential subgraphs B', either $E(B) \cap E(B') = \emptyset$ or $B' \subseteq B$. Fig. 4 specifies two maximal essential subgraphs B_x and B_y. We write $\mathbb{B}(G, P)$ to denote the set of maximal essential subgraphs with respect to G and P. Given a fixed planar embedding of G with $P \subseteq C_O$, $\mathbb{B}_{\text{in}}(G, P)$ is

[7] These terms are related to SPQR tree, as described below. In an SPQR tree of G with any reference edge e in P, an essential subgraph B is a pertinent graph (with poles $\{s, t\}$) associated with either a P-node or an R-node. The reason that B is not associated with an S-node is that B is biconnected. See Section 6 for more details.

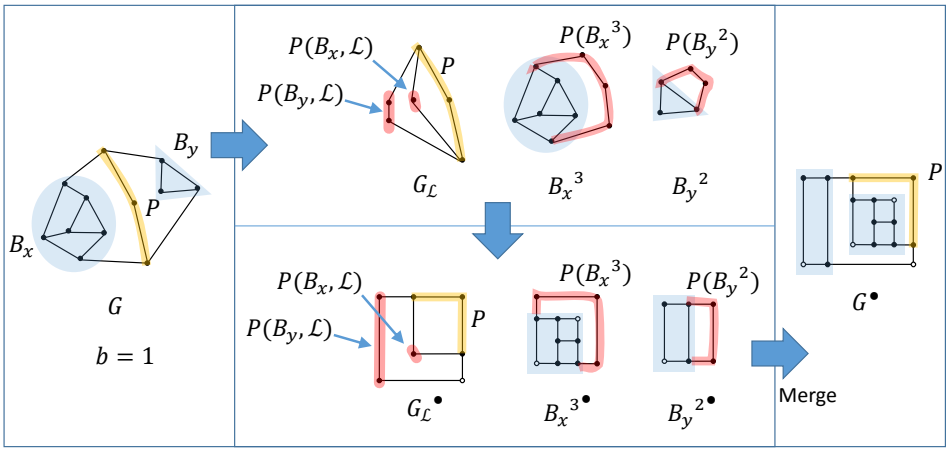


Figure 4 Illustration of Subdiv-Embed.

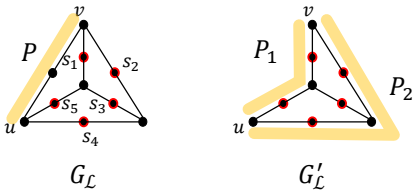


Figure 5 An example illustrating the proof of Lemma 10.

defined as the set of maximal essential subgraphs that does not have any edge in C_O, and $\mathbb{B}_{\text{out}}(G, P)$ is defined as $\mathbb{B}(G, P) \setminus \mathbb{B}_{\text{in}}(G, P)$.

For any essential subgraph B, we write B^b to denote the planar graph resulting from adding a path $(s, r_1, \ldots, r_b, t)$ to B, where $\{s, t\}$ are the poles of B. We write $P(B^b)$ to denote the path $(s, r_1, \ldots, r_b, t)$. The upper figure of Fig. 4 shows B_x^3, $P(B_x^3)$, B_y^2 and $P(B_y^2)$. Intuitively, the path $(s, r_1, \ldots, r_b, t)$ is an abstraction for the sub-drawing (with b convex corners) to which the drawing of B will eventually be attached, and $4 - b$ represents the number of convex corners that the drawing of B is capable to contribute in forming the final orthogonal drawing.

Given a function $\mathcal{L}$ that maps $\mathbb{B}_{\text{out}}(G, P)$ to $\{1, 2, 3\}$, we write $G_{\mathcal{L}}$ to denote the plane graph resulting from replacing each $B \in \mathbb{B}_{\text{out}}(G, P)$ with a path of $4 - \mathcal{L}(B)$ 2-vertices $(r_1, \ldots, r_{4-\mathcal{L}(B)})$ and replacing each $B \in \mathbb{B}_{\text{in}}(G, P)$ with a 2-vertex r. See $G_{\mathcal{L}}$ in the upper figure of Fig. 4. Let $\{s, s'\}$ and $\{t, t'\}$ be the two (uniquely defined) edges in $E(G) \setminus E(B)$, where $\{s, t\}$ are the poles of B. We write $P(B, \mathcal{L})$ to denote the path of $4 - \mathcal{L}(B)$ 2-vertices or the single 2-vertex in $G_{\mathcal{L}}$ that replaces B, depending on whether $B \in \mathbb{B}_{\text{out}}(G, P)$ or $B \in \mathbb{B}_{\text{in}}(G, P)$.

▶ **Definition 9.** A function $\mathcal{L}$ that maps $\mathbb{B}_{\text{out}}(G, P)$ to $\{1, 2, 3\}$ is a *valid labeling* if and only if $|\mathbb{B}_{\text{out}}(G, P)| \geq 3$ implies $\mathcal{L}(B) = 3$ for all $B \in \mathbb{B}_{\text{out}}(G, P)$.

The intuition behind Definition 9 is as follows. Recall that 4 convex corners are needed in the outer boundary of an orthogonal drawing. The path P can supply at least 1 convex corner (P has at least one 2-vertex). Thus, if $|\mathbb{B}_{\text{out}}(G, P)| \geq 3$, it suffices that each $B \in \mathbb{B}_{\text{out}}(G, P)$ supplies 1 convex corner.

▶ **Lemma 10.** *Among all embeddings of G with $P \subseteq C_O$, there are at most two ways of partitioning $\mathbb{B}(G,P)$ into $\mathbb{B}_{in}(G,P)$ and $\mathbb{B}_{out}(G,P)$. Moreover, given (1) a labeling $\mathcal{L}$ that maps $\mathbb{B}_{out}(G,P)$ to $\{1,2,3\}$, and (2) a set $\mathbb{B}_{out}(G,P)$, the plane graph $G_{\mathcal{L}}$ (if it exists) is uniquely determined.*

Proof. First of all, we can restrict our consideration to the function $\mathcal{L}$ such that $\mathcal{L}(B) = 1$ for all B, as taking subdivisions does not affect the number of planar embeddings.

To understand the proof better, the reader is referred to Fig. 5 for an illustrating example, in which $s_1, \ldots, s_5$ are 2-vertices corresponding to essential subgraphs $B_1, \ldots, B_5$, respectively. Let $G'_{\mathcal{L}}$ be the planar graph resulting from removing the intermediate vertices of the u–v path P in $G_{\mathcal{L}}$ and neglecting the planar embedding of $G_{\mathcal{L}}$. It is clear that in any planar embedding of $G'_{\mathcal{L}}$ such that $u, v \in V(C_O)$, there is no 2-legged cycle C in $G'_{\mathcal{L}}$ such that C contains both u and v. The existence of such a cycle C violates the definition of $\mathbb{B}(G,P)$, as C would have been contracted into a 2-vertex in $G_{\mathcal{L}}$.

Therefore, under the constraint that $u, v \in V(C_O)$ in an embedding of $G'_{\mathcal{L}}$, there is no way to flip $G'_{\mathcal{L}}[C]$ in $G'_{\mathcal{L}}$, where C is any 2-legged cycle. With respect to any embedding of $G'_{\mathcal{L}}$ such that $u, v \in V(C_O)$, let P_1 and P_2 be the two u–v paths along the outer boundary, and define $S_i = \{B | P(B,\mathcal{L}) \subseteq P_i\}$, where $i \in \{1,2\}$. The unordered pair $\{S_1, S_2\}$ is independent of the chosen embedding (since no flipping operation can be performed).

The outer boundary of $G_{\mathcal{L}}$ is either P together with P_1 (i.e., $\mathbb{B}_{\text{out}}(G,P)$=$\{B_1, B_5\}$) or P together with P_2 (i.e., $\mathbb{B}_{\text{out}}(G,P)$=$\{B_2, B_4\}$). Therefore, $\mathbb{B}_{\text{out}}(G,P)$ can only be S_1 or S_2, and once $\mathbb{B}_{\text{out}}(G,P)$ is fixed, the planar embedding of $G_{\mathcal{L}}$ is fixed. ◀

The procedure Subdiv-Embed(G,P) is defined as Algorithm 4. The procedure uses a subroutine Merge which constructs a plane graph $G^\bullet$ from the following plane graphs:

- ${G_{\mathcal{L}}}^\bullet \leftarrow$ a subdivision of $G_{\mathcal{L}}$ that admits a no-bend orthogonal drawing.
- ${B^3}^\bullet \leftarrow$ Subdiv-Embed$(B^3, P(B^3))$, for each $B \in \mathbb{B}_{\text{in}}(G,P)$.
- ${B^{\mathcal{L}(B)}}^\bullet \leftarrow$ Subdiv-Embed$(B^{\mathcal{L}(B)}, P(B^{\mathcal{L}(B)}))$, for each $B \in \mathbb{B}_{\text{out}}(G,P)$.

Recall that the plane graph $G^\bullet$ is a planar embedding of a subdivision of G that admits a no-bend orthogonal drawing. In addition, we require $|V(G^\bullet)| - |V(G)|$ to be $|V({G_{\mathcal{L}}}^\bullet)| - |V(G_{\mathcal{L}})| + \left(\sum_{B \in \mathbb{B}_{\text{in}}(G,P)} |V({B^3}^\bullet)| - |V(B^3)|\right) + \left(\sum_{B \in \mathbb{B}_{\text{out}}(G,P)} |V({B^{\mathcal{L}(B)}}^\bullet)| - |V(B^{\mathcal{L}(B)})|\right)$. Intuitively, $G^\bullet$ is constructed by merging these plane graphs in such a way that maintains the property of having a no-bend orthogonal drawing without making any new subdivision (see Fig. 4). As guaranteed by Lemma 10, there are two ways of partitioning $\mathbb{B}(G,P)$, and the upper middle figure shows one of the two in which $\mathbb{B}_{\text{out}}(G,P) = \{B_y\}$ and $\mathbb{B}_{\text{in}}(G,P) = \{B_x\}$. Procedure Subdiv-Embed$(G,P)$ produces ${G_{\mathcal{L}}}^\bullet$ (which is a subdivision of $G_{\mathcal{L}}$), $B_x^{3\bullet}$ (i.e., Subdiv-Embed$(B_x^3, P(B_x^3))$ which is a subdivision of B_x^3), and $B_y^{2\bullet}$ (i.e., Subdiv-Embed$(B_y^2, P(B_y^2))$, which is a subdivision of B_y^2). Note that the white dots in the drawing indicate 2-vertices created by subdivisions. Also note that displaying the orthogonal drawings of $B_x^{3\bullet}$, $B_y^{2\bullet}$, and $G^\bullet$ in Fig. 4 are simply for illustrating purposes. We only compute their planar embeddings and subdivisions, and no specific drawing is fixed. The description of the procedure Merge is left to the next section.

The P-orthogonal drawing of the plane graph $G_{\mathcal{L}}$ in Algorithm 4 can be computed using any orthogonal drawing algorithm for plane 3-graphs that allows us to restrict some edges to have no bend. This can be done in time $T(|V(G_{\mathcal{L}})|) = \tilde{O}(|V(G_{\mathcal{L}})|^{10/7})$ using the min-cost flow described in Sec. 2.2. It is straightforward to modify the min-cost flow to restrict some edges to have no bend. The proof of the correctness of Subdiv-Embed(G,P) is omitted due to the space limit.

Algorithm 4: Subdiv-Embed(G, P).

Input: (1) a biconnected planar 3-graph G that is not a cycle, and (2) a u–v path P in G such that $V(P) \setminus \{u, v\}$ consists of exactly b 2-vertices, where $b \in \{1, 2, 3\}$

Output: a plane graph $G^\bullet$ which is a planar embedding of a subdivision of G such that no subdivision is made on P, and any no-bend orthogonal drawing of $G^\bullet$ is also a bend-minimized P-orthogonal drawing of G

1 **for** *the at most 2 possibilities of partitioning* $\mathbb{B}(G, P)$ *into* $\mathbb{B}_{in}(G, P)$ *and* $\mathbb{B}_{out}(G, P)$, *and the at most 9 possibilities of valid labelings* $\mathcal{L}$ **do**

2 Construct a bend-minimized P-orthogonal drawing D of the plane graph $G_\mathcal{L}$ subject to the constraint that no bend is made on the path $(r_1, \ldots, r_{4-\mathcal{L}(B)})$ that replaces B, for each $B \in \mathbb{B}_{\text{out}}(G, P)$. Let ${G_\mathcal{L}}^\bullet$ be the subdivision of $G_\mathcal{L}$ resulting from replacing each bend in D by a 2-vertex.

3 For each $B \in \mathbb{B}_{\text{in}}(G, P)$, let ${B^3}^\bullet \leftarrow$ Subdiv-Embed$(B^3, P(B^3))$.

4 For each $B \in \mathbb{B}_{\text{out}}(G, P)$, let ${B^{\mathcal{L}(B)}}^\bullet \leftarrow$ Subdiv-Embed$(B^{\mathcal{L}(B)}, P(B^{\mathcal{L}(B)}))$.

5 Use Merge to construct a planar embedding of a subdivision of G from ${G_\mathcal{L}}^\bullet$, $\{{B^3}^\bullet\}_{B \in \mathbb{B}_{\text{in}}(G,P)}$, and $\{{B^{\mathcal{L}(B)}}^\bullet\}_{B \in \mathbb{B}_{\text{out}}(G,P)}$.

6 Among all planar embeddings of subdivisions of G computed, return the one that uses the minimum number of subdivisions.

▶ **Theorem 11.** *Procedure Subdiv-Embed(G, P) returns a plane graph $G^\bullet$ such that any no-bend orthogonal drawing of $G^\bullet$ is a bend-minimized P-orthogonal drawing of G.*

We will later see that the procedure Merge admits an implementation that runs in $|\mathbb{B}(G, P)|$ time. This leads to the following lemma.

▶ **Lemma 12.** *Suppose that Subdiv-Embed$(B^i, P(B^i))$ for each $B \in \mathbb{B}(G, P)$, $i \in \{1, 2, 3\}$ are precomputed. Subdiv-Embed(G, P) is in $O(T(|E(G)| - \sum_{B \in \mathbb{B}(G,P)} |E(B)|))$ time.*

Proof. The drawing D of $G_\mathcal{L}$ can be computed in $O(T(|V(G_\mathcal{L})|))$ time. The procedure Merge takes $O(|\mathbb{B}(G, P)|) \leq O(|V(G_\mathcal{L})|)$ time. The lemma follows from the fact that $|V(G_\mathcal{L})|$ and $|E(G)| - \sum_{B \in \mathbb{B}(G,P)} |E(B)|$ differs by at most a constant factor. ◀

5 Merging Subgraphs

In this section we describe the procedure Merge. For each essential subgraph B, we define $B^\bullet$ as its corresponding subgraph in ${B^3}^\bullet$ or ${B^{\mathcal{L}(B)}}^\bullet$, depending on whether $B \in \mathbb{B}_{\text{in}}(G, P)$ or $B \in \mathbb{B}_{\text{out}}(G, P)$. Here $B^\bullet$ is defined as a plane graph instead of a planar graph. Observe that simply replacing each $P(B, \mathcal{L})$ with $B^\bullet$ (i.e., an expansion) yields a planar embedding of a subdivision of G that meets the constraint on the number of subdivisions made (i.e., $|V(G^\bullet)| - |V(G)|$). But this does not guarantee that the resulting plane graph $G^\bullet$ has a no-bend P-orthogonal drawing. A key in our merging procedure is that after replacing $P(B, \mathcal{L})$ with $B^\bullet$, $B^\bullet$ becomes a subgraph of $G^\bullet$ that can be flipped. We will see that, by appropriately flipping some $B^\bullet$, we obtain a planar embedding that has a no-bend P-orthogonal drawing.

Let B be an essential subgraph of G with poles $\{s, t\}$. We let $e_s = \{s, s'\}$ and $e_t = \{t, t'\}$ be the two edges incident to s and t, respectively, from the outside of B. The two edges e_s and e_t are well-defined since both s and t already have two incident edges in B as B is required to be biconnected. Recall that in Subdiv-Embed(G, P) an orthogonal drawing D of $G_\mathcal{L}$ is computed. By definition, D is also a no-bend orthogonal drawing of the plane

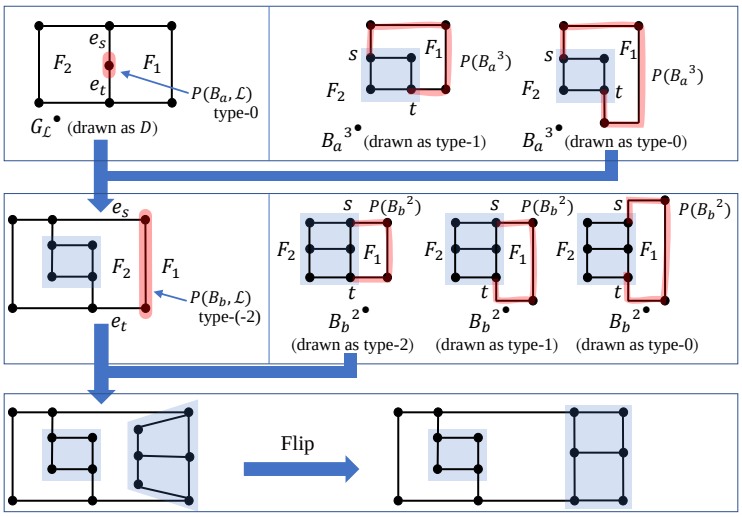


Figure 6 Illustration of Merge.

graph $G_{\mathcal{L}}{}^{\bullet}$. We let F_1 (resp., F_2) be the face in $G_{\mathcal{L}}{}^{\bullet}$ such that s' is followed by s (resp., s is followed by s') in the counter-clockwise ordering of vertices of its facial cycle. We say that $P(B, \mathcal{L})$ is of *type-k* if the number of convex corners minus the number of concave corners of F_1 in $P(B, \mathcal{L})$ is k in D. Note that $P(B, \mathcal{L})$ is of *type-k* if and only if the number of convex corners minus the number of concave corners of F_2 in $P(B, \mathcal{L})$ is $-k$, since a vertex in $P(B, \mathcal{L})$ is convex in F_1 if and only if it is concave in F_2. See Fig. 6 for examples of types of $P(B, \mathcal{L})$.

Next, we define the type of $B^{\bullet}$ with respect to a specific orthogonal drawing D' of $B^{3\bullet}$ or $B^{\mathcal{L}(B)\bullet}$. For notational simplicity, we write $i = 3$ if $B \in \mathbb{B}_{\text{in}}(G, P)$, and $i = \mathcal{L}(B)$ otherwise. Let $(s, x_1, \ldots, x_a, t, y_1, \ldots, y_b)$ be the clockwise ordering of vertices in the cycle surrounding $B^{\bullet}$, and let $P_1 = (s, x_1, \ldots, x_a, t)$ and $P_2 = (s, y_b, \ldots, y_1, t)$. Among the two faces in $B^{i\bullet}$ that are not within the subgraph $B^{\bullet}$, we let F_1 (resp., F_2) be the face that has P_1 (resp., P_2) as a subpath in the facial cycle. Then we say $B^{\bullet}$ is of *type-k* in a given orthogonal drawing of $B^{i\bullet}$ if the number of convex corners minus the number of concave corners of F_1 in P_1 is k. Note that $B^{\bullet}$ is of *type-k* if and only if the number of convex corners minus the number of concave corners of F_2 in P_2 is $-k$. See Fig. 6 for examples of $B^{\bullet}$ of different types and drawings of $B^{i\bullet}$ that realize these types.

Recall that our algorithm does not fix any specific orthogonal drawing of $B^{i\bullet}$. We define $\mathcal{T}(B^{\bullet})$ as the set of integers such that $k \in \mathcal{T}(B^{\bullet})$ if there exists an orthogonal drawing D' of $B^{i\bullet}$ such that (1) $B^{\bullet}$ is of type-k with respect to the drawing D', (2) in the drawing D', no bend is made in the subgraph $B^{\bullet}$ (but it is allowed to have bends in the path $P(B^i)$).

▶ **Lemma 13.** *The type of $P(B, \mathcal{L})$ is within $\{-4+i, \ldots, 4-i\}$, where $i = 3$ if $B \in \mathbb{B}_{in}(G, P)$, and $i = \mathcal{L}(B)$ otherwise.*

Proof. As the path $P(B, \mathcal{L})$ contains exactly $4 - i$ 2-vertices, the result follows. ◀

▶ **Lemma 14.** *If s is followed by t (resp., t is followed by s) in $P(B^i)$ in the counter-clockwise ordering of vertices of the outer cycle of $B^{i\bullet}$, then $\mathcal{T}(B^{\bullet})$ contains all of $0, -1, \ldots, -4+i$ (resp., $0, 1, \ldots, 4-i$).*

Proof. We only focus on the case where t is followed by s in $P(B^i)$ in the counter-clockwise ordering of vertices of the outer cycle of $B^{i\bullet}$. In this case F_1 is an inner face. The proof

Algorithm 5: Merge.

Input: $G_{\mathcal{L}}{}^{\bullet}$ and its no-bend orthogonal drawing D, $\{B^{3\bullet}\}_{B\in\mathbb{B}_{\text{in}}(G,P)}$, and $\{B^{\mathcal{L}(B)\bullet}\}_{B\in\mathbb{B}_{\text{out}}(G,P)}$

Output: a plane graph $G^{\bullet}$ which is a planar embedding of a subdivision of G that admits a no-bend orthogonal drawing

```
Initialize G̃ = G_L•.
for B ∈ 𝔹(G, P) do
    Set i = 3 if B ∈ 𝔹_in(G, P), and i = L(B) otherwise.
    Set b1 as the sign of the type of P(B, L) in D (if the type is 0, then b1 can be
     either −1 or 1).
    Set b2 = 1 if t is followed by s in P(B^i) in the counter-clockwise ordering of
     vertices of the outer cycle of B^i•, and set b2 = −1 otherwise.
    Replace P(B, L) in G̃ with B•.
    Flip the subgraph B• if b1 · b2 < 0.
return G• = G̃.
```

of the other case is similar. To see that $4-i \in \mathcal{T}(B^{\bullet})$, consider any no-bend orthogonal drawing D' of $B^{i\bullet}$ where all the i 2-vertices on the path $P(B^i)$ are drawn as convex corners in F_1. Due to Theorem 3, such a drawing D' exists. The type of $B^{i\bullet}$ with respect to D' is $4-i$ since the number of convex corners minus the number of concave corners of F_1 in P_1 must be $4-i$. In what follows, we prove that $\mathcal{T}(B^{\bullet})$ also contains $0, 1, \ldots, 4-i-1$. For each $x \in \{i+1, \ldots, 4\}$, by adding $x-i$ new 2-vertices (which are treated as bends in a drawing) to the path $P(B^i)$, Theorem 3 allows us to construct an orthogonal drawing D' of $B^{i\bullet}$ where the path $P(B^i)$, excluding the two endpoints, supplies x convex corners in F_1. The type of $B^{i\bullet}$ with respect to D' is $4-x$ since the number of convex corners minus the number of concave corners of F_1 in P_1 must be $4-x$. ◀

Based on Lemma 13 and Lemma 14, we define the Merge procedure as Algorithm 5. In the iteration of the algorithm that processes B, let τ be the type of $P(B,\mathcal{L})$ in D. For the case where b_1 and b_2 have the same sign, there is an orthogonal drawing D' of $B^{i\bullet}$ realizing the type τ (by Lemma 13 and Lemma 14). It is straightforward to see that replacing $P(B,\mathcal{L})$ with the drawing of $B^{\bullet}$ (taken from D') maintains the validity of the orthogonal representation D (since both $P(B,\mathcal{L})$ in D and $B^{\bullet}$ in D' have the same type τ).

For the case where b_1 and b_2 have opposite signs, there is an orthogonal drawing D' of $B^{i\bullet}$ realizing the type $-\tau$, where τ is the type of $P(B,\mathcal{L})$ in D. Similarly, if the replacement is done with the drawing of $B^{\bullet}$ taken from D', then after flipping the subgraph $B^{\bullet}$, the validity of the orthogonal representation D is maintained (the flipping cancels the effect of opposite signs).

Though the correctness of Algorithm 5 is based on the existence of certain drawings of $B^{i\bullet}$, there is no need to compute these drawings. Therefore, the Algorithm 5 takes only $|\mathbb{B}(G,P)|$ time. See Fig. 6 for an illustration of Merge.

6 SPQR-tree Implementation

In this section we show that the three procedures Min-Bend-Draw(G), Min-Bend-Draw(G,s), and Min-Bend-Draw(G,e) admit efficient implementations based on SPQR-trees [10].

We denote the SPQR-tree of G rooted at the edge e as $\mathbb{T}_{G,e}$. Each node in $\mathbb{T}_{G,e}$ is either an S-,P-,Q-, or R-node. Each node is associated with a subgraph of G which is called the *pertinent graph*. Each pertinent graph B is associated with two vertices $\{s,t\} \subseteq V(H)$, called *poles*, such that removal of s and t disconnects B from the rest of the graph. The root node of $\mathbb{T}_{G,e}$ is a Q-node whose pertinent graph is the subgraph of G resulting from removing the edge e. Let ν be a node in $\mathbb{T}_{G,e}$ that is a descendant of another node μ; then the pertinent graph of ν is a proper subgraph of the pertinent graph of μ.

For a given subgraph $H \subseteq G$ and two vertices $s,t \in V(H)$ such that H does not contain the reference edge e, the following two statements are equivalent:

- B is biconnected, and removing s and t disconnects B from the rest of the graph.
- B is the pertinent graph of a P-node or an R-node μ of $\mathbb{T}_{G,e}$, and the poles of μ are $\{s,t\}$.

For any node μ in $\mathbb{T}_{G,e}$, we define $\mathbb{B}(\mu)$ as the set of all pertinent graphs B meeting the following condition: B is associated with a P-node or an R-node ν which is a descendant of μ such that all intermediate nodes in the directed path $(\mu,\ldots,\nu)$ in the SPQR-tree $\mathbb{T}_{G,e}$ contains no P-node and R-node.

Consider the procedure Subdiv-Embed$(\tilde{G},P)$ invoked in an execution of Min-Bend-Draw(G,e) or Min-Bend-Draw(G,s). With respect to the SPQR-tree $\mathbb{T}_{\tilde{G},e'}$ for any arbitrary choice of $e' \in E(P)$, it is clear that $\mathbb{B}(\tilde{G},P)$ is exactly $\mathbb{B}(\mu)$, where μ is the root of $\mathbb{T}_{\tilde{G},e'}$. Therefore, any recursive call Subdiv-Embed$(B^i,P(B^i))$ invoked in the procedure Subdiv-Embed$(\tilde{G},P)$ can be associated with a P-node or an R-node $\nu \in \mathbb{B}(\mu)$ of $\mathbb{T}_{\tilde{G},e}$ in the sense that B is the pertinent graph of ν. Similarly, it is straightforward to see that the set $\mathbb{B}(B^i,P(B^i))$ is exactly $\mathbb{B}(\nu)$, independent of i. Since the SPQR-tree $\mathbb{T}_{\tilde{G},e'}$ can be constructed in linear time, we have the following theorem (which is due to Lemma 12).

▶ **Theorem 15.** *Let $n = |V(G)|$. Both Min-Bend-Draw(G,e) and Min-Bend-Draw (G,s) can be implemented to run in $O(T(n))$ time, and Min-Bend-Draw(G) can be implemented to run in $O(n \cdot T(n))$ time, where $T(n) = \tilde{O}(n^{10/7})$.*

▶ Remark. We comment on the suggestion of an anonymous reviewer regarding the use of the terminology in [9] to derive our result. In the procedure Subdiv-Embed(G,P), the computation of Subdiv-Embed$(B^i,P(B^i))$ for $i \in \{1,2,3\}$ is analogous to the computation of *optimal set* of B in [9]. Theorem 1 actually implies that the *spirality* of a split component of a biconnected planar 3-graph is bounded by a *constant*. This also explains why it suffices to only consider $i \in \{1,2,3\}$. If one goes through the proof details in [9], tracks the dependence on spirality carefully, and incorporates the $\tilde{O}(n^{10/7})$ time algorithm for the fixed-embedding setting (Theorem 4, which is based on [6]) to the approach in [9], an $\tilde{O}(n^{17/7})$ time bend-minimization algorithm can also be obtained using the terminology in [9]. Nonetheless, we feel that our approach (directly based on tools in [19]) is more natural and simpler than [9].

Acknowledgements. We thank the anonymous reviewers for their thoughtful comments.

References

1 Michael A. Bekos, Michael Kaufmann, Stephen G. Kobourov, and Antonios Symvonis. Smooth orthogonal layouts. *JGAA*, 17(5):575–595, 2013.

2 Michael A. Bekos, Michael Kaufmann, Robert Krug, Thorsten Ludwig, Stefan Näher, and Vincenzo Roselli. Slanted orthogonal drawings: Model, algorithms and evaluations. *JGAA*, 18(3):459–489, 2014.

3 Thomas Bläsius, Ignaz Rutter, and Dorothea Wagner. Optimal orthogonal graph drawing with convex bend costs. *ACM Trans. Algorithms*, 12(3):33:1–33:32, 2016.

4 Franz Brandenburg, David Eppstein, Michael T. Goodrich, Stephen Kobourov, Giuseppe Liotta, and Petra Mutzel. Selected open problems in graph drawing. In *Proceedings of the 11th International Symposium on Graph Drawing (GD'03)*, pages 515–539. Springer Berlin Heidelberg, 2004.

5 Yi-Jun Chang and Hsu-Chun Yen. On orthogonally convex drawings of plane graphs. *Computational Geometry*, 62:34–51, 2017.

6 Michael B. Cohen, Aleksander Mądry, Piotr Sankowski, and Adrian Vladu. Negative-weight shortest paths and unit capacity minimum cost flow in $\tilde{O}(m^{10/7} \log W)$ time. In *Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'17)*, pages 752–771. Society for Industrial and Applied Mathematics, 2017.

7 Sabine Cornelsen and Andreas Karrenbauer. Accelerated bend minimization. *JGAA*, 16(3):635–650, 2012.

8 Giuseppe Di Battista, Walter Didimo, Maurizio Patrignani, and Maurizio Pizzonia. Orthogonal and quasi-upward drawings with vertices of prescribed size. In *Proceedings of the 7th International Symposium on Graph Drawing (GD'99)*, pages 297–310. Springer Berlin Heidelberg, 1999.

9 Giuseppe Di Battista, Giuseppe Liotta, and Francesco Vargiu. Spirality and optimal orthogonal drawings. *SIAM Journal on Computing*, 27(6):1764–1811, 1998.

10 Giuseppe Di Battista and Roberto Tamassia. On-line planarity testing. *SIAM Journal on Computing*, 25(5):956–997, 1996.

11 Walter Didimo, Giuseppe Liotta, and Maurizio Patrignani. On the complexity of hv-rectilinear planarity testing. In *Proceedings of the 22nd International Symposium on Graph Drawing (GD'14)*, pages 343–354. Springer Berlin Heidelberg, 2014.

12 Christian A. Duncan and Michael T. Goodrich. Planar orthogonal and polyline drawing algorithms. In Roberto Tamassia, editor, *Handbook of Graph Drawing and Visualization*, chapter 8. CRC Press, 2013.

13 Stephane Durocher, Stefan Felsner, Saeed Mehrabi, and Debajyoti Mondal. Drawing hv-restricted planar graphs. In *Proceedings of the 11th Latin American Theoretical Informatics Symposium (LATIN'14)*, pages 156–167. Springer Berlin Heidelberg, 2014.

14 Ashim Garg and Roberto Tamassia. A new minimum cost flow algorithm with applications to graph drawing. In *Proceedings of the Symposium on Graph Drawing (GD'96)*, pages 201–216. Springer Berlin Heidelberg, 1997.

15 Ashim Garg and Roberto Tamassia. On the computational complexity of upward and rectilinear planarity testing. *SIAM Journal on Computing*, 31(2):601–625, 2001.

16 Gunnar W. Klau and Petra Mutzel. Quasi–orthogonal drawing of planar graphs. Technical Report MPI-I-98-1-013, Max-Planck-Institut für Informatik, Saarbrücken, 1998.

17 Md. Saidur Rahman, Shin-ichi Nakano, and Takao Nishizeki. A linear algorithm for bend-optimal orthogonal drawings of triconnected cubic plane graphs. *JGAA*, 3(4):31–62, 1999.

18 Md. Saidur Rahman and Takao Nishizeki. Bend-minimum orthogonal drawings of plane 3-graphs. In *Proceedings of the 28th International Workshop on Graph-Theoretic Concepts in Computer Science (WG'02)*, pages 367–378. Springer Berlin Heidelberg, 2002.

19 Md. Saidur Rahman, Takao Nishizeki, and Mahmuda Naznin. Orthogonal drawings of plane graphs without bends. *JGAA*, 7(4):335–362, 2003.

20 Roberto Tamassia. On embedding a graph in the grid with the minimum number of bends. *SIAM Journal on Computing*, 16(3):421–444, 1987.

Adaptive Planar Point Location*

Siu-Wing Cheng[1] and Man-Kit Lau[2]

1 **Department of Computer Science and Engineering, HKUST, Hong Kong**
2 **Department of Computer Science and Engineering, HKUST, Hong Kong**

Abstract

We present a self-adjusting point location structure for convex subdivisions. Let n be the number of vertices in a convex subdivision S. Our structure for S uses $O(n)$ space and processes any online query sequence σ in $O(n + \mathrm{OPT})$ time, where OPT is the minimum time required by any linear decision tree for answering point location queries in S to process σ. The $O(n + \mathrm{OPT})$ time bound includes the preprocessing time. Our result is a two-dimensional analog of the static optimality property of splay trees. For connected subdivisions, we achieve a processing time of $O(|\sigma| \log\log n + n + \mathrm{OPT})$.

1998 ACM Subject Classification F.2.2 [Analysis of Algorithms and Problem Complexity] Nonnumerical Algorithms and Problems – Geometrical Problems and Computations

Keywords and phrases point location, planar subdivision, static optimality

Digital Object Identifier 10.4230/LIPIcs.SoCG.2017.30

1 Introduction

Planar point location is a fundamental problem in computational geometry that has been studied extensively. It calls for preprocessing a *planar subdivision* into a data structure so that for any query point, the region in the subdivision that contains the query point can be reported. There are several common types of planar subdivisions. A subdivision is *convex* if the boundary of every region (including the outer boundary) bounds a convex polygon. A subdivision is *connected* if the boundary of every region bounds a simple polygon. A subdivision is *general* if the boundary of every region bounds a polygon possibly with holes. In this paper, we are concerned with point location methods that use point-line comparisons.

Given a general subdivision with n vertices, point location structures with worst-case $O(\log n)$ query time, $O(n \log n)$ preprocessing time, and $O(n)$ space have been obtained [2, 11, 14, 15]. For connected subdivisions, the preprocessing time can be reduced to $O(n)$ [14] after triangulating every region in linear time [6].

When processing a sequence of query points that fall into different regions with vastly different frequencies, one may consider objectives other than minimizing the worst-case time to answer a single query. One scenario is that for every region r, the probability p_r of the query point falling into r is given. In this case, one may want to minimize the expected query time. The entropy $H = \sum_r p_r \log(1/p_r)$, where the sum is over all regions in S, is a lower bound to the expected query time according to Shannon's theory [16]. Arya, Malamatos, and Mount [3] and Iacono [12] studied subdivisions in which all regions have sizes bounded by some constant. They obtained structures that use $O(n)$ space and answer a query in $O(H)$ expected time. Later, Arya, Malamatos, Mount, and Wong [4] improved the expected query time to $H + O(\sqrt{H})$.

* Supported by Research Grants Council, Hong Kong, China (project no. 16201116).

If some regions have non-constant sizes, the entropy is a very weak lower bound. Indeed, Arya, Malamatos, Mount, and Wong [4] showed a convex polygon of n sides and a query distribution so that a query point lies in the polygon with probability $1/2$ and the expected number of point-line comparisons needed to decide whether a query point lies in the polygon is $\Omega(\log n)$. Note that the entropy is only a constant. Several subsequent research works consider comparison against linear decision trees that answer point location queries in planar subdivisions. We call them *point location linear decision tree* for convenience. Given a connected subdivision S with n vertices and the query distribution, Collete et al. [9] designed a structure that uses $O(n)$ space and answers a query in $O(H^*)$ expected time, where H^* is the minimum expected time needed by any point location linear decision tree for S. Afshani, Barbay, and Chan [1] and Bose et al. [5] also obtained optimal solutions (with respect to linear decision trees) for several geometry query problems, including planar point location, when the query distribution is given.

In one dimension, optimal query performance can be obtained without knowing the query distribution. Sleator and Tarjan [17] designed splay trees for storing an ordered set of values such that any online query sequence σ can be processed in $O(|\sigma| + \sum_v f_v \log(|\sigma|/f_v))$ time, where the sum is over all values in the set and f_v denotes the frequency of v being queried in σ, provided that every value is accessed at least once. This result is known as the Static Optimality Theorem [17]. Note that $\sum_v f_v \log(|\sigma|/f_v)$ is the minimum time needed to process σ by any static binary search tree that stores the same set of values.

Does there exist an analog of the Static Optimality Theorem in the context of planar point location? Iacono and Mulzer [13] proposed a self-adjusting point location structure for triangulations. Given a triangulation S, their structure uses $O(n)$ space and processes any online query sequence σ in $O(n + \sum_t f_t \log(|\sigma|/f_t))$ time, where the sum is over all triangles in S and f_t denotes the frequency of a triangle t being hit by a query point in σ. The space usage is $O(n)$. The time bound includes the preprocessing time to construct the first structure before locating the first query point in σ. Note that $\sum_t f_t \log(|\sigma|/f_t)$ is a lower bound to the minimum time needed by a static point location structure for S to process σ. The handling of more general planar subdivisions is posed as an open problem in [13]. Recently, we made progress by designing a self-adjusting point location structure for convex subdivisions [8] based on the result in [13]. Given a convex subdivision S, our structure uses $O(n)$ space and processes any online query sequence σ in $O(|\sigma| \log\log n + n + \mathrm{OPT})$ time, where OPT is the minimum time needed by any point location linear decision tree for S to process σ.

In this paper, we prove an analog of the Static Optimality Theorem for convex subdivisions. We propose a self-adjusting point location structure that processes any online query sequence in $O(n + \mathrm{OPT})$ time, which includes the preprocessing time. The space usage is $O(n)$.

It is known that the optimal point location linear decision tree for an optimally triangulated subdivision has the same asymptotic performance as the optimal point location linear decision tree for the untriangulated subdivision. Therefore, our solution keeps a triangulation of the convex subdivision so that we can invoke Iacono and Mulzer's result [13]. The triangulation also allows us to extract some frequently accessed triangles and keep a separate, smaller point location structure for them. Then, query points in these triangles can be located faster. As observed in [13], the difficulty lies in efficiently computing the optimal triangulation, which depend on the access frequencies. As the access frequencies evolve, the subdivision will need to be retriangulated and the analysis has to address this issue. On the other hand, we showed in [8] that some canonical triangulation methods (independent of the access frequencies) can lower the average extra cost per query to $O(\log\log n)$. Our insight is to *recursively* extract

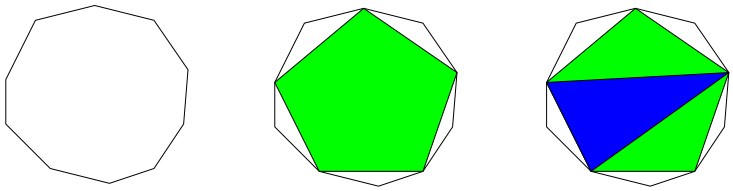


Figure 1 Triangulation of a bounded region in S.

frequently accessed triangles and generate a separate point location structure for them using these canonical triangulation methods. This results in a multi-level structure. The structure at the highest level is queried first and if that fails, we move down the levels. We devise an analysis that handles both successful and unsuccessful queries at each level. Intuitively, the performance improves as the number of levels increases, and thus we circumvent the difficulty of computing an optimal triangulation.

We also observe that the strategy in [8] works for connected subdivisions with the help of balanced geodesic triangulations. This gives a processing time of $O(|\sigma| \log\log n + n + \mathrm{OPT})$.

2 Basics

We state the result of Iacono and Muzler [13] below for future reference.

▶ **Theorem 1** ([13]). *For any planar triangulation T with n vertices, there is a point-line comparison based data structure that uses $O(n)$ space and processes any online sequence of point location queries in T in $O\left(\sum_{t\in T} f(t) \log \frac{N}{f(t)} + n\right)$ time, where N is the number of queries and $f(t)$ is the number of query points that fall into the triangle t in T. The time bounds includes the $O(n)$ preprocessing time.*

We review the canonical triangulation methods in [8] which will be used later.

Let S denote a convex subdivision with n vertices. For each bounded region r in S, the procedure TriReg is called to triangulate r. Figure 1 gives an example. TriReg runs in $O(|r|)$ time and produces a triangulation of $O(|r|)$ size. This triangulation method was first introduced by Dobkin and Kirkpatrick for convex polygon intersection detection [10]. Every line segment in r intersects $O(\log |r|)$ triangles.

TriReg(r)
1. If r is a triangle, then return.
2. Take any maximum subsequence α of vertices of r such that no two vertices in α are adjacent along the boundary of r, except possibly the first and the last ones.
3. Connect the vertices in α to form a convex polygon r'.
4. Call TriReg(r').

Second, we triangulate the exterior region of S. We pick three boundary edges of S such that removing them gives three boundary chains of S of roughly equal sizes. The support lines of these three edges bound a triangle, denoted by $\mathcal{B}_S$, that contains S.[1] We call the three interior-disjoint regions between the boundaries of $\mathcal{B}_S$ and S *boomerangs*. Each boomerang has two straight sides and a reflex chain. Figure 2(a) gives an example. For

[1] It is possible that $\mathcal{B}_S$ is unbounded, but we will assume that $\mathcal{B}_S$ is bounded for simplicity.

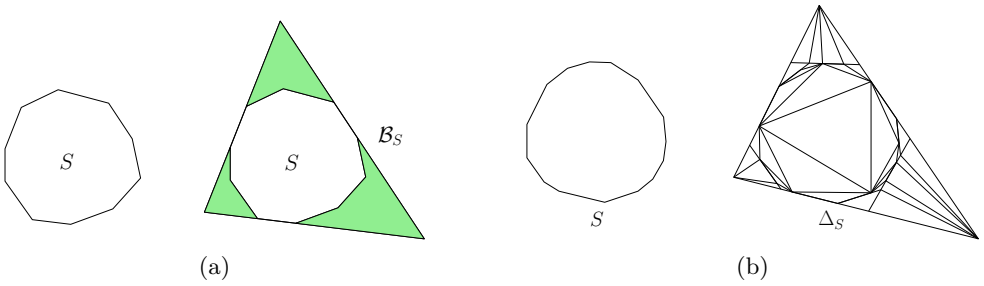


Figure 2 (a) Boomerangs (shown shaded). (b) An example in which S is just one convex polygon.

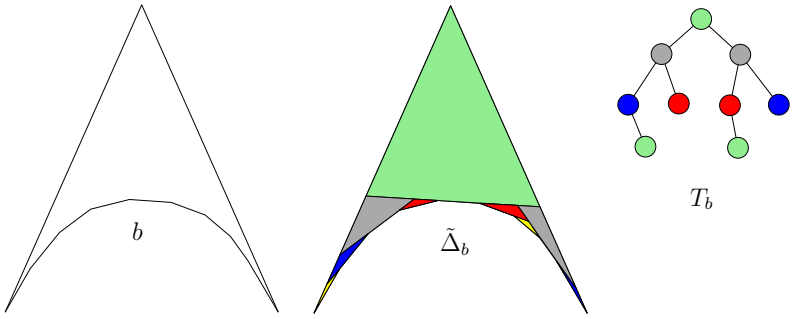


Figure 3 The nodes of T_b are given the same colors as the corresponding regions in $\tilde{\Delta}_b$.

each boomerang b, we call the procedure SplitBR to partition b hierarchically into triangular regions as well as to construct a binary tree that represents this hierarchy. Denote the output partition of b by $\tilde{\Delta}_b$ and the binary tree by T_b. Figure 3 gives an example. The binary tree T_b is not constructed in [8], but we will need it later. SplitBR runs in $O(|b|)$ time, $\tilde{\Delta}_b$ has $O(|b|)$ size, and T_b has $O(\log |b|)$ height.

SplitBR(b)

1. If b is a triangle, then return.
2. Take the middle edge e of the reflex chain of b.
3. Cut b with the support line of e into a triangle t and two smaller boomerangs b_1 and b_2.
4. Call SplitBR(b_1) and SplitBR(b_2) to obtain $\tilde{\Delta}_{b_1}$, T_{b_1}, $\tilde{\Delta}_{b_2}$, and T_{b_2}.
5. $\tilde{\Delta}_b := \{t\} \cup \tilde{\Delta}_{b_1} \cup \tilde{\Delta}_{b_2}$.
6. Create the binary tree T_b with root v containing t. Make T_{b_1} and T_{b_2} left and right subtrees of v.
7. Return $\tilde{\Delta}_b$ and T_b.

Finally, for every triangular region r in $\tilde{\Delta}_b$, there is exactly one side e of r that bounds S. This side e contains $O(\log |b|)$ vertices. We call TriBR(b) to obtain a triangulation of b.

TriBR(b)

1. For each triangular region r in $\tilde{\Delta}_b$, do
   a. take the side e of r that bounds S,
   b. add edges to connect the vertices in e to the vertex of r opposite e.

Every line segment in b intersects $O(\log |b|)$ triangular regions in $\tilde{\Delta}_b$ and there are $O(\log |b|)$ triangles in each triangular region. So each line segment in b intersects $O(\log^2 |b|)$ triangles in the triangulation of b.

We use Δ_S to denote the resulting triangulation of the boomerangs and the bounded regions in S. Figure 2(b) gives an example. There are $O(n)$ vertices in Δ_S and Δ_S can be constructed in $O(n)$ time. Theorem 1 is applied to Δ_S to produce a data structure for answering point location queries in S.

▶ **Theorem 2** ([8]). *Let S be a planar convex subdivision with n vertices. There is a point-line comparison based data structure that uses $O(n)$ space and processes any online sequence σ of point location queries in S in $O(|\sigma| \log\log n + n + \mathrm{OPT})$ time, where* OPT *is the minimum time needed by any point location linear decision tree for S to process σ. The time bound includes the $O(n)$ preprocessing time.*

3 Planar convex subdivision

Let S denote a planar convex subdivision with n vertices. We first present a solution with processing time $O(|\sigma| \log\log\log n + n + \mathrm{OPT})$. Then, we bootstrap from this solution to obtain the optimal result.

3.1 First solution

We first compute Δ_S as described in Section 2. Let D_S be the point location structure in Theorem 2 for S. Note that D_S evolves as σ is processed. Let $f(n)$ denote the function $(\log_2 n)^6$. For any $j \geq 1$, whenever D_S has been used to answer the j-th subset of $f(n)$ queries, we extract a subset of triangles from Δ_S, construct a new triangulation Δ_j using this subset, and then compute a point location structure D_j for Δ_j. Once D_j has been constructed, the next query is answered using D_j first. If D_j locates the query point in a region (bounded or exterior) in S, we are done; otherwise, we use D_S to answer the query. We elaborate on the construction of Δ_j and D_j in the following sections.

3.1.1 Triangulation Δ_j

We extract the subset X of $f(n)(\log_2 n)^{-2}$ triangles in Δ_S that have the highest $f(n)(\log_2 n)^{-2}$ access frequencies currently. Some triangles in X lie in bounded regions in S and some may lie outside S. To extract X quickly, we maintain a doubly linked list A such that the i-th entry of A stores a doubly linked list of triangles with the i-th highest frequency. Whenever we need to output X, we scan the lists in the entries of A in order until we have collected $f(n)(\log_2 n)^{-2}$ triangles. Whenever the frequency of a triangle $t \in \Delta_S$ is incremented, we need to relocate t within A. Suppose that t is currently stored in the list at the i-th entry of A. If the triangles in the list at the $(i-1)$-th entry of A have the same frequency as t, then we move t to the end of the list at the $(i-1)$-th entry. Otherwise, the triangles in the list at the $(i-1)$-th entry have a higher frequency than t, and so we insert a new entry of A between the $(i-1)$-th and the i-th entries and make t a singleton list at this new entry of A. If the list at the i-th entry of A becomes empty after moving t, we delete this entry of A. So each update of A takes $O(1)$ time.

We can assume that for each triangle t in Δ_S, if t lies in a region r in S, then t stores the region id r. Moreover, if r is the exterior region of S, then $t \subseteq \tilde{\Delta}_b$ for some boomerang b and t also stores the id of the triangular region in $\tilde{\Delta}_b$ that contains t. By sorting the triangles in X with respect to their region ids, we can find the triangles in $r \cap X$ for every region r in S.

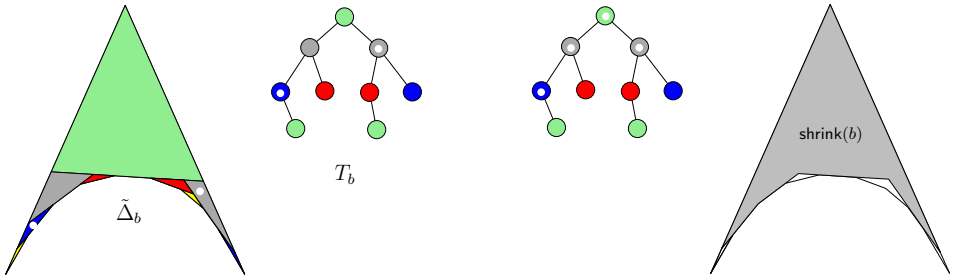


Figure 4 The two triangular regions in $\tilde{\Delta}_b$ with white dots contain some triangles in X. Corresponding nodes in T_b are also marked with white dots. Then, all ancestors of these nodes in T_b are marked, and the union of the corresponding triangular regions in $\tilde{\Delta}_b$ is a boomerang $\mathsf{shrink}(b)$ (shown shaded).

For each bounded region r in S, let $\mathrm{conv}(r \cap X)$ denote the convex hull of the triangles in $r \cap X$ and we can compute $\mathrm{conv}(r \cap X)$ in $O(|r \cap X| \log |r \cap X|)$ time. Then, we call $\mathsf{TriReg}(\mathrm{conv}(r \cap X))$ to triangulate $\mathrm{conv}(r \cap X)$. Each resulting triangle stores the region id r. Summing over all bounded regions in S, the total running time is $O(|X| \log |X|) = O(f(n)/\log n)$ and the total number of triangles produced is $O(|X|) = O(f(n)(\log n)^{-2})$.

Next, we handle the triangles in X outside S. Let b be one of the boomerangs between the boundaries of $\mathcal{B}_S$ and S. For each triangle $t \in b \cap X$, we mark the triangular region r in the binary tree T_b that contains t and we also mark all ancestors of r in T_b. We form the union of the marked triangular regions. Denote the union by $\mathsf{shrink}(b)$. Note that $\mathsf{shrink}(b)$ is a boomerang and every edge in the reflex chain of $\mathsf{shrink}(b)$ supports an outer boundary edge of S. Figure 4 gives an example. Also, $b \cap X \subseteq \mathsf{shrink}(b)$, $|\mathsf{shrink}(b)| = O(|b \cap X| \log |b|)$, and $\mathsf{shrink}(b)$ can be computed in $O(|b \cap X| \log |b|)$ time. We call $\mathsf{SplitBR}(\mathsf{shrink}(b))$ and then $\mathsf{TriBR}(\mathsf{shrink}(b))$ to obtain a triangulation of $\mathsf{shrink}(b)$. Each resulting triangle stores the id of the exterior region of S. Summing over all three boomerangs between the boundaries of $\mathcal{B}_S$ and S, the total running time is $O(|X| \log n) = O(f(n)/\log n)$ and the total number of triangles produced is $O(|X| \log n) = O(f(n)/\log n)$.

Collect all $O(f(n)/\log n)$ triangles computed in the above. By a plane sweep, we can add edges in $O\left(\frac{f(n)}{\log n} \log \frac{f(n)}{\log n}\right)$ time to form a triangulation that contains all triangles collected and has size $O(f(n)/\log n)$. This is the triangulation Δ_j desired. The total construction time of Δ_j is $O\left(\frac{f(n)}{\log n} \log \frac{f(n)}{\log n}\right)$. The extra triangles added by the plane sweep do not store the id of any region in S, and therefore, query points that fall into such triangles are not located successfully in S.

3.1.2 Structure D_j, querying, and frequencies

The access frequencies in Δ_S are initialized to be zero before processing σ. The subscript of Δ_j and D_j increases monotonically as we process σ. When the construction of Δ_j and D_j completes, we forget about Δ_{j-1} and D_{j-1} and reuse their storage. The access frequencies in Δ_j are initialized to be zero.

D_j consists of two point location structures D'_j and D''_j. D'_j is obtained by invoking Theorem 1, the result in [13], on Δ_j. D''_j is a worst-case optimal planar point location structure (e.g. [14]). The querying procedure works as follows. Let q be the next query point. We check in $O(1)$ time whether q lies inside $\mathcal{B}_S$. If not, we just output that q is outside S. Suppose that q lies inside $\mathcal{B}_S$. We query D_j by alternating the search steps in D'_j and D''_j.

We stop as soon as a triangle t in Δ_j containing q is found. If t stores a region (bounded or exterior) of S, then we output that region. Otherwise, we use D_S to locate the triangle t' in Δ_S that contains q, and we output the region of S (bounded or exterior) stored at t'.

After locating q, we update the access frequencies in Δ_S or Δ_j. This update is important because the frequencies govern how the method in [13] will adjust D_S and D_j in order to adapt to incoming queries. If q lies outside $\mathcal{B}_S$, we do not change any frequency in Δ_S and Δ_j. Suppose that q lies inside $\mathcal{B}_S$. If q is located in a triangle $t \in \Delta_j$ and t stores a region of S, then we increment the frequency of t in Δ_j and we are done. The frequencies in Δ_S do not change in this case. On the other hand, if the search in D_j does not report a region of S, then q is subsequently located in S by D_S and we increment the frequency of the triangle in Δ_S that contains q. The frequencies in Δ_j do not change in this case.

There are some consequences due to our frequency update. Consider two online query sequences $\alpha_j \subseteq \alpha$ such that D_j can successfully locate in S the query points in α_j, but not the query points in $\alpha \backslash \alpha_j$. Therefore, query points in $\alpha \backslash \alpha_j$ do not cause any change to D_j. Let $D_j(\alpha)$ denote the running time of D_j on α (excluding the preprocessing time of D_j). We conclude that

$$D_j(\alpha) = O(D_j(\alpha_j) + |\alpha \backslash \alpha_j| \log |\Delta_j|) \tag{1}$$

because each query in $\alpha \backslash \alpha_j$ can be answered by D_j'' in $O(\log |\Delta_j|)$ worst-case time.

3.1.3 Analysis

We first analyze the performance of D_j. Among all point location linear decision trees for S, let D be the one that takes the minimum time to process σ. We convert D to a point location linear decision tree for Δ_j as follows.

Each leaf node v of D corresponds to a convex polygon ρ in a region of S. If ρ has k sides, then v has depth at least k as each node of D applies a cut along a line. Therefore, we can expand v into a linear decision subtree so that the leaf nodes of this subtree correspond to a triangulation of ρ and the height of this subtree is at most $k - 2$.

Let $D^{\sigma}_{\min}$ denote the linear decision tree obtained by expanding D as described above. The triangular regions at the leaf nodes of $D^{\sigma}_{\min}$ form a refinement of S. Locating a query point q in this refinement of S using $D^{\sigma}_{\min}$ has the same asymptotical complexity as locating q in S using D.

Let t be the triangle at a leaf node of $D^{\sigma}_{\min}$. We discuss how to expand this leaf node to a linear decision subtree depending on whether t lies in a bounded or unbounded region of S.

Suppose that t lies in a bounded region r of S. Recall that X is the subset of triangles extracted from Δ_S for constructing Δ_j. All vertices of $\text{conv}(r \cap X)$ lie on the boundary of r, so t intersects $O(\log |r \cap X|) = O(\log \log n)$ triangles in $\Delta_j \cap \text{conv}(r \cap X)$, which refine t into a planar subdivision P_t of size $O(\log \log n)$. We expand the leaf node of $D^{\sigma}_{\min}$ storing t to a linear decision subtree L_t that performs point location in P_t in $O(\log \log \log n)$ worst-case query time. Some leaf nodes of L_t correspond to regions in the refinement of t that are outside $\text{conv}(r \cap X)$. We need to expand such leaf nodes further in order to locate query points that fall into $t \setminus \text{conv}(r \cap X)$ in a triangle in Δ_j. We will not be interested in the query time for such query points, so we can expand these leaf nodes of L_t arbitrarily.

Suppose that t lies in the unbounded region of S. We expand the leaf node storing t into a linear decision subtree L'_t of $O(1)$ height such that each leaf node of L'_t corresponds to a triangle that lies inside or outside $t \cap \mathcal{B}_S$. At each leaf node of L'_t outside $t \cap \mathcal{B}_S$, we can output the exterior of $\mathcal{B}_S$. All leaf nodes of L'_t inside $\mathcal{B}_S$ lie in a boomerang b between the boundaries of $\mathcal{B}_S$ and S. Take such a leaf node of L'_t and let t' be the triangle stored

there. We are concerned with the overlay of t' and $\mathsf{shrink}(b)$. Since every edge of the reflex chain of $\mathsf{shrink}(b)$ supports an outer boundary edge of S, every triangular region in $\mathsf{shrink}(b)$ produced by SplitBR is incident on an outer boundary edge of S. Since the interior of t' cannot intersect the boundary of S or any bounded region in S, the interior of t' contains at most one vertex in the reflex chain of $\mathsf{shrink}(b)$. Thus, the boundary of t' inside $\mathsf{shrink}(b)$ consists of $O(1)$ line segments. Each segment intersects $O(\log|\mathsf{shrink}(b)|)$ triangular regions in $\mathsf{shrink}(b)$ produced by SplitBR, and each triangular region contains $O(\log|\mathsf{shrink}(b)|)$ triangles produced by TriBR. As a result, t' intersects $O(\log^2|\mathsf{shrink}(b)|) = O((\log\log n)^2)$ triangles in the triangulation of $\mathsf{shrink}(b)$. Therefore, we can expand the leaf node storing t' into a linear decision subtree as in the previous paragraph.

Let D' be the linear decision tree obtained by expanding $D^{\sigma}_{\min}$ as described above. Let q be a query point. If q can be located successfully in S by D_j, the search in D' traverses a root-to-leaf path in $D^{\sigma}_{\min}$ and then another path of length $O(\log\log\log n)$ to a leaf of D'. Let $D^{\sigma}_{\min}(\alpha)$ and $D'(\alpha)$ denote the running times of $D^{\sigma}_{\min}$ and D' on an online query sequence α, respectively. Then, for any online query sequence α_j such that query points in α_j can be located successfully in S by D_j,

$$D'(\alpha_j) = O(D^{\sigma}_{\min}(\alpha_j) + |\alpha_j|\log\log\log n). \tag{2}$$

For every pair of online query sequences $\alpha_j \subseteq \alpha$ such that α_j is the maximum subsequence of α that can be located successfully in S by D_j, by (1), $D_j(\alpha) = O(D_j(\alpha_j) + |\alpha\backslash\alpha_j|\log|\Delta_j|)$. By Theorem 1, D_j performs no worse than D' on α_j. Let D^{pre}_j denote the $O(|\Delta_j|\log|\Delta_j|)$ preprocessing time to construct both Δ_j and D_j. Therefore,

$$\begin{aligned}
&D_j(\alpha) + D^{\mathrm{pre}}_j \\
&= O\big(D_j(\alpha_j) + |\alpha\backslash\alpha_j|\log|\Delta_j|\big) + O\big(|\Delta_j|\log|\Delta_j|\big) \\
&= O\big(D'(\alpha_j) + |\Delta_j| + |\alpha\backslash\alpha_j|\log|\Delta_j|\big) + O\big(|\Delta_j|\log|\Delta_j|\big) && (\because \text{Theorem 1}) \\
&= O\big(D^{\sigma}_{\min}(\alpha_j) + |\Delta_j|\log|\Delta_j| + |\alpha_j|\log\log\log n + |\alpha\backslash\alpha_j|\log|\Delta_j|\big). && (\because (2))
\end{aligned}$$

The next result summarizes the discussion above.

▶ **Lemma 3.**

(i) *For every pair of online query sequences $\alpha_j \subseteq \alpha$ such that α_j is the maximum subsequence of α that can be located successfully in S by D_j, $D_j(\alpha) + D^{\mathrm{pre}}_j = O\big(D^{\sigma}_{\min}(\alpha_j) + |\Delta_j|\log|\Delta_j| + |\alpha\backslash\alpha_j|\log\log n + |\alpha_j|\log\log\log n\big)$.*

(ii) *$D^{\sigma}_{\min}(\sigma) = O(\mathrm{OPT})$, where OPT is the minimum time needed by any point location linear decision tree for S to process σ.*

We are ready to analyze the performance of the first solution.

▶ **Lemma 4.** *Let S be a planar convex subdivision with n vertices. There is a point-line comparison based data structure that processes any online sequence σ of point location queries in S in $O(|\sigma|\log\log\log n + n + \mathrm{OPT})$ time. The time bound includes the preprocessing time.*

Proof. Let σ_S denote the subsequence of queries in σ that are answered by D_S. For each $j \geq 1$, we use σ_j to denote the subsequence of σ that are located successfully in S by D_j. Therefore, $\bigcup_{j\geq 1}\sigma_j = \sigma\backslash\sigma_S$. Note that $\sigma_j \cap \sigma_k = \emptyset$ if $j \neq k$. Let Γ denote the total processing time required by all D_j's, including the preprocessing time D^{pre}_j. Note that Γ also includes the time spent on unsuccessfully locating some query points in σ_S by the D_j's. Lemma 3(i) implies that

$$\Gamma = O\Big(\sum_j D^{\sigma}_{\min}(\sigma_j) + \sum_j |\Delta_j|\log|\Delta_j| + |\sigma_S|\log\log n + |\sigma\backslash\sigma_S|\log\log\log n\Big).$$

Each Δ_j has $O(f(n)/\log n)$ size and Δ_j is constructed after answering $f(n)$ new queries using D_S. So $|\Delta_j|\log|\Delta_j|$ can be charged to these queries, i.e., $\sum_j |\Delta_j|\log|\Delta_j| = O(|\sigma|)$. Therefore,

$$\Gamma = O\Big(D^{\sigma}_{\min}(\sigma\backslash\sigma_S) + |\sigma| + |\sigma_S|\log\log n + |\sigma\backslash\sigma_S|\log\log\log n\Big).$$

Let Γ_0 denote the total processing time required by D_S on σ_S, including the $O(n)$ preprocessing time to construct Δ_S and D_S. By Theorem 2, $\Gamma_0 = O(D^{\sigma}_{\min}(\sigma_S)+|\Delta_S|+|\sigma_S|\log\log n)$. Therefore,

$$\Gamma_0 + \Gamma = O\Big(D^{\sigma}_{\min}(\sigma) + n + |\sigma| + |\sigma_S|\log\log n + |\sigma\backslash\sigma_S|\log\log\log n\Big). \tag{3}$$

We will show that the term $|\sigma_S|\log\log n$ can be absorbed by other terms in (3). For query points in σ_S that end in leaf nodes of $D^{\sigma}_{\min}$ at depth greater than $\log_2\log_2 n$, their contribution to $|\sigma_S|\log\log n$ can be absorbed by $D^{\sigma}_{\min}(\sigma_S)$. We bound the number of remaining query points in σ_S in Claim 5 below.

▶ **Claim 5.** *Let $\hat{\sigma}_S$ be the subsequence of σ_S such that each query point in $\hat{\sigma}_S$ lies in some triangle (leaf node) in $D^{\sigma}_{\min}$ at depth $\log_2\log_2 n$ or less. Then, $|\hat{\sigma}_S| = O(\log^9 n + |\sigma_S|/\log n)$.*

Proof. We will make use of the following facts:

Fact 1: At most $2^{1+\log_2\log_2 n} - 1 = 2\log_2 n - 1$ nodes in $D^{\sigma}_{\min}$ have depth at most $\log_2\log_2 n$ because $D^{\sigma}_{\min}$ is a binary tree.

Fact 2: For each triangle $t \in \Delta_S$, if the current access frequency of t is at least $|\sigma_S|(\log_2 n)^2/f(n)$, then t must be included in the set X for the next construction of Δ_j and D_j. The reason is that the sum of frequencies in Δ_S is at most $|\sigma_S|$, so there are at most $f(n)(\log_2 n)^{-2}$ triangles in Δ_S with frequencies at least $|\sigma_S|(\log_2 n)^2/f(n)$, implying that t is one of the top $f(n)(\log_2 n)^{-2}$ frequently accessed triangles.

Let Z be the subset of triangles in Δ_S that overlap with some triangle (leaf node) in $D^{\sigma}_{\min}$ at depth $\log_2\log_2 n$ or less. By Fact 1, there are at most $2\log_2 n - 1$ triangles (leaf nodes) in $D^{\sigma}_{\min}$ at depth $\log_2\log_2 n$ or less. Each such triangle must lie inside a region (bounded or exterior) of S in order that $D^{\sigma}_{\min}$ answers a point location query correctly. So each such triangle intersects $O(\log^2 n)$ triangles in Δ_S. It follows that

$$|Z| = O(\log^3 n). \tag{4}$$

Consider a triangle $t \in \Delta_S$ that contains a query point in $\hat{\sigma}_S$. Thus, $t \in Z$ because t must overlap with some triangle (leaf node) in $D^{\sigma}_{\min}$ at depth $\log_2\log_2 n$ or less. If the frequency of t in Δ_S never reaches $|\sigma_S|(\log_2 n)^2/f(n)$, then at most $|\sigma_S|(\log_2 n)^2/f(n)$ query points in t are from $\hat{\sigma}_S$. Suppose that the frequency of t in Δ_S reaches $|\sigma_S|(\log_2 n)^2/f(n)$, say after the construction of Δ_j and before the construction of Δ_{j+1}. At most $f(n)$ queries can be answered by D_S during this period. It means that the frequency of t in Δ_S is at most $f(n) + |\sigma_S|(\log_2 n)^2/f(n)$ before the construction of Δ_{j+1}. By Fact 2, t will be included in Δ_k for all $k > j$. Every query point that falls in t after the construction of Δ_{j+1} will be located successfully in S by D_k for some $k \geq j+1$. Thus, the frequency of t in Δ_S will not be increased further and at most $f(n) + |\sigma_S|(\log_2 n)^2/f(n)$ query points in t are from $\hat{\sigma}_S$. Hence, $|\hat{\sigma}_S| \leq (f(n) + |\sigma_S|(\log_2 n)^2/f(n)) \cdot |Z|$. Recall that $f(n) = (\log_2 n)^6$. Since $|Z| = O(\log^3 n)$ by (4), we obtain $|\hat{\sigma}_S| = O(\log^9 n + |\sigma_S|/\log n)$. ◀

If $|\sigma_S \backslash \hat{\sigma}_S| < |\hat{\sigma}_S|$, we obtain the following from (3):

$$\begin{aligned}
\Gamma_0 + \Gamma &= O\big(D^{\sigma}_{\min}(\sigma) + n + |\sigma| + |\sigma_S \backslash \hat{\sigma}_S| \log\log n + |\hat{\sigma}_S| \log\log n + \\
&\qquad |\sigma \backslash \sigma_S| \log\log\log n\big) \\
&= O\big(D^{\sigma}_{\min}(\sigma) + n + |\sigma| + |\hat{\sigma}_S| \log\log n + |\sigma \backslash \sigma_S| \log\log\log n\big) \\
&= O\big(D^{\sigma}_{\min}(\sigma) + n + |\sigma| + |\sigma_S| + \log^9 n \log\log n + \\
&\qquad |\sigma \backslash \sigma_S| \log\log\log n\big) \qquad (\because \text{Claim 5}) \\
&= O\big(D^{\sigma}_{\min}(\sigma) + n + |\sigma| + |\sigma \backslash \sigma_S| \log\log\log n\big).
\end{aligned}$$

If $|\sigma_S \backslash \hat{\sigma}_S| \geq |\hat{\sigma}_S|$, we obtain the following from (3):

$$\begin{aligned}
\Gamma_0 + \Gamma &= O\big(D^{\sigma}_{\min}(\sigma) + n + |\sigma| + |\sigma_S \backslash \hat{\sigma}_S| \log\log n + |\hat{\sigma}_S| \log\log n + \\
&\qquad |\sigma \backslash \sigma_S| \log\log\log n\big) \\
&= O\big(D^{\sigma}_{\min}(\sigma) + n + |\sigma| + |\sigma_S \backslash \hat{\sigma}_S| \log\log n + |\sigma \backslash \sigma_S| \log\log\log n\big) \\
&= O\big(D^{\sigma}_{\min}(\sigma) + n + |\sigma| + |\sigma \backslash \sigma_S| \log\log\log n\big).
\end{aligned}$$

In the last step above, we use the fact that $D^{\sigma}_{\min}(\sigma_S) = \Omega(|\sigma_S \backslash \hat{\sigma}_S| \log\log n)$, which is true because each query point in $\sigma_S \backslash \hat{\sigma}_S$ lies in a triangle (leaf node) in $D^{\sigma}_{\min}$ at depth greater than $\log_2 \log_2 n$.

As a result, no matter whether $|\sigma_S \backslash \hat{\sigma}_S|$ or $|\hat{\sigma}_S|$ is greater than the other, we have $\Gamma_0 + \Gamma = O\left(D^{\sigma}_{\min}(\sigma) + n + |\sigma| \log\log\log n\right) = O\left(\mathrm{OPT} + n + |\sigma| \log\log\log n\right)$ by Lemma 3(ii). ◀

3.2 Optimal solution

We apply the method in Section 3.1 recursively to obtain a multi-level data structure. To facilitate the description of this new strategy, we revise our notation as follows. We relabel each triangulation Δ_j and each point location structure D_j in Section 3.1 as $\Delta_{1,j}$ and $D_{1,j}$. The extra subscript 1 signifies that these are triangulations and structures at the first level. We use $\Delta_{0,1}$ and $D_{0,1}$ to denote Δ_S and D_S, respectively. At any level $i \geq 1$, a new triangulation $\Delta_{i,j}$ and a new point location structure $D_{i,j}$ will be constructed from time to time to replace $\Delta_{i,j-1}$ and $D_{i,j-1}$. At level 0, $\Delta_{0,1}$ and $D_{0,1}$ will never be replaced, and there are no other triangulation and point location structure at level 0. When it is not important to distinguish the current index j at a level, we use $\Delta_{i,*}$ and $D_{i,*}$ to denote the current triangulation and point location structure at level i.

We have multiple levels of triangulations and point location structures at any time: $(\Delta_{0,1}, D_{0,1})$, $(\Delta_{1,*}, D_{1,*}), \ldots, (\Delta_{m,*}, D_{m,*})$, where m is the highest level currently. Let q be the next query point. We first check if q lies inside $\mathcal{B}_S$ in $O(1)$ time. If not, we output the exterior region of S. Suppose that q lies inside $\mathcal{B}_S$. We first query $D_{m,*}$ with q. If we fail to locate a region in S containing q, then we try $D_{m-1,*}$. If that also fails, we try $D_{m-2,*}$ and so on. The location of q will succeed by $D_{0,1}$ the latest.

After locating q, we need to update the access frequencies in the triangulations. This update is important because the frequencies govern how the method in [13] adapts the point location structures to incoming queries. If q lies outside $\mathcal{B}_S$, we do not change any frequency in any triangulation. If q is located in a triangle that stores a region of S at level i, we increment the frequency of the triangle in $\Delta_{i,*}$ that contains q. The frequencies in triangulations at other levels do not change.

The number of levels increases monotonically as we process σ. The triangulation and point location structure at each level are rebuilt from time to time. We use *i-rebuild* to refer to a rebuild at level i. Use n_0 to denote n and define $n_i = f(n_{i-1})/\log_2 n_{i-1}$ for $i \geq 1$.

For any $i \geq 0$, if $f(n_i)$ query points are located successfully in S by $D_{i,*}$ since the last $(i+1)$-rebuild and no i-rebuild has happened during these $f(n_i)$ queries, then we perform a new $(i+1)$-rebuild. That is, if level $i+1$ does not exist, then we construct $\Delta_{i+1,1}$ from $\Delta_{i,*}$ and then $D_{i+1,1}$ for $\Delta_{i+1,1}$; otherwise, if $\Delta_{i+1,k}$ and $D_{i+1,k}$ are currently stored at level $i+1$, then we replace them by constructing $\Delta_{i+1,k+1}$ from $\Delta_{i,*}$ and then $D_{i+1,k+1}$ for $\Delta_{i+1,k+1}$. The construction works as follows.

We extract the subset X of $f(n_i)(\log_2 n_i)^{-2}$ triangles in $\Delta_{i,*}$ that have the highest $f(n_i)(\log_2 n_i)^{-2}$ access frequencies in $\Delta_{i,*}$. Then, we proceed as in Section 3.1.1 to produce $\Delta_{i+1,k+1}$ from X. Details are given below. For each bounded region r of S, we compute $\mathrm{conv}(r \cap X)$ and then triangulate it by calling $\mathsf{TriReg}(\mathrm{conv}(r \cap X))$. This produces $O(|r \cap X|)$ triangles and takes $O(|r \cap X| \log |r \cap X|)$ time. Each resulting triangle stores the region id r. Summing over all bounded regions, we obtain $O(|X|) = O(f(n_i)(\log n_i)^{-2}) = o(n_{i+1})$ triangles in $O(|X| \log |X|) = O(n_{i+1} \log n_{i+1})$ time. Consider the processing of triangles in X outside S. In a 1-rebuild as described in Section 3.1.1, for each boomerang b between the boundaries of $\mathcal{B}_S$ and S, we compute another boomerang $\mathsf{shrink}(b) \subseteq b$. Suppose that there is a 2-rebuild before the next 1-rebuild. Note that any triangle outside S that is selected in this 2-rebuild must be contained in $\mathsf{shrink}(b')$ for some boomerang b' between the boundaries of $\mathcal{B}_S$ and S. Assume that some triangles lying in $\mathsf{shrink}(b)$ are selected. Note that these triangles belong to the triangulation of $\mathsf{shrink}(b)$ produced by $\mathsf{SplitBR}$ and TriBR. Since $\mathsf{shrink}(b)$ is a boomerang, there is also a hierarchy on the triangular regions produced by $\mathsf{SplitBR}$ as in Figure 4. As in Section 3.1.1, we first mark the triangular regions in the hierarchy that store the selected triangles and then mark their ancestors in the hierarchy. The union of the marked triangular regions is another boomerang $\mathsf{shrink}(\mathsf{shrink}(b)) \subseteq \mathsf{shrink}(b)$ and it is triangulated by calling $\mathsf{SplitBR}$ and TriBR. Each resulting triangle stores the id of the exterior region of S. Each triangle also stores the id of the triangular region containing it, which is produced by the call $\mathsf{SplitBR}(\mathsf{shrink}(\mathsf{shrink}(b)))$. Note that $\mathsf{shrink}(\mathsf{shrink}(b))$ has $O(f(n_1)/\log n_1)$ size and its processing takes $O(f(n_1)/\log n_1)$ time. In general, in an $(i+1)$-rebuild, the processing of triangles in X outside S takes $O(f(n_i)/\log n_i) = O(n_{i+1})$ time and produces at most three boomerangs of $O(n_{i+1})$ size and $O(n_{i+1})$ triangles in these boomerangs.

The triangles computed above may form disconnected components. We apply a plane sweep in $O(n_{i+1} \log n_{i+1})$ time to connect them with triangles. These extra triangles do not store any region in S, so query points that fall into them are not located successfully in S. The resulting triangulation is $\Delta_{i+1,k+1}$. The frequencies of all triangles in $\Delta_{i+1,k+1}$ are initialized to be zero. We apply Theorem 1 to $\Delta_{i+1,k+1}$ to obtain the point location structure $D_{i+1,k+1}$. In summary, the $(i+1)$-rebuild takes $O(n_{i+1} \log n_{i+1})$ time and $|\Delta_{i+1,k+1}| = O(n_{i+1})$.

We do not increase the number of levels anymore when the highest level m reaches the value such that $n_m < 30^5$ for the first time.[2] The triangulation size is $O(n_m) = O(1)$. However, we will still perform i-rebuild for any $i \in [1, m]$. Also, if a query point is located successfully in S at this highest possible level m, we do not change any frequency in $\Delta_{m,*}$. The query time is only $O(1)$ anyway.

▶ Remark. Let m be the highest level currently. When an i-rebuild is performed for some $i < m$, the triangulations at levels $i+1, \ldots, m$ are unaffected. Query answering will still start from level m. The selected triangles on which the construction of $\Delta_{i+1,*}$ was based may not be related to the selected triangles on which the construction of the new $\Delta_{i,*}$ is based.

[2] This particular choice goes well with the proof of Claim 8.

▶ **Theorem 6.** *Let S be a planar convex subdivision with n vertices. There is a point-line comparison based structure that uses $O(n)$ space and processes any online sequence σ of point location queries in S in $O(n + \mathrm{OPT})$ time, where* OPT *is the minimum time required by any point location linear decision tree for S to process σ. The time bound includes the $O(n)$ preprocessing time.*

Proof. For any online query sequence α, we use $D_{i,j}(\alpha)$ to denote the time needed by $D_{i,j}$ to process α and $D^{\mathrm{pre}}_{i,j}$ to denote the preprocessing time to construct both $\Delta_{i,j}$ and $D_{i,j}$. Define the following subsets of query points:

$$\begin{aligned}\sigma_{i,j} &= \{q \in \sigma : q \text{ is located successfully in } S \text{ using } D_{i,j}\ \},\\ \sigma_i &= \{q \in \sigma : q \text{ is located successfully in } S \text{ at level } i\},\\ \sigma_{<i} &= \{q \in \sigma : q \text{ is located successfully in } S \text{ at some level less than } i\}.\end{aligned}$$

By definition, $\sigma = \bigcup_{i,j} \sigma_{i,j}$, the $\sigma_{i,j}$'s are mutually disjoint, $\sigma_i = \bigcup_j \sigma_{i,j}$, and $\sigma_{<i} = \bigcup_{a=0}^{i-1} \sigma_a$. Claim 7 below is analogous to Lemma 3(i) and it can be proved by the same argument.

▶ **Claim 7.** *For all $i \in [1, m]$ and all online query sequences $\alpha_{i,j} \subseteq \alpha$ such that $\alpha_{i,j}$ is the maximum subsequence of α that can be successfully located in S by $D_{i,j}$, $D_{i,j}(\alpha) + D^{\mathrm{pre}}_{i,j} = O(D^{\sigma}_{\min}(\alpha_{i,j}) + |\Delta_{i,j}| \log |\Delta_{i,j}| + |\alpha \backslash \alpha_{i,j}| \log n_i + |\alpha_{i,j}| \log\log n_i)$.*

Let Γ_i denote the total processing time required by $D_{i,j}$ over all j, including the preprocessing time $D^{\mathrm{pre}}_{i,j}$. Recall that $D^{\mathrm{pre}}_{i,j} = O(|\Delta_{i,j}| \log |\Delta_{i,j}|) = O(n_i \log n_i)$ for $i > 0$ and $D^{\mathrm{pre}}_{0,1} = O(|\Delta_{0,1}|) = O(n_0)$. Note that for $i > 0$, Γ_i includes the time spent on unsuccessfully locating some query points in $\sigma_{<i}$. By Claim 7, for $i \in [1, m]$,

$$\begin{aligned}\Gamma_i &= O\Big(\sum_j D^{\sigma}_{\min}(\sigma_{i,j}) + \sum_j n_i \log n_i + |\sigma_{<i}| \log n_i + \sum_j |\sigma_{i,j}| \log\log n_i\Big)\\ &= O\Big(D^{\sigma}_{\min}(\sigma_i) + \sum_j n_i \log n_i + |\sigma_{<i}| \log n_i + |\sigma_i| \log\log n_i\Big).\end{aligned}$$

By Theorem 2, $\Gamma_0 = D_{0,1}(\sigma_0) + O\big(|\Delta_{0,1}|\big) = O\big(D^{\sigma}_{\min}(\sigma_0) + n_0 + |\sigma_0| \log\log n_0\big)$. Therefore,

$$\sum_{i=0}^{m} \Gamma_i = O\Big(\sum_{i=0}^{m} D^{\sigma}_{\min}(\sigma_i) + n_0 + \sum_{i=1}^{m}\sum_j n_i \log n_i + \sum_{i=0}^{m} |\sigma_i| \log\log n_i + \sum_{i=1}^{m} |\sigma_{<i}| \log n_i\Big).$$

Since $\Delta_{i,j}$ is constructed after answering $f(n_{i-1})$ new queries using $D_{i-1,*}$, the preprocessing time of $O(n_i \log n_i) = o(f(n_{i-1}))$ for constructing $\Delta_{i,j}$ and $D_{i,j}$ can be charged to these new queries. So $\sum_{i=1}^{m}\sum_j n_i \log n_i$ can be charged to the queries in σ, i.e., $\sum_{i=1}^{m}\sum_j n_i \log n_i = O(|\sigma|)$. We rewrite the term $\sum_{i=1}^{m} |\sigma_{<i}| \log n_i = \sum_{i=0}^{m-1}(|\sigma_i| \sum_{l=i+1}^{m} \log n_l)$.

▶ **Claim 8.** *For all $i \in [0, m-1]$, $\sum_{l=i+1}^{m} \log_2 n_l < 35 \log_2\log_2 n_i$.*

Consequently,

$$\begin{aligned}\sum_{i=0}^{m} \Gamma_i &= O\Big(\sum_{i=0}^{m} D^{\sigma}_{\min}(\sigma_i) + n + |\sigma| + \sum_{i=0}^{m} |\sigma_i| \log\log n_i + \sum_{i=0}^{m-1} |\sigma_i| \log\log n_i\Big)\\ &= O\Big(D^{\sigma}_{\min}(\sigma) + n + |\sigma| + \sum_{i=0}^{m} |\sigma_i| \log\log n_i\Big). \qquad (5)\end{aligned}$$

Define the following quantities:

$$\hat{\sigma}_{i,j} = \big\{q \in \sigma_{i,j} : q \text{ lies in some leaf node of } D^{\sigma}_{\min} \text{ at depth } \log_2 \log_2 n_i \text{ or less}\big\},$$
$$\hat{\sigma}_i = \bigcup_j \hat{\sigma}_{i,j}$$

Claim 9 below is analogous to Claim 5 in the proof of Lemma 4. It also has a similar proof.

▶ **Claim 9.** $|\hat{\sigma}_{i,j}| = O(\log^9 n_i + |\sigma_{i,j}|/\log n_i)$.

By Claim 9,

$$|\hat{\sigma}_i| = \sum_j |\hat{\sigma}_{i,j}| = O\Big(\sum_j \log^9 n_i + |\sigma_i|/\log n_i\Big). \tag{6}$$

If $|\sigma_i \backslash \hat{\sigma}_i| \geq |\hat{\sigma}_i|$, then

$$\begin{aligned} |\sigma_i| \log\log n_i &= |\sigma_i \backslash \hat{\sigma}_i| \log\log n_i + |\hat{\sigma}_i| \log\log n_i = O(|\sigma_i \backslash \hat{\sigma}_i| \log\log n_i) \\ &= O(D^{\sigma}_{\min}(\sigma_i)). \end{aligned}$$

In the last step above, we use the fact that $D^{\sigma}_{\min}(\sigma_i) = \Omega(|\sigma_i \backslash \hat{\sigma}_i| \log\log n_i)$, which is true because each query point in $\sigma_i \backslash \hat{\sigma}_i$ lies in a triangle (leaf node) in $D^{\sigma}_{\min}$ at depth greater than $\log_2 \log_2 n_i$. If $|\sigma_i \backslash \hat{\sigma}_i| < |\hat{\sigma}_i|$, then

$$\begin{aligned} |\sigma_i| \log\log n_i &= |\sigma_i \backslash \hat{\sigma}_i| \log\log n_i + |\hat{\sigma}_i| \log\log n_i = O(|\hat{\sigma}_i| \log\log n_i) \\ &= O\Big(\sum_j \log^9 n_i \log\log n_i + |\sigma_i|\Big). && (\because (6)) \\ &= O\Big(\sum_j n_i + |\sigma_i|\Big). \end{aligned}$$

Combining the two cases above and the fact that $D^{\sigma}_{\min}(\sigma_i) = \Omega(|\sigma_i|)$, we obtain $|\sigma_i| \log\log n_i = O\big(D^{\sigma}_{\min}(\sigma_i) + \sum_j n_i\big)$. Substituting this equation into (5) gives

$$\sum_{i=0}^{m} \Gamma_i = O\Big(D^{\sigma}_{\min}(\sigma) + n + |\sigma| + \sum_{i=0}^{m} D^{\sigma}_{\min}(\sigma_i) + \sum_{i=0}^{m} \sum_j n_i\Big).$$

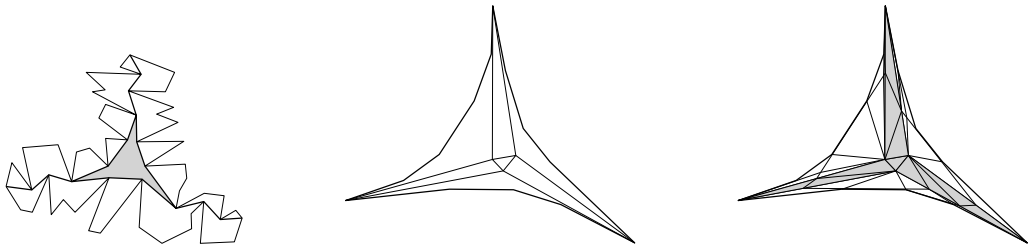
We have shown previously that $\sum_{i=1}^{m} \sum_j n_i \log n_i = O(|\sigma|)$. Note that $\sum_j n_0 = n_0 = n$ as there are only one triangulation and one structure at level 0. Also, $\sum_{i=0}^{m} D^{\sigma}_{\min}(\sigma_i) = D^{\sigma}_{\min}(\sigma)$ and $D^{\sigma}_{\min}(\sigma) = \Omega(|\sigma|)$. Therefore, by Lemma 3(ii), $\sum_{i=0}^{m} \Gamma_i = O\big(D^{\sigma}_{\min}(\sigma)+n\big) = O(\mathrm{OPT}+n)$.

To bound the size of our data structure, observe that $\Delta_{i,j}$ and $D_{i,j}$ have $O(n_i)$ size and $(\Delta_{i,j}, D_{i,j})$ replace $(\Delta_{i,j-1}, D_{i,j-1})$. Therefore, the total size is $O(\sum_{i=0}^{m} n_i)$. Since $n_i = O(f(n_{i-1})/\log n_{i-1}) = O(\log^5 n_{i-1})$, it is clear that $n_i = O(n/2^{i-1})$ by an inductive argument. As a result, the total size is $O(\sum_{i=0}^{m} n_i) = O(\sum_{i=0}^{m} n/2^{i-1}) = O(n)$. ◀

4 Planar connected subdivision

We describe a point location structure for a connected subdivision S with n vertices. It uses $O(n)$ space and processes any online query sequence σ in $O(|\sigma| \log\log n + n + \mathrm{OPT})$ time.

We need a *balanced geodesic triangulation* of a simple polygon P [7]. Let k be the number of vertices of P. Pick three vertices $v_1, v_{k/3}, v_{2k/3}$ of P (which divide the boundary of P into chains of roughly equal sizes). The three geodesic paths among $v_1, v_{k/3}, v_{2k/3}$ bound the



(a) (b) (c)

Figure 5 (a) Kite and the geodesic triangle inside (shown shaded). (b) Divide into triangles and boomerangs. (c) Triangulation.

so-called *kite*. Refer to Figure 5(a) for an example. The part of the kite with an non-empty interior is a *geodesic triangle* τ, whose boundary consists of three reflex chains. Next, compute the geodesic paths from v_1 and $v_{k/3}$ to $v_{k/6}$, the middle vertex in the chain between v_1 and $v_{k/3}$. This creates another kite joining v_1, $v_{k/6}$ and $v_{k/3}$ and hence another geodesic triangle τ' inside this kite. The same process is repeated to other parts of P recursively. In the end, we obtain a balanced geodesic triangulation, which can be computed in $O(|P|)$ time [7].

We simulate the decomposition as sketched in Section 2 using balanced geodesic triangulations. Let $\text{conv}(S)$ denote the convex hull of S. We divide the exterior face of $\text{conv}(S)$ into triangles as described in Section 2. Each region r inside $\text{conv}(S)$ is a simple polygon. We first compute a balanced geodesic triangulation $\tilde{\Delta}_r$ of r. We triangulate each geodesic triangle τ in $\tilde{\Delta}_r$ as follows. We shoot two rays inward from each vertex of τ. They intercept each other and form four triangles. Figure 5(b) shows an example. These four triangles are surrounded by three boomerangs. The boomerangs are triangulated as described in Section 3.1, and this process places $O(\log n)$ vertices on the boundaries of three of the triangles in the middle. We connect these vertices to triangulate these three triangles. Figure 5(c) shows an example. This complete the triangulation of τ. The triangulations of all geodesic triangles in $\tilde{\Delta}_r$ form the triangulation Δ_r. Any triangle that lies in r intersects $O(\log^2 n)$ triangles in τ, implying that any triangle that lies in r intersects $O(\log^3 n)$ triangles in Δ_r. The collection of all triangles obtained above form the triangulation Δ_S. It takes $O(n)$ time to compute Δ_S. We apply Theorem 1 to Δ_S to obtain a point location structure D_S.

Define $D^\sigma_{\min}$ for S as in Section 3.1.3. A leaf of $D^\sigma_{\min}$ corresponds to a triangle t that lies in a region r of S, so t intersects $O(\log^3 n)$ in Δ_S. It means that we can expand the leaf nodes of $D^\sigma_{\min}$ into linear decision subtrees of height $O(\log\log n)$ so that the expanded linear decision tree D' takes $D^\sigma_{\min}(\sigma) + O(|\sigma|\log\log n)$ time to locate the query points in σ in Δ_S. The total processing time by D_S (including the preprocessing time) is $D_S(\sigma) + O(n)$, which by Theorem 1 is $O(D'(\sigma) + n) = O(D^\sigma_{\min}(\sigma) + n + |\sigma|\log\log n) = O(\text{OPT} + n + |\sigma|\log\log n)$.

▶ **Theorem 10.** *Let S be a planar connected subdivision with n vertices. There is a point-line comparison based data structure that uses $O(n)$ space and processes any online sequence σ of point location queries in S in $O(|\sigma|\log\log n + n + \text{OPT})$ time, where* OPT *is the minimum time needed by any point location linear decision tree for S to process σ. The time bound includes the $O(n)$ preprocessing time.*

5 Conclusion

The performance of our data structure is asymptotically optimal when compared with static point location linear decision trees. It is an open problem to obtain optimal performance when

compared with linear decision trees that may reorganize themselves. This open problem may be difficult as it is related to the dynamic optimality conjecture by Sleator and Tarjan [17], which conjectures that the performance of a splay tree is no more than $O(n)$ plus a constant times the time required by any binary search tree algorithm. The dynamic optimality conjecture is still open after over thirty years.

References

1 P. Afshani, J. Barbay, and T. Chan. Instance optimal geometric algorithms. *Proc. 50th Annu. IEEE Symp. Found. Computer Sci.*, 2009, 129–138.

2 U. Adamy and R. Seidel. On the exact worst case query complexity of planar point location. *Proc. 9th Annu. ACM-SIAM Symp. Discrete Alg.*, 1998, 609–618.

3 S. Arya, T. Malamatos, and D. M. Mount. A simple entropy-based algorithm for planar point location. *ACM Trans. Alg.*, vol. 3, no. 2, 2007, article 17.

4 S. Arya, T. Malamatos, D. Mount, and K. Wong. Optimal expected-case planar point location. *SIAM J. Comput.*, 37 (2007), 584–610.

5 P. Bose, L. Devroye, K. Douïeb, V. Dujmovic, J. King, and P. Morin. Odds-On Trees, arXiv:1002.1092v1 [cs.CG], 5 February 2010.

6 B. Chazelle. Triangulating a simple polygon in linear time. *Discrete and Computational Geometry*, 6 (1991), 485–524.

7 B. Chazelle, H. Edelsbrunner, M. Grigni, L. Guibas, J. Hershberger, M. Sharir, and J. Snoeyink. Ray shooting in polygons using geodesic triangulations. *Algorithmica*, 12 (1994), 54–68.

8 S.-W. Cheng and M.-K. Lau. Adaptive Point Location in Planar Convex Subdivisions. Preliminary version appeared in *Proc. 26th Int'l Symp. Algorithms and Computation*, 2015, 14-22. The full version is to appear in *Int'l J. Comput. Geom. Theory and Appl.*

9 S. Collette, V. Dujmović, J. Iacono, S. Langerman, and P. Morin. Entropy, triangulation, and point location in planar subdivisions. *ACM Trans. Alg.*, vol. 8, no. 3, 2012, article 29.

10 D. P. Dobkin and D. G. Kirkpatrick. Determining the separation of preprocessed polyhedra – a unified approach, *Proc. 17th Int'l Colloq. Automata, Languages and Programming*, 1990, 400–413.

11 H. Edelsbrunner, L. J Guibas, and J. Stolfi. Optimal point location in a monotone subdivision. *SIAM J. Comput.*, 15 (1986), 317–340.

12 J. Iacono. Expected asymptotically optimal planar point location. *Comput. Geom.: Theory and Appl.*, 29 (2004), 19–22.

13 J. Iacono and W. Mulzer. A static optimality transformation with applications to planar point location. *Int'l J. Comput. Geom. Appl.*, 22 (2012), 3270–340.

14 D. G. Kirkpatrick. Optimal search in planar subdivisions. *SIAM J. Comput.*, 12 (1983), 28–35.

15 N. Sarnak and R. E. Tarjan. Planar point location using persistent search trees. *Comm. ACM*, 29 (1986), 669–679.

16 C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 1948.

17 D. D. Sleator and R. E. Tarjan. Self-adjusting binary search trees, *J. ACM*, 32 (1985), 652–686.

High Dimensional Consistent Digital Segments

Man-Kwun Chiu[1] and Matias Korman*[2]

1 **National Institute of Informatics (NII), Tokyo, Japan; and JST, ERATO, Kawarabayashi Large Graph Project, Japan**
chiumk@nii.ac.jp

2 **Tohoku University, Sendai, Japan**
mati@dais.is.tohoku.ac.jp

Abstract

We consider the problem of digitalizing Euclidean line segments from $\mathbb{R}^d$ to $\mathbb{Z}^d$. Christ *et al.* (DCG, 2012) showed how to construct a set of *consistent digital segments* (CDS) for $d = 2$: a collection of segments connecting any two points in $\mathbb{Z}^2$ that satisfies the natural extension of the Euclidean axioms to $\mathbb{Z}^d$. In this paper we study the construction of CDSs in higher dimensions.

We show that any total order can be used to create a set of *consistent digital rays* CDR in $\mathbb{Z}^d$ (a set of rays emanating from a fixed point p that satisfies the extension of the Euclidean axioms). We fully characterize for which total orders the construction holds and study their Hausdorff distance, which in particular positively answers the question posed by Christ *et al.*.

1998 ACM Subject Classification I.3.5 Computational Geometry and Object Modeling, I.4.1 Digitization and Image Capture

Keywords and phrases Consistent Digital Line Segments, Digital Geometry, Computer Vision

Digital Object Identifier 10.4230/LIPIcs.SoCG.2017.31

1 Introduction

Computation in Ancient Greece was rigorously done with ruler and compass using the five axioms of Euclidean geometry. The study of these axioms has had a drastic influence in the development of mathematics. Indeed, the removal of one of them (the fifth one) created non-Euclidean geometries, which have had huge influence on science and technology.

Computers and digital data have nowadays replaced the ruler and compass methods of computation. In order to have a rigorous system of geometric computation in the digital world, it is desirable to establish a set of axioms similar to those of the Euclidean geometry, where we need to replace a line by a Manhattan path in the micro scale that in a macro scale can be seen as a straight line.

There have been several attempts to define digital segments in a two dimensional $n \times n$ grid. The two dimensional bounded space is the most popular case to consider given its many applications in computer vision and computer graphics. Solutions have been proposed from a robust finite-precision geometric computation point of view [6, 8], snap rounding [5], and many more.

A pioneering work by Michael Luby in 1987 [7] introduced an axiomatic approach of the set of digital rays emanating from the origin. He showed that lines should curve by $\Theta(\log n)$ to satisfy a set of axioms analogous to Euclid's axioms (the lower bound proof was given

* M. K. was supported in part by the ELC project (MEXT KAKENHI No. 12H00855, 15H02665, and 24106007).

by Håstad). The theory was recently re-discovered by Chun *et al.* [4] and Christ *et al.* [3]. Using these results we can define a geometry that satisfies Euclid-like axioms in the two dimensional grid, and only a small bend of the lines will be needed (i.e., $\Theta(\log n)$ in an $n \times n$ grid, a formal definition is given below).

Chun *et al.* and Christ *et al.* proposed a d-dimensional version of the set of axioms, but unfortunately it is not constructive. That is, they left open how to find a system to generate a complete set of digital segments in d-dimensional space that resembles the Euclidean segments. In this paper we provide the first significant step towards answering the question for high dimensions. For the purpose we extend the constructive algorithm of Christ *et al.* [3] to spaces of arbitrary dimension and study how much of a bend it creates.

2 Preliminaries

Let $x_1, x_2, \ldots, x_d$ denote the coordinate axes in $\mathbb{Z}^d$, and p_i denote the i-th coordinate of a point $p \in \mathbb{Z}^d$ (for simplicity, from now on all indices are in the set $\{1, \ldots, d\}$). Our aim is to construct a digital path for any two points $p, q \in \mathbb{Z}^d$ (we denote such a path by $R(p, q)$). Ideally, we want R to be constructive and defined in the whole domain, but sometimes we will consider subsets of $\mathbb{Z}^d \times \mathbb{Z}^d$ instead.

▶ **Definition 1.** For any $S \subseteq \mathbb{Z}^d \times \mathbb{Z}^d$, let $DS(S)$ be a set of digital segments such that $R(p, q) \in DS(S)$ for all $(p, q) \in S$. We say that $DS(S)$ forms a *partial set of consistent digital segments* on S (partial CDS for short) if for every pair $(p, q) \in S$ it satisfies the following five axioms:

(S1) Grid path property: $R(p, q)$ is a path between p and q under the $2d$-neighbor topology[1].

(S2) Symmetry property: $R(p, q) = R(q, p)$.

(S3) Subsegment property: For any $r \in R(p, q)$, we have $R(p, r) \in DS(S)$ and $R(p, r) \subseteq R(p, q)$.

(S4) Prolongation property: There exists $r \in \mathbb{Z}^d$, such that $R(p, r) \in DS(S)$ and $R(p, q) \subset R(p, r)$.

(S5) Monotonicity property: For all $i \leq d$ such that $p_i = q_i$, it holds that every point $r \in R(p, q)$ satisfies $r_i = p_i = q_i$.

These axioms give nice properties of digital line segments analogous to Euclidean line segments. For example, (S1) and (S3) imply that the non-empty intersection of two digital line segments is connected under the $2d$-neighbor topology. In particular, the intersection between two digital segments is a digital line segment that could degenerate to a single point or even to an empty set. (S5) implies that the intersection with any axis-aligned halfspace is connected, and so on.

Ideally, we want the set S to be as large as possible. A subset of $\mathbb{Z}^d \times \mathbb{Z}^d$ that is often used for constructions is $S = \{p\} \times \mathbb{Z}^d$ (for some $p \in \mathbb{Z}^d$). We say that a partial CDS on such a set is a *consistent digital ray* system (CDR for short). Note that this means that we have a method to connect a fixed point p to any other point of $\mathbb{Z}^d$. A partial CDS for $S = \mathbb{Z}^d \times \mathbb{Z}^d$ is called a set of *consistent digital segments* (CDS for short). Our aim is to create a CDS in $\mathbb{Z}^d$, since it is a constructive way to connect any pairs of points.

It is not straightforward how to create a CDS, even when $d = 2$. For example, the simple rounding scheme of Euclidean segments to the digital world that is often used in

[1] The $2d$-neighbor topology is the natural one that connects to your predecessor and successor in each dimension. Formally speaking, two points are connected if and only if their L_1 distance is exactly one.

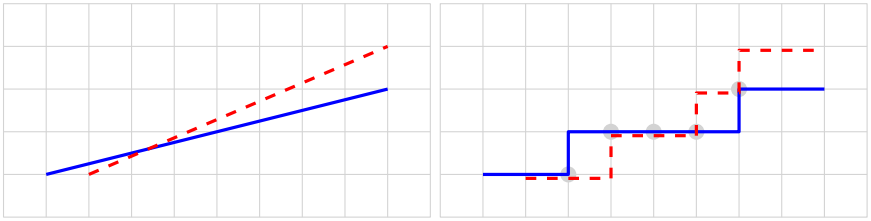


Figure 1 Two different Euclidean line segments and their corresponding digital line segments via a rounding scheme. Note that their intersection in $\mathbb{Z}^2$ (highlighted with grey disks) is not connected under the 4-neighbor topology, which implies that the rounding scheme is not consistent.

computer graphics, does not generate a CDS (since axioms are not always preserved, see Figure 1). Another alternative is to use the bounding box approach that makes all moves in one dimension before moving in another one. Although this set of segments is consistent, it will be visually very different from the Euclidean line segments. Thus, the objective is to create a CDS that resembles the Euclidean segments.

The straightness or resemblance between the digital line segment $R(p, q)$ and the Euclidean segment $\overline{pq}$ is often measured using the *Hausdorff* distance. The Hausdorff distance $H(A, B)$ of two objects A and B is defined by $H(A, B) = \max\{h(A, B), h(B, A)\}$, where $h(A, B) = \max_{a\in A} \min_{b\in B} \delta(a, b)$, and $\delta(a, b)$ is the natural Euclidean distance, given by $||\cdot||_2$ norm.

▶ **Definition 2.** Let $DS(S)$ be a partial CDS. We say that $DS(S)$ has Hausdorff distance $f(n)$ if for all $p, q \in S$ such that $||p - q||_1 \leq n$, it holds that $H(\overline{pq}, R(p, q)) = O(f(n))$.

Constructions with smaller Hausdorff distance resemble more the Euclidean segments and thus, are more desirable. Hence, the big open problem in the field is what is the (asymptotically speaking) smallest $f(n)$ function so that we can have a CDS in $\mathbb{Z}^d$? Or equivalently: what is the asymptotic behavior of the Hausdorff distance of the CDS that best approximates the Euclidean segments?[2]

2.1 Previous work

Although the concept of consistent digital segments was first studied by Luby [7], it received renewed interest by the community when it was rediscovered by Chun *et al.* [4]. The latter showed how to construct a set of consistent digital rays (CDR) in any dimension. The construction satisfies all axioms, including the Hausdorff distance bound:

▶ **Theorem 3** (Theorem 4.4 of [4], rephrased)**.** *For any $d \geq 2$ and $p \in \mathbb{Z}^d$ we can construct a CDR with $O(\log n)$ Hausdorff distance.*

Håstad[3] and Chun *et al.* [4] showed that any CDR in two dimensions must have $\Omega(\log n)$ Hausdorff distance. Thus $\log n$ is the smallest possible distance one can hope to achieve. This result was generalized by Christ *et al.* [3], who shows a correspondence between CDRs in $\mathbb{Z}^2$ and total orders on the integers (details on this correspondence is given in Section 3).

[2] Note that in the original definition of Christ *et al.* [3], the requirement is for points $p, q \in S$ such that $||p - q||_2 \leq n$. The focus of interest is the two dimensional case, and for any fixed dimension both metrics are equivalent (since $\frac{\sqrt{d}}{d}||p - q||_1 \leq ||p - q||_2 \leq ||p - q||_1$. However, since we are interested in bounds that depend in the dimension d, it is more accurate to measure the distance between p and q with the L_1 metric.

[3] The lower bound was published by Luby, but credit given to Håstad (see Theorem 19 of [7]).

In particular, this correspondence can be used to create a CDS in $\mathbb{Z}^2$ that has $O(\log n)$ Hausdorff distance. Note that the $\Omega(\log n)$ lower bound also holds for CDS, so this result is asymptotically tight.

This answers the question of how well can CDSs approximate Euclidean segments in the two dimensional case. However, the question for higher dimensions remains largely open. Although the method of Christ *et al.* [3] cannot be used to construct CDSs or CDRs in higher dimensions, they show that it can create partial CDSs.

▶ **Theorem 4** (Theorem 16 of [3], rephrased). *Let $S = \{(x, y) \colon x_i \geq y_i\} \subset \mathbb{Z}^d \times \mathbb{Z}^d$. We can construct arbitrarily many partial CDSs on S.*

Note that S contains segments of positive slope (that is, only for the pairs (p, q) such that q is in the first orthant of p), hence it is roughly a small fraction (roughly $1/2^{d-1}$) of all possible segments. Other than Theorems 3 and 4, little or nothing is known for three or higher dimensions. Up to date, the only CDS known in three or higher dimensions is the naive bounding box approach (described in Section 3) that has $\Omega(n)$ Hausdorff distance. In particular, it still remains open whether one can create a CDS in $\mathbb{Z}^d$ with $o(n)$ Hausdorff distance (for $d > 2$).

Other research in the topic has focused in the characterization of CDSs in two dimensions. Chowdhury and Gibson [1] gave necessary and sufficient conditions so that the union of CDRs forms a CDS. This characterization heavily uses the correspondence between CDRs and total orders, and thus it was stated in terms of total orders. In a companion paper, the same authors [2] afterwards provided an alternative characterization together with a constructive algorithm. Specifically, they gave an algorithm that, given a collection of segments in an $n \times n$ grid that satisfies the five axioms, computes a CDS that contains those segments. The algorithm runs in polynomial time of n. Both results only hold for the two dimensional case.

Other definitions

Given two points $p, q \in \mathbb{Z}^d$ such that $p \neq q$, the *slope* of $R(p, q)$ is the sign vector $\mathtt{t} = (t_1, t_2, \ldots, t_d) \in \{+1, -1\}^d$, where $t_i = +1$ if $p_i \leq q_i$ and is -1 if $p_i \geq q_i$. Note that two points have more than one slope when $p_i = q_i$ for some index i. For simplicity, we refer to *the* slope of $R(p, q)$ (whenever p and q have more than one slope we pick one arbitrarily). Let T be the set containing all 2^d slopes of $\mathbb{Z}^d$. For any two vectors $u, v \in \mathbb{Z}^d$, let $u \cdot v$ denote their dot product.

A *total order* θ of $\mathbb{Z}$ is a binary relation on all pairs of integers. We denote that a is smaller than b with respect to θ by $a \prec_\theta b$. We say that two elements a and b are *consecutive* if there is no number between them (i.e., no integer c satisfies $a \prec_\theta c \prec_\theta b$).

We define three operations on total orders: shift, flip and reverse. The *shift* operation is denoted by $\theta + c$ and is the result of adding a constant value c to each integer without changing their binary relations (that is, $a \prec_\theta b$ if and only if $a + c \prec_{\theta+c} b + c$). Similarly, the *flip* of θ is denoted by $-\theta$ and is the result of changing the sign of all binary relations (that is, $a \prec_\theta b$ if and only if $-a \prec_{-\theta} -b$). The *reverse* operation of θ (denoted by θ^{-1}) is the total order resulting in inverting all relationships (that is, $a \prec_\theta b$ if and only if $b \prec_{\theta^{-1}} a$).

For any $a < b \in \mathbb{Z}$, we will restrict a total order θ to an interval $[a, b]$ (and denote it by $\theta[a, b]$). For these subsets we also use the same shift, flip and reverse operations whose definitions follow naturally. In particular, observe that $(\theta[a, b])^{-1} = (\theta^{-1})[a, b]$, $(\theta[a, b]) + c = (\theta + c)[a + c, b + c]$, and $-(\theta[a, b]) = (-\theta)[-b, -a]$. Due to lack of space some proofs are deferred to the extended version of the document.

2.2 Results overview and paper organization

We study properties that CDRs and CDSs must satisfy in high dimensions (i.e., $d \geq 3$), and show that they behave very differently from the two-dimensional counterparts. In Section 3 we introduce the concept of *axis-order*. Although not needed in two dimensions, it allows us to extend the total order construction of Christ *et al.* to higher dimensions.

Given a point $p \in \mathbb{Z}^d$, a total order θ on the integers, and a slope t, we construct a partial CDS which we denote by $TOC(\theta, p, \mathsf{t})$. This partial CDS contains segments having an endpoint p and slope t. In two dimensions it generates a tree that covers a quadrant whose corner is p (analogously, in higher dimensions it covers an orthant whose apex is p).

In two dimensions we have $2^d = 4$ different slopes. Christ *et al.* [3] showed that we can pick any four total orders, apply each order to a different slope, and the union of the four constructions will be a CDR. In this paper we show that the analogous result does not hold in higher dimensions: fixing the total order for a single orthant uniquely determines the behavior of other orthants.

▶ **Theorem 5** (Necessary and sufficient condition for CDRs). *For any $d > 2$, point $p \in \mathbb{Z}^d$ and set $\{\theta_{\mathsf{t}} : \mathsf{t} \in T\}$ of 2^d total orders, $\bigcup_{\mathsf{t}\in T} TOC(\theta_{\mathsf{t}}, p, \mathsf{t})$ forms a CDR at p if and only if for any $\mathsf{t}, \mathsf{t}' \in T$ it holds that $\theta_{\mathsf{t}}[\mathsf{t} \cdot p, \infty) = \theta_{\mathsf{t}'}[\mathsf{t}' \cdot p, \infty) - \mathsf{t}' \cdot p + \mathsf{t} \cdot p$, where T is the set containing all possible slopes of $\mathbb{Z}^d$.*

In particular, there is a unique way of completing the partial CDS $TOC(\theta, p, \mathsf{t})$ to a CDR which we denote by $TOC(\theta, p)$. The next step is to consider the union of several CDRs to obtain a CDS. For the two dimensional case, Christ *et al.* showed that we can pick $2^{d-1} = 2$ total orders, and if we use them consistently for all points of $\mathbb{Z}^2$, the result will always be a CDS. Theorem 5 already implies that only one total order can be involved in the construction of CDSs. In Section 4 we observe that not all total orders will create one, and fully characterize which total orders do so.

▶ **Theorem 6** (Necessary and sufficient condition for CDSs). *θ is a total order such that $\bigcup_{p\in\mathbb{Z}^d} TOC(\theta, p)$ forms a CDS if and only if $\theta = \theta + 2$ and $\theta = -(\theta + 1)^{-1}$.*

In particular, this result positively answers the question posed by Christ *et al.* of whether their approach can be extended to create CDSs in higher dimensions (posed in a preliminary version of [3]).

The main difference between two dimensional and higher dimensional spaces is that the construction for two different slopes has a larger portion in common. In two dimensions, two quadrants share at most a line (whose behavior is unique because of the monotonicity axiom), but in general orthants may share a subspace of dimension $d - 1$. The total orders associated to each orthant must behave similarly within the subspace, which creates some dependency between the total orders. More importantly, each orthant shares subspaces with other orthants, and so on. This cascades creating common dependencies that cycle back to the original orthant and highly constrain the total orders. In Section 6 we discuss this dependency and argue that variations of this construction will also have the same necessary and sufficient conditions.

3 Extending the total order construction to higher dimensions

In this section we use a total order to construct a CDR in $\mathbb{Z}^d$. We start by reviewing the construction of Christ *et al.* [3] for $\mathbb{Z}^2$. Given a total order θ and two points $p = (p_1, p_2), q = (q_1, q_2) \in \mathbb{Z}^2$ such that $q_1 \geq p_1$ and $q_2 \geq p_2$, we view the digital segment $R(p, q)$ as a collection

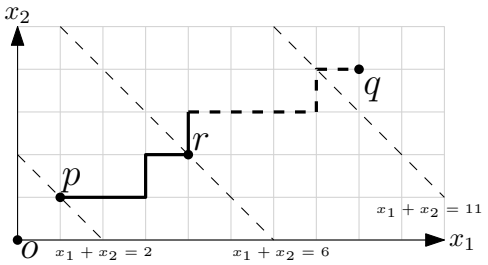


Figure 2 Example of the construction of Christ *et al.* in $\mathbb{Z}^2$. Given $p = (1, 1)$, $q = (8, 4)$ and a total order θ such that $\theta[2, 11] = 5 \prec 3 \prec 2 \prec 7 \prec 9 \prec 8 \prec 11 \prec 10 \prec 6 \prec 4$. The path must perform $q_1 - p_1 = 7$ steps in the x_1 direction and $q_2 - p_2 = 3$ steps in the x_2 direction. Since $p_1 + p_2 = 2$ and 2 is among the 7 smallest elements in $\theta[2, 11]$, it moves in the x_1 direction. Similarly, at point $r = (4, 2)$, the path will move in x_2 direction because $r_1 + r_2 = 6$ is among the 3 largest elements of $\theta[2, 11]$. Observe that, for any $c \in [2, 11]$ there is a unique point m in the path such that $m_1 + m_2 = c$.

of steps that form a path from p to q. Due to the monotonicity property, in each step the path increases either the first or second coordinate by one. Clearly, this path must do $q_1 + q_2 - p_1 - p_2$ steps, out of which $q_1 - p_1$ are in the x_1 coordinate (and the remaining ones in the x_2 coordinate). The choice of which steps we move in which coordinate depends on θ: assume that after moving several steps we have reached some intermediate point (r_1, r_2). Then, we check whether or not the number $r_1 + r_2$ is among the $q_1 - p_1$ smallest elements of $\theta[p_1 + p_2, q_1 + q_2 - 1]$. If so, we move from (r_1, r_2) in the x_1 coordinate. Otherwise we do so in the x_2 coordinate (see an example in Figure 2).

All of the segments created this way have slope $(+1, +1)$. In a similar way, we can pick a total order to define the segments emanating from p with slope $(+1, -1)$, $(-1, +1)$ and $(-1, -1)$. We emphasize that there is no dependency between the total orders: the choice of total order for one slope has no impact on the available options for the others. Moreover, any four choices will result in a CDR (similarly, any CDR in $\mathbb{Z}^d$ is associated with 2^d total orders of $\mathbb{Z}$, one for each slope). As mentioned before, this independence between quadrants does not hold in higher dimensions.

3.1 Constructing a CDR in $\mathbb{Z}^d$ from a total order

The construction of Christ *et al.* explains how to construct segments of slope $(+1, +1)$ in $\mathbb{Z}^2$ (or equivalently, for points in the first quadrant). The segments of different slopes are obtained via symmetry. In higher dimensions it will be useful to have an explicit way to construct segments of any slope. Thus, we first generalize the method of Christ *et al.* for any orthant.

In order to get an idea of our approach, we first look at the folklore bounding box approach to construct a CDS. When defining the path between point p and point q, we consider the minimum bounding box formed by the two points. The point with smaller x_1 coordinate will move in the x_1 coordinate until reaching the x_1 coordinate of another point. Afterwards, the one with smaller x_2 coordinate will move in the x_2 coordinate, and so on until the two points meet (see Figure 3).

So, if $d = 3$, for any segment whose slope is $(+1, +1, +1)$ we first do all the movements in the x_1 coordinate, then x_2 coordinate, and finally in the x_3 coordinate. However, if the segment has slope $(+1, -1, -1)$, then the bounding box CDS will travel first in the x_1 coordinate, then x_3 and finally x_2. Intuitively speaking, even though in both cases we are

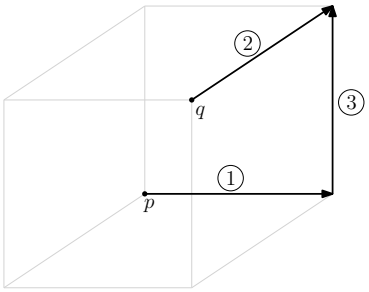


Figure 3 Example of the bounding box approach in $\mathbb{Z}^3$. $p = (0, 3, 0)$ and $q = (3, 0, 3)$. The number in each circle indicates the order in which we execute the movements.

performing the same steps (i.e, we use the natural order $0 \prec 1 \prec 2 \prec 3 \prec \ldots$), the order in which we execute each dimension is slightly different (or equivalently, the total order is being interpreted differently). We model this difference in interpretation through a new concept which we call *axis-order*.

Given a slope $(t_1, t_2, \ldots, t_d)$, let $a_1, \ldots a_k$ be the indices of the coordinates with positive value in increasing order (that is, $t_i = +1$ if and only if $i = a_j$ for some $j \leq k$). Similarly, let $b_1, \ldots b_{d-k}$ be the indices of the coordinates with negative value in decreasing order. Then, the *axis-order* of $(t_1, t_2, \ldots, t_d)$ is $x_{a_1}, x_{a_2}, \ldots, x_{a_k}, x_{b_1}, \ldots, x_{b_{d-k}}$. For example, the axis-order of $(-1, +1, +1)$ is x_2, x_3, x_1, and the axis-order of $(+1, -1, +1)$ is x_1, x_3, x_2. As we will see later, it will be useful to consider subspaces of $\mathbb{Z}^d$. We observe a property that follows from the definition of axis-order.

▶ **Observation 7.** *Let $a_1, \ldots a_k$ be a sequence of indices such that $a_1 < \ldots < a_k$, and let $\mathtt{t}, \mathtt{t}' \in \{-1, 1\}^d$ be two slopes such that $\mathtt{t}_{a_i} = \mathtt{t}'_{a_i}$ (for all $i \leq k$). Then, $\mathtt{t}$ and $\mathtt{t}'$ have the same axis-order τ restricted to a subspace $\mathcal{H}$ spanned by $\{x_{a_1}, x_{a_2}, \ldots, x_{a_k}\}$. Moreover, the axis-order of $-\mathtt{t}$ and $-\mathtt{t}'$ restricted to $\mathcal{H}$ is the reverse of τ.*

With the help of axis-order we can extend the two dimensional construction to higher dimensions. Given a point $p = (p_1, \ldots, p_d) \in \mathbb{Z}^d$, a total order θ and a slope $\mathtt{t}$, we construct the set of rays emanating from p with that slope. Define the orthant $\mathcal{O}_{\mathtt{t}}(p) = \{q \in \mathbb{Z}^d \colon \mathtt{t}_i \cdot q_i \geq \mathtt{t}_i \cdot p_i\}$: by definition, the segment from p to any point in $\mathcal{O}_{\mathtt{t}}(p)$ has slope $\mathtt{t}$. Also, let $x_{a_1}, x_{a_2}, \ldots$ be the axis-order of $\mathtt{t}$.

For any point $q = (q_1, \ldots, q_d) \in \mathcal{O}_{\mathtt{t}}(p)$ we construct the segment $R(p, q)$. Similar to the two dimensional case, the path from p to q must do $\mathtt{t} \cdot q - \mathtt{t} \cdot p$ steps, out of which $|p_1 - q_1|$ will be in the first coordinate, $|p_2 - q_2|$ in the second, and so on. We traverse through intermediate points, each time increasing the inner product with $\mathtt{t}$ by one. At each intermediate point r, we check the position of $\mathtt{t} \cdot r$ in $\theta[\mathtt{t} \cdot p, \mathtt{t} \cdot q - 1]$; if it is among the $|p_{a_1} - q_{a_1}|$ smallest elements in $\theta[\mathtt{t} \cdot p, \mathtt{t} \cdot q - 1]$ then we move in the x_{a_1} coordinate. Otherwise, if it is among the smallest $|p_{a_1} - q_{a_1}| + |p_{a_2} - q_{a_2}|$ elements we move in x_{a_2}, and so on.

For example, if the total order θ satisfies $3 \prec_\theta 1 \prec_\theta 5 \prec_\theta 7 \prec_\theta 9 \prec_\theta 8 \prec_\theta 6 \prec_\theta 4 \prec_\theta 2 \prec_\theta 0$, $p = (0, 0, 0)$ and $q = (2, -3, 5)$, the slope is $(+1, -1, +1)$, axis-order is x_1, x_3, x_2. So we must look at $\theta[p \cdot (+1, -1, +1), q \cdot (+1, -1, +1) - 1] = \theta[0, 9]$. In this total order the number $(+1, -1, +1) \cdot (0, 0, 0) = 0$ is the largest element in $\theta[0, 9]$, so we move from $(0, 0, 0)$ in the x_2 coordinate to point $(0, -1, 0)$. At point $(0, -1, 0)$ the number $(+1, -1, +1) \cdot (0, -1, 0) = 1$ is the second smallest element in $\theta[0, 9]$, so we move in the x_1 coordinate, and so on. Overall the path is $(0, 0, 0) \to (0, -1, 0) \to (1, -1, 0) \to (1, -2, 0) \to (2, -2, 0) \to (2, -3, 0) \to (2, -3, 1) \to (2, -3, 2) \to (2, -3, 3) \to (2, -3, 4) \to (2, -3, 5)$.

▶ **Definition 8.** For any point $p \in \mathbb{Z}^d$, slope $\mathtt{t}$, and total order θ, we call the collection of segments $\{R(p,q)\colon q \in \mathcal{O}_{\mathtt{t}}(p)\}$ the *total order* construction of θ (centered at p) for the slope $\mathtt{t}$, and denote it by $TOC(\theta, p, \mathtt{t})$.

3.2 Properties of the total order construction

▶ **Lemma 9** (Translation Lemma)**.** *For any $p \in \mathbb{Z}^d$, slope $\mathtt{t}$ and total order θ, the set of segments in $TOC(\theta, p, \mathtt{t})$ is the translated copy of the set of segments in $TOC(\theta - \mathtt{t} \cdot p, o, \mathtt{t})$, where o is the origin.*

▶ **Lemma 10.** *For any $p \in \mathbb{Z}^d$, slope $\mathtt{t}$ and total order θ, the set of segments in $TOC(\theta, p, \mathtt{t})$ forms a partial CDS on $\{p\} \times \mathcal{O}_{\mathtt{t}}(p)$.*

Proof. This statement is a particular case of Theorem 4: we are interested in segments of a single slope emanating from a fixed point, whereas Theorem 4 only requires segments of a fixed slope. The proof given by Christ *et al.* [3] is for slope $(+1, \ldots, +1)$, but the arguments extend naturally for the general case. ◀

Let θ_0 be the natural order on the integers (that is, $\theta_0 = \{\ldots \prec -1 \prec 0 \prec 1 \prec 2 \prec \ldots\}$). Fix any point $p \in \mathbb{Z}^d$ and apply the total order construction $TOC(\theta, p, \mathtt{t})$ to all slopes. Similarly, let θ_1 be result of swapping the position of -1 and -2 in θ_0 (i.e., $\theta_1 = \{\ldots \prec -1 \prec -2 \prec 0 \prec 1 \prec 2 \ldots\}$). Let $\mathcal{C}_0(p)$ and $\mathcal{C}_1(p)$ the union of segments created with each total order, respectively.

▶ **Proposition 11.** *$\mathcal{C}_0(p)$ is a CDR that is included in the bounding box CDS whereas $\mathcal{C}_1(p)$ is not a CDR.*

3.3 Gluing orthants to obtain CDRs

The second example of Proposition 11 shows an example of a total order that cannot be applied everywhere to form a CDR. Theorem 5 stated in Section 2.2 shows the relationship that total orders in different slopes must satisfy in order to create a CDR. Intuitively speaking, this correlation is so strong that choosing one total order effectively fixes the rest. The remainder of this section is dedicated to proving this interdependency. We start by showing the proof of one implication of the equivalence.

▶ **Lemma 12** (Necessary condition for CDRs)**.** *Let $p \in \mathbb{Z}^d$ and $\{\theta_{\mathtt{t}}\colon t \in T\}$ be a set of 2^d total orders such that $\bigcup_{\mathtt{t} \in T} TOC(\theta_{\mathtt{t}}, p, \mathtt{t})$ forms a CDR. Then, for any $\mathtt{t}, \mathtt{t}' \in T$, it holds that $\theta_{\mathtt{t}}[\mathtt{t} \cdot p, \infty) = \theta_{\mathtt{t}'}[\mathtt{t}' \cdot p, \infty) - \mathtt{t}' \cdot p + \mathtt{t} \cdot p$.*

Proof (Sketch). We prove the statement by contradiction. That is, assume that there exist two slopes $\mathtt{t}, \mathtt{t}'$ such that $v \prec_{\theta_{\mathtt{t}}} v'$ but $v' - \mathtt{t} \cdot p + \mathtt{t}' \cdot p \prec_{\theta_{\mathtt{t}'}} v - \mathtt{t} \cdot p + \mathtt{t}' \cdot p$. Without loss of generality, we can choose $\mathtt{t}$ and $\mathtt{t}'$ so that the corresponding orthants share a two-dimensional plane (pick a sequence of intermediate orthants so that pairwise they do, and look at the first time in which the equality is not satisfied). We pick a point q such that $R(p,q)$ has both slope $\mathtt{t}$ and $\mathtt{t}'$, and look at $R(p,q)$ from both the viewpoints of $TOC(\theta_{\mathtt{t}}, p, \mathtt{t})$ and $TOC(\theta_{\mathtt{t}'}, p, \mathtt{t}')$.

Along the path $R(p,q)$ we look at two intermediate points r and r'. The main feature of these points is that the behavior of $R(p,q)$ at those points depends on the positions of v and v' in $\theta_{\mathtt{t}}$ (if we look at it from the viewpoint of $TOC(\theta_{\mathtt{t}}, p, \mathtt{t})$). Since $v \prec_{\theta_{\mathtt{t}}} v'$, we can choose q in a way that the path will move in different directions at the two points. Then, we study the same segment from the viewpoint of the other orthant. In this case, the behavior of the same intermediate points will depend on the positions of $v' - \mathtt{t} \cdot p + \mathtt{t}' \cdot p$ and $v - \mathtt{t} \cdot p + \mathtt{t}' \cdot p$ in the shifted total order instead. Thus, if the relationships are reversed, the two paths behave differently and in particular we cannot have a CDR. ◀

▶ **Lemma 13** (Sufficient condition for CDRs). *For any point $p \in \mathbb{Z}^d$, let $\{\theta_{\mathsf{t}} : t \in T\}$ be a set of 2^d total orders such that $\theta_{\mathsf{t}}[\mathsf{t} \cdot p, \infty) = \theta_{\mathsf{t}'}[\mathsf{t}' \cdot p, \infty) - \mathsf{t}' \cdot p + \mathsf{t} \cdot p$ for any $\mathsf{t}, \mathsf{t}' \in T$. Then, $\bigcup_{\mathsf{t} \in T} TOC(\theta_{\mathsf{t}}, p, \mathsf{t})$ forms a CDR.*

This completely characterizes the CDRs that can be made with the total order construction in $\mathbb{Z}^d$. For any point p, slope t and total order θ, there is a unique CDR that can be created in this way and contains $TOC(\theta, p, \mathsf{t})$. Since the choice of slope is not important, let $TOC(\theta, p)$ be the unique CDR that contains $TOC(\theta, p, (+1, \ldots, +1))$.

▶ **Corollary 14.** *For any $p \in \mathbb{Z}^d$ there exist arbitrarily many CDRs with $O(\log n)$ Hausdorff distance.*

Proof. An explicit construction of a single CDR in $\mathbb{Z}^d$ with $O(\log n)$ Hausdorff distance was given by Chun *et al.* [4]. They showed that the CDR generated using the Van der Corput sequence [9] as total order has low Hausdorff distance (for any dimension). Christ *et al.* [3] extended the result showing that the straightness is asymptotically same as the discrepancy of the permutation corresponding to the total order, which is known to be $\Theta(\log n)$. Moreover, for any total order θ with low discrepancy, it holds that $\theta + k$ has low discrepancy (for any $k \in \mathbb{Z}$), so the arguments for $d = 2$ extend directly to the higher dimension construction. Thus, we omit them. ◀

4 Necessary and sufficient conditions for CDSs

Next we focus our attention to constructing CDSs. Christ *et al.* [3] showed that if we apply the same total order construction to all points of $\mathbb{Z}^2$ we get a collection of CDRs whose union is always a CDS. For any total order θ, let $TOC(\theta) = \bigcup_{p \in \mathbb{Z}^d} TOC(\theta, p)$. Unlike the two dimensional case, the construction $TOC(\theta)$ does not always yield a CDS in higher dimensions. Theorem 6 stated in Section 2.2 gives necessary and sufficient conditions that the total order must satisfy.

Recall that in principle, we allow different orthants (except $(+1, \ldots, +1)$) to have different total orders in this construction. For any point $p \in \mathbb{Z}^d$ and slope t, let θ^p_{t} be the total order associated to point p and slope t in $TOC(\theta)$. Since $TOC(\theta)$ in particular contains $TOC(\theta, p)$, Theorem 5 gives a relationship between θ and θ^p_{t}. We give a stronger bound on that relationship as well.

▶ **Theorem 15.** *If θ is a total order such that $TOC(\theta)$ forms a CDS, then for any $p \in \mathbb{Z}^d$ and slope t it holds that $\theta^p_{\mathsf{t}}[\mathsf{t} \cdot p, \infty) = \theta[\mathsf{t} \cdot p, \infty)$. In particular, $TOC(\theta^p_{\mathsf{t}}, p, \mathsf{t}) = TOC(\theta, p, \mathsf{t})$.*

This shows that, if we want to create a CDS in this fashion, we must use the same total order θ for all points and all slopes. Again, this contrasts with the $d = 2$ case where we can combine any two total orders for slopes $(+1, +1)$ and $(+1, -1)$. Christ *et al.* [3] showed that if we repeat the construction for all points of $\mathbb{Z}^2$ the union will form a CDS. The remainder of this section is dedicated to showing Theorems 6 and 15.

4.1 Two dimensional preliminaries

We will often consider two dimensional subspaces and find some requirements that extend to the whole space. Thus, we first show a subtree property that CDS in $\mathbb{Z}^2$ must satisfy. Consider any point $p \in \mathbb{Z}^2$, slope t, point $q \in TOC(\theta^p_{\mathsf{t}}, p, \mathsf{t})$ such that $q \neq p$, and all points $r \in \mathbb{Z}^2$ such that $R(p, r)$ passes through q. This set of points (and their paths to q) form a

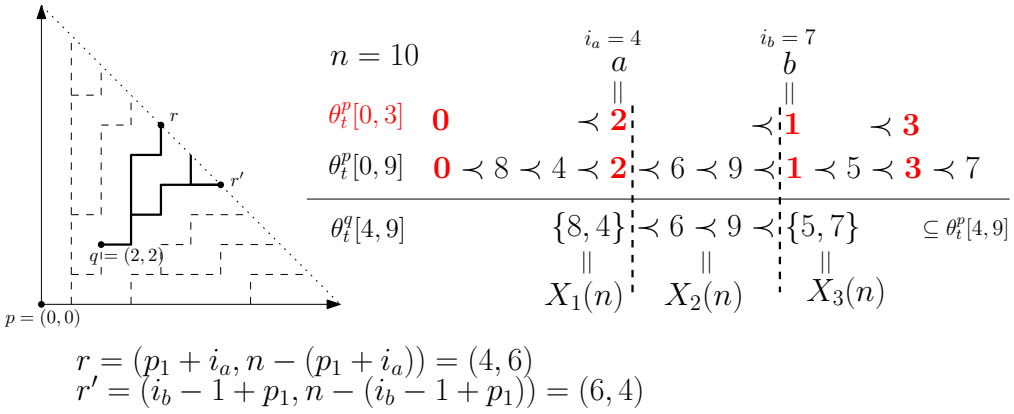


Figure 4 Example of the subtree property. (left) geometric interpretation of the subtree property. The paths to p that pass through q impose a constraint on θ^q_{t}. In particular, a point in the diagonal $x_1 + x_2 = n$ will pass through q if and only if it is between r and r' (highlighted points in the figure). (right) implications in the total order of θ^q_{t}. In red bold we highlight the points that belong to the left interval. The points in the right interval are classified into the three sets $X_1(n)$, $X_2(n)$ and $X_3(n)$ according to their positions (left of a, right of b, or in between). The fact that the subtree of q (black in the left figure) has to be preserved in q implies many relationships for θ^q_{t} that are shown in the third line.

subtree of $TOC(\theta^p_{\mathsf{t}}, p, \mathsf{t})$. The same tree must be part of $TOC(\theta^q_{\mathsf{t}}, q, \mathsf{t})$ or it would violate (S3) (see Figure 4, left).

We express this subtree property in terms of total orders θ^p_{t} and θ^q_{t}. Assume $\mathsf{t} = (+1, +1)$, let $s_1, s_2 \geq 0$ be integers such that $q = p + (s_1, s_2)$, and let n be any number such that $n > s_1 + s_2$. We will consider the restriction of the total order θ^p_{t} to three intervals: $[\mathsf{t}\cdot p, \mathsf{t}\cdot q - 1]$, $[\mathsf{t}\cdot q, \mathsf{t}\cdot p + n - 1]$, and $[\mathsf{t}\cdot p, \mathsf{t}\cdot p + n - 1]$. Note that the union of the first two forms the third one. In order to reduce notation we call them the left, the right, and the complete intervals. Similarly, we call $\theta^p_{\mathsf{t}}[\mathsf{t}\cdot p, \mathsf{t}\cdot q - 1]$, $\theta^p_{\mathsf{t}}[\mathsf{t}\cdot q, \mathsf{t}\cdot p + n - 1]$, and $\theta^p_{\mathsf{t}}[\mathsf{t}\cdot p, \mathsf{t}\cdot p + n - 1]$ the *left order*, the *right order* and the *complete order*. The subtree property says that many inequalities in the right order must also hold in θ^q_{t}.

First assume that $s_1, s_2 \neq 0$; let a and b be the s_1-th and $(s_1 + 1)$-th smallest numbers in the left order, respectively. By definition, these two numbers are consecutive in the left order, but they need not be in the complete order (i.e., there could be numbers from the right interval).

Let i_a and i_b be the positions of a and b in the complete order, respectively. We partition the numbers of the right interval into three groups, depending on whether they are (i) smaller than a, (ii) larger than a and smaller than b, or (iii) larger than b (all these comparisons are with respect to θ^p_{t}). Let $X_1(n)$, $X_2(n)$, and $X_3(n)$ be the three sets, respectively (see Figure 4).

Before giving the subtree property we extend the definitions of these three sets for the cases in which s_1 and s_2 can be zero. If $s_1 = 0$ then a and i_a are not well defined (similarly, b and i_b are not defined when $s_2 = 0$). In the first case we set $i_a = 0$, $X_1(n) = \emptyset$ and classify the numbers of the right interval into $X_2(n)$ and $X_3(n)$ depending on whether they are smaller or larger than b. Similarly, if i_b is not defined, we set $i_b = n + 1$, $X_3(n) = \emptyset$, and numbers are be split into the two sets $X_1(n)$ and $X_2(n)$.

The following lemma characterizes the points whose path to/from p passes through q in the quadrant of $(+1, +1)$.

▶ **Lemma 16.** *For any $n > s_1 + s_2$, let $r \in \mathbb{Z}^2$ be a point such that $r_1 + r_2 = p_1 + p_2 + n$. The path $R(p, r)$ passes through q if and only if $r_1 \geq q_1$, $r_2 \geq q_2$ and $i_a \leq r_1 - p_1 \leq i_b - 1$.*

▶ **Lemma 17** (The subtree property)**.** *For any $n > s_1 + s_2$ and $u, v \in [\mathsf{t} \cdot q, \mathsf{t} \cdot p + n - 1]$, the following relationships must hold in θ^q_{t}:*

- *$u \prec_{\theta^q_{\mathsf{t}}} v$ for all $u \in X_1(n)$ and $v \in X_2(n)$,*
- *$u \prec_{\theta^q_{\mathsf{t}}} v$ for all $u \in X_1(n) \cup X_2(n)$ and $v \in X_3(n)$,*
- *$u \prec_{\theta^q_{\mathsf{t}}} v$ for all $u, v \in X_2(n)$ such that $u \prec_{\theta^p_{\mathsf{t}}} v$.*

▶ Remark. Although we have stated the subtree property for slope $(+1, +1)$, it is straightforward to see that this result extends to other ones. We stick to this notation for simplicity of exposition, although we will afterwards use it for negative slope as well.

4.2 Application in high dimensional spaces

With the subtree property we can show the first necessary condition of Theorem 6.

▶ **Lemma 18.** *Let θ be a total order such that $TOC(\theta)$ forms a CDS. Then, $\theta = \theta + 2$.*

Proof. We first give a birdseye overview of the proof: choose an arbitrary $\lambda \in \mathbb{Z}$ and consider the affine plane $\mathcal{H} = \{x_3 = \lambda, x_4 = 0, \ldots, x_d = 0\}$. In this plane we look at the origin $p = (0, 0)$, and points $q = (0, -1)$ and $r = (-1, 0)$ (see Figure 5, left). In particular, we look at the third quadrant (the one with slope $(-1, -1)$): first, from Theorem 5 we know that $\theta^p_{(-1,-1)}$ must coincide with θ (on the interval $[\lambda, \infty)$).

We apply the subtree property from p to q and r; the key property is that both $\theta^q_{(-1,-1)}$ and $\theta^r_{(-1,-1)}$ coincide with $\theta + 2$ on the interval $[\lambda + 1, \infty)$. Moreover, all paths to p must pass through either q or r, which in particular implies that all inequalities from $\theta^p_{(-1,-1)}$ must also be preserved in either $\theta^q_{(-1,-1)}$ or $\theta^r_{(-1,-1)}$. By combining all of these properties, we show that θ coincides with $\theta + 2$ on the interval $[\lambda + 1, \infty)$. The result works for any value of λ, so when $\lambda \to -\infty$ we get $\theta = \theta + 2$ as claimed.

More formally, pick any $\lambda \in \mathbb{Z}$ and consider the points $p = (0, 0, \lambda, 0, \ldots 0)$, $q = (0, -1, \lambda, 0, \ldots, 0)$ and $r = (-1, 0, \lambda, 0, 0, \ldots, 0)$. By construction, these points lie on the affine plane $\mathcal{H} = \{x_3 = \lambda, x_4 = 0, \ldots, x_d = 0\}$ as claimed.

Let $\mathsf{t} = (+1, \ldots, +1)$ and $\mathsf{t}' = (-1, -1, +1, \ldots, +1)$. By definition of $TOC(\theta)$ we have $\theta^p_{\mathsf{t}} = \theta^q_{\mathsf{t}} = \theta^r_{\mathsf{t}} = \theta$. We use Theorem 5 to determine the total order used at slope t' for the three points: $\theta^p_{\mathsf{t}'}[\mathsf{t}' \cdot p, \infty) = \theta^p_{\mathsf{t}}[\mathsf{t} \cdot p, \infty) - \mathsf{t} \cdot p + \mathsf{t}' \cdot p = \theta[\mathsf{t} \cdot p, \infty) - \mathsf{t} \cdot p + \mathsf{t}' \cdot p = \theta[\lambda, \infty)$. Similarly, at point q we have $\theta^q_{\mathsf{t}'}[\lambda + 1, \infty) = \theta[\lambda - 1, \infty) + 2 = (\theta + 2)[\lambda + 1, \infty)$ and at point r we have $\theta^r_{\mathsf{t}'}[\lambda + 1, \infty) = (\theta + 2)[\lambda + 1, \infty)$ (The six total orders and their relevant orthants are depicted in Figure 5, right).

For any $n > 0$ consider the bounded interval $[\lambda, \lambda + n - 1]$. We apply Lemma 17 in the third quadrant to obtain relationships between $\theta^p_{\mathsf{t}'}$, $\theta^q_{\mathsf{t}'}$ and $\theta^r_{\mathsf{t}'}$. Let $X^{pq}_1(n), X^{pq}_2(n)$, and $X^{pq}_3(n)$ be the partition in the three sets obtained when applying the subtree property to p and q (similarly, we define the sets X^{pr}_i). Since we are applying it to the third quadrant and in particular the axis-order is x_2, x_1, we must swap the definitions of s_1 and s_2 (i.e., s_1 will be equal to the difference in the x_2 coordinate of p and q).

For the pair p, q we have $s_1 = 1$, $s_2 = 0$. Thus the left interval consists of the singleton $[\lambda, \lambda]$, the right interval is $[\lambda + 1, \lambda + n - 1]$, $X^{pq}_3(n) = \emptyset$ and we are splitting the numbers of the right interval into sets $X^{pq}_1(n)$ and $X^{pq}_2(n)$ depending on whether or not they are larger than λ. That is,

$$X^{pq}_1(n) = [\lambda + 1, \lambda + n - 1] \cap \{i \in \mathbb{Z} \colon i \prec_{\theta^p_{\mathsf{t}'}} \lambda\},$$
$$X^{pq}_2(n) = [\lambda + 1, \lambda + n - 1] \cap \{i \in \mathbb{Z} \colon \lambda \prec_{\theta^p_{\mathsf{t}'}} i\}.$$

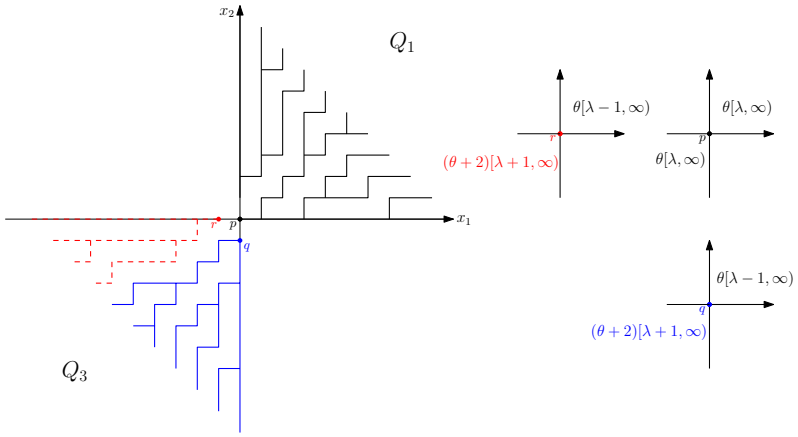


Figure 5 An example of the CDR at p is shown on the left hand side and the relationships between the total orders for the different quadrants at p, q and r on the right hand side. The subtrees at q and at r in Q_3 are represented by solid blue and dashed red segments respectively. In the example $\theta[0,8] = \{2 \prec 8 \prec 4 \prec 0 \prec 6 \prec 9 \prec 1 \prec 5 \prec 3 \prec 7\}$.

Applying the subtree property to the pair p, r gives a similar partition. In this case, the three sets become $X_1^{pr}(n) = \emptyset$, $X_2^{pr}(n) = [\lambda+1, \lambda+n-1] \cap \{i \in \mathbb{Z} \colon i \prec_{\theta^p_{\mathsf{t}'}} \lambda\} = X_1^{pq}(n)$, and $X_3^{pr}(n) = [\lambda+1, \lambda+n-1] \cap \{i \in \mathbb{Z} \colon \lambda \prec_{\theta^p_{\mathsf{t}'}} i\} = X_2^{pq}(n)$.

The sets X_i^{pq} imply some constraints on $\theta^q_{\mathsf{t}'}$ (similarly, X_i^{pr} gives constraints on $\theta^r_{\mathsf{t}'}$). Recall that we previously observed that $\theta^q_{\mathsf{t}'}[\lambda+1,\infty) = \theta^p_{\mathsf{t}'}[\lambda+1,\infty) = (\theta+2)[\lambda+1,\infty)$, which in particular implies that all constraints of the subtree property apply to $\theta+2$.

$X_2^{pq}(n)$ says that all relationships in $\theta^p_{\mathsf{t}'}[\lambda+1, \lambda+n-1]$ are preserved for numbers that are larger than λ in $\theta^p_{\mathsf{t}'}$. Similarly, $X_2^{pr}(n)$ says that relationships for numbers *smaller* than λ must also be preserved. Thus, we conclude that all relationships (both larger and smaller than λ) must be preserved. Hence, we conclude that $\theta^p_{\mathsf{t}'}[\lambda+1, \lambda+n-1] \subset (\theta+2)[\lambda+1,\infty)$. This reasoning applies for any values of $\lambda \in \mathbb{Z}$, and $n > 0$. In particular, when $\lambda \to -\infty$ and $n \to \infty$ we get $\theta = \theta + 2$ as claimed. ◀

With this result we can now show Theorem 15.

(Proof of Theorem 15). Let $\mathsf{t}' = (+1, \ldots, +1)$ and note that, by definition, we have $\theta^p_{\mathsf{t}'} = \theta$. We apply Theorem 5 and obtain $\theta^p_{\mathsf{t}}[\mathsf{t}\cdot p, \infty) = \theta^p_{\mathsf{t}'}[\mathsf{t}'\cdot p, \infty) - \mathsf{t}'\cdot p + \mathsf{t}\cdot p = \theta[\mathsf{t}'\cdot p, \infty) - \mathsf{t}'\cdot p + \mathsf{t}\cdot p$. The term $-\mathsf{t}'\cdot p + \mathsf{t}\cdot p$ must be an even number (since each coordinate of vector $\mathsf{t} - \mathsf{t}'$ is either a zero or a two). Thus, we can apply $\theta = \theta + 2$ repeatedly until we get $\theta[\mathsf{t}'\cdot p, \infty) - \mathsf{t}'\cdot p + \mathsf{t}\cdot p = \theta[\mathsf{t}\cdot p, \infty)$ as claimed. ◀

Specifically, we give two necessary conditions that together are also sufficient. The two conditions are derived from the axioms S1-S5. The first necessary condition is $\theta = \theta + 2$, which is already proved in Lemma 18.

The other necessary condition derives from the symmetry axiom (S2) of CDSs.

▶ **Lemma 19** (Necessary condition 2 for CDSs). *Any total order such that $TOC(\theta)$ forms a CDS satisfies that $\theta = -(\theta+1)^{-1}$.*

Proof (Sketch). This proof follows the same spirit as Theorem 5, but using the symmetry axiom instead. For any two numbers a, b such that such that $a \prec_\theta b$ we choose two points $p, q \in \mathbb{Z}^d$ and look at $R(p,q)$. In particular, we look at two specific intermediate points r

and s. The key property of these two points is that the behavior of $R(p,q)$ around those points is determined by the positions of a and b in θ. Then, we look at the symmetric path $R(q,p)$ and show that the behavior around the same intermediate points now depends on the positions of $-b-1$ and $-a-1$. In order to satisfy the symmetry axiom, the return path $R(q,p)$ has to be the same and thus we must have $-b-1 \prec_\theta -a-1$. ◀

This completes one side of the implication of Theorem 6. In order to complete the proof we show that the two requirements for θ are also sufficient.

▶ **Lemma 20** (Sufficient condition for CDSs). *Let θ be a total order that satisfies $\theta+2=\theta$ and $\theta=-(\theta+1)^{-1}$. Then, $TOC(\theta)$ forms a CDS.*

5 Characterization of necessary and sufficient conditions

Let $\mathcal{F}$ be the collection of total orders of $\mathbb{Z}$ that satisfy the necessary and sufficient conditions of Theorem 6. In order to bound the Hausdorff distance of the CDS associated to these constructions, we must give properties of total orders in $\mathcal{F}$.

▶ **Observation 21.** *All odd numbers appear monotonically in any total order θ that satisfies $\theta=\theta+2$. The same holds for even numbers.*

The above result follows from repeatedly applying the fact that $a \prec_\theta b \Leftrightarrow a+2 \prec_\theta b+2$. The second necessary condition also gives a strong relationship between odd and even numbers.

▶ **Observation 22.** *Let θ be a total order such that $\theta=-(\theta+1)^{-1}$. Then, it holds that $0 \prec_\theta 2 \Leftrightarrow -3 \prec_\theta -1$.*

By combining the previous two observations we get that either both odd and even numbers increase monotonically for any $\theta \in \mathcal{F}$ or both decrease monotonically. Next we study the relationship between odd and even numbers.

▶ **Lemma 23.** *Let $\theta \in \mathcal{F}$ be a total order in which two numbers of the same parity are consecutive in θ. Then, it holds that $1 \prec_\theta 2 \Leftrightarrow 2q+1 \prec_\theta 2q'$ for all $q,q' \in \mathbb{Z}$.*

▶ **Corollary 24.** *There are exactly four total orders in $\mathcal{F}$ in which two numbers of the same parity are consecutive.*

Proof. Let $\theta \in \mathcal{F}$ be any such total order. By Lemma 23 either all odd numbers appear before all even numbers or vice versa. There are four cases depending on whether $0 \prec_\theta 2$ or $2 \prec_\theta 0$ and $1 \prec_\theta 2$ or $2 \prec_\theta 1$. The first inequality determines whether all even numbers appear monotonically increasing or decreasing in θ (by Observations 21 and 22 this also determines the order of all odd numbers). The second inequality determines whether odd numbers are smaller or larger (with respect to $\prec_\theta$) than the even ones. Thus, under the assumption that two numbers of the same parity are consecutive in θ, only the following four orders exist:

$$\begin{aligned}
\tau_{o^+e^+} &= &&\{\ldots \prec 1 \prec 3 \prec 5 \prec \ldots \prec 0 \prec 2 \prec 4 \prec \ldots\},\\
\tau_{o^-e^-} &= &&\{\ldots \prec 5 \prec 3 \prec 1 \prec \ldots \prec 4 \prec 2 \prec 0 \prec \ldots\},\\
\tau_{e^+o^+} &= (\tau_{o^-e^-})^{-1} = &&\{\ldots \prec 0 \prec 2 \prec 4 \prec \ldots \prec 1 \prec 3 \prec 5 \prec \ldots\},\\
\tau_{e^-o^-} &= (\tau_{o^+e^+})^{-1} = &&\{\ldots \prec 4 \prec 2 \prec 0 \prec \ldots \prec 5 \prec 3 \prec 1 \prec \ldots\}.
\end{aligned}$$

◀

It remains to consider the case in which $\theta \in \mathcal{F}$ is a total order in which no two numbers of the same parity appear consecutively. That is, we have an odd number followed by an even number, followed by an odd number, and so on. For any $q \in \mathbb{Z}$, let α_q be the unique total order satisfying $0 \prec_{\alpha_q} 2q+1 \prec_{\alpha_q} 2 \prec_{\alpha_q} 2q+3$ and $\alpha_q = \alpha_q + 2$.

▶ **Theorem 25.** $\mathcal{F} = \{\tau_{o^+e^+}, \tau_{o^-e^-}, \tau_{e^+o^+}, \tau_{e^-o^-}\} \cup \{\alpha_q \colon q \in \mathbb{Z}\} \cup \{(\alpha_q)^{-1} \colon q \in \mathbb{Z}\}$

This completely characterizes the set $\mathcal{F}$ of total orders, and allows us to find a lower bound on the Hausdorff distance of the associated CDSs.

▶ **Theorem 26.** *For any $p = (p_1, \ldots, p_d) \in \mathbb{Z}^d$, total order $\theta \in \mathcal{F}$ and $n > 0$, there exists a point $q \in \mathbb{Z}^d$ such that $||p - q||_1 = 6n$ and $H(\overline{pq}, R(p,q)) \geq \frac{2\sqrt{5}n}{5}$.*

Proof (Sketch). Pick a point q sufficiently far from p and look at one every other step in the path $R(p,q)$. The way in which the path behaves will depend on the position of the odd numbers of θ (or even numbers depending on the parity of the starting point). Since odd and even numbers appear monotonically in θ, the path will do all steps in one direction before moving into a different one. Intuitively speaking, the movements in the odd numbers will form a bounding box and so will the movements in the even numbers (although the path is not necessarily the bounding box CDS). ◀

▶ Remark. First notice that a linear upper bound in the Hausdorff distance trivially follows from the monotonicity axiom. Although asymptotically speaking our construction has the same Hausdorff distance as the bounding box CDS, it can be seen that our leading constant is roughly twice smaller: for points whose L_1 distance is at most n, the bounding box CDS has an error of $\frac{\sqrt{2}n}{4} \approx 0.3n$ whereas, say, $TOC(\tau_{o^+e^+})$ has an error of $\frac{\sqrt{5}n}{15} \approx 0.15n$.

6 Conclusions

Increasing the dimension from two to three brings a significant change in the associated constraints for creating CDRs and CDSs. Although we have not been able to create a CDS with $o(n)$ Hausdorff distance, we believe that the results presented in this paper provide the first significant step towards this goal. The next natural step would be to consider constructions that apply different total orders to different points of $\mathbb{Z}^d$.

For simplicity of exposition, we have defined the CDS as the union of CDRs at all points. The construction of Christ *et al.* [3] considers the union of *half CDRs* instead (CDRs that are defined for only half of the slopes, such as slopes that satisfy $t_1 = +1$). We note that the same result would follow if we use their approach. Indeed, in order to derive the two necessary conditions, we have only looked at two slopes. For simplicity we have used $(+1, \ldots, +1)$ and $(-1, -1, +1, \ldots, +1)$, but the same result follows for any two slopes that differ in two coordinates. Thus, constructing CDSs by gluing half CDRs would result in the same necessary and sufficient constraints.

Similarly, one could consider using some kind of priority between slopes (say, lexicographical) so that if p and q are in more than one orthant, only the definition of $R(p,q)$ in the lexicographically smallest slope is considered. This removes the dependency between orthants (Theorem 5), but has a consistency problem: we can find three points $p, q, q' \in \mathbb{Z}^d$ such that $R(p,q)$ and $R(p,q')$ have different slopes, but the intersection of the two segments is not connected (such as in Figure 6).

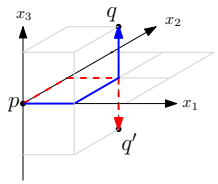


Figure 6 Removing dependency between orthants can create inconsistencies between them.

Acknowledgements. The authors would like to thank Takeshi Tokuyama and Matthew Gibson for their valuable comments during the creation of this paper, as well as the anonymous reviewers whose comments have helped in improving the quality of the paper.

References

1 Iffat Chowdhury and Matt Gibson. A characterization of consistent digital line segments in $\mathbb{Z}^2$. In *Proceedings of the 23rd Annual European Symposium on Algorithms*, pages 337–348, 2015.

2 Iffat Chowdhury and Matt Gibson. Constructing consistent digital line segments. In *Proceedings of the 12th Latin American Theoretical Informatics Symposium*, pages 263–274, 2016.

3 Tobias Christ, Dömötör Pálvölgyi, and Miloš Stojaković. Consistent digital line segments. *Discrete & Computational Geometry*, 47(4):691–710, 2012.

4 Jinhee Chun, Matias Korman, Martin Nöllenburg, and Takeshi Tokuyama. Consistent digital rays. *Discrete and Computational Geometry*, 42(3):359–378, 2009.

5 Michael T. Goodrich, Leonidas J. Guibas, John Hershberger, and Paul J. Tanenbaum. Snap rounding line segments efficiently in two and three dimensions. In *Proceedings of the 13th Annual Symposium on Computational Geometry*, pages 284–293, 1997.

6 Daniel H. Greene and F. Frances Yao. Finite-resolution computational geometry. In *Proceedings of the 27th Annual Symposium on Foundations of Computer Science*, pages 143–152, 1986.

7 M. G. Luby. Grid geometries which preserve properties of Euclidean geometry: A study of graphics line drawing algorithms. In *NATO Conference on Graphics/CAD*, pages 397–432, 1987.

8 Kokichi Sugihara. Robust geometric computation based on topological consistency. In *Proceedings of the 9th International Conference on Computational Science*, pages 12–26, 2001.

9 Johannes van der Corput. Verteilungsfunktionen I & II (in german). *Proceedings of the Koninklijke Nederlandse Akademie van Wetenschappen*, 38:813–820, 1058–1066, 1935.

TSP With Locational Uncertainty: The Adversarial Model*†

Gui Citovsky[1], Tyler Mayer[2], and Joseph S. B. Mitchell[3]

1 **Google Manhattan, New York, NY, USA**
gcitovsky@gmail.com
2 **Dept. of Applied Mathematics and Statistics, Stony Brook University, Stony Brook, NY, USA**
tyler.mayer@stonybrook.edu
3 **Dept. of Applied Mathematics and Statistics, Stony Brook University, Stony Brook, NY, USA**
joseph.mitchell@stonybrook.edu

Abstract

In this paper we study a natural special case of the Traveling Salesman Problem (TSP) with point-locational-uncertainty which we will call the *adversarial TSP* problem (ATSP). Given a metric space (X, d) and a set of subsets $R = \{R_1, R_2, ..., R_n\} : R_i \subseteq X$, the goal is to devise an ordering of the regions, σ_R, that the tour will visit such that when a single point is chosen from each region, the induced tour over those points in the ordering prescribed by σ_R is as short as possible. Unlike the classical locational-uncertainty-TSP problem, which focuses on minimizing the expected length of such a tour when the point within each region is chosen according to some probability distribution, here, we focus on the *adversarial model* in which once the choice of σ_R is announced, an adversary selects a point from each region in order to make the resulting tour as long as possible. In other words, we consider an offline problem in which the goal is to determine an ordering of the regions R that is optimal with respect to the "worst" point possible within each region being chosen by an adversary, who knows the chosen ordering. We give a 3-approximation when R is a set of arbitrary regions/sets of points in a metric space. We show how geometry leads to improved constant factor approximations when regions are parallel line segments of the same lengths, and a polynomial-time approximation scheme (PTAS) for the important special case in which R is a set of disjoint unit disks in the plane.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems, G.2.2 Graph Theory

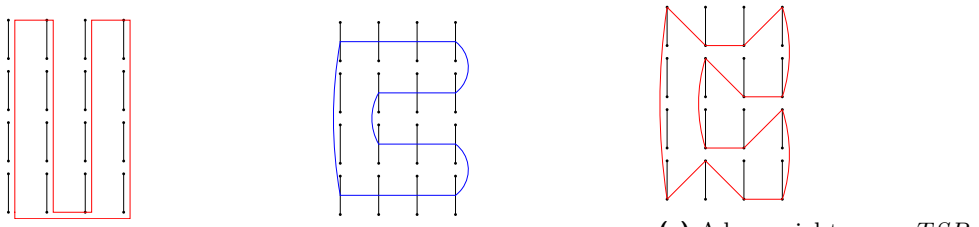
Keywords and phrases traveling salesperson problem, TSP with neighborhoods, approximation algorithms, uncertainty

Digital Object Identifier 10.4230/LIPIcs.SoCG.2017.32

1 Introduction

We consider the travelling salesperson problem (TSP) on uncertain sites. We are given as input a set of n uncertainty regions $R = \{R_1, R_2, \ldots, R_n\}$, each of which is known to contain exactly one site that must be visited by the tour. In the standard TSP, the regions R_i are singleton points. In the *TSP with neighborhoods* (TSPN), or *one-of-a-set TSP*, model, the

* A full version of the paper is available at https://arxiv.org/abs/1705.06180.
† This research was partially supported by the National Science Foundation (CCF-1526406).



(a) Optimal solution $\approx n$ **(b)** TSP_c **(c)** Adversarial tour on TSP_c ordering $\approx \sqrt{2}n$

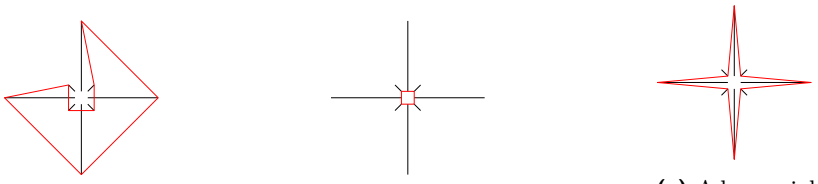
Figure 1 TSP on center points ordering does not always provide an optimal solution to ATSP.

goal is to compute an optimal tour that visits some point of each region R_i, and we are allowed to pick any point $p_i \in R_i$ to visit, making this choice in the most advantageous way possible, to minimize the length of the resulting tour that we compute. In models of TSP with locational uncertainty, the regions R_i model the support sets of probability distributions for the uncertain locations of the (random variable) sites p_i. The objective, then, may be to optimize some statistic of the tour length; e.g., we may wish to minimize the expected tour length, or minimize the probability that the tour length is greater than some threshold, etc. In this paper, we study the version of the stochastic TSP model in which our goal is to optimize for the *worst case* choice of p_i within each R_i. We call this problem the *adversarial TSP*, or ATSP, as one can think of the choice of p_i within each R_i as being made by an adversary. Our goal is to compute a permutation σ_R on the regions R_i so that we minimize the length of the resulting tour on the points p_i, assuming that an adversary makes the choice of $p_i \in R_i$, given our announced permutation σ_R on the regions. While the TSPN seeks an optimal tour for the *best* choices of $p_i \in R_i$, the ATSP seeks an optimal tour for the *worst* choices of $p_i \in R_i$.

Another motivation for the ATSP solution is that one may seek a single permutation of the set of input sets R_i so that the permutation is "good" (controls the worst-case choices of $p_i \in R_i$) for any of the numerous $(|R_1| \cdot |R_2| \cdot |R_3| \cdots |R_n|)$ instances of TSP associated with the sets R_i, thereby avoiding repeated computations of TSP tours. In certain vehicle routing applications, it may also be beneficial to establish a fixed ordering of visits to clients, even if the specific locations of these visits may vary in the sets R_i. Further, in locationally uncertain TSP one may expect that probability distributions over the regions R_i are imperfect and not known precisely, and that customer locations are known imprecisely (possibly for privacy concerns, with deliberate noise added for protecting the identity/privacy of users), making it important to optimize over all possible choices of site locations.

In Figure 1 we give a simple example showing that the ordering given by a TSP on center points (TSP_c) can be suboptimal, by at least a factor of $\sqrt{2}$. The input R is a $\sqrt{n} \times \sqrt{n}$ grid of vertical unit-length line segments with distance 1 between midpoints of horizontally adjacent segments and with distance $1 + \epsilon$ between midpoints of vertically adjacent segments. In Figure 2 we show that the ordering prescribed by a TSPN over the input regions can be at least a factor 2 away from optimal. The input is a set of n segments, $n/2$ of which have length 1, and the remainder have length ϵ; they are arranged in alternating order radially around a point or the boundary of a small circle.

In this paper, we initiate the study of the ATSP. We give a 3-approximation when R is a set of arbitrary regions/sets of points in a metric space. We exploit geometry to give an improved approximation bound for the case of regions that are unit line segments of the same orientation in the plane; we compute a permutation with adversarial tour length at



(a) Optimal solution $\approx n/2$ **(b)** TSPN **(c)** Adversarial tour on TSPN ordering $\approx n$

Figure 2 TSPN ordering does not always provide an optimal solution to ATSP.

most $(7/3+\epsilon)|OPT|+1$, where $|OPT|$ is the length of an optimal solution. We further exploit geometry to give a polynomial time approximation scheme (PTAS) for the important special case when R is a set of disjoint unit disks in the plane.

Related Work

Geometric problems on imprecise points have been the subject of many recent investigations. Löffler et al. [11] study, given a set of n uncertainty regions in the plane, the problem of *selecting* a single point within each region so that the area of the resulting convex hull is as large/small as possible. They show a number of results, including an $O(n^3)$-time and an $O(n^7)$-time exact algorithm for maximizing the area of the convex hull of selected points when the uncertainty regions are parallel line segments and disjoint axis aligned squares respectively. They show that this problem is NP-Hard when the regions are line segments with arbitrary orientations. In the same paper, Löffler et al. show that the problem of selecting a point within each region so that the resulting minimum spanning tree over those points is as small as possible is NP-Hard when the uncertainty regions are overlapping disks and squares. In his thesis [6], Fraser extends the prior minimum spanning tree result to show that the problem is still NP-Hard even when the regions are pairwise disjoint. He provides several constant factor approximation algorithms for the special case of disjoint disks in the plane. Dorrigiv et al. [4] show that neither the minimization nor the maximization version of this problem admit an FPTAS when the regions are disjoint disks. Yang et al. [16] give a PTAS for the minimization version. In a thesis by Montanari [15], it is shown that the minimization version when the input regions are vertically or horizontally aligned segments is NP-Hard and that this problem does not admit a FPTAS. Interestingly, in another paper by Liu and Montanari [10] it is shown that selecting a point from each region so that *diameter* of a minimum spanning tree on the selected points is minimized is polynomially solvable when the regions are arbitrary sized (possibly overlapping) disks in the plane.

Montanari [15] also studies the problem of placing a single point within each region so that the resulting shortest s,t path is either maximized or minimized. They show that the minimization version of this problem can be solved in polynomial time in the L_1 metric when the polygons are rectilinear (not necessarily disjoint, or convex). They also show that the maximization version of the problem is NP-Hard to approximate to any factor $(1-\epsilon):\epsilon<1/4$ even in the case where the polygons are vertically aligned segments.

There has been a considerable amount of work done on studying TSP variants with point-existential uncertainty. Two main models in the literature are the *a priori model* proposed by Bertsimas et al. [2] and Jaillet [7], in which each point x_i (with a known, fixed, location) is independently present with probability p_i, and the *universal model* [8], which asks for a tour over the entire data set such that for *any* subset of active requests, the master tour restricted to this active subset is *close* to optimal.

The TSP with neighborhoods (TSPN) problem was introduced by Arkin and Hassin [1] and has been studied extensively from the perspective of approximation algorithms, particularly in geometric domains (see, e.g.,[13]). Kamousi et al [9] study a stochastic TSPN model where each client lies within a region, a disk with a fixed center and stochastic radius.

Preliminaries

We are given regions $R = \{R_1, R_2, \ldots, R_n\}$, with each R_i a subset of a metric space (X, d). We seek a cyclic permutation $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_n)$ (an *ordering*) of the regions R, in order to minimize the length, $\max_{p_i \in R_i}[d(p_{\sigma_1}, p_{\sigma_2}) + d(p_{\sigma_2}, p_{\sigma_3}) + \cdots + d(p_{\sigma_{n-1}}, p_{\sigma_n}) + d(p_{\sigma_n}, p_{\sigma_1})]$, of a cycle on adversarial choices of the points in the respective regions. We let σ^*_R denote an optimal ordering for R, and we let $|OPT|$ denote the length of the corresponding cycle, OPT, that is based on the optimal adversarial choices of the points $p_i \in R_i$, for the ordering σ^*_R. The following lemmas are shown in the full paper [3].

▶ **Lemma 1.** *The length, $|OPT|$, of OPT satisfies $TSPN^* \leq |OPT| \leq TSPN^* + \sum_{R_i \in R} 2 \cdot diam(R_i)$, where $TSPN^*$ is the length of an optimal TSPN tour on the regions R, and $diam(R_i)$ denotes the diameter of region $R_i \in R$.*

▶ **Lemma 2.** *For a set R of convex regions in the Euclidean plane, and any ordering σ of the regions R, any longest cycle corresponding to an adversarial choice of points $p_i \in R_i$ is a polygonal cycle, with edges $(p_{\sigma_i}, p_{\sigma_{i+1}})$ and with each point p_{σ_i} an extreme point of its corresponding region, R_{σ_i}.*

2 3-Approximation for Arbitrary Regions in a Metric Space

We begin by giving a 3-approximation to the ATSP problem when R is a set of arbitrary regions in a metric space.

Consider the complete graph $\hat{G}$ whose nodes are the regions R and whose edges join every pair of regions with an edge, (R_i, R_j), whose weight is defined to be $w(R_i, R_j) = \max_{s \in R_i, t \in R_j}\{d(s,t)\}$, the maximum distance between a point $s \in R_i$ and a point $t \in R_j$. For distinction, we will speak of edge "weights" in the graph $\hat{G}$ and of edge "lengths" in the original metric space (X, d). It is not hard to see that the edge-weighted graph $\hat{G}$ defines a metric (see the full paper [3]).

▶ **Lemma 3.** *An optimal TSP tour in $\hat{G}$ yields a 2-approximation to the ATSP on R.*

Proof. Let $\sigma^*_R =< R^*_1, R^*_2, ..., R^*_n >$ be an optimal (cyclic) permutation of the regions R for the adversarial TSP on R, and let $p^*_i \in R^*_i$ be the adversary's choice of points corresponding to σ^*_R. Then, $|OPT| = d(p^*_1, p^*_2) + d(p^*_2, p^*_3) + \cdots + d(p^*_n, p^*_1)$ is the length of the cycle $C =< p^*_1, p^*_2, \ldots, p^*_n >$, an optimal adversarial TSP solution.

Let $w_{\sigma^*_R} = w(R^*_1, R^*_2) + w(R^*_2, R^*_3) + \cdots + w(R^*_n, R^*_1)$ be the total weight of the cycle σ^*_R in $\hat{G}$. Let w^*_{TSP} be the total weight of a minimum-weight Hamiltonian cycle, given by (cycle) permutation σ_{TSP}, in $\hat{G}$; then, $w^*_{TSP} \leq w_{\sigma^*_R}$.

Our goal is to show that the permutation σ_{TSP} yields a 2-approximation for the adversarial TSP on R. Since the length of the adversarial cycle corresponding to σ_{TSP} is at most w_{TSP}, and since $w^*_{TSP} \leq w_{\sigma^*_R}$, it suffices to show that $w_{\sigma^*_R} \leq 2|OPT|$.

Consider the cycle $C =< p^*_1, p^*_2, \ldots, p^*_n >$ whose length is $|OPT|$. If we modify C by choosing points within each region R^*_i differently from $p^*_i \in R^*_i$, the length of C can only go down, since the points p^*_i were chosen adversarially to make the cycle C as long as possible (for the given permutation σ^*_R). Consider two copies of C (of total length $2|OPT|$); we will

modify these two cycles into two (possibly shorter) cycles, C_1 and C_2, by making different choices for the points in each region R_i^*.

Consider first the case that n is even. Then, we define C_1 to be the modification of cycle C in which the points are chosen in regions R_i^* in order to maximize the lengths of the "odd" edges, $(R_1^*, R_2^*), (R_3^*, R_4^*), \ldots, (R_{n-1}^*, R_n^*)$, and we define C_2 to be the modification of cycle C in which the points are chosen in regions R_i^* in order to maximize the lengths of the "even" edges, $(R_2^*, R_3^*), (R_4^*, R_5^*), \ldots, (R_n^*, R_1^*)$. The cycle C_1, then, has length at least $w(R_1^*, R_2^*) + w(R_3^*, R_4^*) + \cdots + w(R_{n-1}^*, R_n^*)$, the total weights of the odd edges in the cycle in $\hat{G}$ corresponding to σ_R^*. Similarly, the cycle C_2 has length at least $w(R_2^*, R_3^*) + w(R_4^*, R_5^*) + \cdots + w(R_n^*, R_1^*)$, the total weights of the even edges in the cycle in $\hat{G}$ corresponding to σ_R^*. Together, then, the lengths of the two cycles C_1 and C_2 total at least the weight, $w_{\sigma_R^*}$, of the cycle σ_R^* in the graph $\hat{G}$. Since each of the weights of C_1 and C_2 are at most $|OPT|$ (the weight of C), we conclude that $w_{\sigma_R^*} \leq 2|OPT|$, as claimed.

The case in which n is odd is handled similarly; details appear in the full paper [3]. ◀

▶ **Theorem 4.** *The permutation σ_R corresponding to a Christofides 3/2-approximate TSP tour in $\hat{G}$ yields a 3-approximation to the adversarial TSP on R.*

3 Unit Line Segments of the Same Orientation in the Plane

In this section, we assume that R consists of a set of n unit-length segments of the same orientation; without loss of generality, we assume the segments are vertical. We show that the ordering, TSP_c, given by an optimal TSP tour on the segment center points yields an adversarial tour of length at most $(7/3)|OPT| + 1$; thus, a PTAS to approximate TSP_c yields an algorithm with adversarial tour length at most $(7/3 + \epsilon)|OPT| + 1$, for any fixed ϵ.

▶ **Lemma 5.** *For the ATSP on a set R of unit vertical segments in the plane, $|OPT| \geq TSP_c^*$, where TSP_c^* is the length of an optimal TSP tour on the segment center points.*

Proof. Consider an ATSP optimal ordering σ_R^* of the vertical segments R. The cycle γ_c that visits the center points of segments R in the order σ_R^* has length at least TSP_c^*, and the length, $|OPT|$, of an adversarial cycle for σ_R^* is at least the length of γ_c. ◀

▶ **Lemma 6.** *$|OPT| \geq \frac{3}{4}(n-1)$ when n is odd and $|OPT| \geq \frac{3}{4}n$ when n is even.*

Proof. It suffices to show the claim for ATSP paths; an ATSP cycle is at least as long. The proof is by induction on $n = |R|$. First, suppose that n is odd. The base case is trivially true. Assume that the claim holds for $n \leq k$, for k odd. Next, consider an instance S' with $k+2$ segments. We know that for the first k segments in an optimal permutation for S', that $|OPT| \geq \frac{3}{4}(k-1)$. Next, we show that regardless of the placement of the next two unit segments in S', s_{k+1} and s_{k+2}, an adversary can make us pay at least 3/2 units for every independent pair of consecutive segments in σ_S^*. We can assume that (vertical) segments s_{k+1} and s_{k+2} are vertically collinear. Next we assume, without loss of generality, that s_{k+2} is above s_{k+1}. Let a be the point on s_k that the adversary chose; refer to Figure 3. Let b_1 (resp., c_1) be the top endpoint of s_{k+1} (resp., s_{k+2}). Let b_2 (resp. c_2) be the bottom endpoint of s_{k+1} (resp., s_{k+2}). Now, let $|b_1 a| = x$ and $|c_2 b_1| = y$. This implies that $|ab_2| = 1 - x$, $|c_2 b_2| = 1 - y$ and $|c_1 b_1| = 1 - y$. The three candidate routes for the adversary to take are (a, b_2, c_1) or (a, b_1, c_2) or (a, b_1, c_1). These paths have lengths $3 - x - y$, $x + y$, $x + 1 - y$, respectively. Thus, we solve $min - max_{x,y}\{\{3-x-y, x+y, x+1-y\} : 0 \leq x \leq 1, 0 \leq y \leq 1\}$ to find the minimum possible length of the adversarial route; the solution is 3/2. Thus, the adversary can make us pay 3/2 for each pair of segments; thus, $|OPT| \geq \frac{3}{4}(n-1)$ for n odd.

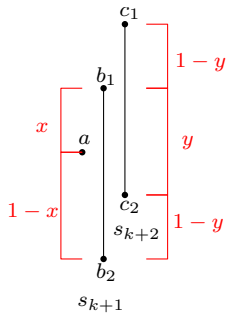


Figure 3 Illustration of the induction step in the proof of Lemma 6.

In the case that n is even, the adversary can make us pay at least $3/2$ between every consecutive pair of segments in the optimal ordering; thus, $|OPT| \geq \frac{3}{4}n$. ◀

▶ **Theorem 7.** *For the ATSP on a set R of unit-length vertical segments, the ordering given by an optimal TSP on the segment center points yields an adversarial tour of length at most $(7/3)|OPT| + 1$. Thus, a PTAS for the TSP on center points yields an approximation algorithm for ATSP, with tour length at most $(7/3 + \epsilon)|OPT| + 1$.*

Proof. Let APX_c be the ordering in which the segments are visited by a $(1+\epsilon)$-approximate TSP tour on their center points, and let $|APX_c|$ be the cost of the resulting adversarial tour for this ordering. We know that $|APX_c| \leq |TSP_c| + n$, where $|TSP_c|$ is the length of an optimal TSP on center points, since a tour on the center points can be made to detour to either endpoint and back, for each segment, at a total increase in length of n. Since $|TSP_c| \leq |OPT|$ (by Lemma 5) and $n \leq 4/3|OPT| + 1$ (by Lemma 6), we have that $|APX| \leq (7/3 + \epsilon)|OPT| + 1$. ◀

4 PTAS for Disjoint Unit Disks in the Plane

In this section we give a PTAS for the adversarial TSP problem when the regions $R = \{d_1, \ldots, d_n\}$ are n disjoint unit-diameter disks in the plane. We employ the m-guillotine method [12], which has been applied to give approximation schemes for a wide variety of geometric network optimization problems, including the Euclidean TSP and the *TSP with Neighborhoods* (TSPN) when the regions are disjoint disks or fat regions in the plane [5, 14].

The challenge in applying known PTAS techniques is being able to handle the adversarial nature of the tour. For the TSPN problem, one computes (using dynamic programming) a shortest connected m-guillotine, Eulerian, spanning subgraph of the regions; a tour visiting each region can then be extracted from this network. A structure lemma shows that an optimal TSPN solution can be converted to an m-guillotine solution whose weight is at most $(1+\epsilon)|OPT|$. Since m-guillotine networks have a recursive structure, we can apply dynamic programming in order to find the cheapest such structure over the input. Then, by extracting a tour from the optimal m-guillotine network, we obtain a permutation of the input disks, as well as a *particular* point within each region that the tour visits.

For the ATSP problem, we require new ideas and a new structure theorem to account for the fact that our algorithm must search for a permutation of the input disks that is good with respect to an adversarial path through the ordered disks. We seek to optimize a network that has a recursive structure (to allow dynamic programming to be applied) and that yields an ordering of the disks so that the length of the adversary's tour is "very

close" to optimal among all possible permutations. We do this by searching for a shortest (embedded) network having an m-guillotine structure that has additional properties that guarantee that the adversary's path through the sequence of regions we compute is not much longer than that of the network we compute. To accomplish this, we will require several structural results about an optimal solution to ATSP.

4.1 Discretization and a Structural Theorem

In order to make our problem and our algorithm discrete, for a fixed integer $m = O(1/\epsilon)$, we place m *sample points* evenly spaced around the boundaries of each of the n disks $d_i \in R$. Let $\mathcal{G}$ be the set of all nm sample points. Let $E_{\mathcal{G}}$ denote the set of edges (line segments) between two sample points of $\mathcal{G}$ that lie on the boundaries of different disks of R. The following lemma shows that for any adversarial (polygonal) tour T associated with σ_R there is a polygonal tour T' visiting the sequence σ_R whose vertices are among the sample points $\mathcal{G}$ and whose length is at least $(1 - O(1/m))|T|$.

▶ **Lemma 8.** *Given an adversarial (polygonal) path/cycle, T, associated with a sequence σ_R of input disks, there is a polygonal path/cycle T' that visits sample points $\mathcal{G}$, exactly one per disk, in the order σ_R, such that $|T| \leq (1 + O(1/m))|T'|$.*

Proof. We let T' be the path/cycle obtained from T by rounding each of its vertices to the closest sample point of the associated (unit-diameter) disk. This rounding results in each edge decreasing in length by at most $2 \cdot \frac{\pi}{m}$, since the sample points are spaced on the disk boundary at distance (along the boundary) of $\frac{2\pi(1/2)}{m}$. Thus, $|T| \leq |T'| + \frac{2\pi}{m}n$. We obtain a lower bound on $|T'|$, in terms of n, using an area argument (as done in [5], but included here for completeness). Let $A(T')$ be the area swept by a disk of radius 1 whose center traverses T'; it is well known that the area swept by a disk of radius δ whose center moves on a curve of length λ is at most $2\delta\lambda + \pi\delta^2$, implying that $A(T') \leq 2|T'| + \pi$. Since T' meets all n of the unit-diameter disks d_i, we know that $A(T') \geq n \cdot \pi(1/2)^2$. Thus, $n \leq (8/\pi)|T'| + 4 \leq O(|T'|)$ (assuming that $|T'| \geq c$, for some constant c, which holds if $n \geq 2$). Since $n \leq O(|T'|)$, the inequality $|T| \leq |T'| + \frac{2\pi}{m}n$ implies that $|T| \leq (1 + O(1/m))|T'|$, as desired. ◀

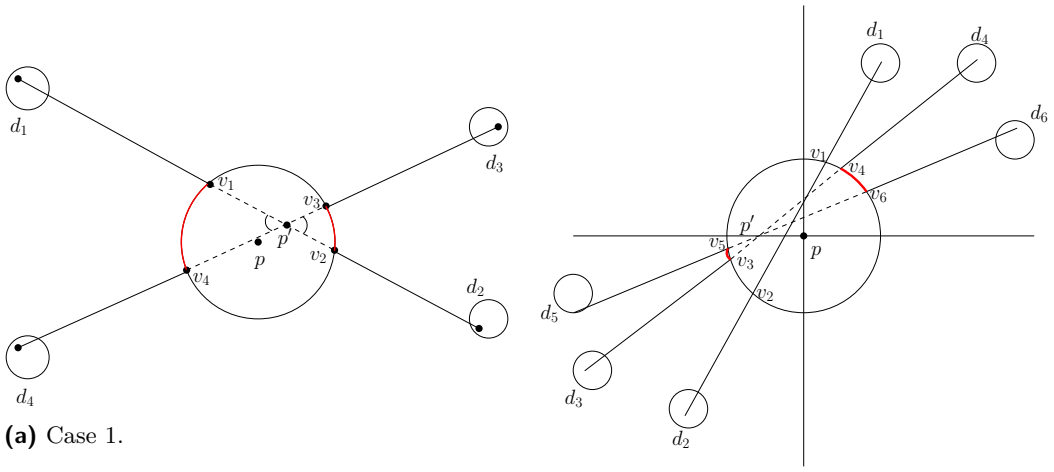
A corollary of Lemma 8 is that, for purposes of obtaining a PTAS, it suffices to search for an optimal adversarial tour in the discrete graph of edges $E_{\mathcal{G}}$ on sample points.

For two consecutive disks, d_i and d_{i+1} in an ordering σ_R, we refer to the convex hull of d_i and d_{i+1} as the *fat edge* associated with (d_i, d_{i+1}). The collection of such fat edges will be called the *convex hull tour* associated with σ_R.

▶ **Theorem 9.** *No point $p \in \mathbb{R}^2$ in the plane lies within more than a constant number of fat edges of the convex hull tour, OPT, associated with an optimal ordering σ_R^*.*

Proof. Consider an arbitrary point $p \in \mathbb{R}^2$ and consider its intersection with the convex hull tour of OPT. Center a disk D_p, of radius K centered at p, with $K = O(1)$ a constant to be determined later. Since the disks d_i are disjoint, there are only a constant number ($O(K^2)$) that intersect D_p. We remove from R these disks, as well as the (at most two) disks adjacent to them in the tour OPT. Let R' be the remaining set of disks after these (constant number of) disks are removed from R.

We claim that p is contained in no more than a constant number of the fat edges of OPT joining two disks of R'. Assume to the contrary that more than a constant number of remaining fat edges of the convex hull tour of OPT connecting disks of R' contain p. Consider two such fat edges, (d_1, d_2) and (d_3, d_4), containing p in the region where they



(a) Case 1.

(b) Case 2.

Figure 4 Case analysis for fat edge swapping.

properly cross. Each of these fat edges must pass "nearly" diametrically across D_p. That is they must cross D_p in such a way that they contain its center point p. We will show that by uncrossing these two fat edges we obtain a strictly shorter adversarial tour, thereby contradicting the optimality assumption. Suppose, without loss of generality, that, in order to preserve connectivity, the uncrossing replaces (d_1, d_2) and (d_3, d_4) with (d_1, d_4) and (d_3, d_2). Let v_i be the point of intersection closest to d_i where the adversarial edge incident on d_i crosses the boundary of D_p. There are two cases.

Case 1: First suppose that $\angle v_1 p' v_4 = \angle v_3 p' v_2 \leq \pi/2$, where p' is the point where the adversarial edges correspond to (d_1, d_2) and (d_3, d_4) cross; refer to Figure 4a. Note that we could delete the portions of adversarial edges (v_1, v_2), and (v_3, v_4) crossing D_p and replace these with the two portions of the circumference of D_p connecting points v_1, v_4 and v_2, v_3 (see Figure 4a). In deleting the portions of the adversarial edges which intersect the interior of D_p, we saved at least $4\sqrt{K^2 - 1}$. This value comes from the fact that the adversarial edge is contained within the fat edge connecting these two disks, which needs only "nearly" pass diametrically across D_p; it could be the case that p is contained within a fat edge on its boundary. In replacing the deleted portions of adversarial edges with two arcs comprising at most half of the circumference of D_p (with arc length at most πK) we still get an overall savings of at least $4\sqrt{K^2 - 1} - \pi K$. Thus, we need to choose K so that $4\sqrt{K^2 - 1} - \pi K \geq 9$ implying $K \geq 11 = O(1)$. We will show later that this savings of 9 units of tour length is more than enough to compensate for the adversarial increase in the new proposed ordering.

Case 2: Next, suppose that $\angle v_1 p' v_4 = \angle v_3 p' v_2 > \pi/2$ for any pair of fat edges still containing p. We will begin by breaking the plane into quadrants whose origin is p and now consider triples of fat edges that contain p. We will only consider those triples of fat edges whose disk endpoints lie in quadrants I, and III, as we can repeat this process a finite number of times, each with a new perturbed (rotated) set of quadrants who's origin is p so that eventually all remaining fat edges containing p have this property.

Let (d_1, d_2) be some remaining fat edge containing p whose disk endpoints are in quadrants I, and III. Let (d_3, d_4), (d_5, d_6) be the second and third fat edge respectively that contain p,

and have an endpoint in each of quadrants I and III, found in order by walking along the optimal tour from d_2 away from d_1. As in case 1, let v_i be the point of intersection of the adversarial edge emanating from disk d_i and the boundary of D_p. We have that all of the v_i are in quadrants I or III as well (see Figure 4b). Given that v_1, v_2 are in opposite quadrants, as well as points v_3, v_4, and v_5, v_6, a simple case analysis will show that we can delete two edges (v_i, v_{i+1}), (v_j, v_{j+1}) that cross the interior of D_p, and replace them with two arcs of D_p, lying strictly within quadrants I and III, which make up at most half the circumference of D_p, while preserving connectivity of the tour. This case analysis is independent of the specifics of which quadrant contains disk d_i, and only requires that each triple of edges we try to uncross go between opposite quadrants.

Thus, as in Case 1, we can argue that in replacing two edges crossing D_p (saving at least $4\sqrt{K^2-1}$ in length) and replacing these with the two arcs of D_p (which comprise at most half the circumference of D_p) we have a net savings of at least $4\sqrt{K^2-1} - \pi K$, which is at least 9 when $K \geq 11$.

Each round of uncrossing (Case 1 or Case 2) reduces the tour length by a positive amount and reduces the depth of p by at least one. Therefore, this process will terminate in a finite number of rounds. The number of fat edges containing p remaining after the process (Case 2) terminates will be at most (another) constant.

Finally, we argue that the constant 9 we save in tour length in each local uncrossing is enough to compensate for whatever global increase in adversarial tour length may occur due to the new proposed ordering (since the adversary gets to re-optimize his selection of points).

Again, consider an uncrossing of the original, hypothesized optimal tour, replacing (d_1, d_2) and (d_3, d_4) with (d_1, d_4) and (d_2, d_3). Let x, y, u, v be the (original) points adversarially chosen in disks d_1, d_2, d_3, d_4. After performing the uncrossing, we get a new tour, and thus the adversary gets to re-optimize by choosing a different set of points. From the adversarial property of the initial solution, we have that the initial paths from x to u and from v to y were as long as possible over the intermediate choice of disks if we fix points x, y, u, v. The new path chosen between disks d_1 and d_3 is at most that of the original path between x and u, plus two diameters, one per disk. That is, suppose the adversary chose new points x', u' in disks d_1, d_3 respectively. We can model the new path as traveling from x' to x in d_1 following the original path from x to u and then traveling from u to u' in d_3 costing at most two diameters. Similarly for the path between v, and y. Finally, in arguing about the additional length reconnecting the tour after the swap, we can upper bound, by triangle inequality, the length of the edge (x', v') and (u', y') as at most four diameters, one per disk $d_1, d_2, ..., d_4$ the portions of the edges (x, y) (u, v) strictly exterior to D_p as well as at most half the circumference of D_p. However, we have the savings of removing those portions of edges (x, y) and (u, v) that were strictly interior to D_p. Recall that the diameter of D_p was chosen such that removing two edges that pass "nearly" diametrically across D_p and replacing them with two arcs comprising at most half of its circumference results in a net savings of 9 units. Therefore in adding at most 8 diameters (or 8 units) upper bounding the adversarial increase, we still have a net savings of at least 1 unit. Thus, we have a strictly shorter adversarial tour after performing the uncrossing, thereby contradicting the optimality assumption of the original tour. ◀

4.2 The m-Guillotine Structure Theorem

We begin with some notation largely following [5, 12]. Let G be an embedded planar straight line graph (PSLG) with edge set E of total length L, and let $R = \{d_1, \ldots, d_n\}$ be a set of

disjoint unit-diameter disks d_i in the plane. (In our setting, there will be exactly one vertex of G within each disk $d_i \in R$.) Let $\mathcal{B}$ be an axis-aligned bounding square of R. We refer to an axis-aligned box $W \subset \mathcal{B}$ as a *window*, which will correspond to a particular subproblem of our dynamic program. We refer to an axis-parallel line ℓ that intersects window W as a *cut* of window W.

Consider a cut ℓ for window W; assume, without loss of generality, that ℓ is vertical. The intersection $\ell \cap (E \cap W)$ of ℓ with the edge set contained in W consists of a, possibly empty, set of subsegments (which include, as a degenerate case, singleton points) along ℓ. We let ξ be the number of endpoints of subsegments along ℓ, and let these endpoints along ℓ be denoted by $\beta_1, \beta_2, \ldots, \beta_\xi$ ordered by decreasing y coordinate. For a positive integer m we define the *m-span* $\sigma_m(\ell)$ of ℓ to be $\emptyset$ if $\xi \leq 2(m-1)$, and the possibly zero length segment $\beta_m, \beta_{\xi-m+1}$, joining the mth and the mth from the last endpoints along ℓ otherwise.

The intersection of $\ell \cap R \cap W$ consists of a possibly empty set of $\xi_R \leq |R \cap W|$ subsegments of ℓ, one subsegment for each disk (bounding box) intersected by $\ell \cap W$. Let these disk/boxes be $d_1, d_2, \ldots, d_{\xi_R}$ in order of decreasing y coordinate. For a positive integer m we define the *m-disk-span* $\sigma_{m,R}(\ell)$ of ℓ to be the (possibly empty) line segment joining the bottom endpoint of $d_m \cap \ell$ to the top endpoint of $d_{\xi_R - m+1} \cap \ell$. In fact, as observed in [14], it suffices to consider the m-disk-span of the set of axis-aligned bounding squares of the input disks, since the charging scheme charges the perimeters of the regions, which are, within a constant factor, the same whether we deal with circular disks or square (L_∞) disks.

As in [5] we define a line (cut) ℓ to be an *m-good cut* with respect to W if $\sigma_m(\ell) \subseteq E$ and $\sigma_{m,R} \subseteq E$. Finally, we say that *E satisfies the m-guillotine property with respect to W* if either (1) W does not fully contain any disk; or (2) there exists an m-good cut ℓ that splits W into W_1, and W_2 and, recursively, E satisfies the m-guillotine property with respect to W_1, and W_2. The following is shown in [5], using a variant of the charging scheme of [12]:

▶ **Theorem 10** ([5]). *Let G be an embedded connected planar graph with edge set E of total length L, and let R be a given set of pairwise-disjoint equal-radius disks (of radius δ) each of which intersects E. Assume that E and R are contained in the square $\mathcal{B}$. Then for any positive integer m there exists a connected planar graph G' that satisfies the m-guillotine property with respect to $\mathcal{B}$ and has edge set $E' \supseteq E$ of length $L' \leq (1 + O(1/m))L + O(\delta/m)$.*

In the constructive proof of Theorem 10, m-spans are added to E, whose lengths are charged off to a small fraction ($O(1/m)$) of the length L of E. Consider the edges of E that cross an m-span, ab that is added: By Theorem 9 we know that the associated fat edges (of width 1) have constant depth. This implies that the number of edges of E that cross an m-span, ab, that arises in the constructive proof of Theorem 10 is $O(|ab|)$.

▶ **Theorem 11.** *In the graph G' that is obtained from G according to Theorem 10, the segments of E' that arise as m-span edges for the input edges E are such that the number of edges of E intersecting an m-span edge ab is at most $O(|ab|)$.*

Provided that the input R is nontrivial ($n \geq 2$), the length L^* of an optimal solution OPT (path or cycle) to ATSP is at least 2; thus, Theorem 10 shows that there exists an m-guillotine supergraph of OPT of length $L' \leq (1 + O(1/m))L^*$. Further, as shown in [5, 12, 14], one can make the m-guillotine conversion using cuts whose coordinates are from among a discrete set of $O(n)$ candidate x- and y-coordinates, for fixed m. We will show how to use this fact, along with the structure of an optimal adversarial solution, to construct via dynamic programming an m-guillotine structure from which we can extract an approximation to OPT, with approximation factor $(1 + \epsilon)$, for any $\epsilon > 0$. (Here, $m = O(1/\epsilon)$.)

4.3 The Dynamic Program

A subproblem of our dynamic program (DP) is responsible for computing a shortest total length connected network that spans the input set R of disks (at their sample points) while satisfying a constant-size, $O(m)$, set of boundary conditions. The boundary conditions specify $O(m)$ disks that the subproblem is responsible for interconnecting, as well as conditions on how the computed network within this subproblem should interact with optimal solutions computed within abutting subproblems. As we cannot afford to keep track of all (potentially $\Omega(n)$) interconnections of the optimal ATSP solution, OPT, between two rectangles that bound subproblems, the m-guillotine structure theorem, together with our additional structural results, allow us to compactly summarize the interconnection information well enough to ensure approximation within factor $(1+\epsilon)$ of optimal.

Unlike the PTAS for TSPN, where the DP can choose any point within each region of R, in computing a minimum-weight connected, Eulerian, m-guillotine spanning subgraph over $\mathcal{G}$, in the ATSP we have no control over the point being spanned within each region: Once we produce an ordering σ_R, the adversary gets to solve an offline longest path problem to choose the ("worst possible") point $p_i \in d_i$ within each region $d_i \in R$ our tour must visit. Thus, we need to create a minimum weight connected spanning Eulerian subgraph over $\mathcal{G}$ that satisfies the m-guillotine property *and* satisfies a certain *adversarial subpath property*, which allows us to show that in the resulting network computed by DP, we can extract a polygonal tour of R that satisfies the adversarial property. In essence, we need the DP subproblems to be able to estimate (approximately) what the cost of an *adversarial* solution will be, if we extract from the optimized m-guillotine network a tour through R.

In particular, each DP subproblem is specified by a window $W \subseteq \mathcal{B}$, along with the following additional information:

1. An m-span (possibly empty) on each of the 4 sides of W, each with a parity bit indicating whether the number of edges incident to the m-span from outside of W is even or odd;
2. $O(m)$ *specified edges*, which are the network edges crossing the boundary of W that are not crossing one of the (up to 4) m-spans;
3. An m-disk span (possibly empty) on each of the 4 sides of W, with a specified sample point given for the first and for the last disk along the m-disk span;
4. $O(m)$ *specified input disks* (i.e., disks of R not intersecting an m-disk span) intersecting the boundary of W;
5. A specified sample point of $\mathcal{G}$ on the boundary of each of the $O(m)$ specified input disks, where the network is required to visit the associated disks (these are the "guessed" positions of the adversarial visitation points for the specified disks);
6. For each of the $O(m)$ specified input disks, we indicate whether the specified sample point of the disk is visited by the network being computed for the subproblem, and, if so, whether its degree in that network is 1 or 2. (The total degree of the sample point, using edges associated with subproblems on both sides of the cut, will be 2.)
7. An interconnection pattern specifying the subsets of the $O(m)$ boundary elements (specified input disks, specified edges, m-spans, and m-disk spans) that form connected components within W.

There are only $O(n^4)$ choices for W, $n^{O(m)}$ choices for the specified edges/disks, and a constant ($O(g(m))$, for some function g) number of choices of the $O(m)$ bits and the interconnection patterns. Thus, there are a polynomial number of subproblems for the DP.

A subproblem in the dynamic program requires one to compute a minimum-length m-guillotine network satisfying the following constraints:

- **(i)** The network is comprised of edges of the following types: (a). edges from the set $E_{\mathcal{G}}$ of edges linking a sample point of $\mathcal{G}$ on one disk to a sample point of $\mathcal{G}$ on another disk; (b). edges of type (a), $E_{\mathcal{G}}$, truncated at a (Steiner) attachment point on an m-span where the edge crosses the m-span or passes through an endpoint of the m-span; and (c) m-spans and m-disk spans that lie along cuts in the decomposition (recall that cuts lie along $O(n)$ discrete horizontal/vertical lines). The attachment points and the endpoints of m-spans and m-disk spans constitute a set, $\mathcal{H}$, of *Steiner points*, distinct from the sample points $\mathcal{G}$ on the boundaries of the disks.
- **(ii)** Each sample point of $\mathcal{G}$ within W that is visited by the network has degree 2.
- **(iii)** The number of edges of type (b) (i.e., edges of $E_{\mathcal{G}}$ truncated at an m-span) incident on an m-span segment ab is even or odd, according to whether the parity bit of the m-span is even or odd, so that the total sum of the degrees of the Steiner points $\mathcal{H}$ along an m-span is even. Further, the number of edges of type (b) incident on an m-span segment ab is bounded by $c_0 \cdot |ab|$, where c_0 is a constant arising from the structure Theorem 9.
- **(iv)** The network must be m-guillotine with respect to W, and, for each cut in the recursive partitioning of W, in the total length of the network we count each m-span twice; these doubled m-spans allow us to augment the resulting network to be Eulerian [12], and thereby to extract a tour (see below). Further, we count the length of each m-disk span a constant ($O(1)$) times as well; this will allow the m-disk spans to be converted into adversarial subpaths visiting the set of disks that are spanned.
- **(v)** The network must utilize the specified edges (which straddle the boundary of W).
- **(vi)** The network must visit, at a sample point, each of the input disks interior to W.
- **(vii)** The network must visit each specified disk whose bit indicates it should be visited by the subproblem, at the specified sample point for that disk. Further, the network must visit the specified sample points for the first and last disk associated with each nonempty m-disk span.
- **(viii)** The network must obey the interconnection pattern of the subproblem.
- **(ix)** The network obeys the *adversarial subpath property*: Any maximal path, endpoints non-inclusive, within the network that goes through only sample points $\mathcal{G}$ is a longest path through the sequence of disks on which the sample points lie (one per disk).

▶ **Lemma 12.** *When an optimal tour OPT is rounded to the grid $\mathcal{G}$ and then converted to become m-guillotine in the process that proves Theorem 10, the network that results from the augmentation of OPT satisfies conditions (i)-(viii) at every level of the recursive process, for appropriate choices of the specified edges, disks, and interconnection patterns.*

Proof. During the process that converts OPT to be m-guillotine, according to the constructive proof of Theorem 10, most of the conditions hold automatically, by construction. Edges of OPT that cross an m-span, ab, do so at a point of $\mathcal{H}$ that has degree 4 (since the crossing edge is partitioned at the crossing point, becoming two truncated type-(b) edges, and the m-span is partitioned at the crossing point as well). An m-span edge ab, by construction, extends between two points (a and b, each a Steiner point) on edges of OPT (each of which is thereby partitioned into two truncated type-(b) edges). Theorem 11 implies that the fat edges associated with the edges of OPT have bounded depth, implying condition (iii) holds. Condition (viii) holds for the choice of interconnection pattern that is implied by OPT. The adversarial subpath property holds because of the adversarial path property of OPT itself. ◀

We now discuss the enforcement of condition (ix), the adversarial subpath property, which is key to our being able to account for the adversary's choices during our optimization of the

network length, assuring that, in the end, we can extract from the computed network a tour that is adversarial and not much longer than the overall network.

Let (W, Σ) denote a subproblem associated with window W, where Σ is a specification of the boundary constraints information (1)-(7). The dynamic programming recursion optimizes the partition of the subproblem (W, Σ) into two subproblems, (W_1, Σ_1) and (W_2, Σ_2), by a horizontal or vertical cut line ℓ (intersecting W and passing through one of the $O(n)$ discrete values of x, y-coordinates that define windows). Crucial to the correctness of the algorithm is that this recursion preserves the properties specified by the conditions (i)-(ix).

The objective function, $f(W, \Sigma)$, measures the total length of the network restricted to the window W; in particular, edges of $E_{\mathcal{G}}$ that are specified in the boundary constraints Σ have their length partitioned and assigned to subproblems through which they pass.

The DP recursion optimizes over the choice of the cut line ℓ that partitions W into W_1 and W_2, as well as the boundary conditions, Σ_ℓ, along the cut, which will be part of the specifications Σ_1 and Σ_2. The conditions Σ_1 and Σ_2 must be compatible with each other and with the choice of boundary conditions, Σ_ℓ, across the cut ℓ. In particular, in order for Σ_1 and Σ_2 to be compatible with each other and with Σ, the specified edges of $E_{\mathcal{G}}$ across ℓ must match, as well as the m-span and m-disk span along the cut ℓ. Further, the interconnection pattern of Σ must specify subsets of boundary elements for W that are yielded by taking the union of interconnection patterns for (W_1, Σ_1) and (W_2, Σ_2).

We let $\Sigma_\ell^{(R)}$ denote the partial specification of the boundary conditions Σ_ℓ, in which we specify which pairs of disks from R constitute the specified edges crossing ℓ, but do not specify the actual sample points on the boundaries of these disks that define the endpoints of the edges from $E_{\mathcal{G}}$ being specified. (In other words, $\Sigma_\ell^{(R)}$ specifies only the equivalence classes of the full set of conditions, Σ_ℓ; the refinement of these equivalence classes will be specified in the optimization within the "max" term of the recursion below.) The DP recursion is

$$f(W, \Sigma) = \min_{\ell, \Sigma_\ell^{(R)}} \{ \max_{\Sigma_\ell \in X(\Sigma_\ell^{(R)})} (f(W_1(\ell), \Sigma_1(\Sigma_\ell)) + f(W_2(\ell), \Sigma_2(\Sigma_\ell))) \}$$

where the outer minimization is over choices of the cut ℓ and the cross-cut boundary conditions $\Sigma_\ell^{(R)}$, and the inner maximization is over choices of Σ_ℓ that are in the set $X(\Sigma_\ell^{(R)})$ of all boundary conditions across the cut ℓ that are refinements of the choice $\Sigma_\ell^{(R)}$, specifying precisely which sample points are utilized for each of the disks of R that are involved in the specification $\Sigma_\ell^{(R)}$ (and not already specified by the "parent" choice, Σ, in cases in which edges crossing ℓ also extend outside of W and have their sample points specified within Σ). In the expression above, $W_1(\ell)$ and $W_2(\ell)$ are the subwindows of W on either side of the cut ℓ, and $\Sigma_1(\Sigma_\ell)$ and $\Sigma_2(\Sigma_\ell)$ are the corresponding boundary conditions on either side of ℓ that are inherited from Σ and consistent with the conditions Σ_ℓ. The fact that we maximize over the choices that the adversary can make, in all choices that cross the cut ℓ, implies that we preserve the adversarial subpath property:

▶ **Lemma 13.** *The DP yields a network satisfying (ix), the adversarial subpath property.*

4.4 Extracting an Approximating ATSP Tour

The output of the DP algorithm is an m-guillotine network G of minimum cost, where cost is total length, taking into account that m-spans are counted twice, and m-disk spans are counted $O(1)$ times (and are each of length at least 1). From the structure Theorem 10, we know that the total length of edges of G is at most $|OPT|(1 + O(1/m))$.

The fact that we accounted for the doubling of the m-spans in the optimization implies that we can afford to augment the edges along each m-span, in order that every Steiner

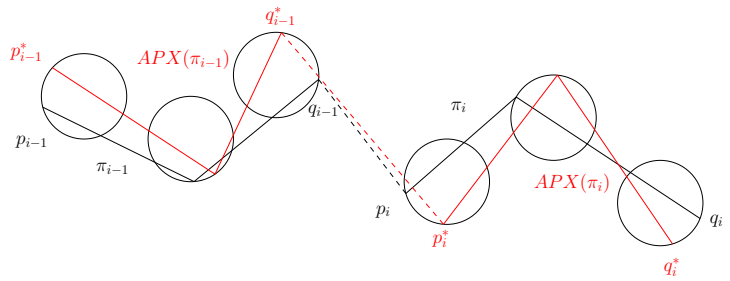


Figure 5 Bounding the adversarial increase over extracted approximate tour T'.

point along an m-span has degree 4: Initially, the points $\mathcal{H}$ along an m-span have degree 3, being either endpoints of the m-span (having a T-junction with an edge between two sample points of $\mathcal{G}$), or being a T-junction where an edge between two sample points of $\mathcal{G}$ is truncated, terminating on the m-span. By the parity condition at the m-span, we know that there are an even number of T-junctions along the m-span, implying that we can add a perfect matching of segments along the m-span, joining consecutive pairs of T-junctions. The total length of this matching is less than the length of the m-span, and is "paid for" by the doubling of the m-span lengths in the DP optimization.

The fact that we accounted for $O(1)$ copies of the m-disk spans in the optimization implies that we can afford to augment the edges of G with an adversarial path through the sequence of disks stabbed by the m-disk span; such a path has length proportional to the length of the m-disk span, assuming the m-disk span is nontrivial in length.

By the above discussion, the result of our algorithm is, then, a connected Eulerian network of length at most $|OPT|(1 + O(1/m))$. From this network, we extract a tour T. The tour T is a cycle consisting of straight line segments joining points that are either sample points, $\mathcal{G}$, or Steiner points, $\mathcal{H}$. Let $\pi_1, \pi_2, \ldots, \pi_k$ be the maximal subpaths along T whose vertices are all sample points $\mathcal{G}$; i.e., each path π_i has only vertices of $\mathcal{G}$ (no Steiner points $\mathcal{H}$), and every sample point of $\mathcal{G}$ that is a vertex of T lies in exactly one path π.

Now, the number, k, of subpaths is at most the number of Steiner points $\mathcal{H}$ along the tour T, and this number is upper bounded by the number of Steiner points along m-spans in the entire network. But, the total length of all m-spans is at most $O(1/m) \cdot |OPT|$, by the proof of Theorem 10. This implies that $k \leq O(1/m) \cdot |OPT|$.

The adversarial subpath property that was enforced in the dynamic programming algorithm implies that the subpaths π_i are each adversarial – their lengths are longest possible, for the given sequence of disks through which it passes (given that the path π_i begins at sample point p_i and ends at sample point q_i as chosen by our dynamic program). We obtain a new tour, T', by chaining together the subpaths π_i, omitting any Steiner points that were along T. The resulting tour T' is not necessarily adversarial, but the following lemma shows that it is close to being so.

▶ **Lemma 14.** *Let σ'_R be the order in which the disks R are visited by the tour T'. Then, the adversarial tour, APX, associated with σ'_R has length at most $|T'| + O(1/m) \cdot |OPT|$.*

Proof. For each subpath π_i in our approximate tour T' let $APX(\pi_i)$ be the adversarial path computed over the sequence of disks associated with π_i in the final adversarial tour associated with σ'_R. Similarly let p^*_i (resp., q^*_i) be the point chosen in the first (resp., last) disk along π_i in APX. Assume that we have chosen the points p_i (resp., q_i) in the first (resp., last) disk along π_i in our extracted tour T'; see Figure 5 for an illustration.

From the adversarial property of our computed solution we know that the length of the path $|\pi_i|$ computed from p_i to q_i is as long as possible over choices in intermediate disks. Therefore we can over estimate the length of APX by walking around T' adding at most two unit diameter detours to the first and last disk in each sub-path π_i. That is we can begin at p_i detour to p_i^* and back, follow π_i until we reach q_i an then detour from q_i to q_i^* and back and follow T' to p_{i+1} and so on. By triangle inequality and the fact that π_i is a longest path for fixed choices of p_i, and q_i this over estimates the length of APX.

We have $|APX| \leq |T'| + 4k$, and therefore $|APX| \leq |T'| + O(1/m)|OPT|$, because, as previously stated $k \leq O(1/m)|OPT|$. ◀

Since we know that the computed T, and thus T', has length at most $|OPT|(1+O(1/m))$, Lemma 14 implies that the overall solution extracted from our computed tour yields a PTAS.

References

1. Esther M. Arkin and Refael Hassin. Approximation algorithms for the geometric covering salesman problem. *Discrete Applied Mathematics*, 55(3):197–218, 1994.
2. Dimitris J. Bertsimas, Patrick Jaillet, and Amedeo R. Odoni. A priori optimization. *Operations Research*, 38(6):1019–1033, 1990.
3. Gui Citovsky, Tyler Mayer, and Joseph S. B. Mitchell. TSP With Locational Uncertainty: The Adversarial Model, March 2017. arXiv:1705.06180 [cs.CG]. URL: `https://arxiv.org/abs/1705.06180`.
4. Reza Dorrigiv, Robert Fraser, Meng He, Shahin Kamali, Akitoshi Kawamura, Alejandro López-Ortiz, and Diego Seco. On minimum-and maximum-weight minimum spanning trees with neighborhoods. *Theory of Computing Systems*, 56(1):220–250, 2015.
5. Adrian Dumitrescu and Joseph S. B. Mitchell. Approximation algorithms for TSP with neighborhoods in the plane. *Journal of Algorithms*, 48:135–159, 2003. Special issue devoted to 12th ACM-SIAM Symposium on Discrete Algorithms, Washington, DC, January, 2001.
6. Robert Fraser. *Algorithms for geometric covering and piercing problems.* PhD thesis, University of Waterloo, 2012.
7. Patrick Jaillet. A priori solution of a traveling salesman problem in which a random subset of the customers are visited. *Operations Research*, 36(6):929–936, 1988.
8. Lujun Jia, Guolong Lin, Guevara Noubir, Rajmohan Rajaraman, and Ravi Sundaram. Universal approximations for TSP, Steiner tree, and set cover. In *Proc. 37th ACM Symposium on Theory of Computing*, pages 386–395. ACM, 2005.
9. Pegah Kamousi and Subhash Suri. Euclidean traveling salesman tours through stochastic neighborhoods. In *International Symposium on Algorithms and Computation*, pages 644–654. Springer, 2013.
10. Chih-Hung Liu and Sandro Montanari. Minimizing the diameter of a spanning tree for imprecise points. In *International Symposium on Algorithms and Computation*, pages 381–392. Springer, 2015.
11. Maarten Löffler and Marc van Kreveld. Largest and smallest convex hulls for imprecise points. *Algorithmica*, 56(2):235–269, 2010.
12. 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.
13. Joseph S. B. Mitchell. Shortest paths and networks. In Jacob E. Goodman and Joseph O'Rourke, editors, *Handbook of Discrete and Computational Geometry (2nd Edition)*, chapter 27, pages 607–641. Chapman & Hall/CRC, Boca Raton, FL, 2004.

14 Joseph S. B. Mitchell. A PTAS for TSP with neighborhoods among fat regions in the plane. In *Proc. 18th ACM-SIAM Symposium on Discrete algorithms*, pages 11–18. Society for Industrial and Applied Mathematics, 2007. URL: `http://www.ams.sunysb.edu/~jsbm/papers/tspn-soda07-rev.pdf`.

15 Sandro Montanari. *Computing routes and trees under uncertainty.* PhD thesis, Dissertation, ETH-Zürich, 2015, No. 23042, 2015.

16 Yang Yang, Mingen Lin, Jinhui Xu, and Yulai Xie. Minimum spanning tree with neighborhoods. In *International Conference on Algorithmic Applications in Management*, pages 306–316. Springer, 2007.

On Planar Greedy Drawings of 3-Connected Planar Graphs*

Giordano Da Lozzo[1], Anthony D'Angelo[2], and Fabrizio Frati[3]

1 **Department of Computer Science, University of California, Irvine, CA, USA**
gdalozzo@uci.edu
2 **School of Computer Science, Carleton University, Ottawa, Canada**
anthonydangelo@cmail.carleton.ca
3 **Department of Engineering, Roma Tre University, Rome, Italy**
frati@dia.uniroma3.it

Abstract

A graph drawing is *greedy* if, for every ordered pair of vertices (x, y), there is a path from x to y such that the Euclidean distance to y decreases monotonically at every vertex of the path. Greedy drawings support a simple geometric routing scheme, in which any node that has to send a packet to a destination "greedily" forwards the packet to any neighbor that is closer to the destination than itself, according to the Euclidean distance in the drawing. In a greedy drawing such a neighbor always exists and hence this routing scheme is guaranteed to succeed.

In 2004 Papadimitriou and Ratajczak stated two conjectures related to greedy drawings. The *greedy embedding conjecture* states that every 3-connected planar graph admits a greedy drawing. The *convex greedy embedding conjecture* asserts that every 3-connected planar graph admits a planar greedy drawing in which the faces are delimited by convex polygons. In 2008 the greedy embedding conjecture was settled in the positive by Leighton and Moitra.

In this paper we prove that every 3-connected planar graph admits a *planar* greedy drawing. Apart from being a strengthening of Leighton and Moitra's result, this theorem constitutes a natural intermediate step towards a proof of the convex greedy embedding conjecture.

1998 ACM Subject Classification G.2.2 Graph Theory

Keywords and phrases Greedy drawings, 3-connectivity, planar graphs, convex drawings

Digital Object Identifier 10.4230/LIPIcs.SoCG.2017.33

1 Introduction

Geographic routing is a family of routing protocols for *ad-hoc networks*, which are networks with no fixed infrastructure – such as routers or access points – and with dynamic topology [15, 27, 28]. In a geographic routing scheme each node of the network actively sends, forwards, and receives packets; further, it does so by only relying on the knowledge of its own geographic coordinates, of those of its neighbors, and of those of the packet destination. *Greedy routing* is the simplest and most renowned geographic routing scheme. In this protocol, a node that has to send a packet simply forwards it to any neighbor that is closer – according to the Euclidean distance – to the destination than itself. The greedy routing scheme might fail

* This article reports on work supported by the U.S. Defense Advanced Research Projects Agency (DARPA) under agreement no. AFRL FA8750-15-2-0092. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government. This research was also partially supported by NSERC, by MIUR-PRIN Project 20157EFM5C – "MODE", and by H2020-MSCA-RISE Project 734922 – "CONNECT".

to deliver packets because of the presence of a *void* in the network; this is a node with no neighbor closer to the destination than itself. For this reason, several variations of the greedy routing scheme have been proposed; see, e.g., [6, 19, 20].

Apart from its failure in the presence of voids, the greedy routing protocol has two disadvantages which limit its applicability. First, in order for the protocol to work, each node of the network has to be equipped with a GPS, which might be expensive and might consume excessive energy. Second, two nodes that are close geographically might be unable to communicate with each other because of the presence of topological obstructions. Rao *et al.* [26] introduced the following brilliant idea for extending the applicability of geographic routing in order to overcome the above issues. Suppose that a network topology is known; then one can assign *virtual coordinates* to the nodes and use these coordinates instead of the geographic locations of the nodes in the greedy routing protocol. The virtual coordinates can then be chosen so that the greedy routing protocol is guaranteed to succeed.

Assigning the virtual coordinate to the nodes of a network corresponds to the following graph drawing problem: given a graph G, construct a *greedy drawing* of G, that is a drawing in the plane such that, for any ordered pair of vertices (x, y), there is a neighbor of x in G that is closer – in terms of Euclidean distance – to y than x. Equivalently, a greedy drawing of G is such that, for any ordered pair of vertices (x, y), there is a *distance-decreasing* path from x to y, that is, a path $(u_1, \ldots, u_m)$ in G such that $x = u_1$, $y = u_m$, and the Euclidean distance between u_{i+1} and u_m is smaller than the one between u_i and u_m, for any $i = 1, \ldots, m-2$.

Greedy drawings experienced a dramatical surge of popularity in the theory community in 2004, when Papadimitriou and Ratajczak [24] proposed the following two conjectures about greedy drawings of 3-connected planar graphs (the convex greedy embedding conjecture has not been stated in the journal version [25] of their paper [24]).

▶ **Conjecture 1** (Greedy embedding conjecture). *Every 3-connected planar graph admits a greedy drawing.*

▶ **Conjecture 2** (Convex greedy embedding conjecture). *Every 3-connected planar graph admits a convex greedy drawing.*

Papadimitriou and Ratajczak [24, 25] provided several reasons why 3-connected planar graphs are central to the study of greedy drawings. First, there exist non-3-connected planar graphs and 3-connected non-planar graphs that do not admit any greedy drawing. Thus, the 3-connected planar graphs form the largest class of graphs that might admit a greedy drawing, in a sense. Second, all the 3-connected graphs with no $K_{3,3}$-minor admit a 3-connected planar spanning graph, hence they admit a greedy drawing, provided the truth of the greedy embedding conjecture. Third, the preliminary study of Papadimitriou and Ratajczak [24, 25] provided evidence for the mathematical depth of their conjectures.

Leighton and Moitra [21] (and, independently and slightly later, Angelini et al. [4]) settled Conjecture 1 in the affirmative. In this paper we show the following result.

▶ **Theorem 3.** *Every 3-connected planar graph admits a planar greedy drawing.*

Given a 3-connected planar graph G, the algorithms in [4, 21] find a spanning subgraph S of G and construct a (planar) greedy drawing of S; then they embed the edges of G not in S as straight-line segments obtaining a, in general, *non-planar* greedy drawing of G. Thus, Theorem 3 strengthens Leighton and Moitra's and Angelini et al.'s results. Furthermore, *convex* drawings, in which all the faces are delimited by convex polygons, are planar, hence Theorem 3 provides a natural step towards a proof of Conjecture 2.

Our proof employs a structural decomposition for 3-connected planar graphs which finds its origins in a paper by Chen and Yu [7]. This decomposition actually works for a super-class of the 3-connected planar graphs known as *strong circuit graphs*. We construct a planar greedy drawing of a given strong circuit graph G recursively: we apply the structural decomposition to G in order to obtain some smaller strong circuit graphs, we recursively construct planar greedy drawings for them, and then we suitably arrange these drawings together to get a planar greedy drawing of G. For this arrangement to be feasible, we need to ensure that the drawings we construct satisfy some restrictive geometric requirements; these are described in the main technical theorem of the paper – Theorem 8.

Related results. Planar greedy drawings always exist for *maximal planar graphs* [11]. Further, every planar graph G with a *Hamiltonian path* $P = (u_1, \dots, u_n)$ has a planar greedy drawing. Namely, construct a planar straight-line drawing Γ of G such that $y(u_1) < \cdots < y(u_n)$; such a drawing always exists [12]; scale Γ down horizontally, so that P is "almost vertical". Then, for any $1 \leq i < j \leq n$, the paths $(u_i, u_{i+1} \dots, u_j)$ and $(u_j, u_{j-1} \dots, u_i)$ are distance-decreasing. A characterization of the *trees* that admit a (planar) greedy drawing is known [22]; indeed, a greedy drawing of a tree is always planar [2].

Algorithms have been designed to construct *succinct* greedy drawings, in which the vertex coordinates are represented with a polylogarithmic number of bits [13, 16, 17]; this has been achieved by allowing the embedding space to be different from the Euclidean plane or the metric to be different from the Euclidean distance.

Planar drawings have been studied in which paths between pairs of vertices are required to exist satisfying properties other than being distance-decreasing. We say that a path $P = (u_1, \dots, u_m)$ in a graph drawing is *self-approaching* [1, 23] if, for any three points a, b, c in this order along P from u_1 to u_m, the Euclidean distance between a and c is larger than the one between b and c – then a self-approaching path is also distance-decreasing. We say that P is *increasing-chord* [1, 10, 23] if it is self-approaching in both directions. We say that P is *strongly monotone* [3, 14, 18] if the orthogonal projections of the vertices of P on the line ℓ through u_1 and u_m appear in the order $u_1, \dots, u_m$. It has been recently proved [14] that every 3-connected planar graph has a planar drawing in which every pair of vertices is connected by a strongly monotone path.

Because of space limitations some proofs are sketched or omitted; they can be found in the complete version of the paper [8].

2 Preliminaries

In this section we introduce some preliminaries.

Subgraphs and connectivity. We denote by $V(G)$ and $E(G)$ the vertex and edge sets of a graph G, respectively. For $U \subseteq V(G)$, we denote by $G - U$ the graph obtained from G by removing the vertices in U and their incident edges. Further, if $e \in E(G)$, we denote by $G - e$ the graph obtained from G by removing the edge e. Let H be a subgraph of G. An *H-bridge* B of G is either an edge of G not in H with both the end-vertices in H (then B is a *trivial H-bridge*), or a connected component of $G - V(H)$ together with the edges from that component to the vertices in $V(H)$ (then B is a *non-trivial H-bridge*); the vertices in $V(H) \cap V(B)$ are the *attachments* of B in H. For a vertex $v \in V(G) - V(H)$, we denote by $H \cup \{v\}$ the subgraph of G composed of H and of the isolated vertex v.

A *vertex k-cut* (in the following simply called *k-cut*) in a connected graph G is a set of k vertices whose removal disconnects G. For $k \geq 2$, a connected graph is *k-connected* if it has no $(k-1)$-cut. A *k-connected component* of a graph G is a maximal k-connected subgraph of G. Given a 2-cut $\{a, b\}$ in a 2-connected graph G, an *$\{a, b\}$-component* is either the edge ab (then the $\{a, b\}$-component is *trivial*) or a subgraph of G induced by a, b, and the vertices of a connected component of $G - \{a, b\}$ (then the $\{a, b\}$-component is *non-trivial*).

Plane graphs and embeddings. A drawing of a graph is *planar* if no two edges cross. A *plane graph* is a planar graph with a plane embedding; a *plane embedding* of a connected planar graph G is an equivalence class of planar drawings of G, where two drawings Γ_1 and Γ_2 are *equivalent* if: (i) the clockwise order of the edges incident to each vertex $v \in V(G)$ coincides in Γ_1 and Γ_2; and (ii) the clockwise order of the edges of the walks delimiting the outer faces of Γ_1 and Γ_2 is the same. When we talk about a planar drawing of a plane graph G, we always mean that it respects the plane embedding of G. We assume that any subgraph H of G is associated with the plane embedding obtained from the one of G by deleting the vertices and edges not in H. A vertex of G is *external* or *internal* depending on whether it is or it is not incident to the outer face of G, respectively. For two external vertices u and v of a 2-connected plane graph G, let $\tau_{uv}(G)$ and $\beta_{uv}(G)$ be the paths composed of the vertices and edges encountered when walking along the boundary of the outer face of G in clockwise and counter-clockwise direction from u to v, respectively. Note that $\tau_{uv}(G)$ and $\beta_{vu}(G)$ have the same vertices and edges, however in reverse linear orders.

Geometry. In this paper every angle is measured in radians. The *slope of a half-line ℓ* is defined as follows. Denote by p the starting point of ℓ and let ℓ' be the vertical half-line starting at p and directed towards decreasing y-coordinates. Then the slope of ℓ is the angle spanned by a counter-clockwise rotation around p bringing ℓ' to coincide with ℓ, minus $\frac{\pi}{2}$. Because of this definition, the slope of any half-line is between $-\frac{\pi}{2}$ (included) and $\frac{3\pi}{2}$ (excluded); in the following there will be very few exceptions to this assumption, which will be however evident from the text. Every angle expressed as $\arctan(\cdot)$ is between $-\frac{\pi}{2}$ and $\frac{\pi}{2}$. We define the *slope of an edge uv* in a graph drawing as the slope of the half-line from u through v. Then the slope of an edge uv is equal to the slope of the edge vu plus or minus π. For a directed line ℓ, we let its slope be equal to the slope of any half-line starting at a point of ℓ and directed as ℓ. We denote by Δpqr a triangle with vertices p, q, r, and we denote by $\measuredangle pqr$ the angle of Δpqr incident to q; note that $\measuredangle pqr$ is between 0 and π.

Let Γ be a drawing of a graph G and let $u, v \in V(G)$. We denote by $d(\Gamma, uv)$ the Euclidean distance between the points representing u and v in Γ. We also denote by $d_V(\Gamma, uv)$ the vertical distance between u and v in Γ, that is, $d_V(\Gamma, uv) = |y(u) - y(v)|$, where the y-coordinates of u and v are taken from Γ; the horizontal distance $d_H(\Gamma, uv)$ between u and v in Γ is defined analogously. With a slight abuse of notation, we will use $d(\Gamma, pq)$, $d_H(\Gamma, pq)$, and $d_V(\Gamma, pq)$ even if p and q are points in the plane (and not vertices of G). A *straight-line* drawing of a graph is such that each edge is represented by a straight-line segment.

The following lemma argues that the planarity and the greediness of a drawing are not lost as a consequence of any sufficiently small perturbation of the vertex positions.

▶ **Lemma 4.** *Let Γ be a planar straight-line drawing of a graph G. There is a value $\varepsilon^*_\Gamma > 0$ such that the following holds. Let Γ' be any straight-line drawing in which, for every vertex $z \in V(G)$, the Euclidean distance between the positions of z in Γ and Γ' is at most ε^*_Γ; then Γ' is planar and any path which is distance-decreasing in Γ is also distance-decreasing in Γ'.*

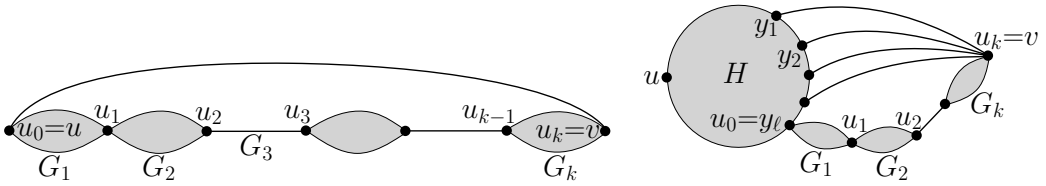


Figure 1 (a) Structure of (G, u, v) in Case A. (b) Structure of (G, u, v) in Case B.

3 Proof of Theorem 3

In this section we prove Theorem 3. Throughout the section, we will work with *plane graphs*. Further, we will deal with a class of graphs, known as *strong circuit graphs* [7], that is wider than 3-connected planar graphs. Strong circuit graphs constitute a subclass of the well-known *circuit graphs*, whose definition is due to Barnette and dates back to 1966 [5]. Here we rephrase the definition of strong circuit graphs as follows.

▶ **Definition 5.** A *strong circuit graph* is a triple (G, u, v) such that either: (i) G is an edge uv or (ii) $|V(G)| \geq 3$ and the following properties are satisfied.

- **(a)** G is a 2-connected plane graph;
- **(b)** u and v are two distinct external vertices of G;
- **(c)** if edge uv exists, then it coincides with the path $\tau_{uv}(G)$; and
- **(d)** for every 2-cut $\{a, b\}$ of G we have that a and b are external vertices of G and at least one of them is an internal vertex of the path $\beta_{uv}(G)$; further, every non-trivial $\{a, b\}$-component of G contains an external vertex of G different from a and b.

Several problems are easier to solve on (strong) circuit graphs than on 3-connected planar graphs. Indeed, (strong) circuit graphs can be easily decomposed into smaller (strong) circuit graphs and hence are suitable for inductive proofs. We now present a structural decomposition for strong circuit graphs whose main ideas can be found in a paper by Chen and Yu [7] (see [9] for an application of this decomposition to *cubic* strong circuit graphs).

Consider a strong circuit graph (G, u, v) such that G is not a single edge. The decomposition distinguishes the case in which the path $\tau_{uv}(G)$ coincides with the edge uv (Case A) from the case in which it does not (Case B).

▶ **Lemma 6.** *Suppose that we are in Case A (refer to Fig. 1(a)). Then the graph $G' = G - uv$ consists of a sequence of graphs $G_1, \ldots, G_k$, with $k \geq 1$, such that:*

- *6a: for $i = 1, \ldots, k-1$, the graphs G_i and G_{i+1} share a single vertex u_i; further, G_i is in the outer face of G_{i+1} and vice versa in the plane embedding of G;*
- *6b: for $1 \leq i, j \leq k$ with $j \geq i+2$, the graphs G_i and G_j do not share any vertex; and*
- *6c: for $i = 1, \ldots, k$ with $u_0 = u$ and $u_k = v$, (G_i, u_{i-1}, u_i) is a strong circuit graph.*

Given a strong circuit graph (G, u, v) that is not a single edge, the vertex u belongs to one 2-connected component of the graph $G - \{v\}$. Indeed, if it belonged to more than one 2-connected component of $G - \{v\}$, then $\{u\}$ would be a 1-cut of $G - \{v\}$, hence $\{u, v\}$ would be a 2-cut of G, which contradicts Property (d) for (G, u, v). We now present the following.

▶ **Lemma 7.** *Suppose that we are in Case B (refer to Fig. 1(b)). Let H be the 2-connected component of the graph $G - \{v\}$ that contains u; then we have $|V(H)| \geq 3$. Let $H' := H \cup \{v\}$. Then G contains ℓ distinct H'-bridges $B_1, \ldots, B_\ell$, for some $\ell \geq 2$, such that:*

- *7a: each H'-bridge B_i has two attachments, namely v and a vertex $y_i \in V(H)$;*
- *7b: the H'-bridges $B_1, \ldots, B_{\ell-1}$ are trivial, while B_ℓ might be trivial or not;*
- *7c: any two among $y_1, \ldots, y_\ell$ are distinct except, possibly, for $y_{\ell-1}$ and y_ℓ; also if $\ell = 2$, then y_1 and y_2 are distinct;*
- *7d: y_1 is an internal vertex of $\tau_{uv}(G)$; further, B_1 is an edge that coincides with $\tau_{y_1 v}(G)$;*
- *7e: y_ℓ is an internal vertex of $\beta_{uv}(G)$ and $\beta_{uy_1}(H)$; further, B_ℓ contains the path $\beta_{y_\ell v}(G)$;*
- *7f: $B_1, \ldots, B_{\ell-1}$ appear in this counter-clockwise order around v and lie in the outer face of B_ℓ in the plane embedding of G;*
- *7g: the triple (H, u, y_1) is a strong circuit graph; and*
- *7h: B_ℓ consists of a sequence of graphs $G_1, \ldots, G_k$, with $k \geq 1$, such that:*
  - *for $i = 1, \ldots, k-1$, the graphs G_i and G_{i+1} share a single vertex u_i; further, G_i is in the outer face of G_{i+1} and vice versa in the plane embedding of G;*
  - *for $1 \leq i, j \leq k$ with $j \geq i+2$, the graphs G_i and G_j do not share any vertex; and*
  - *for $i = 1, \ldots, k$ with $u_0{=}y_\ell$ and $u_k{=}v$, the triple (G_i, u_{i-1}, u_i) is a strong circuit graph.*

We prove that any strong circuit graph (G, u, v) has a planar greedy drawing by exploiting Lemmata 6 and 7 in a natural way. Indeed, if we are in Case A (in Case B) then Lemma 6 (resp. Lemma 7) is applied in order to construct strong circuit graphs (G_i, u_{i-1}, u_i) with $i = 1, \ldots, k$ (resp. strong circuit graphs (H, u, y_1) and (G_i, u_{i-1}, u_i) with $i = 1, \ldots, k$) for which planar greedy drawings are inductively constructed and combined together in order to get a planar greedy drawing of (G, u, v). The base case of the induction is the one in which G is an edge; then a planar greedy drawing of G is directly constructed. In order to be able to combine the planar greedy drawings for the strong circuit graphs (G_i, u_{i-1}, u_i) (and (H, u, y_1) if we are in Case B) to construct a planar greedy drawing of (G, u, v), we need the inductively constructed drawings to satisfy some restrictive geometric requirements. These are expressed in the following theorem, which is the core of the proof of Theorem 3.

▶ **Theorem 8.** *Let (G, u, v) be a strong circuit graph with at least three vertices and let $0 < \alpha < \frac{\pi}{4}$ be an arbitrary parameter. Let $\beta_{uv}(G) = (u = b_1, b_2, \ldots, b_m = v)$. There exists a straight-line drawing Γ of G in the Cartesian plane such that the following holds. For any value $\delta \geq 0$, denote by Γ_δ the straight-line drawing obtained from Γ by moving the position of vertex u by δ units to the left. Then Γ_δ satisfies the following properties (refer to Fig. 2).*

1. *Γ_δ is planar;*
2. *$\tau_{uv}(G)$ lies entirely on a horizontal line ℓ_u with u to the left of v;*
3. *the edge $b_1 b_2$ has slope in the interval $(-\alpha, 0)$ and the edge $b_i b_{i+1}$ has slope in the interval $(0, \alpha)$, for each $i = 2, 3, \ldots, m-1$;*
4. *for every vertex $x \in V(G)$ there is a path $P_x = (x = v_1, v_2, \ldots, v_p = v)$ from x to v in G such that the edge $v_i v_{i+1}$ has slope in the interval $(-\alpha, \alpha)$ in Γ_δ, for each $i = 1, 2, \ldots, p-1$; further, if $x \neq u$, then $u \notin V(P_x)$;*
5. *for every vertex $x \in V(G)$ there is a path $Q_x = (x = w_1, w_2, \ldots, w_q = u)$ from x to u in G such that the edge $w_i w_{i+1}$ has slope in the interval $(\pi - \alpha, \pi + \alpha)$ in Γ_δ, for each $i = 1, 2, \ldots, q-1$; and*
6. *for every ordered pair of vertices (x, y) in $V(G)$ there is a path P_{xy} from x to y in G such that P_{xy} is distance-decreasing in Γ_δ; further, if $x \neq u$ and $y \neq u$, then $u \notin V(P_{xy})$.*

We comment on the statement of Theorem 8. First, let us set $\delta = 0$ and argue about $\Gamma_0 = \Gamma$. Properties 1 and 6 are those that one would expect, as they state that Γ is planar and greedy, respectively. Properties 2 and 3 state that all the edges incident to the outer face of Γ are "almost" horizontal; indeed, the edges of $\tau_{uv}(G)$ are horizontal (this does not

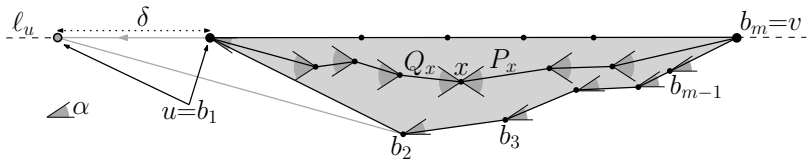


Figure 2 Illustration for the statement of Theorem 8.

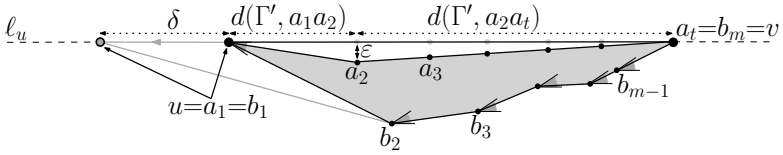


Figure 3 The straight-line drawing Γ of G in Case A if $k = 1$.

compromise the planarity of Γ since G does not contain edges between non-consecutive vertices of $\tau_{uv}(G)$, by Property (d) of (G, u, v)), the edge b_1b_2 has a slightly negative slope, and all the other edges of $\beta_{uv}(G)$ have slightly positive slopes. Then the planarity of Γ implies that Γ is contained in a wedge delimited by two half-lines with slopes 0 and $-\alpha$ starting at u. Properties 4 and 5 argue about the existence of certain paths from any vertex to u and v; these two vertices play an important role in the structural decomposition we employ, since distinct subgraphs are joined on those vertices, and the paths incident to them are inductively combined together in order to construct distance-decreasing paths. Finally, all these properties still hold if u is moved by an arbitrary non-negative amount δ to the left. This is an important feature we exploit in one of our inductive cases.

We now present an inductive proof of Theorem 8. In the **Base Case** the graph G is a single edge. We remark that, although Theorem 8 assumes that the given graph has at least three vertices, for its proof we need to inductively draw certain subgraphs of it; these subgraphs might indeed be single edges. Whenever we need to draw a strong circuit graph (G, u, v) such that G is a single edge uv, we draw it as a horizontal straight-line segment with positive length, with u to the left of v. We remark that, since Theorem 8 assumes that $|V(G)| \geq 3$, we do not need the constructed drawing to satisfy Properties 1–6.

We now discuss the inductive cases. In Case A the path $\tau_{uv}(G)$ coincides with the edge uv, while in Case B it does not. We discuss **Case A** first. Let $G' = G - uv$, where G' consists of a sequence of graphs $G_1, \ldots, G_k$, with $k \geq 1$, satisfying the properties described in Lemma 6. Our construction is different if $k = 1$ and $k \geq 2$.

Suppose first that $\mathbf{k = 1}$; by Lemma 6 the triple $(G' = G_1, u, v)$ is a strong circuit graph (and G_1 is not a single edge, as otherwise we would be in the Base Case). Inductively construct a straight-line drawing Γ' of G' with $\frac{\alpha}{2}$ as a parameter. By Property 2 the path $\tau_{uv}(G') = (u = a_1, \ldots, a_t = v)$ lies on a horizontal line ℓ_u in Γ' with u to the left of v. Let $Y > 0$ be the minimum distance in Γ' of any vertex strictly below ℓ_u from ℓ_u. Let

$$\varepsilon = \frac{1}{2}\min\{\varepsilon^*_{\Gamma'}, Y, \tan(\alpha) \cdot d(\Gamma', a_1a_2), \tan(\alpha) \cdot d(\Gamma', a_2a_t)\}.$$

We construct a straight-line drawing Γ of G from Γ' as follows; refer to Fig. 3. Decrease the y-coordinate of the vertex a_2 by ε; for $i = 3, \ldots, t-1$, decrease the y-coordinate of the vertex a_i so that it ends up on the straight-line segment $\overline{a_2a_t}$. Draw the edge uv as a straight-line segment. We have the following.

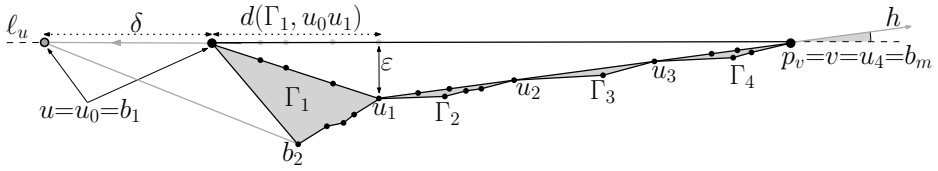


Figure 4 The straight-line drawing Γ of G in Case A if $k \geq 2$. In this example $k = 4$. The gray angle in the drawing is $\frac{\alpha}{2}$.

▶ **Lemma 9.** *For any $\delta \geq 0$, the drawing Γ_δ constructed in Case A if $k = 1$ satisfies Properties 1–6 of Theorem 8.*

Proof Sketch. The planarity of Γ is established due to the inequality $\varepsilon < \varepsilon^*_{\Gamma'}$ and to Lemma 4. Since Γ and Γ_δ coincide, except for the position of u, every crossing in Γ_δ has to involve edges incident to u. The proof that in fact there are no such crossings relies on the fact that Γ'_δ is planar, by induction, and on the inequalities $\varepsilon < \varepsilon^*_{\Gamma'}$ and $\varepsilon < Y$.

The paths P_x and Q_x requested for Properties 4 and 5 are obtained by suitably modifying paths satisfying the same properties for (G', u, v). The paths P_x and Q_x might contain edges in $\tau_{uv}(G')$; however, the slopes of the edges are in the required interval, which is $(-\alpha, \alpha)$ or $(\pi - \alpha, \pi + \alpha)$ depending on whether these edges are traversed towards v or u, respectively. This is due to the inequalities $\varepsilon < \tan(\alpha) \cdot d(\Gamma', a_1a_2)$ and $\varepsilon < \tan(\alpha) \cdot d(\Gamma', a_2a_t)$. ◀

We now discuss the case in which $\mathbf{k \geq 2}$. Refer to Fig. 4. By Lemma 6, for $i = 1, \ldots, k$, the triple (G_i, u_{i-1}, u_i) is a strong circuit graph, where $u_0 = u$, $u_k = v$, and u_i is the only vertex shared by G_i and G_{i+1}, for $i = 1, \ldots, k-1$.

If G_1 is a single edge, then inductively construct a straight-line drawing Γ_1 of G_1 and define $\varepsilon = \frac{1}{2}\min\{\varepsilon^*_{\Gamma_1}, \tan(\alpha) \cdot d(\Gamma_1, u_0u_1)\}$. If G_1 is not a single edge, then inductively construct a straight-line drawing Γ_1 of G_1 with $\frac{\alpha}{2}$ as a parameter. By Property 2 of Γ_1, the path $\tau_{u_0u_1}(G_1)$ lies on a horizontal line ℓ_u. Let $Y > 0$ be the minimum distance in Γ_1 of any vertex strictly below ℓ_u from ℓ_u. Let $\varepsilon = \frac{1}{2}\min\{\varepsilon^*_{\Gamma_1}, Y, \tan(\alpha) \cdot d(\Gamma_1, u_0u_1)\}$. In both cases, decrease the y-coordinate of u_1 by ε. Further, decrease the y-coordinate of every internal vertex of the path $\tau_{u_0u_1}(G_1)$, if any, so that it ends up on the straight-line segment $\overline{u_0u_1}$.

Now consider a half-line h with slope $s = \frac{\alpha}{2}$ starting at u_1. Denote by p_v the point at which h intersects the horizontal line ℓ_u through u. For $i = 2, \ldots, k$, inductively construct a straight-line drawing Γ_i of G_i with $\frac{\alpha}{3}$ as a parameter (if G_i is a single edge, then the parameter does not matter). Uniformly scale the drawings $\Gamma_2, \ldots, \Gamma_k$ so that the Euclidean distance between u_{i-1} and u_i in Γ_i is equal to $\frac{d(\Gamma_1, u_1p_v)}{k-1}$. For $i = 2, \ldots, k$, rotate the scaled drawing Γ_i around u_{i-1} counter-clockwise by s radians. Translate the scaled and rotated drawings $\Gamma_2, \ldots, \Gamma_k$ so that the representations of u_i in Γ_i and Γ_{i+1} coincide, for $i = 1, \ldots, k-1$. Finally, draw the edge uv as a straight-line segment. This completes the construction of a drawing Γ of G. We have the following.

▶ **Lemma 10.** *For any $\delta \geq 0$, the drawing Γ_δ constructed in Case A if $k \geq 2$ satisfies Properties 1–6 of Theorem 8.*

Proof Sketch. The fulfillment of Property 3 for Γ_δ is the reason for the asymmetry of the construction, which shifts vertices in Γ_1, while it rotates $\Gamma_2, \ldots, \Gamma_k$. Indeed, for $i = 1, \ldots, k$, the first edge of $\beta_{u_{i-1}u_i}(G_i)$ has negative slope in Γ_i, while all the other edges have positive slopes; we need to ensure that the same property holds for $\beta_{uv}(G) = \beta_{u_0u_1}(G_1) \cup \cdots \cup \beta_{u_{k-1}u_k}(G_k)$ in Γ_δ. For $i = 2, \ldots, k$, the counter-clockwise rotation of Γ_i by $s = \frac{\alpha}{2}$ radians

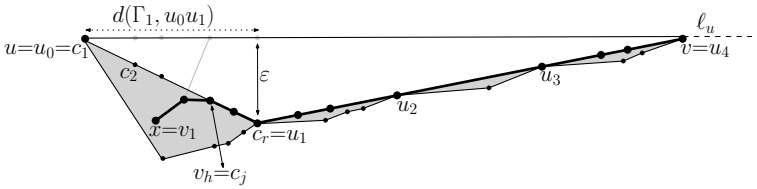


Figure 5 Illustration for the proof that the slope in Γ_δ of every edge in the path P_x is in $(-\alpha, \alpha)$, in the case in which x belongs to G_1. The path P_x is thick.

makes up for the negative slope (at most $\frac{\alpha}{3}$ in absolute value) of the first edge of $\beta_{u_{i-1}u_i}(G_i)$ in Γ_i. On the other hand, the edges of $\beta_{u_0u_1}(G_1)$ do not move when transforming Γ_1 in Γ, except for the edge incident to u_1, which however does not change its slope significantly, due to the inequality $\varepsilon < Y$; hence the slope of the first edge of $\beta_{u_0u_1}(G_1)$ remains negative (or becomes negative if G_1 is a single edge) and the other ones remain positive.

We present a proof that Γ_δ satisfies Property 4. Let $x \in V(G)$. If $x = u$, let $P_x = (u, v)$; then the only edge of P_x has slope $0 \in (-\alpha, \alpha)$ in Γ_δ. If $x = u_i$, for some $i \in \{1, \ldots, k-1\}$, then let $P_x = \bigcup_{j=i+1}^{k} \tau_{u_{j-1}u_j}(G_j)$ and observe that all the edges of P_x have slope $s = \frac{\alpha}{2} \in (-\alpha, \alpha)$; further P_x does not pass through u. If $x \neq u_i$, for every $i \in \{0, \ldots, k\}$, then x belongs to a unique graph G_i, for some $i \in \{1, \ldots, k\}$. Assume that $i = 1$; the case $i \geq 2$ is easier to handle. Refer to Fig. 5. Let $\tau_{u_0u_1}(G_1) = (u_0 = c_1, c_2, \ldots, c_r = u_1)$. Since Γ_1 satisfies Property 4, there exists a path $P_x^1 = (x = v_1, v_2, \ldots, v_p = u_1)$ from x to u_1 in G_1, not passing through u_0, whose edges have slopes in $(-\frac{\alpha}{2}, \frac{\alpha}{2})$ in Γ_1; let h be the smallest index such that $v_h = c_j$, for some $j \in \{1, \ldots, r\}$. Such an index h exists (possibly $h = p$ and $j = r$). Then let P_x consist of the paths $(x = v_1, v_2, \ldots, v_h)$, $(v_h = c_j, c_{j+1}, \ldots, c_r)$, and $\bigcup_{j=2}^{k} \tau_{u_{j-1}u_j}(G_j)$. Since $u \notin V(P_x^1)$, we have that $u \notin V(P_x)$, hence it suffices to argue about the slopes of the edges of P_x in Γ rather than in Γ_δ.

For $l = 1, \ldots, h-2$, the slope of v_lv_{l+1} is in $(-\alpha, \alpha)$ in Γ since it is in $(-\alpha, \alpha)$ in Γ_1 and since neither v_l nor v_{l+1} moves when transforming Γ_1 into Γ. Further, for $l = j, \ldots, r-1$, the slope of the edge c_lc_{l+1} in Γ is $-\arctan\left(\frac{\varepsilon}{d(\Gamma_1, u_0u_1)}\right)$, which is in the interval $(-\alpha, 0) \subset (-\alpha, \alpha)$, given that $\varepsilon, d(\Gamma_1, u_0u_1) > 0$ and that $\varepsilon < \tan(\alpha) \cdot d(\Gamma_1, u_0u_1)$. Moreover, the edges of $\bigcup_{j=2}^{k} \tau_{u_{j-1}u_j}(G_j)$ have slope $s = \frac{\alpha}{2} \in (-\alpha, \alpha)$. Finally, let σ_1 and σ be the slopes of the edge $v_{h-1}v_h$ in Γ_1 and Γ, respectively. Since $v_{h-1}v_h \in E(P_x^1)$, we have $\sigma_1 \in (-\frac{\alpha}{2}, \frac{\alpha}{2})$; since $\alpha \leq \frac{\pi}{4}$, we have $x(v_{h-1}) < x(v_h)$ in Γ_1 and Γ (note that the x-coordinates of the vertices do not change when transforming Γ_1 into Γ). Further, by Properties 1–4 of Γ_1, we have that v_{h-1} lies below ℓ_u, which contains v_h; hence, $y(v_{h-1}) < y(v_h)$ in Γ_1. Since the vertex v_h moves down (while v_{h-1} stays put) when transforming Γ_1 into Γ, and since $\varepsilon \leq \frac{Y}{2} < d_V(\Gamma_1, v_{h-1}v_h)$, it follows that $0 < \sigma < \sigma_1$; hence $\sigma \in (0, \frac{\alpha}{2}) \subset (-\alpha, \alpha)$.

Turning our attention to Property 6, consider any two vertices $x, y \in V(G)$, and assume that $x \in V(G_i)$ and $y \in V(G_j)$. We prove the existence of a path P_{xy} from x to y in G that is distance-decreasing in Γ_δ in the case in which $2 \leq i < j \leq k$; the other cases can be treated similarly. Let P_{xy} consist of a path P_x^i in G_i from x to u_i whose edges have slopes in $(-\frac{\alpha}{3}, \frac{\alpha}{3})$ in Γ_i, of the path $\bigcup_{l=i+1}^{j-1} \tau_{u_{l-1}u_l}(G_l)$, and of a path $P_{u_{j-1}y}^j$ in G_j that is distance-decreasing in Γ_j. By induction, P_x^i and $P_{u_{j-1}y}^j$ exist since Γ_i and Γ_j satisfy Properties 4 and 6, respectively; further, note that $u \notin V(P_{xy})$. Let $P_{xy} = (z_1, z_2, \ldots, z_s)$; we prove that $d(\Gamma_\delta, z_hz_s) > d(\Gamma_\delta, z_{h+1}z_s)$, for $h = 1, 2, \ldots, s-2$, hence P_{xy} is distance-decreasing in Γ_δ. We distinguish three cases.

If z_hz_{h+1} is in G_j, then $(z_h, z_{h+1}, \ldots, z_s)$ is a sub-path of $P_{u_{j-1}y}^j$, hence it is distance-decreasing in Γ_δ since it is distance-decreasing in Γ_j and since the drawing of G_j in Γ_δ is

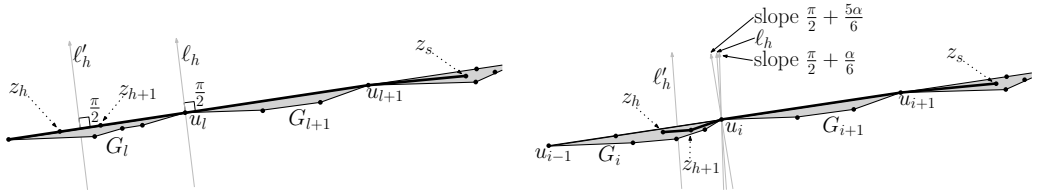


Figure 6 (a) Illustration for the proof that $d(\Gamma_\delta, z_h z_s) > d(\Gamma_\delta, z_{h+1} z_s)$ if $z_h z_{h+1}$ is in $\tau_{u_{l-1}u_l}(G_l)$. (b) Illustration for the proof that $d(\Gamma_\delta, z_h z_s) > d(\Gamma_\delta, z_{h+1} z_s)$ if $z_h z_{h+1}$ is in P_x^i.

congruent to Γ_j, up to affine transformations (a uniform scaling, a rotation, and a translation), which preserve the property of a path to be distance-decreasing.

If $z_h z_{h+1}$ is in $\tau_{u_{l-1}u_l}(G_l)$, for some $l \in \{i+1, i+2, \ldots, j-1\}$, as in Fig. 6(a), then it has slope $s = \frac{\alpha}{2}$. The directed line ℓ_h with slope $\frac{\pi+\alpha}{2}$ through u_l, oriented towards increasing y-coordinates has the drawings of $G_{l+1}, \ldots, G_k$ (and in particular the vertex z_s) to its right; this is because by Property 3 of Γ_δ every edge in $\beta_{u_l v}(G)$ has slope in the interval $(0, \alpha)$, where $\frac{-\pi+\alpha}{2} < 0 < \alpha < \frac{\pi+\alpha}{2}$, and because the path $\bigcup_{m=l+1}^{k} \tau_{u_{m-1}u_m}(G_m)$ has slope $s = \frac{\alpha}{2}$, where $\frac{-\pi+\alpha}{2} < \frac{\alpha}{2} < \frac{\pi+\alpha}{2}$. Then the directed line ℓ'_h parallel to ℓ_h, passing through the midpoint of the edge $z_h z_{h+1}$, and oriented towards increasing y-coordinates has ℓ_h to its right, hence it has z_s to its right. Since the half-plane to the right of ℓ'_h represents the locus of the points of the plane that are closer to z_{h+1} than to z_h, it follows that $d(\Gamma_\delta, z_h z_s) > d(\Gamma_\delta, z_{h+1} z_s)$.

If $z_h z_{h+1}$ is in P_x^i, as in Fig. 6(b), then by Property 4 it has slope in $(-\frac{\alpha}{3}, \frac{\alpha}{3})$ in Γ_i. Since Γ_i is counter-clockwise rotated by s radians in Γ_δ, it follows that $z_h z_{h+1}$ has slope in $(s - \frac{\alpha}{3}, s + \frac{\alpha}{3}) = (\frac{\alpha}{6}, \frac{5\alpha}{6})$ in Γ_δ. Consider the directed line ℓ_h that passes through u_i, that is directed towards increasing y-coordinates and that is orthogonal to the line through z_h and z_{h+1}. Denote by s_h the slope of ℓ_h. Then $s_h \in (\frac{\pi}{2} + \frac{\alpha}{6}, \frac{\pi}{2} + \frac{5\alpha}{6})$. We have that ℓ_h has the drawings of $G_{i+1}, \ldots, G_k$ to its right; this is because by Property 3 of Γ_δ every edge in $\beta_{u_i v}(G)$ has slope in $(0, \alpha)$ with $s_h - \pi < -\frac{\pi}{2} + \frac{5\alpha}{6} < 0 < \alpha < \frac{\pi}{2} + \frac{\alpha}{6} < s_h$ and because the path $\bigcup_{m=i+1}^{k} \tau_{u_{m-1}u_m}(G_m)$ has slope $s = \frac{\alpha}{2}$, where $s_h - \pi < -\frac{\pi}{2} + \frac{5\alpha}{6} < \frac{\alpha}{2} < \frac{\pi}{2} + \frac{\alpha}{6} < s_h$. Further, ℓ_h has the drawings of $G_2, \ldots, G_i$ to its left; this is because by Property 3 of Γ_δ every edge in $\tau_{u_i u_1}(G)$ has slope in $(\pi, \pi+\alpha)$ with $s_h < \frac{\pi}{2} + \frac{5\alpha}{6} < \pi < \pi+\alpha < \frac{3\pi}{2} + \frac{\alpha}{6} < \pi + s_h$ and because the path $\bigcup_{m=2}^{i} \beta_{u_m u_{m-1}}(G_m)$ has slope $s = \pi + \frac{\alpha}{2}$, where $s_h < \frac{\pi}{2} + \frac{5\alpha}{6} < \pi + \frac{\alpha}{2} < \frac{3\pi}{2} + \frac{\alpha}{6} < \pi + s_h$. Now consider the directed line ℓ'_h parallel to ℓ_h, passing through the midpoint of the edge $z_h z_{h+1}$, and oriented towards increasing y-coordinates. This line has ℓ_h to its right, given that the drawing of G_i (and in particular the midpoint of $z_h z_{h+1}$) is to the left of ℓ_h in Γ_δ. Thus, ℓ'_h has the drawings of $G_{l+1}, \ldots, G_k$ (and in particular the vertex z_s) to its right. Since the half-plane to the right of ℓ'_h represents the locus of the points of the plane that are closer to z_{h+1} than to z_h, it follows that $d(\Gamma_\delta, z_h z_s) > d(\Gamma_\delta, z_{h+1} z_s)$. ◀

We now discuss **Case B**, in which (G, u, v) is decomposed according to Lemma 7. Refer to Figs. 7 and 8. First, the triple (H, u, y_1) is a strong circuit graph with $|V(H)| \geq 3$. Inductively construct a straight-line drawing Γ_H of H with $\frac{\alpha}{2}$ as a parameter.

Let $\beta_{uy_1}(H) = (u = b_1, \ldots, b_m = y_1)$. Let ϕ_i be the slope of the edge $b_i b_{i+1}$ in Γ_H and let $\phi = \min_{i=2,\ldots,m-1}\{\phi_i\}$. By Property (c) of (H, u, y_1) if the edge uy_1 belongs to H then it coincides with the path $\tau_{uy_1}(H)$. Hence, $m \geq 3$ and ϕ is well-defined. Further, ϕ is in the interval $(0, \frac{\alpha}{2})$ by Property 3 of Γ_H.

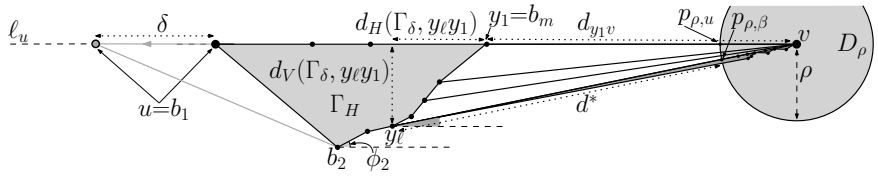


Figure 7 The straight-line drawing Γ of G in Case B. For the sake of readability, ϕ and ρ are larger than they should be. The dark gray angle is β. Fig. 8 shows an enlarged view of D_ρ.

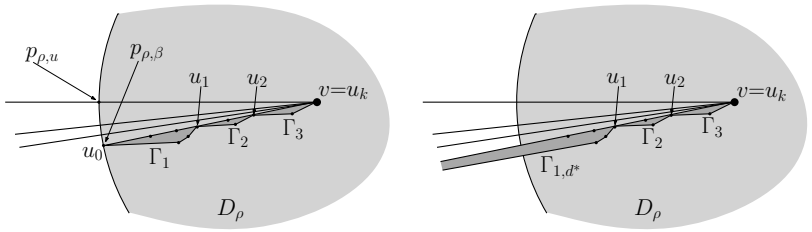


Figure 8 A closer look at D_ρ. Figure (a) represents the drawings $\Gamma_1, \ldots, \Gamma_k$ once they have been uniformly scaled, rotated, and translated, while (b) also has the vertex u_0 moved by d^* units (this movement actually happens before the rotation and translation of Γ_1).

Let $\beta = \frac{1}{2}\min\left\{\phi, \arctan\left(\frac{d_V(\Gamma_H, y_\ell y_1)}{3d_V(\Gamma_H, y_\ell y_1)+3d_H(\Gamma_H, y_\ell y_1)}\right)\right\}$. Note that $\beta > 0$, given that $\phi, d_V(\Gamma_H, y_\ell y_1) > 0$ and $d_H(\Gamma_H, y_\ell y_1) \geq 0$. In particular, $d_V(\Gamma_H, y_\ell y_1) > 0$ because y_1 is a vertex of $\tau_{uy_1}(H)$ and y_ℓ is an internal vertex of $\beta_{uy_1}(H)$ by Lemma 7, and because of Properties 1–3 of Γ_H. Also note that $\beta < \frac{\alpha}{4}$, given that $\phi < \frac{\alpha}{2}$.

Consider a half-line h_β with slope β starting at y_ℓ. Place the vertex v at the intersection point between h_β and the horizontal line ℓ_u through u. Draw all the trivial $(H \cup \{v\})$-bridges of G as straight-line segments. This concludes the construction if every $(H \cup \{v\})$-bridge of G is trivial. Otherwise, B_ℓ is the only non-trivial $(H \cup \{v\})$-bridge of G. Then B_ℓ consists of k strong circuit graphs (G_i, u_{i-1}, u_i), where $u_0 = y_\ell$ and $u_k = v$. With a slight change of notation, in the remainder of the section we assume that, if the edge $y_\ell v$ exists, then it is an edge of B_ℓ (rather than an individual trivial $(H \cup \{v\})$-bridge $B_{\ell-1}$ of G); in this case (B_ℓ, u_0, u_k) is a strong circuit graph.

We claim that v lies to the right of y_1. The polygonal line representing $\beta_{y_\ell y_1}(H)$ in Γ_H and the straight-line segment $\overline{y_\ell v}$ are both incident to y_ℓ. By definition of ϕ and since Γ_H satisfies Property 3, $\beta_{y_\ell y_1}(H)$ is composed of straight-line segments with slopes in the range $[\phi, \frac{\alpha}{2})$, while $\overline{y_\ell v}$ has slope β. The claim then follows from $0 < \beta < \phi < \frac{\pi}{2}$. Let $d_{y_1 v}$ be the distance between y_1 and v. Let $Y > 0$ be the minimum distance in Γ_H of any vertex strictly below ℓ_u from ℓ_u. Let $\rho = \min\{\frac{d_{y_1 v}}{3}, \frac{Y}{2}\}$ and let D_ρ be the disk with radius ρ centered at v. Let $p_{\rho,\beta}$ ($p_{\rho,u}$) be the intersection point of the boundary of D_ρ with h_β (resp. with ℓ_u) that is closer to y_ℓ (resp. to y_1). Let d^* be the Euclidean distance between y_ℓ and $p_{\rho,\beta}$.

Let $\alpha' = \frac{\beta}{2}$. Since $\beta > 0$, we have $\alpha' > 0$; further, $\alpha' < \frac{\alpha}{8}$, given that $\beta < \frac{\alpha}{4}$. For $i = 1, \ldots, k$, inductively construct a straight-line drawing Γ_i of G_i with α' as a parameter (if G_i is a single edge, then the parameter does not matter). Uniformly scale the drawings $\Gamma_1, \ldots, \Gamma_k$ so that the Euclidean distance between u_{i-1} and u_i is equal to $\frac{\rho}{k}$. Move the vertex u_0 in Γ_1 by d^* units to the left, obtaining a drawing Γ_{1,d^*}. Rotate the drawings $\Gamma_{1,d^*}, \Gamma_2, \ldots, \Gamma_k$ counter-clockwise by β radians. Translate $\Gamma_{1,d^*}, \Gamma_2, \ldots, \Gamma_k$ so that, for $i = 1, \ldots, k-1$, the representations of u_i in Γ_i and Γ_{i+1} (in Γ_{1,d^*} and Γ_2 if $i = 1$) coincide

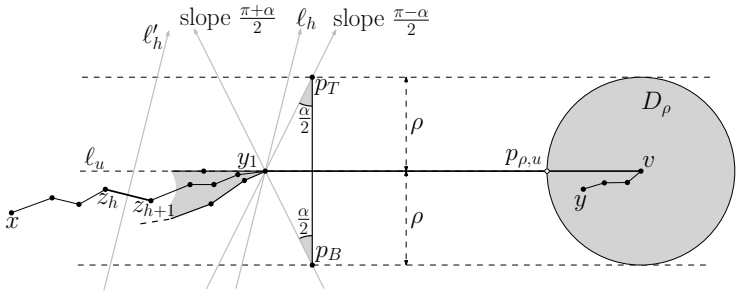


Figure 9 Illustration for the proof that $d(\Gamma_\delta, z_h y) > d(\Gamma_\delta, z_{h+1} y)$ if $z_h z_{h+1}$ is in P_x^H. For the sake of readability, D_ρ is larger than it should be.

and so that the representation of u_0 in Γ_{1,d^*} coincides with the one of y_ℓ in Γ_H. This completes the construction of a straight-line drawing Γ of G. We have the following.

▶ **Lemma 11.** *For any $\delta \geq 0$, the drawing Γ_δ constructed in Case B satisfies Properties 1–6 of Theorem 8.*

Proof Sketch. Let $\Gamma_{H,\delta}$ denote the drawing obtained from Γ_H by moving the vertex u by δ units to the left.

We first prove that every vertex $z \neq u_0$ that belongs to a graph G_i lies inside the disk D_ρ in Γ_δ. A consequence of this statement is a sharp geometric separation between the vertices of G that are in H and those that are not. Consider any drawing Γ_j with $j \in \{1, \ldots, k\}$ (where Γ_1 is considered before moving u_0 by d^* units to the left) and let D_j be the disk centered at u_j with radius $d(\Gamma_j, u_{j-1}u_j)$. By Properties 1 and 2 of Γ_j, the path $\tau_{u_{j-1}u_j}(G_j)$ lies on $\overline{u_{j-1}u_j}$ in Γ_j, hence it lies inside D_j. Further, the edges of $\beta_{u_{j-1}u_j}(G_j)$ have slopes in $(-\alpha', \alpha') \subset (-\frac{\alpha}{8}, \frac{\alpha}{8}) \subset (-\frac{\pi}{32}, \frac{\pi}{32})$; hence $\beta_{u_{j-1}u_j}(G_j)$ also lies inside D_j. By planarity Γ_j lies entirely inside D_j. Hence, u_{j-1} is the farthest vertex of G_j from u_j in Γ_j. This property is true also after the uniform scaling of $\Gamma_1, \ldots, \Gamma_k$; further, after the scaling, the distance between u_{j-1} and u_j is $\frac{\rho}{k}$, by construction. By the triangular inequality, we have that $d(\Gamma_\delta, vz) \leq \sum_{j=i+1}^{k} d(\Gamma_\delta, u_{j-1}u_j) + d(\Gamma_\delta, u_i z)$. Since $d(\Gamma_\delta, u_{j-1}u_j) = \frac{\rho}{k}$ for any $j \in \{2, \ldots, k\}$, and since $d(\Gamma_\delta, u_i z) \leq \frac{\rho}{k}$ (this exploits $z \neq u_0$ and hence $d(\Gamma_\delta, u_i z) = d(\Gamma_i, u_i z)$, where Γ_i is understood as already scaled), we have that $d(\Gamma_\delta, vz) \leq \frac{(k-i+1)\rho}{k} \leq \rho$. Thus z lies inside D_ρ.

We now prove that Property 6 is satisfied by Γ_δ. We devote our attention to the proof of the existence of a distance-decreasing path P_{xy} from a vertex x to a vertex y if: (i) $x \in V(H)$ and $y \in V(G_i)$, for some $i \in \{1, \ldots, k\}$; or (ii) $x \in V(G_i)$, for some $i \in \{1, \ldots, k\}$, and $y \in V(H)$. While the rest of the proof that Property 6 is satisfied by Γ_δ proceeds similarly to the proof of Lemma 10, cases (i) and (ii) above deal with vertices x and y that are "far apart" in Γ_δ, a circumstance that does not occur in the proof of Lemma 10.

In case (i) P_{xy} contains a path P_x^H in H from x to y_1. Assume that $x \neq u$, as the case $x = u$ is easier to handle. By Property 4 of $\Gamma_{H,\delta}$, there is a path $P_x^H = (x = z_1, \ldots, z_s = y_1)$ in H that connects x to y_1, that does not pass through u, and whose edges have slopes in $(-\frac{\alpha}{2}, \frac{\alpha}{2})$ in $\Gamma_{H,\delta}$. We prove that, for $h = 1, \ldots, s-1$, $d(\Gamma_\delta, z_h y) > d(\Gamma_\delta, z_{h+1} y)$; see Fig. 9. Since the drawing of H in Γ_δ coincides with $\Gamma_{H,\delta}$, $z_h z_{h+1}$ has slope in $(-\frac{\alpha}{2}, \frac{\alpha}{2})$ in Γ_δ. Let ℓ_h be the directed line through y_1 directed towards increasing y-coordinates and orthogonal to the line through z_h and z_{h+1}. Denote by s_h the slope of ℓ_h. Then $s_h \in (\frac{\pi-\alpha}{2}, \frac{\pi+\alpha}{2})$.

We prove that ℓ_h has the disk D_ρ to its right. In order to do that, consider the point p_T on the half-line with slope $\frac{\pi-\alpha}{2}$ starting at y_1 and such that $d_V(\Gamma_\delta, y_1 p_T) = \rho$. Further, consider

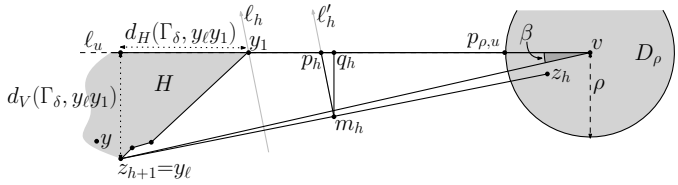


Figure 10 Illustration for the proof that $d(\Gamma_\delta, z_h y) > d(\Gamma_\delta, z_{h+1} y)$ if $z_{h+1} = y_\ell = u_0$.

the point p_B on the half-line with slope $\frac{-\pi+\alpha}{2}$ starting at y_1 and such that $d_V(\Gamma_\delta, y_1 p_B) = \rho$. Note that $\overline{p_T p_B}$ is a vertical straight-line segment with length 2ρ. Consider the infinite closed strip S with height 2ρ that is delimited by the horizontal lines through p_T and p_B. Since D_ρ has its center on ℓ_u and has radius ρ, it lies inside S. The part of ℓ_h inside S is to the left of $\overline{p_T p_B}$, given that $s_h \in (\frac{\pi-\alpha}{2}, \frac{\pi+\alpha}{2})$. Hence, it suffices to show that $p_{\rho,u}$, which is the point of D_ρ with smallest x-coordinate, lies to the right of $\overline{p_T p_B}$. We have $d(\Gamma_\delta, y_1 p_{\rho,u}) = d_{y_1 v} - \rho$. Further, $d_H(\Gamma_\delta, y_1 p_T) = \rho \cdot \tan(\frac{\alpha}{2})$. Hence, it suffices to prove $\rho \cdot \tan(\frac{\alpha}{2}) < d_{y_1 v} - \rho$, that is $\rho < \frac{d_{y_1 v}}{1+\tan(\frac{\alpha}{2})}$; this is true since $\rho < \frac{d_{y_1 v}}{3}$ and $\tan(\frac{\alpha}{2}) < 1$, given that $0 < \alpha < \frac{\pi}{4}$.

The line ℓ_h has $\Gamma_{H,\delta}$ (and in particular the midpoint of $z_h z_{h+1}$) to its left; this is because by Property 2 of $\Gamma_{H,\delta}$ every edge in $\beta_{y_1 u}(H)$ has slope π, where $s_h < \frac{\pi+\alpha}{2} < \pi < \frac{3\pi-\alpha}{2} < \pi + s_h$, and because by Property 3 of $\Gamma_{H,\delta}$ every edge in $\tau_{y_1 u}(H)$ has slope in $(\pi - \frac{\alpha}{2}, \pi + \frac{\alpha}{2})$, where $s_h < \frac{\pi+\alpha}{2} < \pi - \frac{\alpha}{2} < \pi + \frac{\alpha}{2} < \frac{3\pi-\alpha}{2} < \pi + s_h$. Let ℓ'_h be the directed line parallel to ℓ_h, passing through the midpoint of $z_h z_{h+1}$, and oriented towards increasing y-coordinates; ℓ'_h has ℓ_h to its right, as the midpoint of $z_h z_{h+1}$ is to the left of ℓ_h in Γ_δ. Thus, ℓ'_h has D_ρ, and in particular y, to its right. Since the half-plane to the right of ℓ'_h is the locus of the points of the plane that are closer to z_{h+1} than to z_h, it follows that $d(\Gamma_\delta, z_h y) > d(\Gamma_\delta, z_{h+1} y)$.

The path P_{xy} also contains the edge $y_1 v$, which "jumps" from H to D_ρ. Since y lies in D_ρ, we have that $d(\Gamma_\delta, vy) \leq \rho \leq \frac{d_{y_1 v}}{3}$. By the triangular inequality, we have that $d(\Gamma_\delta, y_1 y) > d(\Gamma_\delta, y_1 v) - d(\Gamma_\delta, vy) \geq d_{y_1 v} - \rho \geq \frac{2 d_{y_1 v}}{3}$. Hence, $d(\Gamma_\delta, y_1 y) > d(\Gamma_\delta, vy)$. The third sub-path of P_{xy} is a path P_{vy} from v to y in $\bigcup_{l=i}^{k} G_l$ that is distance-decreasing in Γ_δ. The construction of this path proceeds similarly as in the proof of Lemma 10.

In case (ii) we have that $x \in V(G_i)$, for some $i \in \{1, \ldots, k\}$, and $y \in V(H)$. While in case (i) the connection between H and D_ρ is done via y_1, here it is done via y_ℓ. In particular, the first part of P_{xy} consists of edges with slopes in the range $(\pi - \alpha, \pi + \alpha)$ inside D_ρ. Similarly to case (i), the orthogonal line through the midpoint of each of these edges separates H from D_ρ; hence traversing the edge decreases the distance to y.

We now argue that traversing an edge that "jumps" from D_ρ to H decreases the distance to y. That is, we show that, for a vertex z_h in D_ρ incident to an edge $z_h z_{h+1}$ with $z_{h+1} = y_\ell = u_0$, it holds $d(\Gamma_\delta, z_h y) > d(\Gamma_\delta, z_{h+1} y)$. Refer to Fig. 10. We exploit again the fact that the line ℓ_h passing through y_1 and orthogonal to the line through z_h and z_{h+1} has $\Gamma_{H,\delta}$ (and in particular y) to its left; then consider the directed line ℓ'_h parallel to ℓ_h, oriented towards increasing y-coordinates, and passing through the midpoint m_h of $z_h z_{h+1}$. Since the half-plane to the left of ℓ'_h is the locus of the points of the plane that are closer to z_{h+1} than to z_h, it suffices to show that the intersection point p_h of ℓ'_h and ℓ_u lies to the right of y_1 on ℓ_u; in fact, this implies that ℓ'_h has ℓ_h (and hence y) to its left.

Since z_h lies inside D_ρ, we have $x(z_h) \geq x(p_{\rho,u}) = x(y_1) + d_{y_1 v} - \rho$. Moreover, $x(y_1) = x(y_\ell) + d_H(\Gamma_\delta, y_\ell y_1)$. Thus, we have $x(m_h) = \frac{x(y_\ell)+x(z_h)}{2} \geq \frac{x(y_\ell)+(x(y_\ell)+d_H(\Gamma_\delta, y_\ell y_1)+d_{y_1 v}-\rho)}{2} = x(y_\ell) + \frac{d_H(\Gamma_\delta, y_\ell y_1)+d_{y_1 v}-\rho}{2}$. Translate the Cartesian axes so that $x(y_\ell) = 0$. Thus, $x(m_h) = \frac{d_H(\Gamma_\delta, y_\ell y_1)+d_{y_1 v}-\rho}{2}$. By Lemma 7, y_ℓ is an internal vertex of $\beta_{uv}(G)$, hence y_ℓ lies below ℓ_u.

Since $\rho < Y$ and z_h lies in D_ρ, the y-coordinate of y_ℓ is smaller than the one of z_h. Hence, the slope of $z_h z_{h+1}$ is larger than π. Further, z_h and hence m_h lie on or below h_β, thus the slope of $z_h z_{h+1}$ is at most $\pi + \beta$ and the slope s_h of ℓ'_h is in the interval $(\frac{\pi}{2}, \frac{\pi}{2} + \beta)$.

We now derive a lower bound for the x-coordinate of p_h. Let q_h be the projection of m_h on ℓ_u. Consider the triangle $\Delta m_h p_h q_h$. Since the y-coordinate of y_ℓ is smaller than the one of z_h, it is also smaller than the one of m_h. Thus, $d(\Gamma_\delta, m_h q_h) \leq d_V(\Gamma_\delta, y_\ell y_1)$. Since $s_h \in (\frac{\pi}{2}, \frac{\pi}{2} + \beta)$, the angle $\measuredangle p_h m_h q_h$ is at most β. Hence, $d(\Gamma_\delta, p_h q_h) \leq d_V(\Gamma_\delta, y_\ell y_1) \cdot \tan(\beta)$. Thus, $x(p_h) = x(m_h) - d(\Gamma_\delta, p_h q_h) \geq \frac{d_H(\Gamma_\delta, y_\ell y_1) + d_{y_1 v} - \rho}{2} - d_V(\Gamma_\delta, y_\ell y_1) \cdot \tan(\beta)$. It remains to prove that this quantity is larger than $d_H(\Gamma_\delta, y_\ell y_1)$, which is the x-coordinate of y_1.

Since $\beta < \frac{\alpha}{4} < \frac{\pi}{16}$, we have $\tan(\beta) \leq 1$, hence $\frac{d_H(\Gamma_\delta, y_\ell y_1) + d_{y_1 v} - \rho}{2} - d_V(\Gamma_\delta, y_\ell y_1) \cdot \tan(\beta) \geq \frac{d_H(\Gamma_\delta, y_\ell y_1) + d_{y_1 v} - \rho}{2} - d_V(\Gamma_\delta, y_\ell y_1)$. We want to establish $\frac{d_H(\Gamma_\delta, y_\ell y_1) + d_{y_1 v} - \rho}{2} - d_V(\Gamma_\delta, y_\ell y_1) > d_H(\Gamma_\delta, y_\ell y_1)$, that is, $d_{y_1 v} > 2 d_V(\Gamma_\delta, y_\ell y_1) + d_H(\Gamma_\delta, y_\ell y_1) + \rho$. Since $\rho \leq \frac{d_{y_1 v}}{3}$, we need to prove that $d_{y_1 v} > \frac{6 d_V(\Gamma_\delta, y_\ell y_1) + 3 d_H(\Gamma_\delta, y_\ell y_1)}{2}$.

We now express $d_{y_1 v}$ as a function of β. This is done by looking at the triangle whose vertices are y_ℓ, v, and the projection of y_ℓ on ℓ_u. Since the angle of this triangle at v is β, we get $d_{y_1 v} = \frac{d_V(\Gamma_\delta, y_\ell y_1)}{\tan(\beta)} - d_H(\Gamma_\delta, y_\ell y_1)$. Substituting this into the previous inequality, we need to have $\frac{d_V(\Gamma_\delta, y_\ell y_1)}{\tan(\beta)} - d_H(\Gamma_\delta, y_\ell y_1) > \frac{6 d_V(\Gamma_\delta, y_\ell y_1) + 3 d_H(\Gamma_\delta, y_\ell y_1)}{2}$, hence $\tan(\beta) < \frac{2 d_V(\Gamma_\delta, y_\ell y_1)}{6 d_V(\Gamma_\delta, y_\ell y_1) + 5 d_H(\Gamma_\delta, y_\ell y_1)}$, which is true since $\beta < \arctan\left(\frac{d_V(\Gamma_H, y_\ell y_1)}{3 d_V(\Gamma_H, y_\ell y_1) + 3 d_H(\Gamma_H, y_\ell y_1)}\right)$. This concludes the proof that $d(\Gamma_\delta, z_h y) > d(\Gamma_\delta, z_{h+1} y)$.

The path P_{xy} continues with a path $P_{y_\ell y}$ from y_ℓ to y in H that is distance-decreasing in $\Gamma_{H,\delta}$ (and hence in Γ_δ, since the drawing of H in Γ_δ coincides with $\Gamma_{H,\delta}$). This concludes the proof of the lemma. ◀

Given a strong circuit graph (G, u, v) such that G is not a single edge, we are in Case A or Case B depending on whether the edge uv exists or not, respectively. Thus, Lemmata 9–11 prove Theorem 8. We show how to use Theorem 8 in order to prove Theorem 3. Consider any 3-connected planar graph G and associate any plane embedding to it; let u and v be two consecutive vertices in the clockwise order of the vertices along the outer face of G. We have that (G, u, v) is a strong circuit graph. Indeed: (a) by assumption G is 2-connected – in fact 3-connected – and associated with a plane embedding; (b) by construction u and v are two distinct external vertices of G; (c) edge uv exists and coincides with $\tau_{uv}(G)$, given that v immediately follows u in the clockwise order of the vertices along the outer face of G; and (d) G does not have any 2-cut, as it is 3-connected. Thus, Theorem 8 can be applied in order to construct a planar greedy drawing of G. This concludes the proof of Theorem 3.

4 Conclusions

In this paper we have shown how to construct planar greedy drawings of 3-connected planar graphs. It is tempting to try to use the graph decomposition we employed in this paper for proving that 3-connected planar graphs admit *convex* greedy drawings. However, despite some efforts in this direction, we have not been able to modify the statement of Theorem 8 in order to guarantee the desired convexities of the angles in the drawings. Thus, proving or disproving the convex greedy embedding conjecture remains an elusive goal.

References

1 S. Alamdari, T. M. Chan, E. Grant, A. Lubiw, and V. Pathak. Self-approaching graphs. In Didimo and Patrignani, editors, *GD*, volume 7704 of *LNCS*, pages 260–271, 2012.

2 P. Angelini, G. Di Battista, and F. Frati. Succinct greedy drawings do not always exist. *Networks*, 59(3):267–274, 2012.

3 P. Angelini, E. Colasante, G. Di Battista, F. Frati, and M. Patrignani. Monotone drawings of graphs. *J. Graph Algorithms Appl.*, 16(1):5–35, 2012.

4 P. Angelini, F. Frati, and L. Grilli. An algorithm to construct greedy drawings of triangulations. *J. Graph Algorithms Appl.*, 14(1):19–51, 2010.

5 D. Barnette. Trees in polyhedral graphs. *Canadian J. Math.*, 18:731–736, 1966.

6 P. Bose, P. Morin, I. Stojmenović, and J. Urrutia. Routing with guaranteed delivery in ad hoc wireless networks. *Wireless Networks*, 7(6):609–616, 2001.

7 G. Chen and X. Yu. Long cycles in 3-connected graphs. *J. Comb. Theory, Ser. B*, 86(1):80–99, 2002.

8 G. Da Lozzo, A. D'Angelo, and F. Frati. On planar greedy drawings of 3-connected planar graphs. *CoRR*, 2016. URL: `http://arxiv.org/abs/1612.09277`.

9 G. Da Lozzo, V. Dujmović, F. Frati, T. Mchedlidze, and V. Roselli. Drawing planar graphs with many collinear vertices. In Hu and Nöllenburg, editors, *GD*, volume 9801 of *LNCS*, pages 152–165, 2016.

10 H. R. Dehkordi, F. Frati, and J. Gudmundsson. Increasing-chord graphs on point sets. *J. Graph Algorithms Appl.*, 19(2):761–778, 2015.

11 R. Dhandapani. Greedy drawings of triangulations. *Discr. Comp. Geom.*, 43(2):375–392, 2010.

12 G. Di Battista and R. Tamassia. Algorithms for plane representations of acyclic digraphs. *Theor. Comput. Sci.*, 61:175–198, 1988.

13 D. Eppstein and M. T. Goodrich. Succinct greedy geometric routing using hyperbolic geometry. *IEEE Trans. Computers*, 60(11):1571–1580, 2011.

14 S. Felsner, A. Igamberdiev, P. Kindermann, B. Klemz, T. Mchedlidze, and M. Scheucher. Strongly monotone drawings of planar graphs. In Fekete and Lubiw, editors, *SoCG*, volume 51 of *LIPIcs*, pages 37:1–37:15, 2016.

15 H. Frey, S. Rührup, and I. Stojmenović. Routing in wireless sensor networks. In Misra, Woungang, and Misra, editors, *Guide to Wireless Sensor Networks*, Computer Communications and Networks, chapter 4, pages 81–111. Springer, 2009.

16 M. T. Goodrich and D. Strash. Succinct greedy geometric routing in the Euclidean plane. In Dong, Du, and Ibarra, editors, *ISAAC*, volume 5878 of *LNCS*, pages 781–791, 2009.

17 X. He and H. Zhang. On succinct greedy drawings of plane triangulations and 3-connected plane graphs. *Algorithmica*, 68(2):531–544, 2014.

18 P. Kindermann, A. Schulz, J. Spoerhase, and A. Wolff. On monotone drawings of trees. In Duncan and Symvonis, editors, *GD*, volume 8871 of *LNCS*, pages 488–500, 2014.

19 E. Kranakis, H. Singh, and J. Urrutia. Compass routing on geometric networks. In *CCCG*, 1999. URL: `http://www.cccg.ca/proceedings/1999/c46.pdf`.

20 F. Kuhn, R. Wattenhofer, and A. Zollinger. An algorithmic approach to geographic routing in ad hoc and sensor networks. *IEEE/ACM Trans. Netw.*, 16(1):51–62, 2008.

21 T. Leighton and A. Moitra. Some results on greedy embeddings in metric spaces. *Discr. Comp. Geom.*, 44(3):686–705, 2010.

22 M. Nöllenburg and R. Prutkin. Euclidean greedy drawings of trees. In Bodlaender and Italiano, editors, *ESA*, volume 8125 of *LNCS*, pages 767–778, 2013.

23 M. Nöllenburg, R. Prutkin, and I. Rutter. On self-approaching and increasing-chord drawings of 3-connected planar graphs. *J. Comp. Geom.*, 7(1):47–69, 2016.

24 C. H. Papadimitriou and D. Ratajczak. On a conjecture related to geometric routing. In Nikoletseas and Rolim, editors, *ALGOSENSORS*, volume 3121 of *LNCS*, pages 9–17, 2004.

25 C. H. Papadimitriou and D. Ratajczak. On a conjecture related to geometric routing. *Theor. Comput. Sci.*, 344(1):3–14, 2005.

26 A. Rao, C. H. Papadimitriou, S. Shenker, and I. Stoica. Geographic routing without location information. In Johnson, Joseph, and Vaidya, editors, *MOBICOM*, pages 96–108, 2003.

27 C. Siva Ram Murthy and B. S. Manoj. *Ad Hoc Wireless Networks: Architectures and Protocols*. Prentice Hall, 2004.

28 C. K. Toh. *Ad Hoc Mobile Wireless Networks: Protocols and Systems*. Prentice Hall, 2002.

Origamizer: A Practical Algorithm for Folding Any Polyhedron

Erik D. Demaine[*1] **and Tomohiro Tachi**[†2]

1 **MIT Computer Science and Artificial Intelligence Laboratory, Cambridge, MA, USA**
edemaine@mit.edu

2 **Department of General Systems Studies, The University of Tokyo, Japan**
tachi@idea.c.u-tokyo.ac.jp

Abstract

It was established at SoCG'99 that every polyhedral complex can be folded from a sufficiently large square of paper, but the known algorithms are extremely impractical, wasting most of the material and making folds through many layers of paper. At a deeper level, these foldings get the topology wrong, introducing many gaps (boundaries) in the surface, which results in flimsy foldings in practice. We develop a new algorithm designed specifically for the practical folding of real paper into complicated polyhedral models. We prove that the algorithm correctly folds any oriented polyhedral manifold, plus an arbitrarily small amount of additional structure on one side of the surface (so for closed manifolds, inside the model). This algorithm is the first to attain the *watertight* property: for a specified cutting of the manifold into a topological disk with boundary, the folding maps the boundary of the paper to within ε of the specified boundary of the surface (in Fréchet distance). Our foldings also have the geometric feature that every convex face is folded seamlessly, i.e., as one unfolded convex polygon of the piece of paper. This work provides the theoretical underpinnings for Origamizer, freely available software written by the second author, which has enabled practical folding of many complex polyhedral models such as the Stanford bunny.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases origami, folding, polyhedra, Voronoi diagram, computational geometry

Digital Object Identifier 10.4230/LIPIcs.SoCG.2017.34

1 Introduction

The ultimate challenge in computational origami design is to devise an algorithm that tells you the best way to fold anything you want. Several results tackle this problem for various notions of "best" and "anything". We highlight two key such results, from SoCG'96 and SoCG'99 respectively. The tree method [6, 7, 4] finds an efficient folding of a given square of paper into a shape with an orthogonal projection equal to a scaled copy of a given metric tree. We use the term "efficient" because the method works well in practice, being the foundation for most modern origami design, but exact optimization of the scale factor (the usual measure of efficiency) is a difficult computational problem, recently shown NP-hard [3], but one that can be handled reasonably well by heuristics. The strip method [1] finds a folding of a given piece of paper into a scaled copy of any desired polyhedral complex (any

[*] Supported in part by NSF ODISSEI grant EFRI-1240383 and NSF Expedition grant CCF-1138967.
[†] Supported in part by JST PRESTO program and JSPS KAKENHI 16H06106.

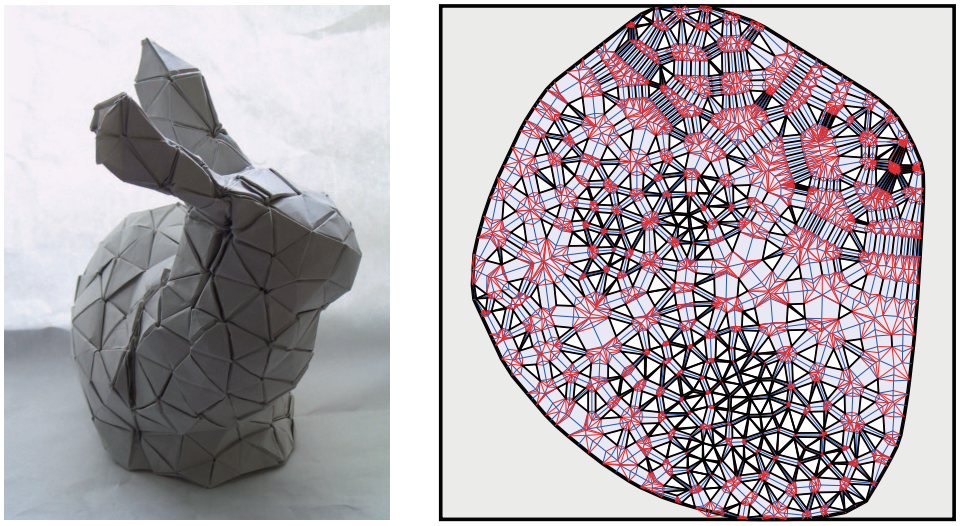


Figure 1 Origamizer software [8, 9] applied to 374-triangle Stanford bunny: real-world folding (left) and computed crease pattern (right).

connected union of polygons in 3D), which is a much broader notion of "anything". But it inherently provides no way to optimize the scale factor, forcing extreme inefficiency in paper usage (unless the piece of paper is a long narrow strip).

Our goal is to achieve the best of both these worlds. On the one hand, we want to fold arbitrary polyhedral complexes. On the other hand, we want to be able to optimize the scale factor to find efficient and ideally practical foldings. The vision is that the origami designer of the future uses 3D modeling software to design their target figure, gives it to an algorithm, and out comes a practical origami design.

We are not far from this vision today. *Origamizer* [8, 9] is freely available computer software implementing a relatively new heuristic for this problem. It achieves surprisingly practical foldings for complicated polyhedral models. For example, Figure 1 shows the result for the classic Stanford bunny, coarsened to 374 triangles. In this design, 22.3% of the paper area makes up the actual surface of the target shape – about a 2:1 scale factor in each dimension – which matches the material usage ratio in most practical origami design. The crease pattern is unlike most representational origami designs today, likely incapable of being designed by hand, yet it is also quite reasonable to fold in practice.[1]

The catch is that the heuristic implemented by the Origamizer software sometimes fails: sometimes it cannot even find a feasible solution, which makes it impossible to optimize even locally. In this paper, we develop an *Origamizer algorithm* that is guaranteed to find a feasible folding, for any orientable polyhedral manifold. In fact, we describe a large family of feasible foldings, with many free parameters, similar to the original Origamizer heuristic. In particular, Figure 1 falls within our family of foldings. The key new guarantee is that our family of foldings is nonempty, and that we can find such a folding (actually many) algorithmically. While we do not consider here how to optimize within the family of foldings (this is likely a computationally difficult problem, like the tree method), the starting points found by our algorithm open the door to a wealth of optimization heuristics. The result

[1] To see an accelerated video of the 10-hour folding process, visit `http://www.youtube.com/watch?v=GAnW-KU2yn4`

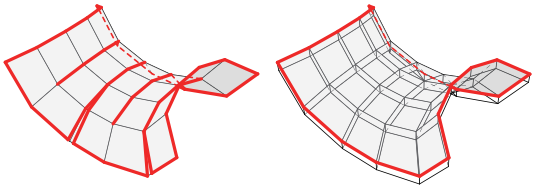


Figure 2 Folding a surface with gaps like the strip method (left) versus a watertight folding with some extra tiny facets like our method (right). The paper boundary is drawn thick.

should be at least as efficient as the existing Origamizer software, because the family of foldings is broader, but now it is also guaranteed never to fail. We plan to implement this provably correct algorithm in a future version of the Origamizer software.

Our Origamizer algorithm proves the existence of a new type of folding, called *watertight*, for any specification of where the boundary of the paper should go. That is, suppose we are told how to cut the given oriented polyhedral manifold into a topological disk with boundary. (If the manifold is itself a disk, no cutting is necessary.) Informally, a watertight folding has no holes, gaps, or slits internal to this boundary – only paper. Formally, the boundary of the

piece of paper (which can be any convex polygon) maps to within Fréchet distance ε of the boundary of the polyhedral surface, for a specified $\varepsilon > 0$. Thus the rest of the polyhedral surface must be covered entirely by the interior of the paper. By contrast, this property is violated violently in the strip method [1], which places the boundary of the paper on every face. Indeed, the lack of the watertight property seems a natural formalization of how the strip method felt like "cheating". Figure 2 shows a comparison.

Because the paper is homeomorphic to a disk, disk surfaces are the best we could hope to make watertight. Although we believe the watertight property is an essential feature of what makes Origamizer's foldings practical, it has one theoretical downside: it cannot fold exactly a desired polyhedral complex. By watertightness, a negative curvature vertex must be covered by an interior point of the piece of paper, which has a disk neighborhood. In other words, an entire neighborhood of a point of the piece of paper must fold to an entire neighborhood of a vertex of negative curvature. But such a folding is impossible by the Gauss-Bonnet Theorem.

Therefore Origamizer aims to fold a slight variation of the polyhedral complex, which adds small additional features at the vertices and along the edges. These features all have Hausdorff distance at most ε from the polyhedral complex, for any specified $\varepsilon > 0$. Furthermore, for orientable polyhedral surfaces, the features can all be placed on one side. In the case of an orientable closed polyhedron, like the Stanford bunny, we can place all of these features on the inside, so that they become invisible to the spectator. Such hidden features are standard in origami, and we believe they are well worth it to achieve the watertight property.

2 Problem Statement and Overview

The *Origamizer problem* is to find a convex polygon of paper P and a "watertight", "seamless", "ε-extra folding" of P into a given "polyhedral manifold" Q.[2] We need to define the four notions in quotes.

[2] Any convex polygon of paper can be folded into any other convex polygon after suitable scaling [1], so we can view the convex shape of the piece of paper as a free choice by the algorithm.

A *polyhedral manifold* Q is an embedded polyhedral manifold with strictly convex facets homeomorphic to a disk. Such Q could come from any polyhedral complex using standard techniques: for nonorientable or nonmanifold complexes, doubling every face to make them orientable manifolds; for orientable manifolds not homeomorphic to a disk, cutting each handle; and for nonconvex facets, subdividing into convex pieces. The polyhedral manifold Q has two specified sides, the *clean side* and the *tuck side*. For defining clockwise/counterclockwise orientations, we view the tuck side as the *top* side of the manifold. We require that the manifold does not touch itself, at least on the tuck side, other than the two boundary edges that come from each cut edge.

For any $\varepsilon > 0$, an *ε-extra folding* of a polygon of paper, P, into a manifold with specified boundary, Q, is a folded state of P (as defined, e.g., in [4, chapter 11]) whose image includes all of Q and otherwise is within ε of Q on the tuck side of Q. More precisely, we construct a *tuck-side ε offset* of Q, by unioning the portion of a radius-ε ball, centered at every point of Q, that lies on the tuck side of Q; when Q does not separate the ball into two portions (i.e., within ε of the original boundary of Q), we include the entire ball. Then the image of an ε-extra folding must lie entirely within the union of Q and its tuck-side ε offset. In particular, the Hausdorff distance between Q and the image of the folded state is at most ε.

Such a folding is *seamless* if the clean side of every facet of Q is covered by a single facet of the crease pattern of P, at the outermost layer of the folded state. Intuitively, this condition means that all visible creases and boundary edges of the piece of paper lie on edges and the tuck side of Q. In our foldings, we will satisfy the stronger property that each facet of Q is represented by a single uncreased face of paper, with no additional layers of paper even on the tuck side.

Such a folding is *watertight* if there is a closed curve C on P (the *effective boundary* of P) that folds to a 3D curve having Fréchet distance at most ε to the closed loop of boundary edges of Q, such that the folded image of the interior of C covers the interior of Q. In other words, the exterior of C is extraneous to the folding, and every point on the folded C is within distance ε of a corresponding point on the boundary of Q, where this correspondence proceeds monotonically around both curves (according to some parameterization). In our foldings, we will further guarantee that C is convex; indeed, C will be the boundary of the piece of paper P output by our algorithm.

Overview. Figure 3 gives a visual overview of the entire Origamizer algorithm. The algorithm first attaches ε-thin faces to the target polyhedral surface Q to form a target folded structure called a *waffle*, effectively splitting negative-curvature vertices into multiple positive-curvature *pockets* of the waffle [Section 3]. Next the algorithm locally "squashes" these pockets into the piece of paper, with angles large enough that they can be folded down to the desired angles [Section 4]. All that remains is to fold away the excess material between these squashed pockets, and thereby form the waffle. To guide this process, the algorithm first draws *streams* (smooth constant-width channels) that connect together corresponding edges and vertices of different squashed pockets [Section 5].

Ultimately, Origamizer uses a *Voronoi diagram* as the basis for its crease pattern: for any set of sites, we show how to fold an abstract waffle (not necessarily embedded in 3D) with exactly one pocket for every Voronoi cell [Section 7]. The challenge is to choose sites defining the Voronoi diagram so that the resulting abstract waffle can be folded into the desired waffle. The algorithm places one site at each squashed pocket, several sites along each stream, and additional sites to fill the rest of the paper [Section 6]. The resulting abstract waffle can be folded into the desired waffle by collapsing each stream's pockets to bring together the pockets at either end of the stream, and by collapsing all additional pockets [Section 7].

Given the page limit, we focus here on high-level sketches and figures of the required properties and algorithms. Refer to the full version of the paper [5] for the details and proofs.

3 Tuck Proxy and Waffles

Given the target polyhedral manifold Q, the first step of the algorithm computes a "tuck proxy" T, which is a special type of "polyhedral waffle". The *tuck proxy* T is a polyhedral complex that will contain our folding and which contains (and lies very close to) the target polyhedral surface Q. Roughly speaking, a *polyhedral waffle* consists of *floor* polygons, which together form a topological disk, and *wall* polygons – wall *quadrangles* glued along floor edges, and wall *triangles* glued at floor vertices. The *top* edges of the wall polygons, opposite their attachment to their floor, must form an edge-2-connected planar graph called the *waffle graph*; refer to Figure 4. The *waffle dual* is the "modified dual" of the waffle graph. Roughly speaking, the *modified dual* is the usual planar dual plus a boundary node and incident edge for each edge of the outside face. (Thus, each boundary node has exactly one incident edge.)

In the tuck proxy, the floor polygons must be exactly the facets of Q. Any single floor vertex, floor edge, or floor polygon, together with its incident wall polygons, must form a polyhedral manifold homeomorphic to a disk (called a *waffle pocket*) that is intrinsically convex, meaning that no point has more than 360° of material. Furthermore, every wall polygon must have height at most ε, so that the tuck proxy is within ε of Q. In addition, the algorithm outputs a parameter ε' with $0 < \varepsilon' < \varepsilon$ that lower bounds how far the tuck proxy extends beyond Q, which guarantees no global self intersection up to that distance.

Figure 5 illustrates this step of the algorithm. The main idea is to inset the edges of Q on the tuck side (like the beginning of the formation of the 3D straight skeleton) by an amount ε' small enough that no collision events occur. This insetting bisects all dihedral angles, so in particular, it divides every reflex dihedral angle into two convex dihedral angles. This consequence is the key to how we guarantee that the waffle pockets are intrinsically convex. We then connect these edge offsets around each vertex of Q, which is equivalent to drawing a connected graph on a small sphere around the vertex. We first connect these offsets by a cycle on the sphere. The pocket formed by the floor polygon, two offset edges, and one edge of the cycle has a perimeter of at most 360° because of the strict convexity of the incident polygons and strict convexity of the angle between the offset and the floor. Then, to make the remaining pocket intrinsically convex, we subdivide the cycle by overlaying a regular tetrahedron and triangulating (if necessary). The resulting faces have edge lengths of at most 109.5°, so perimeter at most $328.5° < 360°$.

4 Waffle Pocket Squashing

Given the tuck proxy T and parameter ε', the next step of the algorithm computes a *local squashing* of each waffle pocket of the tuck proxy. Roughly speaking, local squashing consists of adding material between polygons in the waffle pocket until they lie flat and nonoverlapping in the plane. More formally, a local squashing draws each floor and wall polygon of the waffle pocket in the plane subject to only increasing the bottom angles of wall polygons, preserving convexity of the wall polygons, preserving the connectivity between the polygons, leaving no angular gaps between polygons at floor vertices, keeping the squashed wall polygons within ε' of the floor vertex/polygon, and preserving the cyclic order of the wall polygons around the floor vertex/polygon. The top edges of the squashed wall polygons must form a convex polygon in the plane, called the *cell*, so that the *cell perpendiculars* (normal to each cell edge,

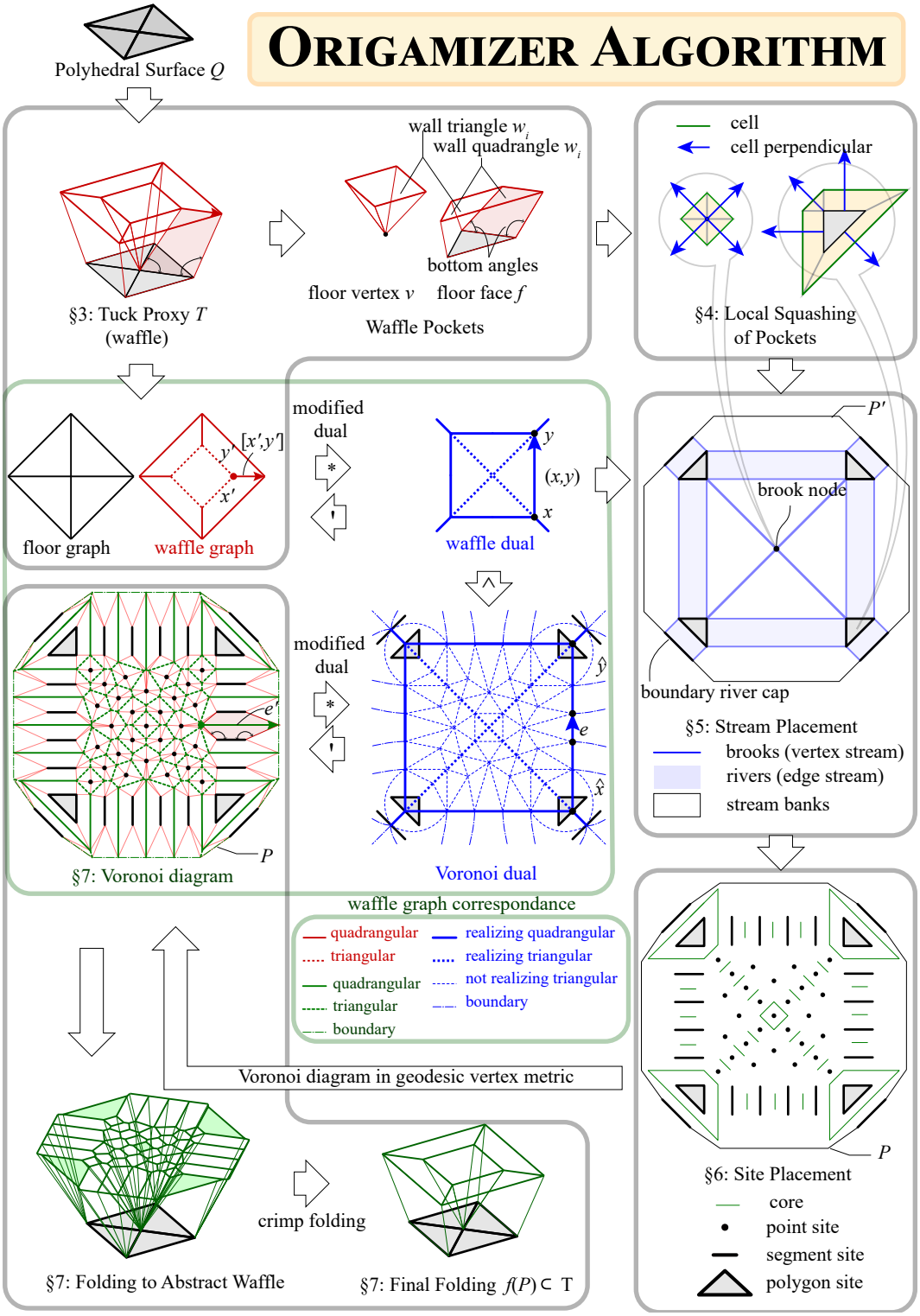


Figure 3 Visual overview of entire Origamizer algorithm, including intermediate data structures.

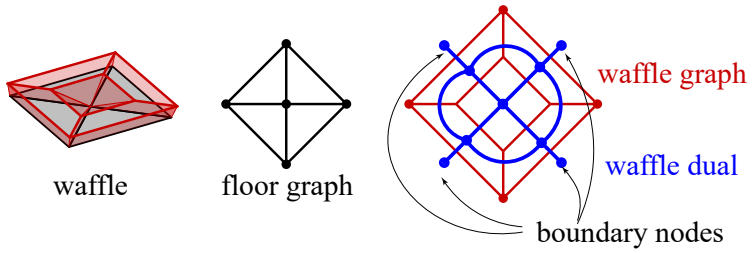


Figure 4 A waffle, the waffle graph, and the waffle dual. Wall faces are red; floor faces are grey.

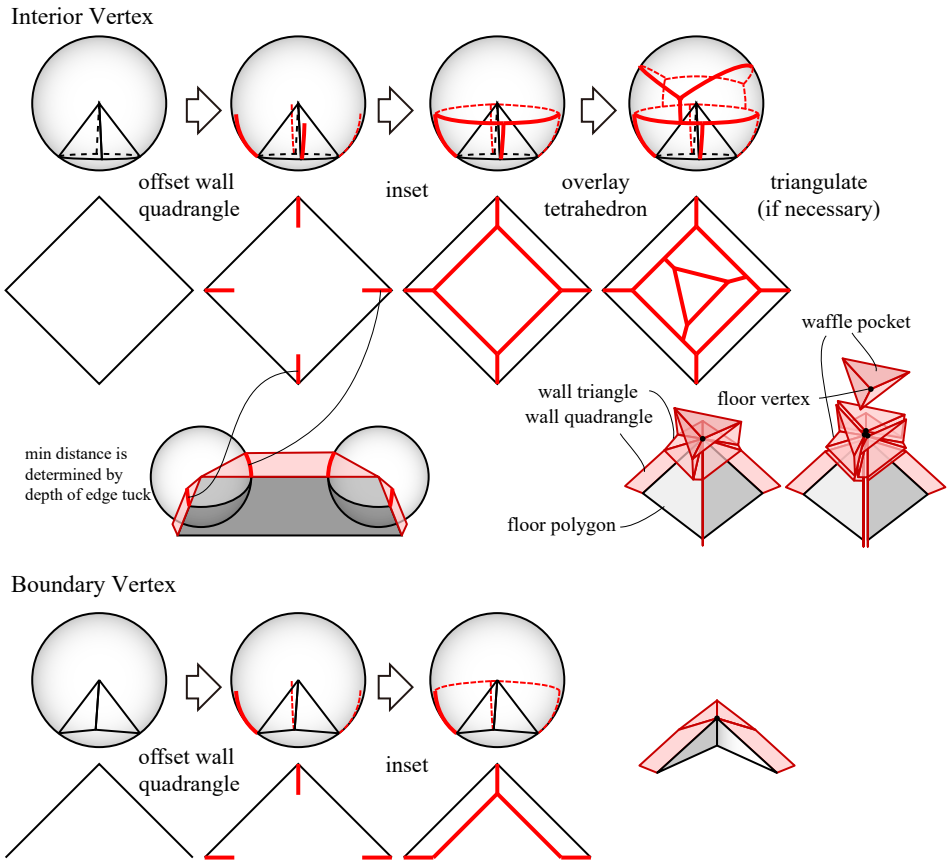


Figure 5 Construction of the tuck proxy for an interior vertex (top) and boundary vertex (bottom). The interior-vertex construction consists of three rows. Top: spherical view of behavior around a vertex; Middle: planar projection of behavior on the sphere; and Bottom: broader 3D view, around an edge and its two endpoints (left) and actual tuck proxy around the vertex (right).

and starting from the midpoint of the bottom of the corresponding squashed wall polygon) proceed counterclockwise around the floor vertex/polygon.

Figures 6 and 7 illustrate the two cases of this step of the algorithm, which get applied to all waffle pockets of the tuck proxy corresponding to floor vertices and floor polygons, respectively. The algorithm first vertex-unfolds [2] the wall polygons into the plane, leaving angular gaps evenly distributed between squashed wall polygons. We then place rays for each gap from the vertex, such that consecutive rays sandwich the squashed wall polygons and form an angle strictly less than 180°. (Specifically, each ray is the angular bisector between

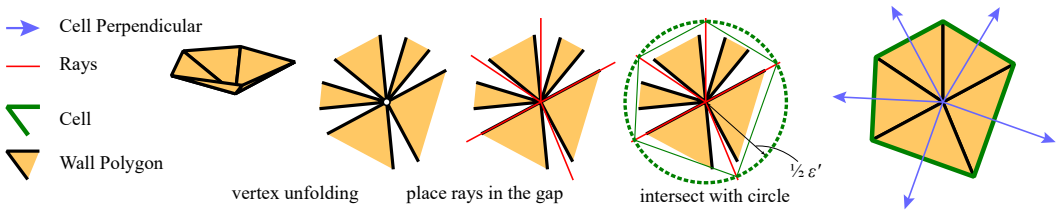


Figure 6 Local squashing algorithm for a waffle pocket corresponding to a floor vertex.

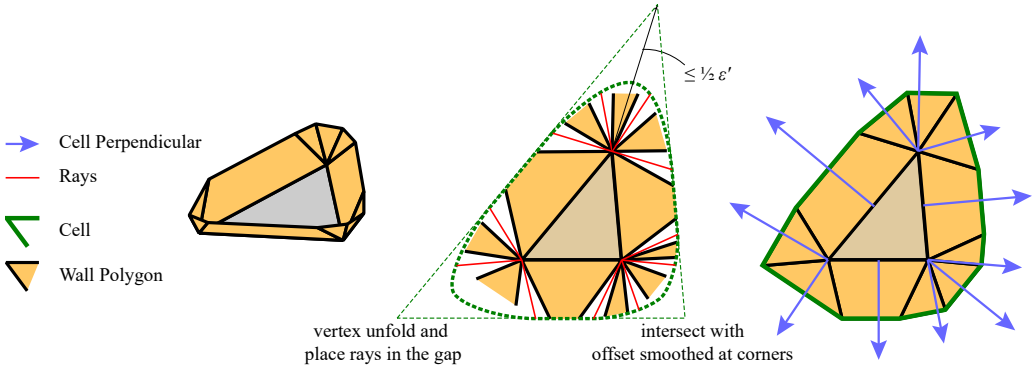


Figure 7 Local squashing algorithm for a waffle pocket corresponding a floor polygon.

the angular bisectors of two consecutive squashed wall polygons, possibly rounded to one of those wall polygon's boundaries.) We connect the intersection of the rays with a smooth convex curve (for the floor vertex case, a circle; and for the floor polygon case, the offset of the floor polygon smoothed at the corner with quadratic Bézier curve), placed close enough to and surrounding the floor vertex/polygon, to obtain a strictly convex cell. The resulting squashed wall polygons are bounded by the rays and the cell, and thus the bottom angles only increase. Because we use an offset curve to intersect the rays, wall quadrangles attached to a floor polygon squash into trapezoids.

5 Placing Streams

Given the tuck proxy T, parameter ε', and a local squashing of its waffle pockets, the next step of the algorithm computes a *stream placement*, which consists of a convex polygon P' and a mapping from various features of the waffle dual into geometric structures drawn on P'. Roughly speaking, each waffle dual node maps to either a point in P' (for a waffle pocket corresponding to a floor vertex) or an isometric embedding of a facet of Q in P' (for a waffle pocket corresponding to that floor polygon); and each waffle dual edge maps to a *stream* – a C^1 curve consisting of line segments and circular arcs, possibly *thickened* orthogonally. Specifically, if the waffle dual edge corresponds to a wall quadrangle, then the stream connects two equal-length edges of the placed floor polygons, and the stream is thickened by an amount equal to that common edge length, forming a *river* (as in [6, 7]). On the other hand, if the waffle dual edge corresponds to a wall triangle, then the stream connects two points, either placed floor vertices or vertices of placed floor polygons, and the stream has zero thickness, forming a *brook*.

This step of the algorithm also outputs a number $\delta > 0$ that is a lower bound on the "clearance" of the output structures, that is, the critical radius at which a disk Minkowski-

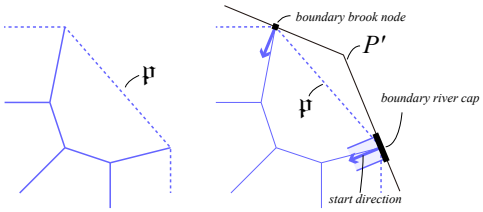


Figure 8 Left: Tutte embedding with outside face $\mathfrak{p}$. Right: Construction of start ray and convex polygon P'.

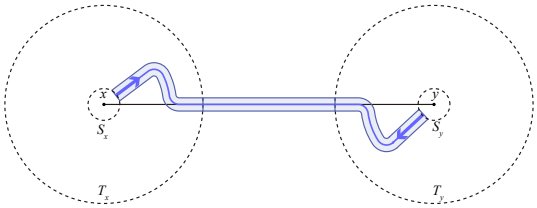


Figure 9 Connecting two embedded waffle dual nodes x and y with a river.

summed with the structures causes a collision event. This lower bound is important for bounding the number of creases in the ultimate Origamizer design.

This step of the algorithm consists of two major parts. In the first part, we embed the waffle dual in the plane using a Tutte embedding [10], and construct a convex polygon P' so that the boundary nodes lie on the boundary of P'; refer to Figure 8. During this construction, we guarantee that every boundary node attached to a river has enough clearance to be thickened parallel to its edge of P', to its desired width, without leaving that edge of P' and without intersecting other boundary nodes.

In the second part, we place floor vertices and polygons at the nodes of the embedded waffle dual graph, and connect them by rivers and brooks, respectively, along the edges of the embedded graph; see Figure 9. The main challenge is that the directions of the edges of the embedded waffle dual graph do not (in general) match the start directions of the streams defined by the cell perpendiculars of the local squashing. We construct a local structure around each embedded waffle dual node to adjust these directions without self-intersection. Specifically, this local structure consists of three nested disks, centered at each waffle dual node x and having radii $r_x < s_x < t_x$, each helping to adjust the directions of streams incident to x; refer to Figure 10. The radius-r_x disk R_x separates the streams from the floor vertex/polygon; the radius-s_x disk S_x bends the streams to meet the disk boundary orthogonally; and the radius-t_x disk T_x twists the streams to match the directions of the incident edges of the embedded waffle dual graph, while carefully avoiding collisions by having k tracks for a degree-k vertex x. In the last twisting step, we rotate the embedded floor polygon so that one edge normal does not require twisting, conceptually cut the disk there, and look at the $T_x - S_x$ annulus in polar view; then we make each connection from inside (p_i) to outside (q_i) in counterclockwise order, using the outermost unused track for counterclockwise (leftward) connections and the innermost unused track for clockwise (rightward) connections. To make the streams C^1 and obtain positive clearance δ, we *fillet* each corner of these streams, replacing each sharp corner by a circular arc of sufficient radius. To guarantee that these disks are local to their corresponding nodes, we scale up the Tutte embedding by a sufficient (but finite) factor.

6 Placing Sites

Given the output from the previous three steps (the tuck proxy T and parameter ε' from Section 3, a local squashing of waffle pockets from Section 4, and a stream placement from Section 5), the next step of the algorithm computes a *site placement*: the final piece of paper P, and a set of point, segment, and polygon *sites* on P. This site placement satisfies several properties which we state in terms of their generalized Voronoi diagram. The Voronoi diagram we use is in the *geodesic vertex metric*, which measures the geodesic (shortest-path)

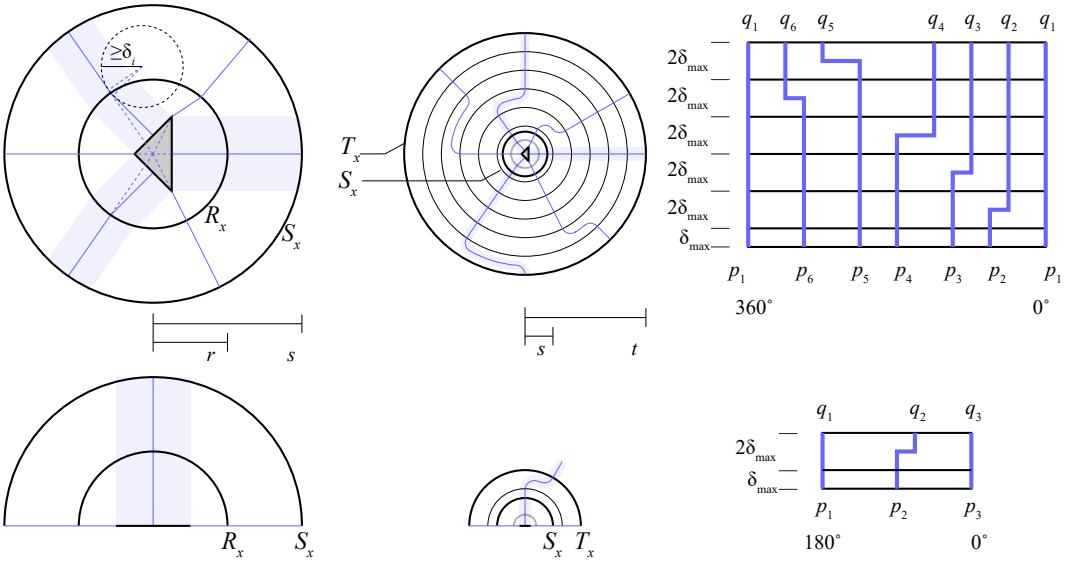


Figure 10 Attaching rivers to a facet of Q. Top: interior case. Bottom: boundary case. Left: filling disks R_x and S_x of radii $r_x < s_x$. Middle: filling the annulus $T_x \setminus S_x$ of outer radius $t_x > s_x$. Right: polar view.

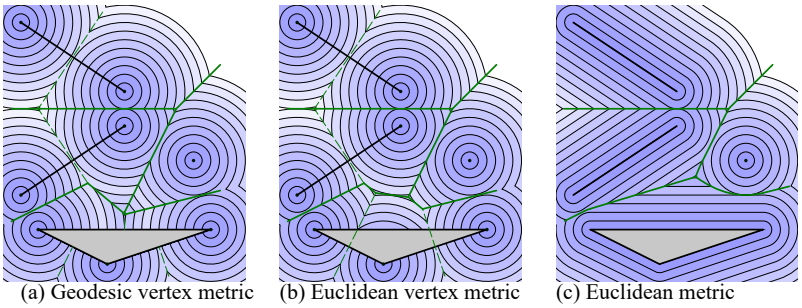
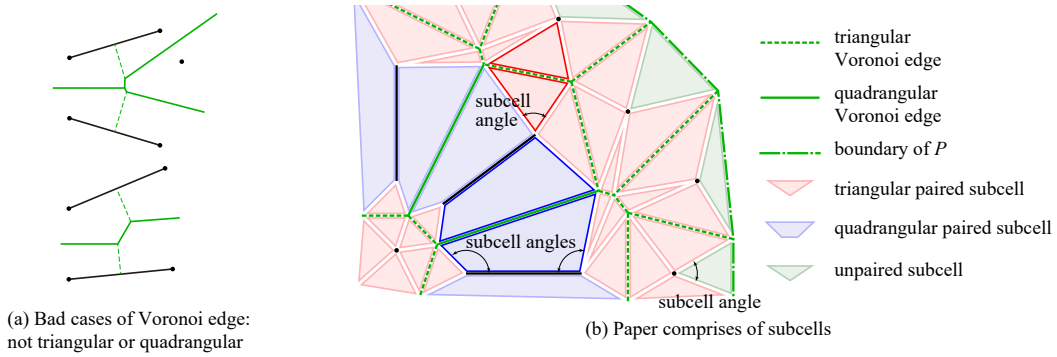


Figure 11 The geodesic vertex metric and the resulting Voronoi diagram of point, segment, and polygon sites, compared with the usual Euclidean metric (where we measure the minimum distance to any point of a site) and an intermediate "Euclidean vertex metric" (where we measure the minimum Euclidean distance to a vertex of a site).

distance between a point of the paper and the nearest vertex of a site, viewing all sites as planar obstacles that cannot be crossed (and thus must be routed around) by a shortest path. Refer to Figure 11.

The site placement requirements are the following:

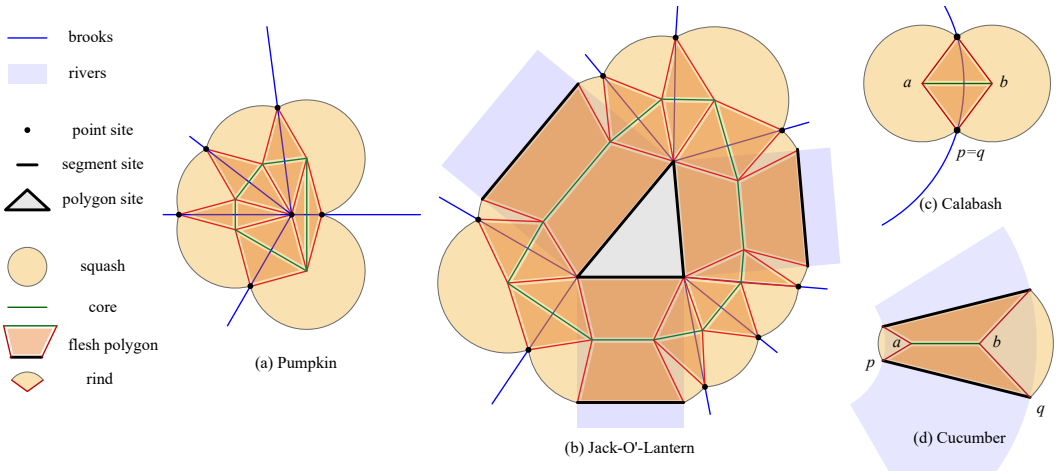
1. The Voronoi diagram can be contracted into the waffle graph of T, i.e., the modified dual of the Voronoi diagram is a supergraph of a subdivision of the waffle dual (Figure 3 middle). Thus each edge of the waffle dual is *realized* by a path of edges in the Voronoi dual, and the corresponding waffle graph edges *realize* the corresponding Voronoi edges.
2. Each Voronoi edge is a straight segment which mirror-reflects its defining site vertices (avoiding cases like Figure 12(a), which can generally happen when using the geodesic vertex metric). As shown in Figure 12(b), we obtain a mirror-reflected pair of triangles or quadrangles called *paired subcells*, where a paired subcell is quadrangular if and only if the Voronoi edge realizes a wall quadrangle.



(a) Bad cases of Voronoi edge: not triangular or quadrangular

(b) Paper comprises of subcells

Figure 12 Triangular and quadrangular edges, subcells, and subcell angles.



(a) Pumpkin

(b) Jack-O'-Lantern

(c) Calabash

(d) Cucumber

Figure 13 Different types of squashes generated by the site placement algorithm: pumpkins around brook nodes, jack-o'-lanterns around placed facets, calabashes along brooks, and cucumbers along rivers.

3. Each Voronoi edge realizing an edge of the waffle graph must have subcell angles (formed with the site) that are at least the bottom angles of the corresponding wall polygons.
4. Every point of the paper P is within ε' (Euclidean) distance of a site vertex.

To guarantee these properties of the Voronoi diagram of the placed sites, we also place a collection of (open set) planar regions called *squashes*, of four different types (see Figure 13): *pumpkins* at brook nodes (points connecting multiple brooks), *jack-o'-lanterns* around placed floor polygons, *calabashes* along brooks, and *cucumbers* along rivers. The sites defining a squash lie on the boundary of the squash. Within each squash, there is a *core*, which is part of the Voronoi diagram of the generating sites, and *flesh polygons*, which are mirror-reflected triangles and quadrangles between sites and the core. We design so that the flesh polygons have angles (against the sites) that are at least the bottom angles of corresponding wall polygons.

A key property is that each squash is the continuous union of geodesic (open) disks centered at points on the core and passing though the nearest vertices of the generating sites. This property ensures that, if there are no other sites in the squash, then the core is guaranteed to be part of the global Voronoi diagram, and thus the flesh polygons will be contained in paired subcells. Thus the algorithm places sites and squashes tightly enough

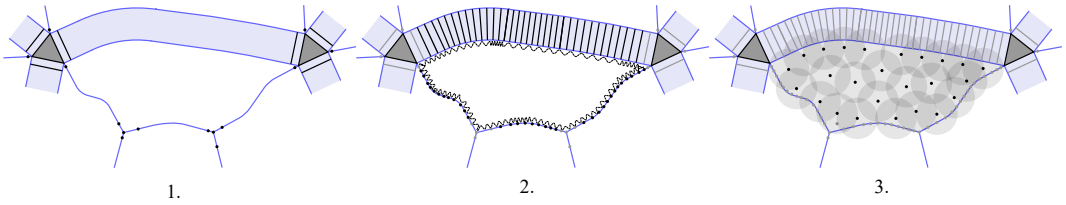


Figure 14 Site placement algorithm overview. 1. Adding sites at brook nodes and placed floor faces. 2. Adding point sites along brooks and segment sites along rivers. 3. Filling stream banks with point sites.

(so any point of P has distance at most ε' to a site) while using the union of squashes as a protective region in which we forbid placing any new sites. Specifically, the site-placement algorithm consists of following three steps; refer to Figure 14.

1. **Node sites:** Add a pumpkin at each brook node, and a jack-o'-lantern at each placed floor polygon. The core of each pumpkin and jack-o'-lantern is exactly the cell of the local squashing of the corresponding waffle pocket.
2. **Stream sites:** Place a sequence of point sites along each brook, and calabashes between consecutive pairs of placed point sites, such that the angle of each flesh polygon equals the bottom angle of the wall triangle of T corresponding to the brook. Place a sequence of segment sites along each river, and cucumbers between consecutive pairs of placed segment sites, such that the angles of each flesh polygon equal the bottom angles of the wall quadrangle of T corresponding to the river. Because streams are (thickened) line segments and circular arcs, we can design calabashes and cucumbers to have mirror symmetry between consecutive sites along each stream. If the gap between two consecutive sites is too big, other streams (and thus stream sites) might intersect the squash. In this case, we subdivide by bisecting the gap, adding an additional site along the stream, which converges to squashes intersecting only the streams they belong to (by the smoothness and positive clearance δ of streams obtained in Section 5). Also subdivide sufficiently so that each point in the squash has distance at most $\frac{1}{2}\varepsilon'$ from the core.
3. **Bank sites:** Repeatedly add a point site wherever there is a point ε' away from the closest site. Here we use that squashes are contained in the Minkowski sum of each site with an ε'-radius disk, so that this process terminates without placing sites on squashes.

We claim that the resulting site placement gives a Voronoi diagram that can contract to the waffle graph of T. This claim follows because, for each stream following the modified dual of the waffle graph of T, there is a sequence of sites that are adjacent to each other through Voronoi edge containing the cores of calabashes or cucumbers, which are sandwiched by flesh triangles or quadrangles, respectively. In the Voronoi dual, such a sequence realizes an edge of waffle dual. (Refer to the waffle graph correspondence in the middle of Figure 3.)

The piece of paper P is defined to be the convex hull of the paired subcells (within P'), or equivalently, the convex hull of the sites and the Voronoi diagram (clipped to P'). Polygon P differs only slightly from P' from the previous step, possibly removing "nonsubcell" portions of P' near its vertices. There may still be regions of P that are not in any paired subcell, which we call *unpaired subcells*; see Figure 12(b). Unpaired subcells are all triangles, with two edges defined by legs of adjacent paired subcells and one edge defined by the boundary of P.

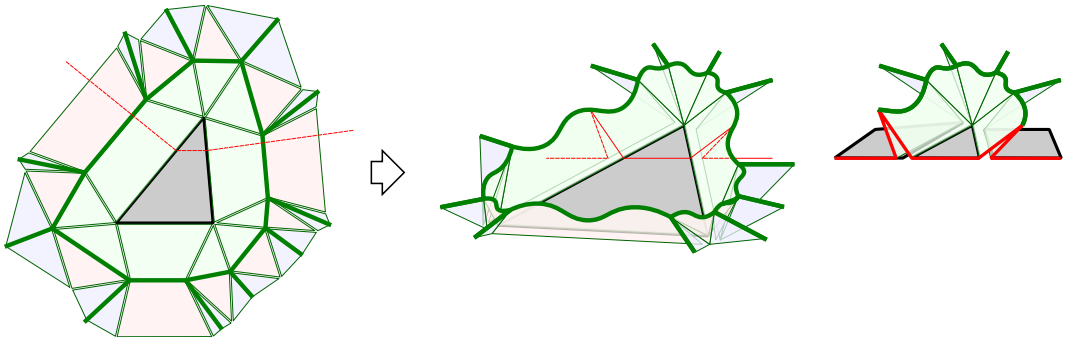


Figure 15 Folding each Voronoi cell (left) into a pocket of an abstract waffle (middle). Each wall is double covered by a pair of mirror-reflected paired subcells as shown in the cross section (right).

7 Voronoi Folding

Given the tuck proxy T and the site placement from Section 6, the final step of the algorithm computes a watertight seamless ε-extra folding of the piece of paper P into Q. In particular, the computed folding contains all facets of Q and lies on the tuck proxy T. We construct the folding by a sequence of *folding steps*, where each folding step treats the folded image resulting from previous steps as the "sheet of paper" to start from (never separating layers of paper that have been brought together by previous steps). In fact, each folding step produces an abstract metric polyhedral complex called an *abstract waffle*, which is topologically equivalent to a waffle and has an intrinsic metric for each floor and wall polygon. The last folding step's abstract waffle embeds directly on the tuck proxy, and thus we can realize the abstract structure isometrically in 3D. Validity of each folding step guarantees a consistent layer ordering in the final folded state (without paper crossing itself).

Initial folding step into abstract waffle: Define the graph G to consist of the following edges: the Voronoi edge of every paired subcell (within P) and the boundary edge of every unpaired subcell. The initial folding step folds P into an abstract waffle A whose waffle graph is G; refer to Figure 15. Specifically, for every Voronoi edge of G, the two paired subcells of P (which are reflections of each other) fold onto each other to doubly cover the corresponding wall polygon of A; and for every boundary edge of G, the unpaired subcell of P fold to singly cover the corresponding wall polygon of A. This folding gives a consistent metric to every wall polygon of the abstract waffle A. In particular, each cell of the Voronoi diagram (within P) becomes a pocket of A.

The floor polygons of A are the placed floor polygons in P. Because the Voronoi diagram realizes the waffle graph of the tuck proxy T, the floor polygons in A are connected together in the same way as the floor polygons in T. In other words, the floor of A is isometric to Q.

The remaining challenge is that A has excess wall polygons and also larger angles compared to T. The main idea of the following steps is to remove excess wall triangles that do not realize the waffle graph of T, reduce the bottom angles of walls realizing the waffle graph of T, and then glue isometric walls together to have a simplified topology. Each of the following folding steps involves folding one or two creases that radiate from a single floor vertex, so they can easily be viewed as foldings of the 1D waffle graph, where edge lengths represent bottom angles and each quadrangular edge is split into two subedges to represent its two bottom angles.

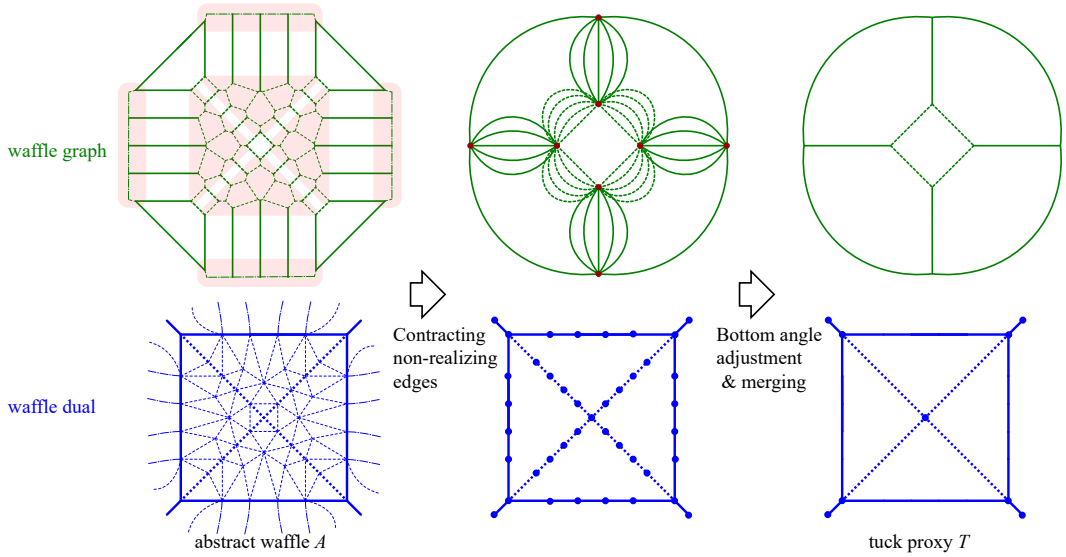


Figure 16 The waffle graph of the abstract waffle, reduced to the waffle graph of the tuck proxy.

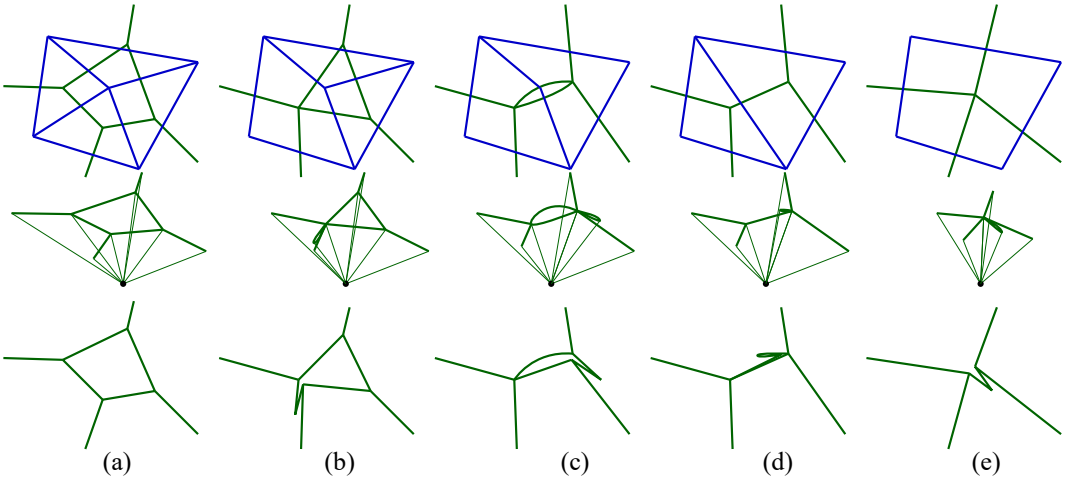


Figure 17 Contraction in Voronoi diagram / deletion in Voronoi dual to form the waffle graph / waffle dual of the tuck proxy.

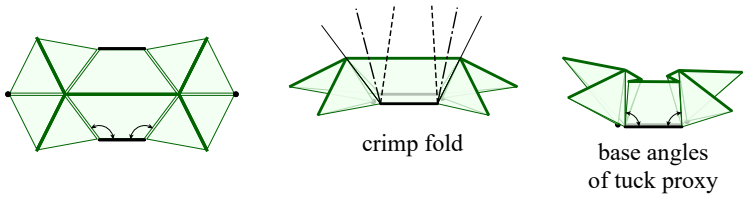


Figure 18 Angle adjustment to fit quadrangular wall of T.

Contracting and merging nonrealizing edges: First we contract each edge of the Voronoi diagram that does not realize an edge of the waffle graph of T (Figure 16 from left to middle). Each edge contraction can be achieved by folding the corresponding wall triangle in half (along an angular bisector) and gluing it against an adjacent wall polygon, possibly wrapping around the walls of the same pocket (Figure 17, (a) $\to$ (b) and (b) $\to$ (c)). If we ever have multiple edges connecting the same two nodes, we can unify the angles to the smallest angle by crimp folding the larger angles (Figure 17, (c) $\to$ (d)), making these multiple edges isometric to each other, and thus the wall polygons can be stacked on top of each other. This fused edge can be contracted again (Figure 17, (d) $\to$ (e)) to collapse a pocket. By applying these folding steps to all edges not realizing the waffle graph of T, we obtain a simplified waffle graph whose dual consists of just the paths of edges realizing each edge of the waffle dual of T.

Merging realizing edges: Each dual path corresponds to a sequence of wall polygons that represent the same edge in the waffle graph. To fuse them together, we crimp their bottom angles to match the bottom angles of the corresponding wall polygons of T; see Figure 18. Once they have the same base angles, we can stack the wall polygons on top of each other and regard them as a single wall polygon (Figure 16 from middle to right). Now each wall polygon can be isometrically embedded into the corresponding wall polygon of T, which gives the resulting folded state being a subset of T. Therefore, the final folded state f is the desired ε-extra folding of Q.

Remaining desired properties: The seamless property follows because the floor polygons are isometric to Q, and no other part of the paper folds strictly interior to any facet of Q. Watertightness follows by considering a point p moving along the boundary of P in counterclockwise order and a point q moving along the boundary of Q in counterclockwise order. The correspondence between p and q is given by (1) if p is on a boundary site, then $f(p) = q$; and (2) if p is between two consecutive boundary sites, then q stays at the corresponding vertex, which stays within distance ε from $f(p)$.

References

1. Erik D. Demaine, Martin L. Demaine, and Joseph S. B. Mitchell. Folding flat silhouettes and wrapping polyhedral packages: New results in computational origami. *Computational Geometry: Theory and Applications*, 16(1):3–21, 2000.
2. Erik D. Demaine, David Eppstein, Jeff Erickson, George W. Hart, and Joseph O'Rourke. Vertex-unfolding of simplicial manifolds. In *Discrete Geometry: In Honor of W. Kuperberg's 60th Birthday*, pages 215–228. Marcer Dekker Inc., 2003.
3. Erik D. Demaine, Sándor P. Fekete, and Robert J. Lang. Circle packing for origami design is hard. In *Origami^5: Proceedings of the 5th International Conference on Origami in Science, Mathematics and Education*, pages 609–626. A K Peters, Singapore, July 2010.
4. Erik D. Demaine and Joseph O'Rourke. *Geometric Folding Algorithms: Linkages, Origami, Polyhedra*. Cambridge University Press, July 2007.
5. Erik D. Demaine and Tomohiro Tachi. Origamizer: A practical algorithm for folding any polyhedron. Manuscript, 2017. URL: `http://erikdemaine.org/papers/Origamizer/`.
6. Robert J. Lang. A computational algorithm for origami design. In *Proceedings of the 12th Annual ACM Symposium on Computational Geometry*, pages 98–105, Philadelphia, PA, May 1996.

7 Robert J. Lang and Erik D. Demaine. Facet ordering and crease assignment in uniaxial bases. In *Origami*^{4}*: Proceedings of the 4th International Conference on Origami in Science, Mathematics, and Education*, Pasadena, California, September 2006.

8 Tomohiro Tachi. Software: Origamizer, 2008. URL: `http://www.tsg.ne.jp/TT/software/`.

9 Tomohiro Tachi. Origamizing polyhedral surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 16(2):298–311, 2010. `doi:10.1109/TVCG.2009.67`.

10 W. T. Tutte. How to draw a graph. *Proceedings of the London Mathematical Society*, 13:743–767, 1963. URL: `http://plms.oxfordjournals.org/cgi/pdf_extract/s3-13/1/743`, `doi:doi:10.1112/plms/s3-13.1.743`.

Computing the Geometric Intersection Number of Curves*†

Vincent Despré[1] and Francis Lazarus[2]

1 **LIP, ENS-Lyon, Lyon, France**
`Vincent.Despre@ens-lyon.fr`
2 **GIPSA-Lab, CNRS, Grenoble, France**
`Francis.Lazarus@gipsa-lab.fr`

Abstract

The geometric intersection number of a curve on a surface is the minimal number of self-intersections of any homotopic curve, i.e., of any curve obtained by continuous deformation. Given a curve c represented by a closed walk of length at most ℓ on a combinatorial surface of complexity n we describe simple algorithms to (1) compute the geometric intersection number of c in $O(n+\ell^2)$ time, (2) construct a curve homotopic to c that realizes this geometric intersection number in $O(n+\ell^4)$ time, (3) decide if the geometric intersection number of c is zero, i.e., if c is homotopic to a simple curve, in $O(n+\ell\log^2\ell)$ time.

To our knowledge, no exact complexity analysis had yet appeared on those problems. An optimistic analysis of the complexity of the published algorithms for problems (1) and (3) gives at best a $O(n+g^2\ell^2)$ time complexity on a genus g surface without boundary. No polynomial time algorithm was known for problem (2). Interestingly, our solution to problem (3) is the first quasi-linear algorithm since the problem was raised by Poincaré more than a century ago. Finally, we note that our algorithm for problem (1) extends to computing the geometric intersection number of two curves of length at most ℓ in $O(n+\ell^2)$ time.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases computational topology, curves on surfaces, combinatorial geodesic

Digital Object Identifier 10.4230/LIPIcs.SoCG.2017.35

1 Introduction

Let S be a surface. Two closed curves $\alpha,\beta:\mathbb{R}/\mathbb{Z}\to S$ are **freely homotopic**, written $\alpha\sim\beta$, if there exists a continuous map $h:[0,1]\times\mathbb{R}/\mathbb{Z}$ such that $h(0,t)=\alpha(t)$ and $h(1,t)=\beta(t)$ for all $t\in\mathbb{R}/\mathbb{Z}$. Assuming the curves are in general position, their number of intersections is

$$|\alpha\cap\beta| = |\{(t,t') \mid t,t'\in\mathbb{R}/\mathbb{Z} \text{ and } \alpha(t)=\beta(t')\}|.$$

Their **geometric intersection number** only depends on their free homotopy classes and is defined as

$$i(\alpha,\beta) = \min_{\alpha'\sim\alpha,\beta'\sim\beta} |\alpha'\cap\beta'|.$$

Likewise, the number of self-intersections of α is given by

$$\frac{1}{2}|\{(t,t') \mid t\neq t'\in\mathbb{R}/\mathbb{Z} \text{ and } \alpha(t)=\alpha(t')\}|,$$

* A full version of the paper is available at `http://arxiv.org/pdf/1511.09327`.
† This work was partially supported by the LabEx PERSYVAL-Lab ANR-11-LABX-0025-01.

and its minimum over all the curves freely homotopic to α is its **geometric self-intersection number** $i(\alpha)$. Note the one half factor that comes from the identification of (t, t') with (t', t).

The geometric intersection number is an important parameter that allows to stratify the set of homotopy classes of curves on a surface. The surface is usually endowed with a hyperbolic metric, implying that each homotopy class is identified by its unique geodesic representative. Extending a former result by Mirzakhani [24], Sapir [31, 25] has recently provided upper and lower bounds for the number of closed geodesics with bounded length and bounded geometric intersection number. Chas and Lalley [6] also proved that the distribution of the geometric intersection number with respect to the word length approaches the Gaussian distribution as the length grows to infinity. Other more experimental results were obtained with the help of a computer to show the existence of length-equivalent homotopy classes with distinct geometric intersection numbers [5]. Hence, for both theoretical and practical reasons various aspects of the computation of geometric intersection numbers have been studied in the past including the algorithmic ones. Nonetheless, all the previous approaches rely on rather complex mathematical arguments and to our knowledge no exact complexity analysis has yet appeared. In this paper, we make our own the words of Dehn who noted that the metric on words (on some basis of the fundamental group of the surface) can advantageously replace the hyperbolic metric [12]. We propose a combinatorial framework that leads to simple algorithms of low complexity to compute the geometric intersection number of curves or to test if this number is zero. Our approach is based on the computation of canonical forms as recently introduced in the purpose of testing whether two curves are homotopic [22, 15]. Canonical forms are instances of combinatorial geodesics who share nice properties with the geodesics of a hyperbolic surface. On such surfaces each homotopy class contains a unique geodesic that moreover minimizes the number of self-intersections. Although a combinatorial geodesic is generally not unique in its homotopy class, it must stay at distance one from its canonical representative and a careful analysis of its structure leads to the first result of the paper.

▶ **Theorem 1.** *Given two curves represented by closed walks of length at most ℓ on an orientable combinatorial surface of complexity n we can compute the geometric intersection number of each curve or of the two curves in $O(n + \ell^2)$ time.*

As usual the complexity of a combinatorial surface stands for its total number of vertices, edges and faces. A key point in our algorithm is the ability to compute the primitive root of a canonical curve c in linear time. This is a curve r that is not homotopic to a proper power of any other curve and such that $c \sim r^k$ for some integer k. We next provide an algorithm to compute an actual curve immersion – its combinatorial description is part of our combinatorial framework – that minimizes the number of self-intersections in its homotopy class.

▶ **Theorem 2.** *Let c be a closed walk of length ℓ in canonical form. We can compute a combinatorial immersion with $i(c)$ crossings in $O(\ell^4)$ time.*

We also propose a nearly optimal algorithm that answers an old problem studied by Poincaré [29, §4]: decide if the geometric intersection number of a curve is null, that is if the curve is homotopic to a simple curve.

▶ **Theorem 3.** *Given a curve represented by a closed walk of length ℓ on an orientable combinatorial surface of complexity n we can decide if the curve is homotopic to a simple curve in $O(n + \ell \log^2 \ell)$ time.*

We emphasize that our results represent significant progress with respect to the state of the art. No precise analysis appeared in the previously proposed algorithms [2, 7, 8, 9, 23, 11, 28, 18] concerning Theorems 1 or 3. An optimistic analysis of what seems the most efficient approach [23, Th. 3.7], although particularly complex, gives at best a quadratic time complexity for computing the geometric intersection number on a genus g surface without boundary, assuming that the curves are primitive. Schaefer et al. [32] propose an efficient computation of the geometric intersection number of curves represented by normal coordinates in a triangulated surface. However, their approach is limited to *simple* input curves. Apart from a recent algorithm by Aretinnes [1], which is restricted to surfaces with *nonempty* boundary, we know of no polynomial time algorithm for Theorem 2. Finally, Theorem 3 states the first quasi-linear algorithm for detecting homotopy classes of simple curves since the problem was raised by Poincaré more than a century ago [29, §4].

Section 2 presents our general simple strategy to compute the geometric intersection number. We introduce our combinatorial framework in Sections 3 - 5. The proof of Theorem 1 is given in the next three sections where the case of non-primitive curves is also treated. The computation of a minimally crossing immersion is presented in Section 9. We finally propose a simple algorithm to detect and embed curves that are homotopic to simple curves (Theorem 3) in Section 10. Due to space limitations most proofs are deferred to the full arXiv version of the paper.

2 Our strategy for counting intersections

Following Poincaré's original approach we represent the surface S as the hyperbolic quotient surface $\mathbb{D}/\Gamma$ where Γ is a discrete group of hyperbolic motions of the Poincaré disk $\mathbb{D}$. We denote by $p : \mathbb{D} \to \mathbb{D}/\Gamma = S$ its universal covering map. Any closed curve $\alpha : \mathbb{R}/\mathbb{Z} \to S$ gives rise to its infinite power $\alpha^\infty : \mathbb{R} \to \mathbb{R}/\mathbb{Z} \to S$ that wraps around α infinitely many times. A **lift** of α is any curve $\tilde{\alpha} : \mathbb{R} \to \mathbb{D}$ such that $p \circ \tilde{\alpha} = \alpha^\infty$ where the parameter of $\tilde{\alpha}$ is defined up to an integer translation (we thus identify the curves $t \mapsto \tilde{\alpha}(t + k)$, $k \in \mathbb{Z}$). Note that $p^{-1}(\alpha)$ is the union of all the images $\Gamma \cdot \tilde{\alpha}$ of $\tilde{\alpha}$ by the motions in Γ. The curve $\tilde{\alpha}$ has two limit points on the boundary of $\mathbb{D}$ which can be joined by a unique hyperbolic line L. The projection $p(L)$ covers infinitely many times the unique geodesic homotopic to α. In particular, the set of pairs of limit points of all lifts of α only depends on the homotopy class of α.

No two motions of Γ have a limit point in common unless they are powers of the same motion. This can be used to show that when α is primitive its lifts are uniquely identified by their limit points [16]. Let α and β be two primitive curves. We fix a lift $\tilde{\alpha}$ of α and denote by $\tau \in \Gamma$ the hyperbolic motion sending $\tilde{\alpha}(0)$ to $\tilde{\alpha}(1)$. Let $\Gamma \cdot \tilde{\beta}$ be the set of lifts of β. We consider the subset of lifts

$$B = \{\tilde{\beta}' \in \Gamma \cdot \tilde{\beta} \mid \text{ the limit points of } \tilde{\beta}' \text{ and } \tilde{\alpha} \text{ alternate along } \partial\mathbb{D}\},$$

and we denote by B/τ the set of equivalence classes of lifts generated by the relations $\tilde{\beta}' \sim \tau(\tilde{\beta}')$.

▶ **Lemma 4** (Reinhart [30]). $i(\alpha, \beta) = |B/\tau|$.

In the ideal situation of hyperbolic geodesics, each intersection point of α and β corresponds precisely to a class in B/τ. When α and β are not geodesic the situation is more ambiguous and their lifts may have multiple intersection points. When dealing with combinatorial geodesics, the situation is more constrained and somehow intermediate between the hyperbolic case and the most general situation. See Figure 1.

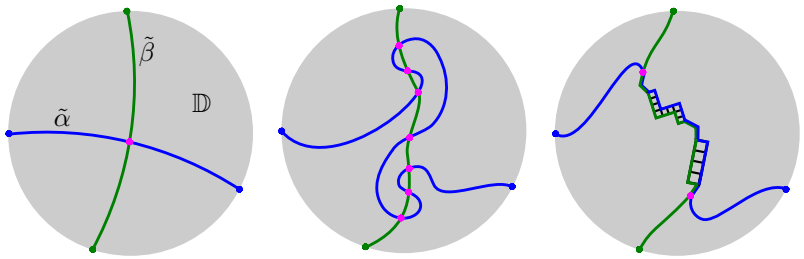


Figure 1 Left, two intersecting hyperbolic lines. Middle, two lifts of non-geodesic curves may intersect several times. Right, lifts of combinatorial geodesics.

Our strategy to compute the geometric intersection number consist of identifying B/τ with certain pairs of homotopic subpaths of α and β. We shall work in a combinatorial framework as described in the next section. In order to mimic at best hyperbolic geometry we restrict our framework to system of quads as introduced in Section 4. The structure of combinatorial geodesics in a system of quads is analysed in Section 5.

3 Combinatorial framework

Combinatorial surfaces. As usual in computational topology, we model a surface by a cellular embedding of a graph G in a compact topological surface S. Such a cellular embedding can be encoded by a **combinatorial surface** composed of the graph G itself together with a rotation system [26] that records for every vertex of the graph the clockwise cyclic order of the incident arcs. The **facial walks** are obtained from the rotation system by the face traversal procedure as described in [26, p.93]. In order to handle surfaces with boundaries we allow every face of G in S to be either an open disk or an annulus (open on one side). In other words G is a cellular embedding in the closure of S obtained by attaching a disk to every boundary of S. We record this information by storing a boolean for every facial walk of G indicating whether the associated face is perforated or not. All the considered graphs may have loop and multiple edges. A directed edge will be called an **arc** and each edge corresponds to two opposite arcs. We denote by a^{-1} the arc opposite to an arc a. **We only consider orientable surfaces in this paper**. Every combinatorial surface Σ can be reduced by first contracting the edges of a spanning tree and then deleting edges incident to distinct faces. The resulting **reduced surface** has a single vertex and a single face. The combinatorial surface Σ and its reduced version encode different cellular embeddings on a same topological surface.

Combinatorial curves. Consider a combinatorial surface with its graph G. A combinatorial curve (or path) c is a walk in G, i.e., an alternating sequence of vertices and arcs, starting and ending with a vertex, such that each vertex in the sequence is the target vertex of the previous arc and the source vertex of the next arc. We generally omit the vertices in the sequence. A combinatorial curve is closed when additionally the first and last vertex are equal. When no confusion is possible we shall drop the adjective combinatorial. The **length** of c is its total number of arc occurrences, which we denote by $|c|$. If c is closed, we write $c(i)$, $i \in \mathbb{Z}/|c|\mathbb{Z}$, for the vertex of index i of c and $c[i, i+1]$ for the arc joining $c(i)$ to $c(i+1)$. For convenience we set $c[i+1, i] = c[i, i+1]^{-1}$ to allow the traversal of c in reverse direction. In order to differentiate the arcs with their occurrences we denote by $[i, i \pm 1]_c$ the corresponding occurrence of the arc $c[i, i \pm 1]$ in $c^{\pm 1}$, where c^{-1} is obtained by traversing $c^1 := c$ in the

opposite direction. More generally, for any non-negative integer ℓ and any sign $\varepsilon \in \{-1, 1\}$, the sequence of indices $(i, i+\varepsilon, i+2\varepsilon, \dots, i+\varepsilon\ell)$ is called an **index path** of c of length ℓ. The index path can be **forward** ($\varepsilon = 1$) or **backward** ($\varepsilon = -1$) and can be longer than c, so that an index may appear more than once in the sequence. We denote this path by $[i \overset{\varepsilon\ell}{\to}]_c$. Its **image path** is given by the arc sequence

$$c[i \overset{\varepsilon\ell}{\to}] := (c[i, i+\varepsilon], c[i+\varepsilon, i+2\varepsilon], \dots, c[\varepsilon(\ell-1), \varepsilon\ell]).$$

The image path of a length zero index path is just a vertex. A **spur** of c is a subsequence of arcs of the form (a, a^{-1}). A closed curve is **contractible** if it is homotopic to a trivial curve (i.e., a curve reduced to a single vertex). We will implicitly assume that a homotopy has fixed endpoints when applied to paths and is free when applied to closed curves.

Combinatorial immersions. A combinatorial curve may be seen as a continuous curve in general position snapped to the graph of the combinatorial surface. When doing so several parts of the continuous curve may be mapped to the same edge. In order not to lose information, one needs to record their ordering. Following the notion of a combinatorial set of loops as in [10], we thus define a **combinatorial immersion** of a set $\mathcal{C}$ of combinatorial curves as the data for each arc a in G of a left-to-right order $\preceq_a$ over all the occurrences of a or a^{-1} in the curves of $\mathcal{C}$. The only requirement is that opposite arcs should be associated with inverse orders. Let A_v be the set of occurrences of a or a^{-1} in the curves of $\mathcal{C}$, where a runs over all arcs of G with origin v. A combinatorial immersion induces for each vertex v of G a circular order $\preceq_v$ over A_v; if $a_1, \dots, a_k$ is the clockwise-ordered list of arcs of G with origin v, then $\preceq_v$ is the cyclic concatenation of the orders $\preceq_{a_1}, \dots, \preceq_{a_k}$.

Combinatorial crossings. Given an immersion $\mathcal{I}$ of two combinatorial closed curves c and d we define a **double point** of (c, d) as a pair of indices $(i, j) \in \mathbb{Z}/|c|\mathbb{Z} \times \mathbb{Z}/|d|\mathbb{Z}$ such that $c(i) = d(j)$. Likewise, a double point of c is a pair $(i, j) \in \mathbb{Z}/|c|\mathbb{Z} \times \mathbb{Z}/|c|\mathbb{Z}$ with $i \neq j$ and $c(i) = c(j)$. The double point (i, j) is a **crossing** in $\mathcal{I}$ if the pairs of arc occurrences $([i-1, i]_c, [i, i+1]_c)$ and $([j-1, j]_d, [j, j+1]_d)$ are linked in the $\preceq_{c(i)}$-order, i.e., if they appear in the cyclic order

$$\cdots [i, i-1]_c \cdots [j, j-1]_d \cdots [i, i+1]_c \cdots [j, j+1]_d \cdots,$$

with respect to $\preceq_{c(i)}$ or the opposite order. An analogous definition holds for a self-crossing of a single curve, taking $c = d$ in the above definition. The number of crossings of c and d and of self-crossings of c in $\mathcal{I}$ is denoted respectively by $i_{\mathcal{I}}(c, d)$ and $i_{\mathcal{I}}(c)$. Note that every combinatorial immersion can be realized by continuous curves with the same number of crossings. The **combinatorial self-crossing number** of c, denoted by $i(c)$, is the minimum of $i_{\mathcal{I}}(c')$ over all the combinatorial immersions $\mathcal{I}$ of any combinatorial curve c' freely homotopic[1] to c. The **combinatorial crossing number** of two combinatorial curves c and d is defined the same way taking into account crossings between c and d only. It is easily proved that $i(c)$ and $i(c, d)$ coincide with the geometric (self-)intersection number of continuous realizations of c and d.

[1] Homotopy of combinatorial curves can be defined equivalently via their continuous realizations or thanks to combinatorial homotopies based on elementary moves. See Appendix A in the arXiv version.

4 Systems of quads

Reduction to a system of quads. Let Σ be a combinatorial surface with negative Euler characteristic. We describe the construction of a system of quads for a surface without boundary. A similar construction applies when Σ has perforated faces. Following Lazarus and Rivaud [22] we start putting Σ into a standard form called a **system of quads** by Erickson and Whittlesey [15]. After reducing Σ to a surface Σ' with a single vertex v and a single face f this system of quads is obtained by adding a vertex w at the center of f, adding edges between w and all occurrences of v in the facial walk of f, and finally deleting the edges of Σ'. The graph of the resulting system of quads, called the **radial graph** [22], is bipartite. It contains two vertices, namely v and w, and $4g$ edges, where g is the genus of Σ. All its faces are quadrilaterals. Note that this system of quads is deduced from Σ by a sequence of edge contractions, deletions or insertions, including one edge subdivision to insert w. Every cycle of Σ can be modified accordingly to give a homotopic cycle in the system of quads.

▶ **Lemma 5** ([13, 22]). *Let n be the number of edges of Σ. The above construction of a system of quads can be performed in $O(n)$ time so that for every closed curve c of length ℓ in Σ, we can compute in $O(\ell)$ time a homotopic curve of length at most 2ℓ in the system of quads.*

For the rest of the paper we shall assume that **all surfaces have negative Euler characteristic**. The case of surfaces with non-negative Euler characteristic is handled in the proof of Theorem 1.

Diagrams. A **disk diagram** over the combinatorial surface Σ is a combinatorial sphere Δ with one perforated face together with a labelling of the arcs of Δ by the arcs of Σ such that

1. opposite arcs receive opposite labels,
2. the facial walk of each non-perforated face of Δ is labelled by the facial walk of some non-perforated face of Σ.

The diagram is **reduced** when no edge of Δ is incident to two non-perforated faces labelled by the same facial walk (with opposite orientations) of Σ. An **annular diagram** is defined similarly by a combinatorial sphere with two distinct perforated faces. A vertex of a diagram that is not incident to any perforated face is said **interior**.

▶ **Lemma 6** (van Kampen, See [15, Sec. 2.4]). *A cycle of Σ is contractible if and only if it is the label of the facial walk of the perforated face of a reduced disk diagram over Σ. Two cycles are freely homotopic if and only if the facial walks of the two perforated faces of a reduced annular diagram over Σ are labelled by these two cycles respectively.*

Note that two non-perforated faces that are adjacent and consistently oriented in a reduced diagram are labelled by adjacent faces that are consistently oriented in Σ. Moreover, the degree of an interior vertex of the diagram is a multiple of the degree of the corresponding vertex in Σ. In the sequel, **all the considered diagrams will be supposed reduced**.

Spurs, brackets and canonical curves. Thanks to Lemma 5 we may assume that our combinatorial surface Σ is a system of quads. Moreover, the construction of this system of quads with the assumption on the Euler characteristic implies that **all interior vertices have degree at least 8**. Following the terminology of Erickson and Whittlesey [15], we define the **turn** of a pair of arcs (a_1, a_2) sharing their origin vertex v as the number of face

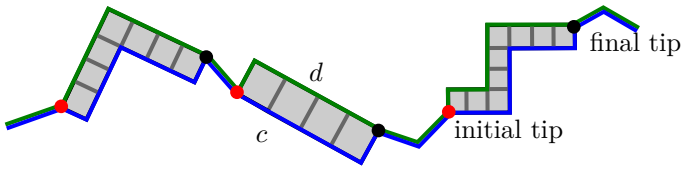


Figure 2 A disk diagram for two homotopic paths c and d composed of paths and staircases.

corners between a_1 and a_2 in clockwise order around v. Hence, if v is a vertex of degree d in Σ, the turn of (a_1, a_2) is an integer modulo d that is zero when $a_1 = a_2$. The **turn sequence** of a subpath $(a_i, a_{i+1}, \ldots, a_{i+j-1})$ of a closed curve of length ℓ is the sequence of $j+1$ turns of $(a_{i+k}^{-1}, a_{i+k+1})$ for $-1 \leq k < j$, where indices are taken modulo ℓ. The subpath may have length ℓ, thus leading to a sequence of $\ell+1$ turns. Note that the turn of $(a_{i+k}^{-1}, a_{i+k+1})$ is zero precisely when (a_{i+k}, a_{i+k+1}) is a spur. A **bracket** is any subpath whose turn sequence has the form 12^*1 or $\bar{1}\bar{2}^*\bar{1}$, where t^* stands for a possibly empty sequence of turns t and $\bar{x}$ stands for $-x$. It follows from Lemma 6 and a simple combinatorial Gauss-Bonnet theorem [17] that

▶ **Theorem 7** ([17, 15]). *A nontrivial contractible closed curve on a system of quads must have either a spur or four brackets. Moreover, if the curve is the label of the boundary walk (i.e., of the facial walk of the perforated face) of a disk diagram with at least one interior vertex, then the curve must have either a spur or five brackets.*

Lazarus and Rivaud [22] have introduced a canonical form for every nontrivial free homotopy class of closed curves in a system of quads. In particular, two curves are freely homotopic if and only if their **canonical forms** are equal (up to a circular shift of their vertex indices). It was further characterized by Erickson and Whittlesey [15] in terms of turns and brackets. It is the unique homotopic curve that contains no spurs or brackets and whose turning sequence contains no -1's and contains at least one turn that is not -2.

▶ **Theorem 8** ([22, 15]). *The canonical form of a closed curve of length ℓ on a system of quads can be computed in $O(\ell)$ time.*

5 Geodesics

The canonical form is an instance of a **combinatorial geodesic**, i.e., a curve that contains no spurs or brackets. The canonical form is the rightmost homotopic geodesic. The definitions of a geodesic and of a canonical form extend trivially to paths. In particular, the canonical form of a path is the unique homotopic path that contains no spurs or brackets and whose turning sequence contains no -1's. Although we cannot claim in general the uniqueness of geodesics in a homotopy class, homotopic geodesics are almost equal and have the same length. Specifically, define a **(quad) staircase** as a planar sequence of quads obtained by stitching an alternating sequence of rows and columns of quads to get the shape of a staircase. See Fig. 2. Assuming that the staircase goes up from left to right, we define the **initial tip** of a quad staircase as the lower left vertex of the first quad in the sequence. The **final tip** is defined as the upper right vertex of the last quad.

A **closed staircase** is obtained by identifying the two vertical arcs incident to the initial and final tips of a staircase.

▶ **Theorem 9.** *Let c, d be two non-trivial homotopic combinatorial geodesics. If c, d are closed curves, then they label the two boundary cycles of an annular diagram composed of a*

unique closed staircase or of an alternating sequence of paths (possibly reduced to a vertex) and quad staircases connected through their tips. Likewise, if c, d are paths, then the closed curve $c \cdot d^{-1}$ labels the boundary of a disk diagram composed of an alternating sequence of paths (possibly reduced to a vertex) and quad staircases connected through their tips.

▶ **Corollary 10.** *With the hypothesis of Theorem 9, c and d have equal length which is minimal among homotopic curves. Moreover, c and d have no index path whose image path is contractible.*

The next two lemmas follow directly from the characterization of geodesics and canonical forms in terms of spurs, brackets and turns.

▶ **Lemma 11.** *The image path of any index path of a combinatorial geodesic is geodesic. If the combinatorial geodesic is in canonical form, so is the image path.*

▶ **Lemma 12.** *Likewise, any power c^k of a combinatorial closed geodesic c is also a combinatorial geodesic. Moreover, if c is in canonical form, so is c^k.*

6 Crossing Double-paths

Let c, d be two combinatorial closed curves on a combinatorial surface. A **double-path** of (c, d) of length ℓ is a pair of forward index paths $([i \overset{\ell}{\to}]_c, [j \overset{\ell}{\to}]_d)$ with the same image path $c[i \overset{\ell}{\to}] = d[j \overset{\ell}{\to}]$. If $\ell = 0$ then the double path is just a double point. A double path of c is defined similarly, taking $c = d$ and assuming $i \neq j$. The next Lemma follows from Lemma 11.

▶ **Lemma 13.** *Let $[i \overset{\ell}{\to}]_c$ and $[j \overset{k}{\to}]_d$ be forward index paths of two canonical curves c and d such that the image paths $c[i \overset{\ell}{\to}]$ and $d[j \overset{k}{\to}]$ are homotopic. Then $k = \ell$ and $([i \overset{\ell}{\to}]_c, [j \overset{\ell}{\to}]_d)$ is a double path.*

A double path $([i \overset{\ell}{\to}]_c, [j \overset{\ell}{\to}]_d)$ gives rise to a sequence of $\ell + 1$ double points $(i + k, j + k)$ for $k \in [0, \ell]$. A priori a double point could occur several times in this sequence. The next two lemmas claim that this is not possible when the curves are primitive. Recall that a curve is **primitive** if its homotopy class cannot be expressed as a proper power of another class.

▶ **Lemma 14.** *A double path of a primitive combinatorial curve c cannot contain a double point more than once in its sequence. In particular, a double path of c must be strictly shorter than c.*

▶ **Lemma 15.** *Let c and d be two non-homotopic primitive combinatorial curves. A double path of (c, d) cannot contain a double point more than once in its sequence. Moreover, the length of a double path of (c, d) must be less than $|c| + |d| - 1$.*

A double path whose index paths cannot be extended is said **maximal**. As an immediate consequence of Lemmas 14 and 15 we have:

▶ **Corollary 16.** *The maximal double paths of a primitive curve or of two primitive curves in canonical form induce a partition of the double points of the curves.*

Let (i, j) and $(i + \ell, j + \ell)$ be the first and the last double points of a maximal double path of (c, d), possibly with $c = d$. When $\ell \geq 1$ the arcs $c[i, i - 1]$, $d[j, j - 1]$, $c[i, i + 1]$ must be pairwise distinct because canonical curves have no spurs, and similarly for the three arcs $c[i + \ell, i + \ell + 1]$, $d[j + \ell, j + \ell + 1]$, $c[i + \ell, i + \ell - 1]$. We declare the maximal double

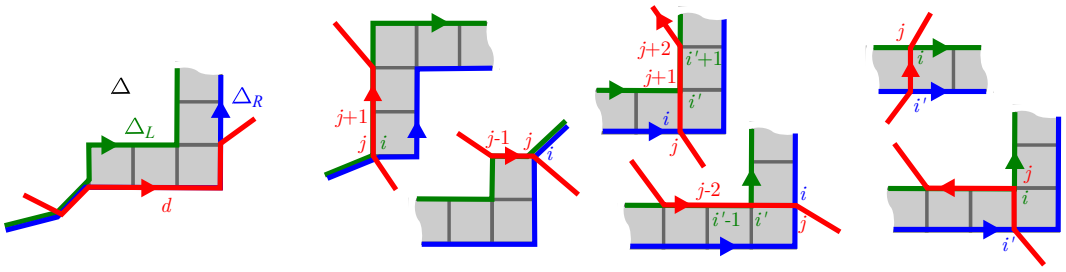


Figure 3 Left, a typical crossing double path in $\mathcal{D}_+$. Middle, four configurations in $\mathcal{D}_0$. Right, two configurations in $\mathcal{D}_-$.

path to be a **crossing double path** if the circular ordering of the first three arcs at $c(i)$ and the circular ordering of the last three arcs at $c(i+\ell)$ are either both clockwise or both counterclockwise with respect to the rotation system of the system of quads. When $\ell = 0$, that is when the maximal double path is reduced to the double point (i, j), we require that the arcs $c[i, i-1], d[j, j-1], c[i, i+1], d[j, j+1]$ are pairwise distinct and appear in this circular order, or its opposite, around the vertex $c(i) = d(j)$.

7 Counting intersections combinatorially: the primitive case

Let c, d be primitive combinatorial curves such that d is canonical and let c_R and c_L^{-1} be the canonical curves homotopic to c and c^{-1} respectively. We denote by Δ the annular diagram with left and right boundaries Δ_L and Δ_R corresponding to c_L and c_R given by Theorem 9. We consider the following set of double paths:

- $\mathcal{D}_+$ is the set of crossing double paths of positive length of c_R and d,
- $\mathcal{D}_0$ is the set of crossing double paths (i, j) of zero length of c_R and d such that either
  - the two boundaries of Δ coincide at $\Delta_L(i) = \Delta_R(i)$ and $d[j-1, j] = c_L[i-1, i]$ or $d[j, j+1] = c_L[i, i+1]$, or
  - one of $d[j, j-1]$ or $d[j, j+1]$ is the label of a spoke $(\Delta_R(i), \Delta_L(i'))$ of Δ and $d[j-2, j-1] = c_L[i'-1, i']$ in the first case or $d[j+1, j+2] = c_L[i', i'+1]$ in the other case.
- $\mathcal{D}_-$ is the set of crossing double paths $([i \xrightarrow{\ell}]_{c_L^{-1}}, [j \xrightarrow{\ell}]_d)$ $(\ell \geq 0)$ of c_L^{-1} and d such that *none* of the following situations occurs:
  - the two boundaries of Δ coincide at $\Delta_L^{-1}(i) = \Delta_R(i')$ and $d[j-1, j] = c_R[i'-1, i']$,
  - the two boundaries of Δ coincide at $\Delta_L^{-1}(i+\ell) = \Delta_R(i')$ and $d[j+\ell, j+\ell+1] = c_R[i', i'+1]$,
  - $d[j-1, j]$ is the label of a spoke $(\Delta_L^{-1}(i), \Delta_R(i'))$ of Δ and $d[j-2, j-1] = c_R[i'-1, i']$,
  - $d[j+\ell, j+\ell+1]$ is the label of a spoke $(\Delta_L^{-1}(i+\ell), \Delta_R(i'))$ of Δ and $d[j+\ell+1, j+\ell+2] = c_R[i', i'+1]$.

Those definitions allow the case $c \sim d$, recalling that the index paths of a double path of c must be distinct by definition. Figure 3 depicts some configurations.

Referring to Section 2, we view the underlying surface of the system of quads Σ as a quotient $\mathbb{D}/\Gamma$ of the Poincaré disk. The system of quads lifts to a quadrangulation of $\mathbb{D}$ and the lifts of a combinatorial curve in Σ are combinatorial bi-infinite paths in this quadrangulation. By Lemma 12, if the combinatorial curve is geodesic (resp. canonical) so are its lifts. In this case, each lift is simple by Corollary 10. We fix a lift $\tilde{c_R}$ of c_R and consider the set B/τ of Lemma 4 corresponding to the classes of lifts of d whose limit points alternate with the limit points of $\tilde{c_R}$ along $\partial\mathbb{D}$.

▶ **Proposition 17.** *B/τ is in 1-1 correspondence with the disjoint union $\mathcal{D}_+ \cup \mathcal{D}_0 \cup \mathcal{D}_-$.*

The proof relies on a careful analysis of canonical paths inside a disk diagram.

8 Non-primitive curves and proof of Theorem 1

In order to finish the proof of Theorem 1, we need to handle the case of non-primitive curves. Thanks to canonical forms, computing the primitive root of a curve becomes extremely simple[2].

▶ **Lemma 18.** *Let c be a combinatorial curve of length $\ell > 0$ in canonical form. A primitive curve d such that c is homotopic to d^k for some integer k can be computed in $O(\ell)$ time.*

Proof. By Theorem 8, we may assume that c and d are in canonical form. By Lemma 12, the curve d^k is also in canonical form. The uniqueness of the canonical form implies that $c = d^k$, possibly after some circular shift of d. It follows that d is the smallest prefix of c such that c is a power of this prefix. It can be found in $O(\ell)$ time using a variation of the Knuth-Morris-Pratt algorithm to find the smallest period of a word [21]. ◀

The geometric intersection number of non-primitive curves is related to the geometric intersection number of their primitive roots. The next result is part of the folklore although we could only find references in some relatively recent papers.

▶ **Proposition 19** ([11, 18])**.** *Let c and d be primitive curves and let p, q be positive integers. Then,*

$$i(c^p) = p^2 \times i(c) + p - 1 \quad \text{and} \quad i(c^p, d^q) = \begin{cases} 2pq \times i(c) & \text{if } c \sim d \text{ or } c \sim d^{-1}, \\ pq \times i(c,d) & \text{otherwise.} \end{cases}$$

Proof of Theorem 1. Let c, d and Σ be the two combinatorial curves and the combinatorial surface as in the Theorem. We first assume that the Σ has negative Euler characteristic. We can compute the canonical forms of c, c^{-1} and d in $O(\ell)$ time after $O(n)$ time preprocessing by Lemma 5. Thanks to Lemma 18 we can further determine primitive curves c' and d' and integers p, q such that $c \sim c'^p$ and $d \sim d'^p$ in $O(\ell)$ time. We then use the formulas in Proposition 19 to deduce $i(c,d)$ and $i(c)$ from $i(c',d')$ and $i(c')$. We can thus assume that c and d are primitive. According to Proposition 17 and Lemma 4, we have

$$i(c,d) = |\mathcal{D}_+| + |\mathcal{D}_0| + |\mathcal{D}_-|$$

The set $\mathcal{D}_+$ can be constructed in $O(\ell^2)$ time. Indeed, since the maximal double paths of c and d form disjoint sets of double points by Corollary 16, we just need to traverse the grid $\mathbb{Z}/|c|\mathbb{Z} \times \mathbb{Z}/|d|\mathbb{Z}$ and group the double points into maximal double paths. Those correspond to diagonal segments in the grid that can be computed in time proportional to the size of the grid. We can also determine which double paths are crossing in the same amount of time. Likewise, we can construct the sets $\mathcal{D}_0, \mathcal{D}_-$ in $O(\ell^2)$ time. We can also compute $i(c)$ in quadratic time using that $i(c,c) = 2i(c)$.

If Σ is a sphere or a disk, then every curve is contractible and $i(c,d) = i(c) = 0$. If Σ is a cylinder, then every two curves can be made non crossing so that $i(c,d) = 0$ while $i(c) = p-1$. Finally, if Σ is a torus, the radial graph of the system of quads can be decomposed into two loops α, β such that $c \sim \alpha^x \cdot \beta^y$ and $d \sim \alpha^{x'} \cdot \beta^{y'}$. We may then use the classical formulas: $i(c) = \gcd(x,y) - 1$ and $i(c,d) = |\det\big((x,y),(x',y')\big)|$. ◀

[2] Compare with the short-lex straight normal form and its use by Epstein and Holt [14, Sec. 3.2].

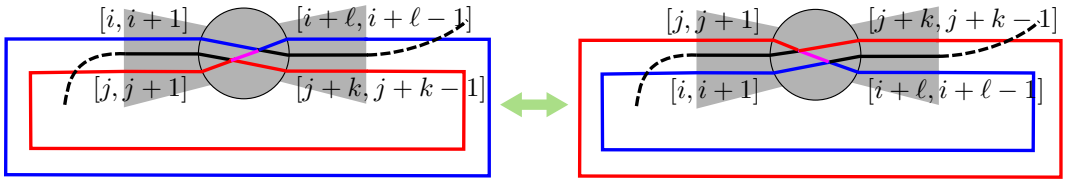


Figure 4 Right, the realization of the bigon $([i \xrightarrow{\ell}], [j \xrightarrow{k}])$ when $j = i + \ell$ and Condition 2 in the definition of a singular bigon is not satisfied. The small purple part is at the same time the beginning of the red side of the bigon and the end of the blue side. Swapping this bigon does not reduce the number of crossings.

9 Computing a minimal immersion

In the subsequent sections we deal with the self-intersection number of a single curve. We thus drop the subscript c to denote an index path $[i \xrightarrow{\ell}]$ or an arc occurrence $[i, i+1]$.

Bigons and monogons. A **bigon** of an immersion $\mathcal{I}$ of c is a pair of index paths $([i \xrightarrow{\ell}], [j \xrightarrow{k}])$ whose **sides** $c[i \xrightarrow{\ell}]$ and $c[j \xrightarrow{k}]$ have strictly positive lengths, are homotopic, and whose **tips** (i, j) and $(i+\ell, j+k)$ are combinatorial crossings for $\mathcal{I}$. A **monogon** of $\mathcal{I}$ is an index path $[i \xrightarrow{\ell}]$ of strictly positive length such that $(i, i+\ell)$ is a combinatorial crossing and the image path $c[i \xrightarrow{\ell}]$ is contractible. In agreement with the terminology of Hass and Scott [19], a bigon $([i \xrightarrow{\ell}], [j \xrightarrow{k}])$ is said **singular** if

1. its two index paths have disjoint interiors, i.e., they do not share any arc occurrence;
2. when $j = i + \ell$ the following arc occurrences

$$[i, i-1], [j, j-1], [i, i+1], [j+k, j+k+1], [j, j+1], [j+k, j+k-1]$$

   do *not* appear in this order or its opposite in the circular ordering induced by $\mathcal{I}$ at $c(j)$;
3. when $i = j + k$ the following arc occurrences

$$[i, i-1], [j, j-1], [i+\ell, i+\ell-1], [i, i+1], [i+\ell, i+\ell+1], [j, j+1]$$

   do *not* appear in this order or its opposite in the circular ordering induced by $\mathcal{I}$ at $c(i)$.

Remark that when the above conditions 2 and 3 are not satisfied, the bigon maps to a non-singular bigon in the continuous realization of $\mathcal{I}$. See Figure 4.

When the bigon is singular we can **swap** its two sides by exchanging the two arc occurrences $[i+p, i+p+1]$ and $[j+\varepsilon p, i+\varepsilon(p+1)]$ in $\mathcal{I}$, for $0 \leq p < \ell$ and $k = \varepsilon\ell$. By performing the swap on a continuous realization of $\mathcal{I}$, then projecting back to a combinatorial version, we obtain the following

▶ **Lemma 20.** *Swapping the two sides of a singular bigon of an immersion of a geodesic primitive curve decreases its number of crossings by at least two.*

Hence, by swapping singular bigons we may decrease the number of crossings until there is no more singular bigons. Since a combinatorial immersion of a primitive geodesic c cannot have a monogon by Corollary 10, it follows from the next theorem that the resulting immersion has no excess crossing.

▶ **Theorem 21** (Hass and Scott [19, Th. 4.2])**.** *An immersion of a primitive curve has excess crossing if and only if it contains a monogon or a singular bigon.*

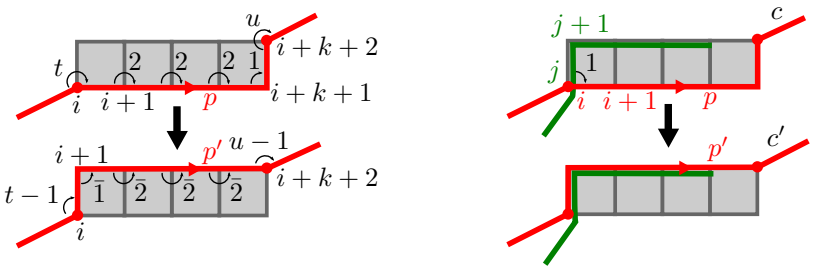


Figure 5 Left, the arc $[i, i+1]$ is switchable. Right, a switch may avoid a crossing.

In the full version of the paper it is shown that we can find a singular bigon of a given combinatorial immersion of a curve of length ℓ with excess intersection in $O(\ell^2)$ time. This allows to conclude the proof of Theorem 2 since such an immersion contains $O(\ell^2)$ crossings.

10 The unzip algorithm

We now turn to the original problem of Poincaré [29, §4], deciding whether a given curve c is homotopic to a simple curve. In the affirmative we know by Lemma 20 and Theorem 21 that some geodesic homotopic to c must have a (combinatorial) **embedding**, i.e., an immersion without crossings. Rather than swapping the sides of a singular bigon as in Lemma 20 we can choose to switch one side along the other side. This will also decrease the number of crossings if the bigon contains no other interior bigons. By considering interior-most bigons only, we can thus enforce a given edge of c to stay fixed as we remove crossings. This suggests an incremental computation of an embedding in which the image of the first arc occurrence is left unchanged: we assume that c is canonical and consider the trivial embedding of its first arc occurrence $[0, 1]$. We next insert the successive arc occurrences incrementally to obtain an embedding of the path formed by the already inserted arcs. When inserting the occurrence $[i, i+1]$ we need to compare its left-to-right order with each already inserted arc occurrence β of its supporting arc. If $\beta \neq [0, 1]$ we can use the comparison of the occurrence $[i-1, i]$ with the occurrence γ preceding β (or succeeding β if it is a backward occurrence). If $[i-1, i]$ and γ have the same supporting arc, we just propagate their relative order to $[i, i+1]$ and β. Otherwise, we use the circular ordering of the supporting arcs of $[i-1, i]$, γ and $[i, i+1]$ in order to conclude. When $\beta = [0, 1]$, we cannot use the occurrence preceding $[0, 1]$ as it is not yet inserted. We rather compare $[i, i+1]$ and $[0, 1]$ as follows. In the Poincaré disk, we consider two lifts $\tilde{d}_i$ and $\tilde{d}_0$ of c such that $\tilde{d}_i[i, i+1] = \tilde{d}_0[0, 1]$. We decide to insert $[i, i+1]$ to the left (right) of $[0, 1]$ if one of the limit points of $\tilde{d}_i$ lies to the left (right) of $\tilde{d}_0$.

After comparing $[i, i+1]$ with all the occurrences of its supporting arc, we can insert it in the correct place. If no crossings were introduced this way, we proceed with the next occurrence $[i+1, i+2]$. It may happen, however, that no matter how we insert $[i, i+1]$ in the left-to-right order of its supporting arc, the resulting immersion of $[0 \overset{i+1}{\to}]$ will have a combinatorial crossing. In order to handle this case, we first check if $[i, i+1]$ is **switchable**, i.e., if for some $k \geq 0$ and some turns t, u the subpath $p := c[i \overset{k+2}{\to}]$ has turn sequence $t2^k1u$ and the index path $[i \overset{k+2}{\to}]$ does not contain the arc occurrence $[0, 1]$. When $[i, i+1]$ is switchable we can switch p to a new subpath p' with turn sequence $(t-1)\bar{1}\bar{2}^k(u-1)$ such that p and p' bound a diagram composed of a single horizontal staircase. See Figure 5.

We then insert the arc occurrence $[i, i+1]$ and proceed with the algorithm using c' in place of c. The successive switches in the course of the computation untangle c incrementally and we call our embedding procedure the **unzip algorithm**.

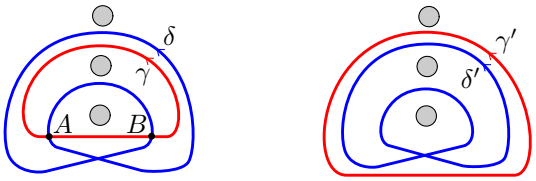


Figure 6 The plain circles represent non-contractible curves. The two curves γ and δ on the left have homotopic disjoint curves γ' and δ'. They thus have excess intersection although there is no singular bigon between the two. If $A = \gamma(0) = \delta(0)$ and $B = \gamma(u) = \delta(v)$ we nonetheless have $\delta|_{[0,v]} \sim \gamma|_{[0,1+u]}$ where $\gamma|_{[0,1+u]}$ is the concatenation of γ with $\gamma|_{[0,u]}$. In particular, $\gamma|_{[0,1+u]}$ wraps more that once around γ.

▶ **Lemma 22.** *The unzip algorithm applied to a canonical primitive curve c of length ℓ can be implemented to run in $O(\ell \log^2 \ell)$ time.*

▶ **Proposition 23.** *If $i(c) = 0$ the unzip algorithm returns an embedding of a geodesic homotopic to c.*

The proof of the proposition is far from trivial. Assuming that the the unzip algorithm returns an immersion with crossings, the rough idea is to show that c has two lifts in $\mathbb{D}$ whose limit points alternate along $\partial\mathbb{D}$.

Proof of Theorem 3. Let c be a combinatorial curve of length ℓ on a combinatorial surface of size n. We compute its canonical form in $O(n + \ell)$ time and check in linear time that c is primitive. In the negative, we conclude that either c is contractible, hence reduced to a vertex, or that c has no embedding by Proposition 19. In the affirmative, we apply the unzip algorithm to compute an immersion $\mathcal{I}$ of some geodesic c' homotopic to c. According to Proposition 23, we have $i(c) = 0$ if and only if $\mathcal{I}$ has no crossings. This is easily verified in $O(n + \ell)$ time by checking for each vertex v of the system of quads that the set of paired arc occurrences with v as middle vertex form a well-parenthesized sequence with respect to the local ordering $\prec_v$ induced by $\mathcal{I}$. We conclude the proof thanks to Lemma 22. ◀

A related problem was tackled by Chang *et al.* [4, Th. 8.2] who can decide if a given closed walk on a combinatorial surface has an embedding. In their formulation, though, the combinatorial path is fixed and they only look for the existence of a combinatorial immersion without crossing. They suggest a linear time complexity for this problem and it seems likely that we could also eliminate the $\log^2 \ell$ factor in our complexity.

11 Concluding remarks

The existence of a singular bigon claimed in Theorem 21 relies on Theorem 4.2 of Hass and Scott [19]. As noted by the authors themselves this result is "surprisingly difficult to prove". Except for this result and the recourse to some hyperbolic geometry in the general strategy of Section 2, our algorithms and proofs are purely combinatorial. Concerning Theorem 21, the existence of an immersion without bigon could be achieved in our combinatorial viewpoint by showing that if an immersion has bigons, then one of them can be swapped to reduce the number of crossings. One difficulty is that such bigons need not be singular as shown by our example in Figure 6.

If those swappable bigons could be found easily this would provide an algorithm to compute a minimally crossing immersion of two curves by iteratively swapping bigons as

in Section 9. Note that in the approaches based on Reidemeister-like moves by de Graff and Schrijver [11] or by Paterson [28], the number of moves required to reach a minimal configuration is unknown. Even though Chang and Erickson [3] conjectured that a minimal configuration could be reached in a quadratic number of moves, it would remain to construct the corresponding sequence of moves efficiently. Comparatively, the number of bigon swapping would be just half the excess crossing of a given immersion. We would thus obtain a polynomial time algorithm for computing a minimally crossing immersion of two curves (there is an exponential time algorithm by a result of Neumann-Coto [27, Prop. 2.2]).

It would be interesting to see if the unzip algorithm of Section 10 yields minimally crossing curves even with curves that are not homotopic to simple curves, thus improving Theorem 2. It is also tempting to check whether the unzip algorithm applies to compute the geometric intersection number of two curves rather than a single curve. Finally, say that two curves are in the **same configuration** if there is an ambient isotopy of the surface where they live that brings one curve to the other. It was shown by Neumann-Coto [27] that every minimally crossing immersion is in the configuration of shortest geodesics for some Riemaniann metric μ, but Hass and Scott [20] gave counterexamples to the fact that we could always choose μ to be hyperbolic. *Is there an algorithm to construct or detect combinatorial immersions that have a realization in the configuration of geodesics for some hyperbolic metric?*

Acknowledgements. We thank the anonymous referees for their insightful comments.

References

1 Chris Arettines. A combinatorial algorithm for visualizing representatives with minimal self-intersection. *J. Knot Theor. Ramif.*, 24(11):1–17, 2015.

2 Joan S. Birman and Caroline Series. An algorithm for simple curves on surfaces. *J. London Math. Soc.*, 29(2):331–342, 1984.

3 Hsien-Chih Chang and Jeff Erickson. Untangling planar curves. In *Proc. 32nd Int'l Symp. Comput. Geom. (SoCG)*, volume 51, pages 29:1–15, 2016.

4 Hsien-Chih Chang, Jeff Erickson, and Chao Xu. Detecting weakly simple polygons. In *Proc. 26th ACM-SIAM Symp. Discrete Alg. (SODA)*, pages 1655–1670, 2015.

5 Moira Chas. Self-intersection numbers of length-equivalent curves on surfaces. *Exp. Math.*, 23(3):271–276, 2014.

6 Moira Chas and Steven P. Lalley. Self-intersections in combinatorial topology: statistical structure. *Invent. Math.*, 188(2):429–463, 2012.

7 David R. J. Chillingworth. Simple closed curves on surfaces. *Bull. London Math. Soc.*, 1(3):310–314, 1969.

8 David R. J. Chillingworth. Winding numbers on surfaces. II. *Math. Ann.*, 199(3):131–153, 1972.

9 Marshall Cohen and Martin Lustig. Paths of geodesics and geometric intersection numbers: I. In *Combinatorial group theory and topology*, volume 111 of *Ann. Math. Stud.*, pages 479–500. Princeton Univ. Press, 1987.

10 Éric Colin de Verdière and Francis Lazarus. Optimal System of Loops on an Orientable Surface. *Discrete Comput. Geom.*, 33(3):507–534, 2005.

11 Maurits de Graaf and Alexander Schrijver. Making curves minimally crossing by Reidemeister moves. *J. Com. Theory B*, 70(1):134–156, 1997.

12 Pierre De La Harpe. Topologie, théorie des groupes et problèmes de décision: célébration d'un article de max dehn de 1910. *Gazette des mathématiciens*, 125:41–75, 2010.

13 Tamal K. Dey and Sumanta Guha. Transforming Curves on Surfaces. *J. Comput. and Syst. Sci.*, 58(2):297–325, 1999.

14 David Epstein and Derek Holt. The linearity of the conjugacy problem in word-hyperbolic groups. *Int'l J. Algebr. Comput.*, 16(02):287–305, 2006.

15 Jeff Erickson and Kim Whittelsey. Transforming curves on surfaces redux. In *Proc. 24th ACM-SIAM Symp. Discrete Alg. (SODA)*, 2013.

16 Benson Farb and Dan Margalit. *A primer on mapping class groups*. Princeton Univ. Press, 2012.

17 Steve M. Gersten and Hamish B. Short. Small cancellation theory and automatic groups. *Invent. Math.*, 102:305–334, 1990.

18 Daciberg L. Gonçalves, Elena Kudryavtseva, and Heiner Zieschang. An algorithm for minimal number of (self-)intersection points of curves on surfaces. In *Proc. Seminar on Vector and Tensor Analysis*, volume 26, pages 139–167, 2005.

19 Joel Hass and Peter Scott. Intersections of curves on surfaces. *Isr. J. Math.*, 51(1-2):90–120, 1985.

20 Joel Hass and Peter Scott. Configurations of curves and geodesics on surfaces. *Geometry and Topology Monographs*, 2:201–213, 1999.

21 Donald E. Knuth, James H. Morris, Jr, and Vaughan R. Pratt. Fast pattern matching in strings. *SIAM J. Comput.*, 6(2):323–350, 1977.

22 Francis Lazarus and Julien Rivaud. On the homotopy test on surfaces. In *Proc. 53rd IEEE Symp. Found. Comput. Sci. (FOCS)*, pages 440–449, 2012.

23 Martin Lustig. Paths of geodesics and geometric intersection numbers: II. In *Combinatorial group theory and topology*, volume 111 of *Ann. of Math. Stud.*, pages 501–543. Princeton Univ. Press, 1987.

24 Maryam Mirzakhani. Growth of the number of simple closed geodesies on hyperbolic surfaces. *Ann. Math.*, pages 97–125, 2008.

25 Maryam Mirzakhani. Counting mapping class group orbits on hyperbolic surfaces. Preprint arxiv:1601.03342, January 2016. URL: `http://arxiv.org/pdf/1601.03342`.

26 Bojan Mohar and Carsten Thomassen. *Graphs on Surfaces*. Studies in the Mathematical Sciences. Johns Hopkins Univ. Press, 2001.

27 Max Neumann-Coto. A characterization of shortest geodesics on surfaces. *Algebr. Geom. Topol.*, 1:349–368, 2001.

28 J. M. Paterson. A combinatorial algorithm for immersed loops in surfaces. *Topol. Appl.*, 123(2):205–234, 2002.

29 Henri Poincaré. Cinquième complément à l'analysis situs. *Rendiconti del Circolo Matematico di Palermo*, 18(1):45–110, 1904.

30 Bruce L. Reinhart. Algorithms for Jordan curves on compact surfaces. *Ann. Math.*, pages 209–222, 1962.

31 Jenya Sapir. Bounds on the number of non-simple closed geodesics on a surface. Preprint arxiv:1505.07171, May 2015. URL: `http://arxiv.org/abs/1505.07171`.

32 Marcus Schaefer, Eric Sedgwick, and Daniel Stefankovic. Computing Dehn twists and geometric intersection numbers in polynomial time. In *CCCG*, pages 111–114, 2008.

Topological Analysis of Nerves, Reeb Spaces, Mappers, and Multiscale Mappers*†

Tamal K. Dey[1], Facundo Mémoli[2], and Yusu Wang[3]

1 **Department of Computer Science and Engineering, The Ohio State University, Columbus, OH, USA**
tamaldey@cse.ohio-state.edu

2 **Department of Mathematics and Department of Computer Science and Engineering, The Ohio State University, Columbus, OH, USA**
memoli@math.osu.edu

3 **Department of Computer Science and Engineering, The Ohio State University, Columbus, OH, USA**
yusu@cse.ohio-state.edu

Abstract

Data analysis often concerns not only the space where data come from, but also various types of maps attached to data. In recent years, several related structures have been used to study maps on data, including Reeb spaces, mappers and multiscale mappers. The construction of these structures also relies on the so-called *nerve* of a cover of the domain.

In this paper, we aim to analyze the topological information encoded in these structures in order to provide better understanding of these structures and facilitate their practical usage.

More specifically, we show that the one-dimensional homology of the nerve complex $N(\mathcal{U})$ of a path-connected cover $\mathcal{U}$ of a domain X cannot be richer than that of the domain X itself. Intuitively, this result means that no new H_1-homology class can be "created" under a natural map from X to the nerve complex $N(\mathcal{U})$. Equipping X with a pseudometric d, we further refine this result and characterize the classes of $H_1(X)$ that may survive in the nerve complex using the notion of *size* of the covering elements in $\mathcal{U}$. These fundamental results about nerve complexes then lead to an analysis of the H_1-homology of Reeb spaces, mappers and multiscale mappers.

The analysis of H_1-homology groups unfortunately does not extend to higher dimensions. Nevertheless, by using a map-induced metric, establishing a Gromov-Hausdorff convergence result between mappers and the domain, and interleaving relevant modules, we can still analyze the persistent homology groups of (multiscale) mappers to establish a connection to Reeb spaces.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Topology, nerves, mapper, multiscale mapper, Reeb spaces

Digital Object Identifier 10.4230/LIPIcs.SoCG.2017.36

1 Introduction

Data analysis often concerns not only the space where data come from, but also various types of information attached to data. For example, each node in a road network can contain information about the average traffic flow passing this point, a node in protein-protein interaction network can be associated with biochemical properties of the proteins involved.

* A full version of the paper is available at https://arxiv.org/abs/1703.07387.
† This work was partially supported by National Science Foundation under grant CCF-1526513.

Such information attached to data can be modeled as maps defined on the domain of interest; note that the maps are not necessarily $\mathbb{R}^d$-valued, e.g, the co-domain can be $\mathbb{S}^1$. Hence understanding data benefits from analyzing maps relating two spaces rather than a single space with no map on it.

In recent years, several related structures have been used to study general maps on data, including Reeb spaces [9, 11, 13, 18], mappers (and variants) [4, 8, 21] and multiscale mappers [10]. More specifically, given a map $f : X \to Z$ defined on a topological space X, the Reeb space R_f w.r.t. f (first studied for piecewise-linear maps in [13]), is a generalization of the so-called Reeb graph for a scalar function which has been used in various applications [2]. It is the quotient space of X w.r.t. an equivalence relation that asserts two points of X to be equivalent if they have the same function value and are connected to each other via points of the same function value. All equivalent points are collapsed into a single point in the Reeb space. Hence R_f provides a way to view X from the perspective of f.

The Mapper structure, originally introduced in [21], can be considered as a further generalization of the Reeb space. Given a map $f : X \to Z$, it also considers a cover $\mathcal{U}$ of the co-domain Z that enables viewing the structure of f at a coarser level. Intuitively, the equivalence relation between points in X is now defined by whether points are within the same connected component of the pre-image of a cover element $U \in \mathcal{U}$. Instead of a quotient space, the mapper takes the nerve complex of the cover of X formed by the connected components of the pre-images of all elements in $\mathcal{U}$ (i.e, the cover formed by those equivalent points). Hence the mapper structure provides a view of X from the perspective of both f and a cover of the co-domain Z.

Finally, both the Reeb space and the mapper structures provide a fixed snapshot of the input map f. As we vary the cover $\mathcal{U}$ of the co-domain Z, we obtain a family of snapshots at different granularities. The *multiscale mapper* [10] describes the sequence of the mapper structures as one varies the granularity of the cover of Z through a sequence of covers of Z connected via cover maps.

New work. While these structures are meaningful in that they summarize the information contained in data, there has not been any qualitative analysis of the precise information encoded by them with the only exception of [4] and [14] [1]. In this paper, we aim to analyze the *topological information* encoded by these structures, so as to provide better understanding of these structures and facilitate their practical usage [12, 17]. In particular, the construction of the mapper and multiscale mapper use the so-called *nerve* of a cover of the domain. To understand the mappers and multiscale mappers, we first provide a quantitative analysis of the topological information encoded in the nerve of a reasonably well-behaved cover for a domain. Given the generality and importance of the nerve complex in topological studies, this result is of independent interest.

More specifically, in Section 3, we first obtain a general result that relates the one dimensional homology H_1 of the nerve complex $N(\mathcal{U})$ of a path-connected cover $\mathcal{U}$ (where each open set contained is path-connected) of a domain X to that of the domain X itself. Intuitively, this result says that no new H_1-homology classes can be "created" under a natural map from X to the nerve complex $N(\mathcal{U})$. Equipping X with a pseudometric d, we further

[1] Carrière and Oudot [4] analyzed certain persistence diagram of mappers induced by a real-valued function, and provided a characterization for it in terms of the persistence diagram of the corresponding Reeb graph. Gasparovic et al [14] provides full description of the persistence homology information encoded in the *intrinsic Čech complex* (a special type of nerve complex) of a metric graph.

refine this result and quantify the classes of $H_1(X)$ that may survive in the nerve complex (Theorem 21, Section 4). This demarcation is obtained via a notion of *size* of covering elements in $\mathcal{U}$. These fundamental results about nerve complexes then lead to an analysis of the H_1-homology classes in Reeb spaces (Theorem 27), mappers and multiscale mappers (Theorem 29). The analysis of H_1-homology groups unfortunately does not extend to higher dimensions. Nevertheless, we can still provide an interesting analysis of the persistent homology groups for these structures (Theorem 36, Section 5). During this course, by using a map-induced metric, we establish a Gromov-Hausdorff convergence between the mapper structure and the domain. This offers an alternative to [18] for defining the convergence between mappers and the Reeb space, which may be of independent interest.

All missing proofs in what follows are deferred to the full version of this paper on arXiv.

2 Topological background and motivation

Space, paths, covers. Let X denote a path connected topological space. Since X is path connected, there exists a path $\gamma : [0,1] \to X$ connecting every pair of points $\{x, x'\} \in X \times X$ where $\gamma(0) = x$ and $\gamma(1) = x'$. Let $\Gamma_X(x, x')$ denote the set of all such paths connecting x and x'. These paths play an important role in our definitions and arguments.

By a cover of X we mean a collection $\mathcal{U} = \{U_\alpha\}_{\alpha \in A}$ of open sets such that $\bigcup_{\alpha \in A} U_\alpha = X$. A cover $\mathcal{U}$ is *path connected* if each U_α is path connected. In this paper, we consider only path connected covers.

Later to define maps between X and its nerve complexes, we need X to be *paracompact*, that is, every cover $\mathcal{U}$ of X has a subcover $\mathcal{U}' \subseteq \mathcal{U}$ so that each point $x \in X$ has an open neighborhood contained in *finitely many* elements of $\mathcal{U}'$. Such a cover $\mathcal{U}'$ is called *locally finite*. From now on, we assume X *to be compact* which implies that it is paracompact too.

▶ **Definition 1** (Simplicial complex and maps)**.** A simplicial complex K with a vertex set V is a collection of subsets of V with the condition that if $\sigma \in 2^V$ is in K, then all subsets of σ are in K. We denote the geometric realization of K by $|K|$. Let K and L be two simplicial complexes. A map $\phi : K \to L$ is *simplicial* if for every simplex $\sigma = \{v_1, v_2, \ldots, v_p\}$ in K, the simplex $\phi(\sigma) = \{\phi(v_1), \phi(v_2), \ldots, \phi(v_p)\}$ is in L.

▶ **Definition 2** (Nerve of a cover)**.** Given a cover $\mathcal{U} = \{U_\alpha\}_{\alpha \in A}$ of X, we define the *nerve* of the cover $\mathcal{U}$ to be the simplicial complex $N(\mathcal{U})$ whose vertex set is the index set A, and where a subset $\{\alpha_0, \alpha_1, \ldots, \alpha_k\} \subseteq A$ spans a k-simplex in $N(\mathcal{U})$ if and only if $U_{\alpha_0} \cap U_{\alpha_1} \cap \ldots \cap U_{\alpha_k} \neq \emptyset$.

Maps between covers. Given two covers $\mathcal{U} = \{U_\alpha\}_{\alpha \in A}$ and $\mathcal{V} = \{V_\beta\}_{\beta \in B}$ of X, a *map of covers* from $\mathcal{U}$ to $\mathcal{V}$ is a set map $\xi : A \to B$ so that $U_\alpha \subseteq V_{\xi(\alpha)}$ for all $\alpha \in A$. *By a slight abuse of notation we also use* ξ *to indicate the map* $\mathcal{U} \to \mathcal{V}$. Given such a map of covers, there is an induced simplicial map $N(\xi) : N(\mathcal{U}) \to N(\mathcal{V})$, given on vertices by the map ξ. Furthermore, if $\mathcal{U} \xrightarrow{\xi} \mathcal{V} \xrightarrow{\zeta} \mathcal{W}$ are three covers of X with the intervening maps of covers between them, then $N(\zeta \circ \xi) = N(\zeta) \circ N(\xi)$ as well. The following simple result is useful.

▶ **Proposition 3** (Maps of covers induce contiguous simplicial maps [10])**.** *Let* $\zeta, \xi : \mathcal{U} \to \mathcal{V}$ *be any two maps of covers. Then, the simplicial maps* $N(\zeta)$ *and* $N(\xi)$ *are contiguous.*

Recall that two simplicial maps $h_1, h_2 : K \to L$ are *contiguous* if for all $\sigma \in K$ it holds that $h_1(\sigma) \cup h_2(\sigma) \in L$. In particular, contiguous maps induce identical maps at the homology level [19]. Let $H_k(\cdot)$ denote the k-dimensional homology of the space in its argument. This homology is *singular* or *simplicial* depending on if the argument is a topological space or a

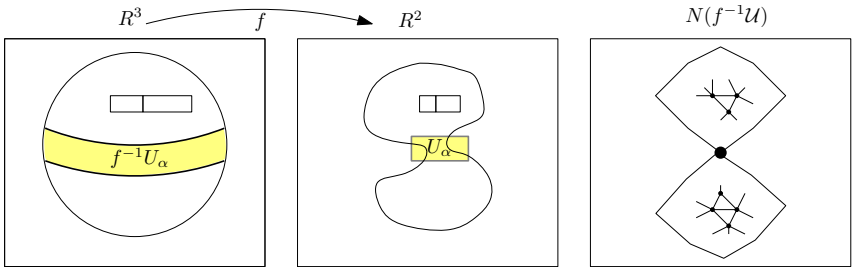


Figure 1 The map $f : \mathbb{S}^2 \subset \mathbb{R}^3 \to \mathbb{R}^2$ takes the sphere to $\mathbb{R}^2$. The pullback of the cover element U_α makes a band surrounding the equator which causes the nerve $N(f^{-1}\mathcal{U})$ to pinch in the middle creating two 2-cycles. This shows that the map $\phi_* : X \to N(*)$ may not induce a surjection in H_2.

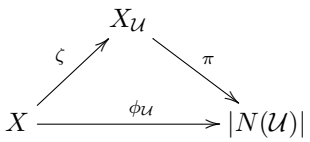
simplicial complex respectively. All homology groups in this paper are defined over the field $\mathbb{Z}_2$. Proposition 3 implies that the map $H_k(N(\mathcal{U})) \to H_k(N(\mathcal{V}))$ arising out of a cover map can be deemed canonical.

3 Surjectivity in H_1-persistence

In this section we first establish a map $\phi_\mathcal{U}$ between X and the geometric realization $|N(\mathcal{U})|$ of a nerve complex $N(\mathcal{U})$. This helps us to define a map $\phi_{\mathcal{U}*}$ from the singular homology groups of X to the simplicial homology groups of $N(\mathcal{U})$ via the singular homology of $|N(\mathcal{U})|$. The famous nerve theorem [3, 16] says that if the elements of $\mathcal{U}$ intersect only in contractible spaces, then $\phi_\mathcal{U}$ is a homotopy equivalence and hence $\phi_{\mathcal{U}*}$ leads to an isomorphism between $H_*(X)$ and $H_*(N(\mathcal{U}))$. The contractibility condition can be weakened to a *homology ball* condition to retain the isomorphism between the two homology groups [16]. In absence of such conditions of the cover, simple examples exist to show that $\phi_{\mathcal{U}*}$ is neither a monophorphism (injection) nor an epimorphism (surjection). Figure 1 gives an example where $\phi_{\mathcal{U}*}$ is not sujective in H_2. However, for one dimensional homology we show that, for any path connected cover $\mathcal{U}$, the map $\phi_{\mathcal{U}*}$ is necessarily a surjection. One implication of this is that the simplicial maps arising out of cover maps induce a surjection among the one dimensional homology groups of two nerve complexes.

3.1 Nerves

The proof of the nerve theorem [15] uses a construction that connects the two spaces X and $|N(\mathcal{U})|$ via a third space $X_\mathcal{U}$ that is a product space of $\mathcal{U}$ and the geometric realization $|N(\mathcal{U})|$.



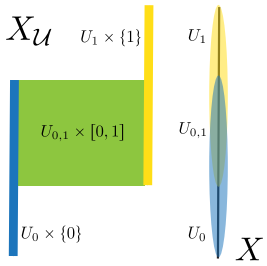
In our case $\mathcal{U}$ may not satisfy the contractibility condition. Nevertheless, we use the same construction to define three maps, $\zeta : X \to X_\mathcal{U}$, $\pi : X_\mathcal{U} \to |N(\mathcal{U})|$, and $\phi_\mathcal{U} : X \to |N(\mathcal{U})|$ where $\phi_\mathcal{U} = \pi \circ \zeta$ is referred to as the *nerve map*. Details about the construction of these maps follow.

Denote the elements of the cover $\mathcal{U}$ as U_α for α taken from some indexing set A. The vertices of $N(\mathcal{U})$ are denoted by $\{u_\alpha, \alpha \in A\}$, where each u_α corresponds to the cover

element U_α. For each finite non-empty intersection $U_{\alpha_0,\ldots,\alpha_n} := \bigcap_{i=0}^n U_{\alpha_i}$ consider the product $U_{\alpha_0,\ldots,\alpha_n} \times \Delta^n_{\alpha_0,\ldots,\alpha_n}$, where $\Delta^n_{\alpha_0,\ldots,\alpha_n}$ denotes the n-dimensional simplex with vertices $u_{\alpha_0},\ldots,u_{\alpha_n}$. Consider now the disjoint union

$$M := \bigsqcup_{\alpha_0,\ldots,\alpha_n \in A:\, U_{\alpha_0,\ldots,\alpha_n}\neq\emptyset} U_{\alpha_0,\ldots,\alpha_n} \times \Delta^n_{\alpha_0,\ldots,\alpha_n}$$

together with the following identification: each point $(x,y) \in M$, with $x \in U_{\alpha_0,\ldots,\alpha_n}$ and $y \in [\alpha_0,\ldots,\widehat{\alpha}_i,\ldots,\alpha_n] \subset \Delta^n_{\alpha_0,\ldots,\alpha_n}$ is identified with the corresponding point in the product $U_{\alpha_0,\ldots,\widehat{\alpha_i},\ldots,\alpha_n} \times \Delta_{\alpha_0,\ldots,\widehat{\alpha_i},\ldots,\alpha_n}$ via the inclusion $U_{\alpha_0,\ldots,\alpha_n} \subset U_{\alpha_0,\ldots,\widehat{\alpha_i},\ldots,\alpha_n}$. Here $[\alpha_0,\ldots,\widehat{\alpha}_i,\ldots,\alpha_n]$ denotes the i-th face of the simplex $\Delta^n_{\alpha_0,\ldots,\alpha_n}$. Denote by $\sim$ this identification and now define the space $X_{\mathcal{U}} := M \,/\sim$. An example for the case when X is a line segment and $\mathcal{U}$ consists of only two open sets is shown below.



▶ **Definition 4.** A collection of real valued continuous functions $\{\varphi_\alpha :\to [0,1], \alpha \in A\}$ is called a *partition of unity* if (i) $\sum_{\alpha\in A} \varphi_\alpha(x) = 1$ for all $x \in X$, (ii) For every $x \in X$, there are only finitely many $\alpha \in A$ such that $\varphi_\alpha(x) > 0$.

If $\mathcal{U} = \{U_\alpha,\, \alpha \in A\}$ is any open cover of X, then a partition of unity $\{\varphi_\alpha,\, \alpha \in A\}$ is *subordinate* to $\mathcal{U}$ if $\mathrm{supp}(\varphi_\alpha)$ is contained in U_α for each $\alpha \in A$.

Since X is paracompact, for any open cover $\mathcal{U} = \{U_\alpha,\, \alpha \in A\}$ of X, there exists a partition of unity $\{\varphi_\alpha, \alpha \in A\}$ subordinate to $\mathcal{U}$ [20]. For each $x \in X$ such that $x \in U_\alpha$, denote by x_α the corresponding copy of x residing in $X_{\mathcal{U}}$. Then, the map $\zeta : X \to X_{\mathcal{U}}$ is defined as follows: for any $x \in X$,

$$\zeta(x) := \sum_{\alpha\in A} \varphi_\alpha(x)\, x_\alpha.$$

The map $\pi : X_{\mathcal{U}} \to |N(\mathcal{U})|$ is induced by the individual projection maps

$$U_{\alpha_0,\ldots,\alpha_n} \times \Delta^n_{\alpha_0,\ldots,\alpha_n} \to \Delta^n_{\alpha_0,\ldots,\alpha_n}.$$

Then, it follows that $\phi_{\mathcal{U}} = \pi \circ \zeta : X \to |N(\mathcal{U})|$ satisfies, for $x \in X$,

$$\phi_{\mathcal{U}}(x) = \sum_{\alpha\in A} \varphi_\alpha(x)\, u_\alpha. \tag{1}$$

We have the following fact [20, pp. 108]:

▶ **Fact 5.** *ζ is a homotopy equivalence.*

3.2 From space to nerves

Now, we show that the nerve maps at the homology level are surjective for one dimensional homology. Interestingly, the result is not true beyond one dimensional homology (see Figure 1)

which is probably why this simple but important fact has not been observed before. First, we make a simple observation that connects the classes in singular homology of $|N(\mathcal{U})|$ to those in the simplicial homology of $N(\mathcal{U})$. The result follows immediately from the isomorphism between singular and simplicial homology induced by the geometric realization; see [19, Theorem 34.3]. In what follows let $[c]$ denote the class of a cycle c.

▶ **Proposition 6.** *Every 1-cycle ξ in $|N(\mathcal{U})|$ has a 1-cycle γ in $N(\mathcal{U})$ so that $[\xi] = [|\gamma|]$.*

▶ **Proposition 7.** *If $\mathcal{U}$ is path connected, $\phi_{\mathcal{U}*} : H_1(X) \to H_1(|N(\mathcal{U})|)$ is a surjection.*

Proof. Let $[\gamma]$ be any class in $H_1(|N(\mathcal{U})|)$. Because of Proposition 6, we can assume that $\gamma = |\gamma'|$, where γ' is a 1-cycle in the 1-skeleton of $N(\mathcal{U})$. We construct a 1-cycle $\gamma_{\mathcal{U}}$ in $X_{\mathcal{U}}$ so that $\pi(\gamma_{\mathcal{U}}) = \gamma$. Recall the map $\zeta : X \to X_{\mathcal{U}}$ in the construction of the nerve map $\phi_{\mathcal{U}}$ where $\phi_{\mathcal{U}} = \pi \circ \zeta$. There exists a class $[\gamma_X]$ in $H_1(X)$ so that $\zeta_*([\gamma_X]) = [\gamma_{\mathcal{U}}]$ because ζ_* is an isomorphism by Fact 5. Then, $\phi_{\mathcal{U}*}([\gamma_X]) = \pi_*(\zeta_*([\gamma_X]))$ because $\phi_{\mathcal{U}*} = \pi_* \circ \zeta_*$. It follows $\phi_{\mathcal{U}*}([\gamma_X]) = \pi_*([\gamma_{\mathcal{U}}]) = [\gamma]$ showing that $\phi_{\mathcal{U}*}$ is surjective.

Therefore, it remains only to show that a 1-cycle $\gamma_{\mathcal{U}}$ can be constructed given γ in $|N(\mathcal{U})|$ so that $\pi(\gamma_{\mathcal{U}}) = \gamma$. See the full version for this construction. ◀

Since we are eventually interested in the simplicial homology groups of the nerves rather than the singular homology groups of their geometric realizations, we make one more transition using the known isomorphism between the two homology groups. Specifically, if $\iota_{\mathcal{U}} : H_k(|N(\mathcal{U})|) \to H_k(N(\mathcal{U}))$ denotes this isomorphism, we let $\bar{\phi}_{\mathcal{U}*}$ denote the composition $\iota_{\mathcal{U}} \circ \phi_{\mathcal{U}*}$. As a corollary to Proposition 7, we obtain:

▶ **Theorem 8.** *If $\mathcal{U}$ is path connected, $\bar{\phi}_{\mathcal{U}*} : H_1(X) \to H_1(N(\mathcal{U}))$ is a surjection.*

3.3 From nerves to nerves

In this section we extend the result in Theorem 8 to simplicial maps between two nerves induced by cover maps. The following proposition is key to establishing the result.

▶ **Proposition 9** (Coherent partitions of unity)**.** *Suppose $\{U_\alpha\}_{\alpha\in A} = \mathcal{U} \xrightarrow{\theta} \mathcal{V} = \{V_\beta\}_{\beta\in B}$ are open covers of the paracompact topological space X and $\theta : A \to B$ is a map of covers. Then there exists a partition of unity $\{\varphi_\alpha\}_{\alpha\in A}$ subordinate to the cover $\mathcal{U}$ such that if for each $\beta \in B$ we define*

$$\psi_\beta := \begin{cases} \sum_{\alpha\in\theta^{-1}(\beta)} \varphi_\alpha & \text{if } \beta \in \mathrm{im}(\theta); \\ 0 & \text{otherwise.} \end{cases}$$

then the set of functions $\{\psi_\beta\}_{\beta\in B}$ is a partition of unity subordinate to the cover $\mathcal{V}$.

Proof is deferred to the full version.

Let $\{U_\alpha\}_{\alpha\in A} = \mathcal{U} \xrightarrow{\theta} \mathcal{V} = \{V_\beta\}_{\beta\in B}$ be two open covers of X connected by a map of covers. Apply Proposition 9 to obtain coherent partitions of unity $\{\varphi_\alpha\}_{\alpha\in A}$ and $\{\psi_\beta\}_{\beta\in B}$ subordinate to $\mathcal{U}$ and $\mathcal{V}$, respectively. Let the nerve maps $\phi_{\mathcal{U}} : X \to |N(\mathcal{U})|$ and $\phi_{\mathcal{V}} : X \to |N(\mathcal{V})|$ be defined as in (1) above. Let $N(\mathcal{U}) \xrightarrow{\tau} N(\mathcal{V})$ be the simplicial map induced by the cover map θ. Then, τ can be extended to a continuous map $\hat{\tau}$ on the image of $\phi_{\mathcal{U}}$ as follows: for $x \in X$, $\hat{\tau}(\phi_{\mathcal{U}}(x)) = \Sigma_{\alpha\in A}\varphi_\alpha(x)\, v_{\theta(\alpha)}$.

▶ **Proposition 10.** *Let $\mathcal{U}$ and $\mathcal{V}$ be two covers of X connected by a cover map $\mathcal{U} \xrightarrow{\theta} \mathcal{V}$. Then, the nerve maps $\phi_{\mathcal{U}}$ and $\phi_{\mathcal{V}}$ satisfy $\phi_{\mathcal{V}} = \hat{\tau} \circ \phi_{\mathcal{U}}$ where $\tau : N(\mathcal{U}) \to N(\mathcal{V})$ is the simplicial map induced by the cover map θ.*

Proof. For any point $p \in \mathrm{im}(\phi_{\mathcal{U}})$, there is $x \in X$ where $p = \phi_{\mathcal{U}}(x) = \Sigma_{\alpha\in A}\varphi_\alpha(x)u_\alpha$. Then,

$$\hat{\tau} \circ \phi_{\mathcal{U}}(x) = \hat{\tau}\left(\sum_{\alpha\in A}\varphi_\alpha(x)u_\alpha\right) = \sum_{\alpha\in A}\varphi_\alpha(x)\tau(u_\alpha) = \sum_{\alpha\in A}\varphi_\alpha(x)\, v_{\theta(\alpha)}$$
$$= \sum_{\beta\in B}\sum_{\alpha\in\theta^{-1}(\beta)}\varphi_\alpha(x)\, v_{\theta(\alpha)} = \sum_{\beta\in B}\psi_\beta(x)v_\beta = \phi_{\mathcal{V}}(x)\,. \qquad \blacktriangleleft$$

An immediate corollary of the above Proposition is:

▶ **Corollary 11.** *The induced maps of $\phi_{\mathcal{U}*} : H_k(X) \to H_k(|N(\mathcal{U})|)$, $\phi_{\mathcal{V}*} : H_k(X) \to H_k(|N(\mathcal{V})|)$, and $\hat{\tau}_* : H_k(|N(\mathcal{U})|) \to H_k(|N(\mathcal{V})|)$ at the homology levels commute, that is, $\phi_{\mathcal{V}*} = \hat{\tau}_* \circ \phi_{\mathcal{U}*}$.*

With transition from singular to simplicial homology, Corollary 11 implies that:

▶ **Proposition 12.** *$\bar{\phi}_{\mathcal{V}*} = \tau_* \circ \bar{\phi}_{\mathcal{U}*}$ where $\bar{\phi}_{\mathcal{V}*} : H_k(X) \to H_k(N(\mathcal{V}))$, $\bar{\phi}_{\mathcal{U}*} : H_k(X) \to H_k(N(\mathcal{U}))$ and $\tau : N(\mathcal{U}) \to N(\mathcal{V})$ is the simplicial map induced by a cover map $\mathcal{U} \to \mathcal{V}$.*

Proposition 12 extends Theorem 8 to the simplicial maps between two nerves.

▶ **Theorem 13.** *Let $\tau : N(\mathcal{U}) \to N(\mathcal{V})$ be a simplicial map induced by a cover map $\mathcal{U} \to \mathcal{V}$ where both $\mathcal{U}$ and $\mathcal{V}$ are path connected. Then, $\tau_* : H_1(N(\mathcal{U})) \to H_1(N(\mathcal{V}))$ is a surjection.*

Proof. Consider the maps

$$H_1(X) \overset{\bar{\phi}_{\mathcal{U}*}}{\to} H_1(N(\mathcal{U})) \overset{\tau_*}{\to} H_1(N(\mathcal{V})), \text{ and } H_1(X) \overset{\bar{\phi}_{\mathcal{V}*}}{\to} H_1(N(\mathcal{V})).$$

By Proposition 12, $\tau_* \circ \bar{\phi}_{\mathcal{U}*} = \bar{\phi}_{\mathcal{V}*}$. By Theorem 8, the map $\bar{\phi}_{\mathcal{V}*}$ is a surjection. It follows that τ_* is a surjection. $\blacktriangleleft$

3.4 Mapper and multiscale mapper

In this section we extend the previous results to the structures called mapper and multiscale mapper. Recall that X is assumed to be compact. Consider a cover of X obtained indirectly as a pullback of a cover of another space Z. This gives rise to the so called *Mapper* and *Multiscale Mapper*. Let $f : X \to Z$ be a continuous map where Z is equipped with an open cover $\mathcal{U} = \{U_\alpha\}_{\alpha\in A}$ for some index set A. Since f is continuous, the sets $\{f^{-1}(U_\alpha), \alpha \in A\}$ form an open cover of X. For each α, we can now consider the decomposition of $f^{-1}(U_\alpha)$ into its path connected components, so we write $f^{-1}(U_\alpha) = \bigcup_{i=1}^{j_\alpha} V_{\alpha,i}$, where j_α is the number of path connected components $V_{\alpha,i}$'s in $f^{-1}(U_\alpha)$. We write $f^*\mathcal{U}$ for the cover of X obtained this way from the cover $\mathcal{U}$ of Z and refer to it as the *pullback* cover of X induced by $\mathcal{U}$ via f. Note that by its construction, this pullback cover $f^*\mathcal{U}$ is path-connected.

Notice that there are pathological examples of f where $f^{-1}(U_\alpha)$ may shatter into infinitely many path components. This motivates us to consider *well-behaved* functions f: we require that for every path connected open set $U \subseteq Z$, the preimage $f^{-1}(U)$ has *finitely* many open path connected components. Henceforth, all such functions are assumed to be well-behaved.

▶ **Definition 14** (Mapper [21])**.** Let $f : X \to Z$ be a continuous map. Let $\mathcal{U} = \{U_\alpha\}_{\alpha\in A}$ be an open cover of Z. The *mapper* arising from these data is defined to be the nerve simplicial complex of the pullback cover: $\mathrm{M}(\mathcal{U}, f) := N(f^*\mathcal{U})$.

When we consider a continuous map $f : X \to Z$ and we are given a map of covers $\xi : \mathcal{U} \to \mathcal{V}$ between covers of Z, we observed in [10] that there is a corresponding map of covers between the respective pullback covers of X: $f^*(\xi) : f^*\mathcal{U} \longrightarrow f^*\mathcal{V}$. Furthermore, if $\mathcal{U} \xrightarrow{\xi} \mathcal{V} \xrightarrow{\theta} \mathcal{W}$ are three different covers of a topological space with the intervening maps of covers between them, then $f^*(\theta \circ \xi) = f^*(\theta) \circ f^*(\xi)$.

In the definition below, *objects* can be covers, simplicial complexes, or vector spaces.

▶ **Definition 15** (Tower)**.** A *tower* $\mathfrak{W}$ with resolution $r \in \mathbb{R}$ is any collection $\mathfrak{W} = \{\mathcal{W}_\varepsilon\}_{\varepsilon \geq r}$ of objects $\mathcal{W}_\varepsilon$ indexed in $\mathbb{R}$ together with maps $w_{\varepsilon,\varepsilon'} : \mathcal{W}_\varepsilon \to \mathcal{W}_{\varepsilon'}$ so that $w_{\varepsilon,\varepsilon} = \mathrm{id}$ and $w_{\varepsilon',\varepsilon''} \circ w_{\varepsilon,\varepsilon'} = w_{\varepsilon,\varepsilon''}$ for all $r \leq \varepsilon \leq \varepsilon' \leq \varepsilon''$. Sometimes we write $\mathfrak{W} = \{\mathcal{W}_\varepsilon \xrightarrow{w_{\varepsilon,\varepsilon'}} \mathcal{W}_{\varepsilon'}\}_{r \leq \varepsilon \leq \varepsilon'}$ to denote the collection with the maps. Given such a tower $\mathfrak{W}$, $\mathrm{res}(\mathfrak{W})$ refers to its resolution.

When $\mathfrak{W}$ is a collection of covers equipped with maps of covers between them, we call it *a tower of covers*. When $\mathfrak{W}$ is a collection of simplicial complexes equipped with simplicial maps between them, we call it *a tower of simplicial complexes*.

The pullback properties described at the end of section 2 make it possible to take the pullback of a given tower of covers of a space via a given continuous function into another space, so that we obtain the following.

▶ **Proposition 16** ([10])**.** *Let $\mathfrak{U} = \{\mathcal{U}_\varepsilon\}$ be a tower of covers of Z and $f : X \to Z$ be a continuous function. Then, $f^*\mathfrak{U} = \{f^*\mathcal{U}_\varepsilon\}$ is a tower of (path-connected) covers of X.*

In general, given a tower of covers $\mathfrak{W}$ of a space X, the nerve of each cover in $\mathfrak{W}$ together with each map of $\mathfrak{W}$ provides a tower of simplicial complexes which we denote by $N(\mathfrak{W})$.

▶ **Definition 17** (Multiscale Mapper [10])**.** Let $f : X \to Z$ be a continuous map. Let $\mathfrak{U}$ be a tower of covers of Z. Then, the *multiscale mapper* is defined to be the tower of the nerve simplicial complexes of the pullback: $\mathrm{MM}(\mathfrak{U}, f) := N(f^*\mathfrak{U})$.

As we indicated earlier, in general, no surjection between X and its nerve may exist at the homology level. It follows that the same is true for the mapper $N(f^*\mathcal{U})$. But, for H_1, we can apply the results contained in previous section to claim the following.

▶ **Theorem 18.** *Consider the following multiscale mapper arising out of a tower of path connected covers:*

$$N(f^*\mathcal{U}_0) \to N(f^*\mathcal{U}_1) \to \cdots \to N(f^*\mathcal{U}_n)\,.$$

- *There is a surjection from $H_1(X)$ to $H_1(N(f^*\mathcal{U}_i))$ for each $i \in [0, n]$.*
- *Consider a H_1-persistence module of a multiscale mapper as shown below.*

$$\mathrm{H}_1\big(N(f^*\mathcal{U}_0)\big) \to \mathrm{H}_1\big(N(f^*\mathcal{U}_1)\big) \to \cdots \to \mathrm{H}_1\big(N(f^*\mathcal{U}_n)\big)\,. \tag{2}$$

All connecting maps in the above module are surjections.

The above result implies that, as we proceed forward through the multiscale mapper, no new homology classes are born. They can only die. Consequently, all bar codes in the persistence diagram of the H_1-persistence module induced by it have the left endpoint at 0.

4 Analysis of persistent H_1-classes

Using the language of persistent homology, the results in the previous section imply that one dimensional homology classes can die in the nerves, but they cannot be born. In this section,

we analyze further to identify the classes that survive. The distinction among the classes is made via a notion of 'size'. Intuitively, we show that the classes with 'size' much larger than the 'size' of the cover survive. The 'size' is defined with the pseudometric that the space X is assumed to be equipped with. Precise statements are made in the subsections.

4.1 H_1-classes of nerves of pseudometric spaces

Let (X, d) be a pseudometric space, that is, d satisfies the axioms of a metric except that $d(x, x') = 0$ may not necessarily imply $x = x'$. Assume X to be compact as before. We define a 'size' for a homology class that reflects how big the smallest generator in the class is in the metric d.

▶ **Definition 19.** The size $s(X')$ of a subset X' of the pseudometric space (X, d) is defined to be its diameter, that is, $s(X') = \sup_{x,x' \in X' \times X'} d(x, x')$. The size of a class $c \in H_k(X)$ is defined as $s(c) = \inf_{z \in c} s(z)$.

▶ **Definition 20.** A set of k-cycles $z_1, z_2, \ldots, z_n$ of $H_k(X)$ is called a generator basis if the classes $[z_1], [z_2], \ldots, [z_n]$ together form a basis of $H_k(X)$. It is called a minimal generator basis if $\Sigma_{i=1}^{n} s(z_i)$ is minimal among all generator bases.

Lebesgue number of a cover. Our goal is to characterize the classes in the nerve of $\mathcal{U}$ with respect to the sizes of their preimages in X via the map $\phi_{\mathcal{U}}$. The Lebesgue number of a cover $\mathcal{U}$ becomes useful in this characterization. It is the largest number $\lambda(\mathcal{U})$ so that any subset of X with size at most $\lambda(\mathcal{U})$ is contained in at least one element of $\mathcal{U}$. Formally,

$$\lambda(\mathcal{U}) = \sup\{\delta \,|\, \forall X' \subseteq X \text{ with } s(X') \leq \delta, \exists U_\alpha \in \mathcal{U} \text{ where } U_\alpha \supseteq X'\}.$$

We observe that a homology class of size no more than $\lambda(\mathcal{U})$ cannot survive in the nerve. Further, the homology classes whose sizes are significantly larger than the maximum size of a cover do necessarily survive where we define the maximum size of a cover as $s_{max}(\mathcal{U}) := \max_{U \in \mathcal{U}}\{s(U)\}$.

Let $z_1, z_2, \ldots, z_g$ be a non-decreasing sequence of the generators with respect to their sizes in a minimal generator basis of $H_1(X)$. Consider the map $\phi_{\mathcal{U}} : X \to |N(\mathcal{U})|$ as introduced in Section 3. We have the following result.

▶ **Theorem 21.** *Let $\mathcal{U}$ be a path-connected cover of X.*

(i) *Let $\ell = g + 1$ if $\lambda(\mathcal{U}) > s(z_g)$. Otherwise, let $\ell \in [1, g]$ be the smallest integer so that $s(z_\ell) > \lambda(\mathcal{U})$. If $\ell \neq 1$, the class $\bar{\phi}_{\mathcal{U}*}[z_j] = 0$ for $j = 1, \ldots, \ell - 1$. Moreover, if $\ell \neq g + 1$, the classes $\{\bar{\phi}_{\mathcal{U}*}[z_j]\}_{j=\ell,\ldots,g}$ generate $H_1(N(\mathcal{U}))$.*

(ii) *The classes $\{\bar{\phi}_{\mathcal{U}*}[z_j]\}_{j=\ell',\ldots,g}$ are linearly independent where $s(z_{\ell'}) > 4s_{max}(\mathcal{U})$.*

The result above says that only the classes of $H_1(X)$ generated by generators of large enough size survive in the nerve. To prove this result, we use a map ρ that sends each 1-cycle in $N(\mathcal{U})$ to a 1-cycle in X. We define a chain map $\rho : \mathcal{C}_1(N(\mathcal{U})) \to \mathcal{C}_1(X)$ among one dimensional chain groups as follows [2]. It is sufficient to exhibit the map for an elementary chain of an edge, say $e = \{u_\alpha, u_{\alpha'}\} \in \mathcal{C}_1(N(\mathcal{U}))$. Since e is an edge in $N(\mathcal{U})$, the two cover elements U_α and $U_{\alpha'}$ in X have a common intersection. Let $a \in U_\alpha$ and $b \in U_{\alpha'}$ be two points that are arbitrary but fixed for U_α and $U_{\alpha'}$ respectively. Pick a path $\xi(a, b)$ (viewed

[2] We note that the high level framework of defining such a chain map and analyzing what it does to homologous cycles is similar to the work by Gasparovic et al. [14]. The technical details are different.

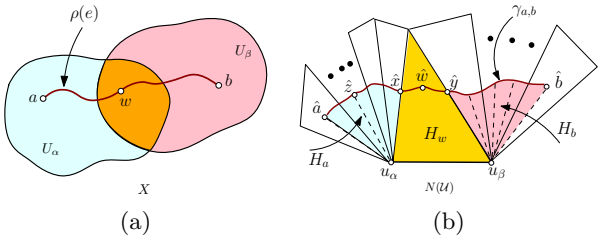


Figure 2 Illustration for proof of Proposition 22.

as a singular chain) in the union of U_α and $U_{\alpha'}$ which is path connected as both U_α and $U_{\alpha'}$ are. Then, define $\rho(e) = \xi(a, b)$. The following properties of $\phi_{\mathcal{U}}$ and ρ turn out to be useful.

▶ **Proposition 22.** *Let γ be any 1-cycle in $N(\mathcal{U})$. Then, $[\phi_{\mathcal{U}}(\rho(\gamma))] = [|\gamma|]$.*

Proof. Let $e = (u_\alpha, u_\beta)$ be an edge in γ with u_α and u_β corresponding to U_α and U_β respectively. Let a and b be the corresponding fixed points for set U_α and U_β respectively. Consider the path $\rho(e) = \xi(a, b)$ in X as constructed above, and set $\gamma_{a,b} = \phi_{\mathcal{U}}(\xi(a, b))$ to be the image of $\rho(e)$ in $|N(\mathcal{U})|$. See Figure 2 for an illustration. Given an oriented path ℓ and two points $x, y \in \ell$, we use $\ell[x, y]$ to denote the subpath of ℓ from x to y. For a point $x \in X$, for simplicity we set $\hat{x} = \phi_{\mathcal{U}}(x)$ to be its image in $|N(\mathcal{U})|$.

Now, let $w \in \rho(e)$ be a point in $U_\alpha \cap U_\beta$, and $\hat{w} = \phi_{\mathcal{U}}(w)$ be its image in $\gamma_{a,b}$. Furthermore, let $\sigma_w \in N(\mathcal{U})$ be the lowest-dimensional simplex containing $\hat{w}$. While u_α and u_β may not be vertices of σ_w, we can show that $Vert(\sigma_w) \cup \{u_\alpha, u_\beta\}$ must span a simplex $\bar{\sigma}_w$, in $N(\mathcal{U})$ (see full version). Let $\gamma_{a,b}[\hat{x}, \hat{y}]$ be the maximal subpath of $\gamma_{a,b}$ containing $\hat{w}$ that is contained within $|\bar{\sigma}_w|$. One can construct a homotopy H_a that takes $\gamma_{a,b}[\hat{a}, \hat{x}]$ to u_α under which any point $\hat{z} \in \gamma_{a,b}[\hat{a}, \hat{x}]$ moves monotonically along the segment $\hat{z}u_\alpha$ within the geometric realization of the simplex containing both $\hat{z}$ and u_α. See the details in the full version.

Similarly, there is a homotopy H_b that takes $\gamma_{a,b}[\hat{y}, \hat{b}]$ to u_β under which any point $\hat{z} \in \gamma_{a,b}[\hat{y}, \hat{b}]$ moves monotonically along the segment $\hat{z}u_\beta$. Finally, for the middle subpath $\gamma_{a,b}[\hat{x}, \hat{y}]$, since it is within simplex $\bar{\sigma}_w$ with $e = (u_\alpha, u_\beta)$ being an edge of it, we can construct a homotopy H_w that takes $\gamma_{a,b}[\hat{x}, \hat{y}]$ to $|u_\alpha u_\beta|$ under which $\hat{x}$ and $\hat{y}$ move monotonically along the segments $\hat{x}u_\alpha$ and $\hat{y}u_\beta$ within the geometric realization of simplex $\bar{\sigma}_w$, respectively. Concatenating H_a, H_w and H_b, we obtain a homotopy $H_{\alpha,\beta}$ taking $\gamma_{a,b}$ to $|e|$. A concatenation of these homotopies $H_{\alpha,\beta}$ considered over all edges in γ, brings $\phi_{\mathcal{U}}(\rho(\gamma))$ to $|\gamma|$ with a homotopy in $|N(\mathcal{U})|$. Hence, their homology classes are the same. ◀

▶ **Proposition 23.** *Let z be a 1-cycle in $\mathcal{C}_1(X)$. Then, $[\phi_{\mathcal{U}}(z)] = 0$ if $\lambda(\mathcal{U}) > s(z)$.*

Proof of Theorem 21.

Proof of (i): By Proposition 23, we have $\phi_{\mathcal{U}*}[z] = [\phi_{\mathcal{U}}(z)] = 0$ if $\lambda(\mathcal{U}) > s(z)$. This establishes the first part of the assertion because $\bar{\phi}_{\mathcal{U}*} = \iota \circ \phi_{\mathcal{U}*}$ where ι is an isomorphism between the singular homology of $|N(\mathcal{U})|$ and the simplicial homology of $N(U)$. To see the second part, notice that $\bar{\phi}_{\mathcal{U}*}$ is a surjection by Theorem 8. Therefore, the classes $\bar{\phi}_{\mathcal{U}*}(z)$ where $\lambda(\mathcal{U}) \not> s(z)$ contain a basis for $H_1(N(\mathcal{U}))$. Hence they generate it.

Proof of (ii): Suppose on the contrary, there is a subsequence $\{\ell_1, \ldots, \ell_t\} \subset \{\ell', \ldots, g\}$ such that $\Sigma_{j=1}^t [\phi_{\mathcal{U}}(z_{\ell_j})] = 0$. Let $z = \Sigma_{j=1}^t \phi_{\mathcal{U}}(z_{\ell_j})$. Let γ be a 1-cycle in $N(\mathcal{U})$ so that $[z] = [|\gamma|]$ whose existence is guaranteed by Proposition 6. It must be the case that there is a 2-chain D in $N(\mathcal{U})$ so that $\partial D = \gamma$. Consider a triangle $t = \{u_{\alpha_1}, u_{\alpha_2}, u_{\alpha_3}\}$ contributing to D. Let $a'_i = \phi_{\mathcal{U}}^{-1}(u_{\alpha_i})$. Since t appears in $N(\mathcal{U})$, the covers $U_{\alpha_1}, U_{\alpha_2}, U_{\alpha_3}$

containing a'_1, a'_2, and a'_3 respectively have a common intersection in X. This also means that each of the paths $a'_1 \rightsquigarrow a'_2$, $a'_2 \rightsquigarrow a'_3$, $a'_3 \rightsquigarrow a'_1$ has size at most $2s_{max}(\mathcal{U})$. Then, $\rho(\partial t)$ is mapped to a 1-cycle in X of size at most $4s_{max}(\mathcal{U})$. It follows that $\rho(\partial D)$ can be written as a linear combination of cycles of size at most $4s_{max}(\mathcal{U})$. Each of the 1-cycles of size at most $4s_{max}(\mathcal{U})$ is generated by basis elements $z_1, \ldots, z_k$ where $s(z_k) \leq 4s_{max}(\mathcal{U})$. Therefore, the class of $z' = \phi_{\mathcal{U}}(\rho(\gamma))$ is generated by a linear combination of the basis elements whose preimages have size at most $4s_{max}(\mathcal{U})$. The class $[z']$ is same as the class $[|\gamma|]$ by Proposition 22. But, by assumption $[|\gamma|] = [z]$ is generated by a linear combination of the basis elements whose sizes are larger than $4s_{max}(\mathcal{U})$ reaching a contradiction. ◀

4.2 H_1-classes in Reeb space

In this section we prove an analogue of Theorem 21 for Reeb spaces, which to our knowledge is new. The Reeb space of a function $f : X \to Z$, denoted R_f, is the quotient of X under the equivalence relation $x \sim_f x'$ if and only if $f(x) = f(x')$ and there exists a continuous path $\gamma \in \Gamma_X(x, x')$ such that $f \circ \gamma$ is constant. The induced quotient map is denoted $q : X \to R_f$ which is of course surjective. We show that q_* at the homology level is also surjective for H_1 when the codomain Z of f is a metric space. In fact, we prove a stronger statement: only 'vertical' homology classes (classes with strictly positive size) survive in a Reeb space which extends the result of Dey and Wang [11] for Reeb graphs.

Let $\mathcal{V}$ be a path-connected cover of R_f. This induces a pullback cover denoted $\mathcal{U} = \{U_\alpha\}_{\alpha \in A} = \{q^{-1}(V_\alpha)\}_{\alpha \in A}$ on X. Let $N(\mathcal{U})$ and $N(\mathcal{V})$ denote the corresponding nerve complexes of $\mathcal{U}$ and $\mathcal{V}$ respectively. It is easy to see that $N(\mathcal{U}) \cong N(\mathcal{V})$ because $U_\alpha \cap U_{\alpha'} \neq \emptyset$ if and only if $V_\alpha \cap V_{\alpha'} \neq \emptyset$. There are nerve maps $\phi_{\mathcal{V}} : R_f \to |N(\mathcal{V})|$ and $\phi_{\mathcal{U}} : X \to |N(\mathcal{U})|$ so that the following holds:

▶ **Proposition 24.** *Consider the sequence* $X \xrightarrow{q} R_f(X) \xrightarrow{\phi_{\mathcal{V}}} |N(\mathcal{V})| = |N(\mathcal{U})|$. *Then,* $\phi_{\mathcal{U}} = \phi_{\mathcal{V}} \circ q$.

Let the codomain of the function $f : X \to Z$ be a *metric space* (Z, d_Z). We first impose a pseudometric on X induced by f; the one-dimensional version of this pseudometric is similar to the one used in [1] for Reeb graphs. Recall that given two points $x, x' \in X$ we denote by $\Gamma_X(x, x')$ the set of all continuous paths $\gamma : [0, 1] \to X$ such that $\gamma(0) = x$ and $\gamma(1) = x'$.

▶ **Definition 25.** We define a pseudometric d_f on X as follows: for $x, x' \in X$,

$$d_f(x, x') := \inf_{\gamma \in \Gamma_X(x,x')} \operatorname{diam}_Z(f \circ \gamma).$$

▶ **Proposition 26.** $d_f : X \times X \to \mathbb{R}_+$ *is a pseudometric.*

Similar to X, we endow R_f with a distance $\tilde{d}_f$ that descends via the map q: for any equivalence classes $r, r' \in R_f$, pick $x, x' \in X$ with $r = q(x)$ and $r' = q(x')$, then define

$$\tilde{d}_f(r, r') := d_f(x, x').$$

The definition does not depend on the representatives x and x' chosen. In this manner we obtain the pseudometric space $(R_f, \tilde{d}_f)$. Let $z_1, \ldots, z_g$ be a minimal generator basis of $H_1(X)$ defined with respect to the pseudometric d_f and $q : X \to R_f$ be the quotient map.

▶ **Theorem 27.** *Let* $\ell \in [1, g]$ *be the smallest integer so that* $s(z_\ell) \neq 0$. *If no such* ℓ *exists,* $H_1(R_f)$ *is trivial, otherwise,* $\{[q(z_i)]\}_{i=\ell,\ldots g}$ *is a basis for* $H_1(R_f)$.

Proof. Consider the sequence $X \xrightarrow{q} R_f \xrightarrow{\phi_{\mathcal{V}}} |N(\mathcal{V})|$ where $\mathcal{V}$ is a cover of R_f. It is shown in the full version that q_* is a surjection for H_1-homology. Then, $\{[q(z_i)]\}_{i=1,\dots,g}$ generate $H_1(R_f)$. First, assume that ℓ as stated in the theorem exists. Let the cover $\mathcal{V}$ be fine enough so that $0 < s_{\max}(\mathcal{U}) \leq \delta$ where $\delta = \frac{1}{4}\min\{s(z_i) \,|\, s(z_i) \neq 0\}$. Then, by applying Theorem 21(ii), we obtain that $[\phi_{\mathcal{U}}(z_i)]_{i=\ell,\dots,g}$ are linearly independent in $H_1(|N(\mathcal{U})|) = H_1(|N(\mathcal{V})|$. Since $[\phi_{\mathcal{U}}(z_i)] = [\phi_{\mathcal{V}} \circ q(z_i)]$ by Proposition 24, $\{[q(z_i)]\}_{i=\ell,\dots,g}$ are linearly independent in $H_1(R_f)$. But, $[q(z_i)] = 0$ for $s(z_i) = 0$ and $\{[q(z_i)]\}_{i=1,\dots,g}$ generate $H_1(R_f)$. Therefore, $\{[q(z_i)]\}_{i=\ell,\dots,g}$ is a basis. In the case when ℓ does not exist, we have $s(z_i) = 0$ for every $i \in [1, g]$. Then, $[q(z_i)] = 0$ for every i rendering $H_1(R_f)$ trivial. ◀

4.3 Persistence of H_1-classes in mapper and multiscale mapper

To apply the results for nerves in section 4.1 to mappers and multiscale mappers, the Lebesgue number of the pullback covers of X becomes important. The following observation in this respect is useful. Remember that the size of a subset in X and hence the cover elements are measured with respect to the pseudometric d_f.

▶ **Proposition 28.** *Let $\mathcal{U}$ be a cover for the codomain Z. Then, the pullback cover $f^*\mathcal{U}$ has Lebesgue number $\lambda(\mathcal{U})$.*

Notice that the smallest size $s_{min}(f^*\mathcal{U})$ of an element of the pullback cover can be arbitrarily small even if $s_{min}(\mathcal{U})$ is not. However, the Lebesgue number of $\mathcal{U}$ can be leveraged for the mapper due to the above proposition.

Given a cover $\mathcal{U}$ of Z, consider the mapper $N(f^*\mathcal{U})$. Let $z_1, \dots, z_g$ be a set of minimal generator basis for $H_1(X)$ where the metric in question is d_f. Then, as a consequence of Theorem 21 we have:

▶ **Theorem 29.**

(i) *Let $\ell = g + 1$ if $\lambda(\mathcal{U}) > s(z_g)$. Otherwise, let $\ell \in [1, g]$ be the smallest integer so that $s(z_\ell) > \lambda(\mathcal{U})$. If $\ell \neq 1$, the class $\phi_{\mathcal{U}*}[z_j] = 0$ for $j = 1, \dots, \ell - 1$. Moreover, if $\ell \neq g + 1$, the classes $\{\phi_{\mathcal{U}*}[z_j]\}_{j=\ell,\dots,g}$ generate $H_1(N(f^*\mathcal{U}))$.*

(ii) *The classes $\{\phi_{\mathcal{U}*}[z_j]\}_{j=\ell',\dots,g}$ are linearly independent where $s(z_{\ell'}) > 4s_{max}(\mathcal{U})$.*

(iii) *Consider a H_1-persistence module of a multiscale mapper induced by a tower of path connected covers:*

$$\mathrm{H}_1\big(N(f^*\mathcal{U}_{\varepsilon_0})\big) \xrightarrow{s_{1*}} \mathrm{H}_1\big(N(f^*\mathcal{U}_{\varepsilon_1})\big) \xrightarrow{s_{2*}} \cdots \xrightarrow{s_{n*}} \mathrm{H}_1\big(N(f^*\mathcal{U}_{\varepsilon_n})\big)\,. \tag{3}$$

Let $\hat{s}_{i*} = s_{i*} \circ s_{(i-1)*} \circ \cdots \circ \bar{\phi}_{\mathcal{U}_{\varepsilon_0}*}$. Then, the assertions in (i) and (ii) hold for $H_1(N(f^*\mathcal{U}_{\varepsilon_i}))$ with the map $\hat{s}_{i*} : X \to N(f^*\mathcal{U}_{\varepsilon_i})$.

▶ Remark (Persistence diagram approximation.). The persistence diagram of the H_1-persistence module considered in Theorem 29(iii) contains points whose birth coordinates are exactly zero. This is because all connecting maps are surjective by (i) and thus every class is born only at the beginning. The death coordinate of a point that corresponds to a minimal basis generator of size s is in between the index ε_i and ε_j where $s \geq 4s_{max}(\mathcal{U}_{\varepsilon_i})$ and $s \leq \lambda(\mathcal{U}_{\varepsilon_j})$ because of the assertions (i) and (ii) in Theorem 29. Assuming covers whose λ and s_{max} values are within a constant factor of each other (such as the ones described in next subsection), we can conclude that a generator of size s dies at some point cs for some constant c. Therefore, by computing a minimal generator basis of $N(\mathcal{U}_{\varepsilon_0})$ and computing their sizes provide a 4-approximation to the persistence diagram of the multiscale mapper in the log scale.

4.4 Two special covers and intrinsic Čech complex

We discuss two special covers, one can be effectively computed and the other one is relevant in the context of the intrinsic Čech complex of a metric space. We say a cover $\mathcal{U}$ of a metric space (Y, d) is (α, β)-cover if $\alpha \leq \lambda(\mathcal{U})$ and $\beta \geq s_{max}(\mathcal{U})$.

A $(\delta, 4\delta)$-cover: Consider a δ-sample P of Y, that is, every metric ball $B(y, \delta)$, $y \in Y$, contains a point in P. Observe that the cover $\mathcal{U} = \{B(p, 2\delta)\}_{p \in P}$ is a $(\delta, 4\delta)$-cover for Z. Clearly, $s_{max}(\mathcal{U}) \leq 4\delta$. To determine $\lambda(\mathcal{U})$, consider any subset $Y' \subseteq Y$ with $s(Y') \leq \delta$. There is a $p \in P$ so that $d_Y(p, Y') \leq \delta$. Let y' be the furthest point in Y' from p. Then, $d_Y(p, y') \leq d_Y(p, Y) + \text{diam}(Y') \leq 2\delta$ establishing that $\lambda(\mathcal{U}) \geq \delta$.

A $(\delta, 2\delta)$-cover: Consider the infinite cover $\mathcal{U}$ of Y where $\mathcal{U} = \{B(y, \delta)\}_{y \in Y}$. These are the set of all metric balls of radius δ. Clearly, $s_{max}(\mathcal{U}) \leq 2\delta$. Any subset $Y' \subseteq Y$ with $s(Y') \leq \delta$ is contained in a ball $B(y, \delta)$ where y is any point in Y'. This shows that $\lambda(\mathcal{U}) \geq \delta$. A consequence of this observation and Theorem 21 is that the intrinsic Čech complexes satisfy some interesting property.

▶ **Definition 30.** Given a metric space (Y, d_Y), its intrinsic Čech complex $C^\delta(Y)$ at scale δ is defined to be the nerve complex of the set of intrinsic δ-balls $\{B(y, \delta)\}_{y \in Y}$.

▶ **Observation 31.** *Let $C^\delta(Y)$ denote the intrinsic Čech complex of a metric space Y at scale δ. Let $\mathcal{U}$ denote the corresponding possibly infinite cover of Y. Let $z_1, \ldots, z_g$ be a minimal generator basis for $H_1(Y)$. Then, $\{\bar{\phi}_{\mathcal{U}*}(z_i)\}_{i=\ell,\ldots,g}$ generate $H_1(C^\delta(Y))$ if ℓ is the smallest integer with $s(z_\ell) > \delta$. Furthermore, $\{\bar{\phi}_{\mathcal{U}*}(z_i)\}_{i=\ell',\ldots,g}$ are linearly independent if $s(z'_\ell) > 8\delta$.*

5 Higher dimensional homology groups

We have already observed that the surjectivity of the map $\phi_{\mathcal{U}*} : H_1(X) \to H_1(|N(\mathcal{U})|)$ in one dimensional homology does not extend to higher dimensional homology groups. This means that we cannot hope for analogues to Theorem 21(i) and Theorem 29 to hold for higher dimensional homology groups. However, under the assumption that $f : X \to Z$ is a continuous map from a compact space to a metric space, we can provide some characterization of the persistent diagrams of the mapper and the multiscale mapper as follows:

- We define a metric d_δ on the vertex set P_δ of $N(\mathcal{U})$ where $s_{\max}(\mathcal{U}) \leq \delta$ and then show that the Gromov-Hausdorff distance between the metric spaces (P_δ, d_δ) and $(R_f, \tilde{d}_f)$ is at most 5δ. The same proof also applies if we replace $(R_f, \tilde{d}_f)$ with the pseudometric space (X, d_f). See the full version.
- Previous result implies that the persistence diagrams of the intrinsic Čech complex of the metric space (X, d_f) and that of the metric space (P_δ, d_δ) have a bottleneck distance of $O(\delta)$. This further implies that the persistence diagram of the mapper structure $N(\mathcal{U})$ (approximated as the metric structure (P_δ, d_δ)) is close to that of the intrinsic Čech complex of the pseudometric space (X, d_f).
- We show that the intrinsic Čech complexes of (X, d_f) interleave with $\text{MM}(\mathfrak{U}, f)$ thus connecting their persistence diagrams. See Section 5.1.
- It follows that the persistence diagrams of the multiscale mapper $\text{MM}(\mathfrak{U}, f)$ and (P_δ, d_δ) are close, both being close to that of (X, d_f). This shows that the multiscale mapper encodes similar information as the mapper under an appropriate map-induced metric.

▶ **Definition 32** (Intrinsic Čech filtration)**.** The *intrinsic Čech filtration of the metric space* (Y, d_Y) is

$$\mathfrak{C}(Y) = \{C^r(Y) \subseteq C^{r'}(Y)\}_{0<r<r'}.$$

The *intrinsic Čech filtration at resolution* s is defined as $\mathfrak{C}_s(Y) = \{C^r(Y) \subseteq C^{r'}(Y)\}_{s\leq r<r'}$.

Whenever (Y, d_Y) is totally bounded, the persistence modules induced by taking homology of this intrinsic Čech filtration become q-tame [7]. This implies that one may define its persistence diagram Dg $\mathfrak{C}(Y)$ which provides one way to summarize the topological information of the space Y through the lens of its metric structure d_Y.

We argue that the pseudometric space (X, d_f) is totally bounded. This requires us to show that for any $\varepsilon > 0$ there is a finite subset of $P \subseteq X$ so that open balls centered at points in P with radii ε cover X. Recall that we have assumed that X is a compact topological space, that (Z, d_Z) is a metric space, and that $f : X \to Z$ is a continuous map. Consider a cover $\mathcal{U}$ of Z where each cover element is a ball of radius most $\varepsilon/2$ around a point in Z. Then, the pullback cover $f^*\mathcal{U}$ of X has all elements with diameter at most ε in the metric d_f. Since X is compact, a finite sub-cover of $f^*\mathcal{U}$ still covers X. A finite set P consisting of one arbitrary point in each element of this finite sub-cover is such that the union of d_f-balls of radius ϵ around points in P covers X. Since $\varepsilon > 0$ was arbitrary, (X, d_f) is totally bounded.

Consider the mapper $N(f^*\mathcal{U})$ w.r.t a cover $\mathcal{U}$ of the codomain Z. We can equip its vertex set, denoted by P_δ, with a metric structure (P_δ, d_δ), where δ is an upper bound on the diameter of each element in $\mathcal{U}$. Hence we can view the persistence diagram Dg $\mathfrak{C}(P_\delta)$ w.r.t. the metric d_δ as a summary of the mapper $N(f^*\mathcal{U})$. Using the Gromov-Hausdorff distance between the metric spaces (P_δ, d_δ) and (X, d_f), we relate this persistent summary to the persistence diagram Dg $\mathfrak{C}(X)$ induced by the intrinsic Čech filtration of (X, d_f). Specifically, we show that $d_{GH}((P_\delta, d_\delta), (X, d_f)) \leq 5\delta$. Theorem 32 in the full version extends to this result. With (X, d_f) being totally bounded, by results of [7], it follows that the bottleneck-distance between the two resulting persistence diagrams satisfies:

$$d_B(\text{Dg } \mathfrak{C}(P_\delta), \text{Dg } \mathfrak{C}(X)) \leq 2 * 5\delta = 10\delta. \tag{4}$$

5.1 MM($\mathfrak{W}, f$) for a tower of covers $\mathfrak{W}$

Above we discussed the information encoded in a certain persistence diagram summary of a single Mapper structure. We now consider the persistent homology of multiscale mappers. Given any tower of covers (TOC) $\mathfrak{W}$ of the co-domain Z, by applying the homology functor to its multiscale mapper MM($\mathfrak{W}, f$), we obtain a persistent module, and we can thus discuss the persistent homology induced by a tower of covers $\mathfrak{W}$. However, as discussed in [10], this persistent module is not necessarily stable under perturbations (of e.g the map f) for general TOCs. To address this issue, Dey et al. introduced a special family of the so-called (c,s)-good TOC in [10], which is natural and still general. Below we provide an equivalent definition of the (c,s)-good TOC based on the Lebesgue number of covers.

▶ **Definition 33** ((c, s)-good TOC)**.** Give a tower of covers $\mathfrak{U} = \{\mathcal{U}_\varepsilon\}_{\varepsilon\geq s}$, we say that it is *(c,s)-good TOC* if for any $\varepsilon \geq s$, we have that (i) $s_{max}(\mathcal{U}_\varepsilon) \leq \varepsilon$ and (ii) $\lambda(\mathcal{U}_{c\varepsilon}) \geq \varepsilon$.

As an example, the TOC $\mathfrak{U} = \{\mathcal{U}_\varepsilon\}_{\varepsilon\geq s}$ with $\mathcal{U}_\varepsilon := \{B_{\varepsilon/2}(z) \mid z \in Z\}$ is an (2,s)-good TOC of the co-domain Z.

We now characterize the persistent homology of multiscale mappers induced by (c,s)-good TOCs. Connecting these persistence modules is achieved via the interleaving of towers of

simplicial complexes originally introduced in [5]. Below we include the slightly generalized version of the definition from [10].

▶ **Definition 34** (Interleaving of simplicial towers, [10]). Let $\mathfrak{S} = \left\{\mathcal{S}_\varepsilon \xrightarrow{s_{\varepsilon,\varepsilon'}} \mathcal{S}_{\varepsilon'}\right\}_{r\le\varepsilon\le\varepsilon'}$ and $\mathfrak{T} = \left\{\mathcal{T}_\varepsilon \xrightarrow{t_{\varepsilon,\varepsilon'}} \mathcal{T}_{\varepsilon'}\right\}_{r\le\varepsilon\le\varepsilon'}$ be two towers of simplicial complexes where $\mathrm{res}(\mathfrak{S}) = \mathrm{res}(\mathfrak{T}) = r$. For some $c \ge 0$, we say that they are *c-interleaved* if for each $\varepsilon \ge r$ one can find simplicial maps $\varphi_\varepsilon : \mathcal{S}_\varepsilon \to \mathcal{T}_{\varepsilon+c}$ and $\psi_\varepsilon : \mathcal{T}_\varepsilon \to \mathcal{S}_{\varepsilon+c}$ so that:

(i) for all $\varepsilon \ge r$, $\psi_{\varepsilon+c} \circ \varphi_\varepsilon$ and $s_{\varepsilon,\varepsilon+2c}$ are contiguous,
(ii) for all $\varepsilon \ge r$, $\varphi_{\varepsilon+\eta} \circ \psi_\varepsilon$ and $t_{\varepsilon,\varepsilon+2c}$ are contiguous,
(iii) for all $\varepsilon' \ge \varepsilon \ge r$, $\varphi_{\varepsilon'} \circ s_{\varepsilon,\varepsilon'}$ and $t_{\varepsilon+c,\varepsilon'+c} \circ \varphi_\varepsilon$ are contiguous,
(iv) for all $\varepsilon' \ge \varepsilon \ge r$, $s_{\varepsilon+c,\varepsilon'+c} \circ \psi_\varepsilon$ and $\psi_{\varepsilon'} \circ t_{\varepsilon,\varepsilon'}$ are contiguous.

Analogously, if we replace the operator '+' by the multiplication '$\cdot$' in the above definition, then we say that $\mathfrak{S}$ and $\mathfrak{T}$ are *c-multiplicatively interleaved*.

Our main results of this section are the following whose proofs are deferred to the full version. First, Theorem 35 states that the multiscale-mappers induced by any two (c, s)-good towers of covers interleave with each other, implying that their respective persistence diagrams are also close under the bottleneck distance. From this point of view, the persistence diagrams induced by any two (c,s)-good TOCs contain roughly the same information. Next in Theorem 36, we show that the multiscale mapper induced by any (c, s)-good TOC interleaves (at the homology level) with the intrinsic Čech filtration of (X, d_f), thereby implying that the persistence diagram of the multiscale mapper w.r.t. any (c, s)-good TOC is close to that of the intrinsic Čech filtration of (X, d_f) under the bottleneck distance.

▶ **Theorem 35.** *Given a map $f : X \to Z$, let $\mathfrak{V} = \{\mathcal{V}_\varepsilon \xrightarrow{v_{\varepsilon,\varepsilon'}} \mathcal{V}_{\varepsilon'}\}_{\varepsilon\le\varepsilon'}$ and $\mathfrak{W} = \{\mathcal{W}_\varepsilon \xrightarrow{w_{\varepsilon,\varepsilon'}} \mathcal{W}_{\varepsilon'}\}_{\varepsilon\le\varepsilon'}$ be two (c, s)-good tower of covers of Z. Then the corresponding multiscale mappers $\mathrm{MM}(\mathfrak{V}, f)$ and $\mathrm{MM}(\mathfrak{W}, f)$ are c-multiplicatively interleaved.*

▶ **Theorem 36.** *Let $\mathfrak{C}_s(X)$ be the intrinsic Čech filtration of (X, d_f) starting with resolution s. Let $\mathfrak{U} = \{\mathcal{U}_\varepsilon \xrightarrow{u_{\varepsilon,\varepsilon'}} \mathcal{U}_{\varepsilon'}\}_{s\le\varepsilon\le\varepsilon'}$ be a (c, s)-good TOC of the compact connected metric space Z. Then the multiscale mapper $\mathrm{MM}(\mathfrak{U}, f)$ and $\mathfrak{C}_s(X)$ are $2c$-multiplicatively interleaved.*

Finally, given a persistence diagram Dg, we denote its *log-scaled version* $\mathrm{Dg}_{\log}$ to be the diagram consisting of the set of points $\{(\log x, \log y) \mid (x, y) \in \mathrm{Dg}\}$. Since interleaving towers of simplicial complexes induce interleaving persistent modules, using results of [5, 6], we have the following corollary.

▶ **Corollary 37.** *Given a continuous map $f : X \to Z$ and a (c, s)-good TOC $\mathfrak{U}$ of Z, let $\mathrm{Dg}_{\log}\mathrm{MM}(\mathfrak{U}, f)$ and $\mathrm{Dg}_{\log}\mathfrak{C}_s$ denote the log-scaled persistence diagram of the persistence modules induced by $\mathrm{MM}(\mathfrak{U}, f)$ and by the intrinsic Čech filtration $\mathfrak{C}_s$ of (X, d_f) respectively. We have that*

$$d_B(\mathrm{Dg}_{\log}\mathrm{MM}(\mathfrak{U}, f), \mathrm{Dg}_{\log}\mathfrak{C}_s) \le 2c.$$

Acknowledgments. We thank the reviewers for helpful comments.

References

1 U. Bauer, X. Ge and Y. Wang. Measuring distance between Reeb graphs. *Proc. 30th Annual Symp. Comput. Geom., SoCG* (2014), 464–473.

2 S. Biasotti, D. Giorgi, M. Spagnuolo, and B. Falcidieno. Reeb graphs for shape analysis and applications. *Theor. Comput. Sci.*, 392(1-3):5–22, 2008.

3 K. Borsuk. On the imbedding of systems of compacta in simplicial complexes. *Fund. Math.* 35 (1948), 217–234.

4 M. Carrière and S. Y. Oudot. Structure and Stability of the 1-Dimensional Mapper. *Proc. 32nd Int'l Symp. Comput. Geom., SoCG* (2016), 25:1–25:16.

5 F. Chazal, D. Cohen-Steiner, M. Glisse, L. Guibas, and S. Oudot. Proximity of persistence modules and their diagrams. *Proc. 25th Annual Symp. Comput. Geom., SoCG* (2009), 237–246.

6 F. Chazal, V. de Silva, M. Glisse, and S. Oudot. The structure and stability of persistence modules. *SpringerBriefs in Mathematics,* eBook ISBN 978-3-319-42545-0, Springer, 2016.

7 F. Chazal, V. de Silva, and S. Oudot. Persistence stability for geometric complexes. *Geometric Dedicata*, 173(1):193–214, 2014.

8 F. Chazal and J. Sun. Gromov-Hausdorff approximation of filament structure using Reeb-type graph. *Proc. 30th Annual Symp. Comput. Geom., SoCG* (2014), 491–500.

9 V. de Silva, E. Munch, and A. Patel. Categorified Reeb graphs. ArXiv preprint arXiv:1501.04147, (2015).

10 T. K. Dey, F. Mémoli, and Y. Wang. Multiscale mapper: Topological summarization via codomain covers. *ACM-SIAM Symp. Discrete Alg., SODA* (2016), 997–1013.

11 T. K. Dey and Y. Wang. Reeb graphs: Approximation and persistence. *Discrete Comput. Geom.* 49 (2013), 46–73.

12 H. Edelsbrunner and J. Harer. *Computational Topology: An Introduction.* Amer. Math. Soc., Providence, Rhode Island, 2009.

13 H. Edelsbrunner, J. Harer, and A. K. Patel. Reeb spaces of piecewise linear mappings. In *Proc. 24th Annual Symp. Comput. Geom., SoCG* (2008), 242–250.

14 E. Gasparovic, M. Gommel, E. Purvine, R. Sazdanovic, B. Wang, Y. Wang and L. Ziegelmeier. A complete characterization of the one-dimensional intrinsic Čech persistence diagrams for metric graphs. *Manuscript, an earlier version appeared as a report for IMA Workshop for Women in Computational Topology (WinCompTop)*, 2016.

15 A. Hatcher. Algebraic Topology. Cambridge U. Press, New York, 2002.

16 J. Leray. L'anneau spectral et l'anneau filtré d'homologie d'un espace localement compact et d'une application continue. *J. Math. Pures Appl.* 29 (1950), 1–139.

17 P. Y. Lum, G. Singh, A. Lehman, T. Ishkhanikov, M. Vejdemo-Johansson, M. Alagappan, J. Carlsson, and G. Carlsson. Extracting insights from the shape of complex data using topology. *Scientific reports* 3 (2013).

18 E. Munch and B. Wang. Convergence between categorical representations of Reeb space and mapper. *32nd Int'l Symp. Comput. Geom., SoCG* (2016), 53:1–53:16.

19 J. R. Munkres, *Topology*, Prentice-Hall, Inc., New Jersey, 2000.

20 V. Prasolov. *Elements of combinatorial and differential topology.* American Mathematical Soc., Vol. 74, 2006.

21 G. Singh, F. Mémoli, and G. Carlsson. Topological Methods for the Analysis of High Dimensional Data Sets and 3D Object Recognition. *Symp. Point Based Graphics*, 2007.

Locality-Sensitive Hashing of Curves*†

Anne Driemel[1] **and Francesco Silvestri**[2]

1 **Department of Mathematics and Computer Science, Eindhoven University of Technology, Eindhoven, The Netherlands**
adriemel@tue.nl

2 **Department of Information Engineering, University of Padova, Padova, Italy; and**
Theoretical Computer Science, IT University of Copenhagen, Copenhagen Denmark
silvestri@dei.unipd.it

Abstract

We study data structures for storing a set of polygonal curves in $\mathbb{R}^d$ such that, given a query curve, we can efficiently retrieve similar curves from the set, where similarity is measured using the discrete Fréchet distance or the dynamic time warping distance. To this end we devise the first locality-sensitive hashing schemes for these distance measures. A major challenge is posed by the fact that these distance measures internally optimize the alignment between the curves. We give solutions for different types of alignments including constrained and unconstrained versions. For unconstrained alignments, we improve over a result by Indyk from 2002 [17] for short curves. Let n be the number of input curves and let m be the maximum complexity of a curve in the input. In the particular case where $m \leq \frac{\alpha}{4d} \log n$, for some fixed $\alpha > 0$, our solutions imply an approximate near-neighbor data structure for the discrete Fréchet distance that uses space in $O(n^{1+\alpha} \log n)$ and achieves query time in $O(n^{\alpha} \log^2 n)$ and constant approximation factor. Furthermore, our solutions provide a trade-off between approximation quality and computational performance: for any parameter $k \in [m]$, we can give a data structure that uses space in $O(2^{2k} m^{k-1} n \log n + nm)$, answers queries in $O(2^{2k} m^k \log n)$ time and achieves approximation factor in $O(m/k)$.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Locality-Sensitive Hashing, Fréchet distance, Dynamic Time Warping

Digital Object Identifier 10.4230/LIPIcs.SoCG.2017.37

1 Introduction

We study nearest-neighbor searching for polygonal curves under the discrete Fréchet distance or the dynamic time warping distance. This problem has various applications in machine learning, information retrieval and classification where the recorded instances are curves. Dynamic time warping has shown to be useful for classification of various types of data: surgical processes [11], whale singing [5], chromosomes [23], fingerprints [22], electrocardiogram (ECG) frames [15], and vessel trajectories [30]. Originally conceived for speech recognition, it is now being deployed as universal similarity measure for time series in the field of data

* A full version of the paper is available at https://arxiv.org/abs/1703.04040.

† Driemel has been supported by NWO Veni project "Clustering time series and trajectories (10019853)". Silvestri has been supported by the European Research Council project "Scalable Similarity Search" (no. 614331) and by MIUR of Italy under project AMANDA.

mining. The Fréchet distance is considered a useful similarity measure for trajectories of moving objects [6, 12, 20, 31].

Indyk and Motwani [19, 14] introduced the idea that hashing could enable faster nearest-neighbor searching in high-dimensional Euclidean spaces using a hashing scheme where near points are more likely to collide than far ones. They showed that such an approach can be used for the (c, r)-*near neighbor problem* which is defined as follows. Preprocess a set S of n points into a data structure that answers queries in the following way: if there exists a point $p \in S$ that lies within distance r from the query point q, then the data structure reports a point $p' \in S$ that lies within distance cr from q. In this paper, we study such locality-sensitive hashing schemes for the space of curves.

1.1 State of the art

In 2002, Indyk gave a deterministic and approximate near-neighbor data structure for the discrete Fréchet distance [17]. This data structure is to date the only result known for this task and represents the state of the art. The data structure achieves approximation factor $O(\log m + \log\log n)$, where m is the maximum length of a curve and n is the maximum number of elements in the data structure. Further, the data structures uses space in $O\Big(|X|^{\sqrt{m}}(m^{\sqrt{m}}n)^2\Big)$, where $|X|$ is the size of the domain on which the curves are defined. The query time is $O\big(m^{O(1)}\log n\big)$. The data structure precomputes all answers to queries with curves of length $\sqrt{m}$, leading to a very high space consumption.[1]

In the group of ℓ_p distances, the Fréchet distance most resembles the ℓ_∞-distance, which is notoriously hard to embed into a low-dimensional ℓ_p-space, see also the discussion by Indyk in [16]. Indyk's data structure for the discrete Fréchet distance is in fact an extension of his data structure for the ℓ_∞-distance [16]. Any subset of ℓ_∞^d can be embedded into the Fréchet metric[2] [18]. This embedding implies that, unless the strong exponential-time hypothesis fails, there exists no data structure for near-neighbor searching under the discrete Fréchet distance that achieves preprocessing time in $O\big(n^{2-\varepsilon}\operatorname{poly} m\big)$, query time in $O\big(n^{1-\varepsilon}\operatorname{poly} m\big)$ for any $\varepsilon > 0$, and approximation factor $c < 3$. This suggests that the problem becomes hard for long curves, i.e., $m \in \omega(\log n)$. Recently, Backurs and Sidiropoulos showed how to embed finite subsets of the Hausdorff distance into ℓ_∞ using constant distortion and constant dimension of the host space [2]. However, for the Fréchet distance, no non-trivial embeddings are known, see also the discussion in [18]. It is possible to embed any finite metric space into ℓ_p, for example, using the embedding due to Bourgain [25]. However, the high cost of computing the embedding makes it unfit for use in a nearest-neighbor data structure. Another known approach to proximity searching in metric spaces is to exploit a low doubling-dimension [1, 13]. However, the doubling dimension of the Fréchet distance is infinite, even if the metric space is restricted to curves of constant length [9]. Recently Bartal *et al.* [3] gave lower bounds for embedding doubling spaces. Their result implies that a metric embedding of the Fréchet distance into an ℓ_p-space would have at least super-constant distortion. However, as noted earlier, it is not even known how to obtain such an embedding.

In general, there is little known in terms of data structures for the Fréchet distance.

---

[1] Indyk also claims (without proof) a slightly different bound using a trade-off parameter $t \geq 2$: approximation factor $O((\log m + \log\log n)^{(t-1)})$, space $O\Big(\big(m^2|X|\big)^{tm^{1/t}} n^{2t}\Big)$ and query time $(m + \log n)^{O(t)}$. The space bound decreases at the cost of approximation and query time as soon as $t < \log m$; however, the trade-off disappear for larger values of t since all bounds increase in t as soon as $t \geq \log m$.

[2] In particular, one can use $3d$ vertices to express each d-dimensional vector as a curve on a real line.

The authors are aware of the following few results which were developed for the classic (continuous) Fréchet distance. De Berg, Cook and Gudmundsson [7] study range counting queries for the set of subcurves that lie within distance r to a query line segment. Their data structure uses a partition tree to store compressed subcurves. For any parameter $n \leq s \leq n^2$, the space used by the data structure is in $O(s \operatorname{poly} \log n)$. The queries are computed in time in $O\left(\frac{n}{\sqrt{s}} \operatorname{poly} \log n\right)$ and uses a constant approximation factor. However, the data structure does not support more complex query curves than line segments. A second data structure is due to Driemel and Har-Peled [8]. This data structure answers queries for the Fréchet distance of a subcurve to a query curve (the subcurve is specified in the query). If the queries are line segments, an approximation factor of $(1+\varepsilon)$ can be achieved with logarithmic query time and linear space. Unlike the ℓ_∞-metric, which can be evaluated in time that is linear in the dimension, evaluating a single Fréchet distance is believed to take time that is at least roughly quadratic in the complexity of the curves (the number of vertices) in the worst case [4]. The high time complexity can be credited to the fact that the distance measure optimizes over all possible monotone alignments of the two input sequences. Computing the discrete Fréchet distance, as well as dynamic time warping, can be solved via dynamic programming. In both cases, the naive linear scan leads to $O(nm^2)$ query time for finding the nearest neighbor. For dynamic time warping (DTW) no data structures exist that give provable guarantees, however there exist many heuristics, see the work of Rakthanmanon *et al.* [26] (and references therein). Since DTW does not satisfy the triangle inequality, it cannot be embedded into an ℓ_p-space.

1.2 Our results

Our first result is a basic LSH scheme for the discrete Fréchet distance, which leads to a very efficient LSH with approximation factor that is linear in the number of curve vertices. The scheme is described in Section 3 and it is surprisingly simple: We snap the curves to a randomly shifted grid and remove consecutive duplicate vertices. It turns out that this simple scheme alleviates the alignment problem which sets the Fréchet distance computation apart from the ℓ_∞-distance. Next, we show in Section 4 that it is even possible to get constant approximation, at the cost of a lower collision probability for near curves. The second scheme randomly perturbs the vertices of the input curves independently and snaps the vertices to a fixed grid instead of a randomly shifted grid. It is natural to ask if there exists an LSH scheme exhibiting a full-spectrum trade-off between collision probability and approximation. We positively answer to this question in Section 5, with a scheme based on a random partition of the input curves, inspired by Indyk's data structure [17], followed by the application of the basic scheme to each subsequence independently.[3] (Due to space constraints, we refer to the full version of our paper [10] for more details.)

All the LSH schemes achieve zero false-positives, meaning that no collisions happen between far curves. When applied to the (c, r)-near neighbor problem, we obtain the results summarized in Table 1 (see also Section 2.3). It is interesting to compare our bounds with the state of the art. The basic scheme of Theorem 7 provides a data structure using almost linear space and $O(m \log n)$ query time by allowing a linear approximation $c = O(m)$. This query time always beats the trivial exact solution of scanning all input curves for each query, which needs $O\left(nm^2\right)$ time. In comparison, Indyk's result [17] provides a better approximation when

[3] Indeed, in Theorem 10, the collision probability for near curves is bounded by 2^{-3M} for $K = M$ (using Stirling's approximation for the binomial coefficient), however when summarizing our bounds we use the simplified bound from Corollary 11.

Table 1 Our approximate near-neighbor data structure results for the discrete Fréchet distance in comparison with the result by Indyk, assuming $d = O(1)$ for simplicity. The first four rows refer to the standard discrete Fréchet distance d_F, while the last two rows $d_{w,\mathrm{aF}}$ and $d_{w,\mathrm{sF}}$ refer to the anchored and speed constraints respectively. The input consists of n polygonal curves in $\mathbb{R}^d$, each of complexity at most m. The corresponding query results are achieved with high probability. The parameters $k \geq 1$ and $\ell \geq 1$ trade-off space/query time and approximation, and parameter w constrains the possible alignments. The first entry in bi-criteria $(\cdot, \cdot)$ denotes the distance approximation, while the second is the alignment approximation.

	Space	Query time	Approximation	Reference
d_F	$O\left(\lvert X\rvert^{\sqrt{m}}(m^{\sqrt{m}}n)^2\right)$	$O\left(m^{O(1)}\log n\right)$	$O(\log m + \log\log n)$	[17]
	$O(n\log n + nm)$	$O(m\log n)$	$O(m)$	Thm. 7
	$O(2^{4md}n\log n + nm)$	$O(2^{4md}m\log n)$	$O(1)$	Thm. 9
	$O(2^{2k}m^{k-1}n\log n + nm)$	$O(2^{2k}m^k\log n)$	$O(m/k)$	Cor. 11
$d_{w,\mathrm{aF}}$	$O\left(\left(\sqrt{2}w\right)^{2m/\ell}n\log n + nm\right)$	$O\left(\left(\sqrt{2}w\right)^{2m/\ell}m\log n\right)$	bi-criteria $\left(4d^{\frac{3}{2}}\ell, 2\ell-2\right)$	Thm. 12
$d_{w,\mathrm{sF}}$	$O\left(\left(\sqrt{2}w\ell\right)^{2m/\ell}n\log n + nm\right)$	$O\left(\left(\sqrt{2}w\ell\right)^{2m/\ell}m\log n\right)$	bi-criteria $\left(4d^{\frac{3}{2}}\ell, \ell\right)$	Thm. 13

$m = \Omega(\log\log n)$ but it uses exponential space and slightly higher query time. More generally, when curves are short $m = o(\log n)$, our basic result provides a good alternative to Indyk's result due to the improved space. In the particular case where $m \leq \frac{\alpha}{4d}\log n$ for some fixed $\alpha > 0$, we can answer queries using a constant approximation factor in $O(n^\alpha \log^2 n)$ time and using $O(n^{1+\alpha}\log n)$ space, using Theorem 9. When curves have constant complexity, the basic LSH gives the first efficient data structure with constant approximation. We recall that a data structure for the (c, r)-approximate near neighbor problem can be used as a building block for solving the c-approximate nearest neighbor problem [28].

We then address LSH for the discrete Fréchet distance under alignment constraints in Section 6. It is natural to constrain the alignments of curves: this preserves important characteristics of the input curves and it also reduces the actual time to compute the distance between curves (see e.g., [27, 21]). We target the anchored and bounded speed constraints that require, respectively, a vertex to be aligned with at most w vertices or to be aligned with vertices whose indices differ by at most $w/2$, for a suitable parameter $w \geq 1$ (for formal definitions see Section 2.2). Our scheme provides the first data structures for the (c, r)-near neighbor problem with alignment constraints. Further, they exhibit a bi-criteria approximation: it is possible to reduce space and query time with a weaker approximation on the distance but also on the alignment parameter w. Bounds are summarized in Table 1.

In Section 7, we study which one of our schemes work for DTW. We show that the basic LSH applies to DTW with the same linear approximation, space and query bounds of the discrete Fréchet distance. In contrast, the techniques to improve the approximation factor under the Fréchet distance do not provide improvements for DTW. The LSH schemes for constrained distances also yields linear approximation for DTW distance, but maintains the trade-off between space/query time and the approximation on the alignment parameter w.

2 Preliminaries

2.1 Distance measures for curves

A *time series* (or *trajectory*)[4] is a series $(p_1, t_1), \ldots, (p_m, t_m)$ of measurements p_i of a signal taken at times t_i. We assume $0 = t_1 < t_2 < \ldots < t_m = 1$ and m is finite. A time series may be

[4] Usually, these are referred to as time series when $d = 1$ and trajectories when $d > 1$.

viewed as a continuous function $P : [0,1] \to \mathbb{R}^d$ by linearly interpolating $p_1, \ldots, p_m$ in order of t_i, $i = 1, \ldots m$. We obtain a polygonal curve with *vertices* $p_1 = P(t_1), \ldots, p_m = P(t_m)$ and segments between p_i and p_{i+1} called *edges* $\overline{p_i p_{i+1}} = \{xp_i + (1-x)p_{i+1} | x \in [0,1]\}$. We will simply refer to P as a *curve*. We denote the space of all curves in $\mathbb{R}^d$ with Δ^d.

We now recall the definitions of discrete Fréchet distance and of the dynamic time warping distance between two curves. To this end we define the concept of traversal. Given two polygonal curves $P = p_1, \ldots, p_{m_1}$ and $Q = q_1, \ldots, q_{m_2}$, a *traversal*

$$T = \{(i_1, j_1), (i_2, j_2), \ldots, (i_\ell, j_\ell)\}$$

is a sequence of pairs of indices referring to a *pairing* of vertices from the two curves with the following properties:

(i) $i_1 = 1$, $j_1 = 1$, $i_\ell = m_1$, and $j_\ell = m_2$
(ii) $\forall (i_k, j_k) \in T : (i_{k+1} - i_k) \in \{0,1\} \wedge (j_{k+1} - j_k) \in \{0,1\}$.
(iii) $\forall (i_k, j_k) \in T : (i_{k+1} - i_k) + (j_{k+1} - j_k) \geq 1$.

Intuitively, one can think of the traversal as a prescribed schedule for simultaneously traversing the two curves, starting at the first vertex of each curve, in every step the traversal advances by one vertex, either on one of the curves, or on both curves simultaneously, finally the traversal has to end at the last vertices of the two curves.

We consider the maximum distance of two vertices paired by a traversal as the cost incurred by this traversal. Let $\mathcal{T}$ be the set of possible traversals for two curves P and Q, then the Fréchet distance corresponds to the minimal cost of a traversal of the two curves. Likewise, if we define the cost of a traversal as the sum of distances between paired vertices, then the traversal with minimum cost corresponds to the dynamic time warping distance.

▶ **Definition 1.** Let $\mathcal{T}$ be the set of possible traversals for two curves P and Q. The *discrete Fréchet distance* $d_F(P,Q)$ between curves P and Q is defined as

$$d_F(P,Q) = \min_{T \in \mathcal{T}} \max_{(i_k, j_k) \in T} \|p_{i_k} - q_{j_k}\|.$$

▶ **Definition 2.** Let $\mathcal{T}$ be the set of possible traversals for two curves P and Q. The *dynamic time warping (DTW) distance* $d_{DTW}(P,Q)$ between curves P and Q is defined as

$$d_{\mathrm{DTW}}(P,Q) = \min_{T \in \mathcal{T}} \sum_{(i_k, j_k) \in T} \|p_{i_k} - q_{j_k}\|.$$

The discrete Fréchet distance satisfies the triangle inequality and is a pseudo-metric. This is not true for the DTW distance, since it does not satisfy the triangle inequality.

We refer to a traversal realizing the distance of two curves as an *optimal traversal*. We can interpret a traversal as the edges of a bipartite graph where the nodes are the vertices of the two curves and the edges connect the pairs. The following simple lemma holds for all distance measures. As a consequence, we assume in the paper that an optimal traversal consists of disconnected stars, that we call *components*.

▶ **Lemma 3.** *For any two curves $P = p_1, \ldots, p_{m_1}$ and $Q = q_1, \ldots, q_{m_2}$, there always exists an optimal traversal T with the following two properties:*

(i) *T consists of at most $m = \min\{m_1, m_2\}$ disconnected components.*
(ii) *Each component is a star, i.e., all edges of this component share a common vertex.*

Proof. The first part is immediate, since we can charge each component to a vertex of the shorter curve that is contained in it. To see the second part of the claim, assume for the sake

of contradiction that an optimal traversal has the pairs $(i,j),(i,j+1)(i+1,j+1)$ for some i,j (or the symmetric configuration $(i,j),(i+1,j)(i+1,j+1)$). In this case, the middle pair $(i,j+1)$ can be removed without increasing the cost and without invalidating the traversal properties. We can apply this reasoning repeatedly until each component is a star. ◀

2.2 Distances measures with constraints

Anchored distances. A traversal T is said *w-anchored traversal* if each vertex is paired with a vertex at distance at most $w/2$ (for simplicity we assume w to be even): namely, $|i-j| \leq w/2$ for each $(i,j) \in T$. Parameter w is called the *width* of the traversal. Such a traversal exists only if $|m_1 - m_2| \leq w/2$, otherwise there would be unpaired vertices (e.g., the last vertex of the longest curve). For two curves P and Q with lengths satisfying $|m_1 - m_2| \leq w/2$, we define the *w-anchored discrete Fréchet distance* $d_{w,aF}(P,Q)$ and *w-anchored DTW distance* $d_{w,aDTW}(P,Q)$ as in Definitions 1 and 2 where $\mathcal{T}$ is defined as the set of all possible w-anchored traversals.

Speed-constrained distances. A traversal T is a *w-speed traversal* if each vertex is aligned with at most w vertices of the other curve: in other terms, the bipartite graph representing the traversal has degree at most w. Parameter w is called the *speed* of the traversal. (We overload the meaning of w since the width and speed parameters play a similar role in our algorithms.) Note that a w-anchored traversal is a $(w+1)$-speed traversal, but the opposite is not necessary true. A w-speed traversal exists only if $1/w \leq m_1/m_2 \leq w$. For two curves P and Q with lengths satisfying $1/w \leq m_1/m_2 \leq w$, we defined the *w-speed discrete Fréchet distance* $d_{w,sF}(P,Q)$ and *w-speed DTW distance* $d_{w,sDTW}(P,Q)$ as in Definitions 1 and 2 where $\mathcal{T}$ is defined as the set of all possible w-speed traversals.

2.3 Locality-sensitive hashing

We use the notion of asymmetric locality-sensitive hashing (see, e.g. [29]), defined as follows:

▶ **Definition 4.** Let $\mathcal{S}$ be the set of curves in $\mathbb{R}^d$ and let $d : \mathcal{S} \times \mathcal{S} \to \mathbb{R}^+$ be a distance measure defined on them. Given real values $r > 0$, $c > 1$, $0 \leq \alpha_1 \leq 1$ and $0 \leq \alpha_2 \leq 1$ with $\alpha_1 > \alpha_2$, a family $\mathcal{H}$ of pairs of hash functions (h_1, h_2) is called $(r, c, \alpha_1, \alpha_2)$-*sensitive* if for any two curves $P, Q \in \mathcal{S}$

(i) if $d(P,Q) \leq r$, then $Pr_{(h_1,h_2)\in\mathcal{H}}(h_1(P) = h_2(Q)) \geq \alpha_1$;
(ii) if $d(P,Q) > cr$, then $Pr_{(h_1,h_2)\in\mathcal{H}}(h_1(P) = h_2(Q)) \leq \alpha_2$.

When $h_1 = h_2$, we have the traditional definition of (symmetric) locality-sensitive hashing. The above scheme is *asymmetric* in the sense that there are two different schemes and the guarantees only hold for curves P and Q where P was hashed using the first scheme and Q was hashed using the second scheme. This is useful, e.g., if the application of the LSH is a nearest neighbor data structure, where comparisons only need to be done between input objects and query objects.

The results reported in Table 1 follow by applying the standard framework for solving the (c,r)-near neighbor problem with an $(r, c, \alpha_1, \alpha_2)$-sensitive hashing scheme $\mathcal{H}$. For the sake of completeness, we sketch this process here.[5] A new family $\mathcal{H}'$ of hashing is constructed by

[5] We observe that the LSH schemes presented in this paper have long hash values (curves or array of curves). However, they can be shortened with traditional hashing (i.e., by mapping each value in $[0, O(n)]$), that allows for a more efficient search in the hash tables generated by the LSH. This technique increases α_2 by an additive $O(1/n)$ term.

concatenating $k = \max\{1, \log_{\alpha_2}(1/n)\}$ hash functions from $\mathcal{H}$, so that the collision probability of far points is at most $1/n$. Then, each point in S is inserted into $L = (1/\alpha_1)^k$ hash tables, each corresponding to a different randomly chosen hash function from $\mathcal{H}'$. For a query point q, the algorithm searches among all points that collide with q in the L hash tables and stops as soon as a cr-near neighbor is found. When $\alpha_2 > 0$, the data structure requires $O\big(n^{1+\rho} + nm\big)$ memory words and query time $O(\Gamma n^{\rho})$, where $\Gamma = \Omega(m)$ is the time required for computing the distance between two curves and $\rho = \log \alpha_1 / \log \alpha_2$. When $\alpha_2 = 0$, the data structure requires $O(n/\alpha_1)$ memory words and query time $O(m/\alpha_1)$. Note that in this case the query time does not include Γ: the algorithm does not need to compute distances between q and points in the buckets since there are no false positives. For a given query, the data structures returns an approximate cr-near neighbor with constant probability. To obtain high probability (i.e., at least $1 - 1/n$) we repeat the above process $\log n$ times, leading to $\log n$ different data structures. This increases space and query time by a $O(\log n)$ term.

3 Linear approximation factor

We first present the basic LSH scheme in Section 3.1, and then in Section 3.2 we analyze its correctness and performance for the discrete Fréchet distance. The basic LSH has an approximation factor that is linear in the number of vertices that a curve can have.

3.1 Algorithm

We use a randomly shifted grid in our hashing scheme. Let the canonical d-dimensional grid of resolution δ be defined as an evenly spaced point set in $\mathbb{R}^d$, as follows:

$$G_\delta = \Big\{ (x_1, \ldots, x_d) \in \mathbb{R}^d \mid \forall\ 1 \le i \le d\ \exists\ j \in \mathbb{N}\ :\ x_i = j \cdot \delta \Big\}.$$

Consider a family of such grids parametrized by a shift t:

$$\widehat{G}^t_\delta = \{p + t \mid p \in G_\delta\}.$$

Choosing t uniformly at random from the half-open hypercube $[0, \delta)^d$ we obtain a family of randomly shifted grids. Let $P \in \Delta^d$ be a polygonal curve with vertices $p_1, \ldots, p_m$ and let $h^t_\delta : \Delta^d \to \Delta^d$ be a hash function. The curve $h^t_\delta(P)$ is defined as the result of the following two-stage construction.

(i) We snap the curve to the grid $\widehat{G}^t_\delta$. More precisely, we replace each vertex p_i with its closest grid point $p'_i = \arg\min_{q \in \widehat{G}^t_\delta} \|p_i - q\|$ to obtain the curve P'.

(ii) We remove consecutive duplicates in P'. That is, we remove the vertex p'_i if it is identical to p'_{i-1}.

Let $\mathcal{H}^{\mathrm{L}}_\delta$ be the family of hash functions h^t_δ constructed this way.

3.2 Analysis

▶ **Lemma 5.** *Let $P, Q \in \Delta^d$ be two curves with m_1 and m_2 points, respectively, and let $m = \min\{m_1, m_2\}$. For any $\delta > 0$, it holds that*

$$Pr_{\mathcal{H}^{\mathrm{L}}_\delta}\big(h^t_\delta(P) = h^t_\delta(Q)\big) \ge 1 - \left(2dm \cdot \frac{d_F(P,Q)}{\delta}\right).$$

Proof. We bound the probability that P and Q do not hash to the same sequence. To this end, consider an optimal traversal T of P and Q with respect to the discrete Fréchet distance.

By Lemma 3, we can assume that $|T| \leq m$ and each component is a star. Let ℓ denote the number of components of T. For $1 \leq k \leq \ell$ denote with E_k the event that not all vertices of the k-th component are snapped to the same grid point. This happens only if at least one pair of vertices is separated in at least one dimension by the random shift t.

Since the component is a star, there exists a vertex v of either P or Q, such that v is involved in all pairs of T in the k-th component. Therefore, all vertices in this component have distance at most $d_F(P,Q)$ to v. Since t is uniformly distributed in $[0,\delta)^d$, the probability that any pair is separated along any fixed dimension is $2d_F(P,Q)/\delta$. As a consequence, event E_k happens with probability at most $2d \cdot d_F(P,Q)/\delta$.

By a union bound over the ℓ components in T, we have that the probability of P and Q not being hashed to the same sequence is bounded by

$$Pr\left(\bigcup_{1\leq k\leq \ell} E_k\right) \leq \sum_{1\leq k\leq \ell} Pr(E_k) = 2dm \cdot \frac{d_F(P,Q)}{\delta}$$

and the lemma follows. ◀

▶ **Lemma 6.** *For any value of δ and for any $P, Q \in \Delta^d$, if there exists a value of $t \in [0,\delta)^d$ such that $h^t_\delta(P) = h^t_\delta(Q)$, then it holds that $d_F(P,Q) \leq \sqrt{d} \cdot \delta$.*

Proof. In the case that $h^t_\delta(P) = h^t_\delta(Q)$, it holds that $d_F(h^t_\delta(P), h^t_\delta(Q)) = 0$. Snapping a curve to the randomly shifted grid changes the position of each vertex by at most $\frac{\sqrt{d}}{2} \cdot \delta$. Therefore, it holds that $d_F(P, h^t_\delta(P)) \leq \frac{\sqrt{d}}{2} \cdot \delta$ and similarly $d_F(Q, h^t_\delta(Q)) \leq \frac{\sqrt{d}}{2} \cdot \delta$. By the triangle inequality, $d_F(P,Q) \leq d_F(h^t_\delta(P), P) + d_F(h^t_\delta(P), h^t_\delta(Q)) + d_F(h^t_\delta(Q), Q) \leq \sqrt{d} \cdot \delta$. ◀

The next theorem follows by plugging in the bounds of Lemmas 5 and 6.

▶ **Theorem 7.** *Let $P, Q \in \Delta^d$ be two curves with m_1 and m_2 points, respectively, and let $m = \min\{m_1, m_2\}$, $\delta = 4dmr$ and $c = 4d^{\frac{3}{2}}m$. It holds that:*
(i) *if $d_F(P,Q) < r$, then $Pr_{\mathcal{H}^L_\delta}(h^t_\delta(P) = h^t_\delta(Q)) > \frac{1}{2}$;*
(ii) *if $d_F(P,Q) > cr$, then $Pr_{\mathcal{H}^L_\delta}(h^t_\delta(P) = h^t_\delta(Q)) = 0$.*

4 Constant approximation factor

In the previous section we analyzed a very efficient LSH with linear approximation factor. On the other end of the spectrum, we can also design an LSH with constant approximation factor, but higher running time. Conceptually, the easiest way to do this is to randomly and independently perturb the vertices of each curve and snap them to a fixed grid.

4.1 Algorithm

The described scheme is asymmetric. We assume that we have two types of curves, which we call input curves and query curves. Consider an input curve $P = p_1, \ldots, p_m$, and let G_δ be the canonical d-dimensional grid of resolution δ defined in the previous section. Let $t_P = t_1, \ldots, t_m$ be a sequence of independent random variables which are uniformly distributed in $\left[-\frac{\delta}{2}, \frac{\delta}{2}\right]^d$. We perturb the vertices of P: Let $P' = p'_1, \ldots, p'_m$ be the perturbed curve with $p'_i = p_i + t_i$. We snap the curve P' to the grid G_δ. More precisely, we replace each vertex p'_i with its closest grid point $p''_i = \arg\min_{q \in G_\delta} \|p'_i - q\|$ to obtain the curve P''. In the next step we remove consecutive duplicates in P''. That is, we remove the vertex p''_i if it is identical to p''_{i-1}. We define $h^{t_P}_\delta(P)$ to be the result of this algorithm.

For a query curve Q, the hash function is the same. However, a different random sequence t_Q is used for randomly perturbing the curve. We let $\mathcal{H}_\delta^{\mathsf{c}}$ denote the LSH scheme defined this way: namely, $\mathcal{H}_\delta^{\mathsf{c}}$ contains all pairs $(h_\delta^{t_P}, h_\delta^{t_Q})$, where vectors t_P and t_Q consist of entries independent and identically distributed in $\left[-\frac{\delta}{2}, \frac{\delta}{2}\right]^d$.

4.2 Analysis

▶ **Lemma 8.** *Let $P, Q \in \Delta^d$ be two curves with m_1 and m_2 points, respectively. Let $m = \min\{m_1, m_2\}$ and let $M = \max\{m_1, m_2\}$. For any $\delta > 0$, it holds that*

$$Pr_{\mathcal{H}_\delta^{\mathsf{c}}}\left(h_\delta^{t_P}(P) = h_\delta^{t_Q}(Q)\right) \geq \left(\frac{1}{2}\right)^{dm} \cdot \left(\frac{1}{2} - \frac{d_F(P,Q)}{\delta}\right)^{dM}$$

In particular, if $\delta > 4d_F(P,Q)$, then the probability is strictly lower bounded by $2^{-2d(m_1+m_2)}$.

Proof. Note that for $d_F(P,Q) \geq \frac{\delta}{2}$ the claim is trivially true. Therefore, assume that $d_F(P,Q) < \frac{\delta}{2}$. For simplicity assume first that $d = 1$. We bound the probability that P and Q do not hash to the same sequence. To this end, consider an optimal traversal T of P and Q with respect to the discrete Fréchet distance. By Lemma 3, we can assume that $|T| \leq m_1 + m_2$ and each component is a star. Let ℓ denote the number of components of T. For $1 \leq k \leq \ell$ denote with E_k the event that not all vertices of the k-th component are snapped to the same grid point. Assume that the center of the k-th star is a vertex p_i of P and that the other vertices of the component are vertices $q_j, \ldots, q_{j+c_k}$ of Q. The analysis for the case where the center is a vertex of Q is analogous. There must be a grid point in either one of the two intervals to the left and to the right of p_i: $I_l = [p_i - \frac{\delta}{2}, p_i)$ and $I_r = [p_i, p_i + \frac{\delta}{2})$. We analyze the case that there is a grid point in I_r, the other case is analogous. Let X_i be the event that $p_i' \in I_r$. Since t_P is uniformly random in $\left[-\frac{\delta}{2}, \frac{\delta}{2}\right]^{m_1}$, it holds that $Pr(X_i) \geq \frac{1}{2}$. Now, let Y_j be the event that $q_j' \in I_r$. If q_j was in p_i's component, then there are two cases. Either q_j lies in I_l or in I_r. In the first case, we have

$$Pr(Y_i) \geq \frac{\frac{\delta}{2} - |p_i - q_j|}{\delta} \geq \frac{1}{2} - \frac{d(P,Q)}{\delta},$$

and in the second case we have $Pr(Y_i) \geq \frac{1}{2}$. We can bound the probability that all vertices in the k-th component snap to the same grid point

$$Pr\left(\overline{E_k}\right) \geq Pr(X_i \cap Y_j \cap \cdots \cap Y_{j+c_k}) \geq \frac{1}{2} \cdot \left(\frac{1}{2} - \frac{d(P,Q)}{\delta}\right)^{c_k}.$$

If all components are preserved, then the two curves will hash to the same sequence, therefore

$$\begin{aligned} Pr\left(h_\delta^{t_P}(P) = h_\delta^{t_Q}(Q)\right) &\geq Pr\left(\bigcap_{1\leq k\leq \ell} \overline{E_k}\right) \geq \prod_{1\leq k\leq \ell} Pr\left(\overline{E_k}\right) \\ &\geq \prod_{1\leq k\leq \ell} \frac{1}{2}\left(\frac{1}{2} - \frac{d(P,Q)}{\delta}\right)^{c_k} \geq \left(\frac{1}{2}\right)^{\ell}\left(\frac{1}{2} - \frac{d(P,Q)}{\delta}\right)^{m_1+m_2-\ell}. \end{aligned}$$

The last inequality follows since $\left(\sum_{1\leq k\leq \ell} c_k\right) = m_1 + m_2 - \ell$. Indeed, each center of a component can be charged to this component and the remaining vertices make up the sum of the leaves of all components. The lemma is now implied for $d = 1$ observing that $\ell \leq \min\{m_1, m_2\}$, as implied by Lemma 3. We get the lemma for general d by observing that the dimensions are independent. ◀

The next theorem follows by plugging in the bounds of Lemma 8 and by using same arguments as in the proof of Lemma 6.

▶ **Theorem 9.** *Let $P, Q \in \Delta^d$ be two curves with m_1 and m_2 points, respectively, and let $\delta = 4dr$ and $c = 4d^{3/2}$. It holds that*

(i) *if $d_F(P,Q) < r$, then $Pr_{\mathcal{H}_\delta^c}\left(h_\delta^{t_P}(P) = h_\delta^{t_Q}(Q)\right) > \left(\frac{1}{2}\right)^{2d(m_1+m_2)}$;*

(ii) *if $d_F(P,Q) > cr$, then $Pr_{\mathcal{H}_\delta^c}\left(h_\delta^{t_P}(P) = h_\delta^{t_Q}(Q)\right) = 0$.*

5 Trade-off between approximation factor and query time

In the previous two sections we have seen schemes with linear and constant approximations. We now suggest a scheme exhibiting a trade-off between the collision probability of near points and the approximation factor. The basic idea is to randomly partition the input curves and to concatenate the outcome of the basic LSH (Section 3) applied to the different parts of the curves.

5.1 Algorithm

The scheme is asymmetric. Again, we assume that we have two types of curves, which we call input and query curves. The difference in how they are handled lies in the way we create the partition. For an input curve $P = p_1, \ldots, p_m$, we randomly sample a partition into K subsequences. To this end, we denote a partition of P with $\Phi^s(P) = \left(\widehat{P}_1, \ldots, \widehat{P}_K\right)$ where the subsequences are defined by a monotone sequence $s \in [m]^{K-1}$ as follows.

$$\widehat{P}_1 = p_1, \ldots, p_{s_1}; \qquad \forall\ 1 < i < K\ :\ \widehat{P}_i = p_{s_{i-1}}, \ldots, p_{s_i}; \qquad \widehat{P}_K = p_{s_{K-1}}, \ldots, p_m.$$

There are at most $\binom{m+K-1}{K-1}$ ways to partition a curve of length m in this way. We denote with $\mathcal{P}_K$ the family of all valid partitions for a given m. Let $t = t_1, \ldots, t_K$ be a sequence of independent random values evenly distributed in $[0, \delta)^d$. Once we have partitioned the input curve P into K (overlapping) subsequences, we apply the basic LSH to each individual subsequence and concatenate the resulting curves:

$$g_{\delta,K}^{t,s}(P) = h_\delta^{t_1}\left(\widehat{P}_1\right) \oplus h_\delta^{t_2}\left(\widehat{P}_2\right) \oplus \cdots \oplus h_\delta^{t_K}\left(\widehat{P}_K\right).$$

A query curve $Q = q_1, \ldots, q_m$ is subdivided into K equal-sized subsequences (deterministically), where the last subsequence may be shorter and two consecutive sequences overlap by one element. We denote with $\Phi^*(Q)$ this partitioning into equal-sized subsequences. For query curves, we define $g_{\delta,K}^{t,*}(Q)$ to be the resulting curve given by applying the basic LSH to each individual subsequence and concatenating the resulting curves. For any given $\delta > 0$ and $K \geq 1$, we denote with $\mathcal{H}_{\delta,K}^{\mathsf{T}}$ the family of asymmetric hash functions created this way: that is, $\mathcal{H}_{\delta,K}^{\mathsf{T}}$ consists of tuples $(g_{\delta,K}^{t,s}, g_{\delta,K}^{t,*})$ where the entries of t are independently and identically distributed in $[0, \delta)^d$ and $\Phi^s(P)$ is uniformly chosen at random from $\mathcal{P}_K$.

5.2 Analysis

We have the following theorem which generalizes Theorem 7. Using the parameter K we get a tradeoff between approximation factor and query time.

▶ **Theorem 10.** *Let $P, Q \in \mathcal{S}$ be two curves with m_1 and m_2 points, respectively, and let $M = \max\{m_1, m_2\}$. Let $K \geq 1$ be a given integer and let $\delta = 4dr \cdot \lceil \frac{M}{K} \rceil$ and $c = 4d^{\frac{3}{2}} \cdot \lceil \frac{M}{K} \rceil$. It holds that*

(i) *if* $d_F(P,Q) < r$, *then* $Pr_{\mathcal{H}^\tau_{\delta,K}}\left(g^{t,s}_{\delta,K}(P) = g^{t,*}_{\delta,K}(Q)\right) \geq \left(\frac{1}{2}\right)^K \cdot \binom{M+K-1}{K-1}^{-1}$;

(ii) *if* $d_F(P,Q) > cr$, *then* $Pr_{\mathcal{H}^\tau_{\delta,K}}\left(g^{t,s}_{\delta,K}(P) = g^{t,*}_{\delta,K}(Q)\right) = 0$.

Proof. We first prove (i). Let T be an optimal traversal of P and Q. We say two partitions $\Phi^s(P)$ and $\Phi^r(Q)$ are *consistent* with respect to T if and only if $(s_i, r_i) \in T$ for all $1 \leq i \leq K-1$. Let E denote the event that the partition $\Phi^s(P)$ used in the hash functions is consistent with $\Phi^*(Q)$ with respect to T. By construction this happens for at least one of the partitions in $\mathcal{P}_K$. Therefore, $Pr(E) \geq \frac{1}{|\mathcal{P}_K|}$. Now, let E_i be the event that $h^{t_i}_\delta\left(\widehat{P}_i\right) = h^{t_i}_\delta\left(\widehat{Q}_i\right)$. By Lemma 5 we have that

$$Pr(E_i \mid E) \geq 1 - \left(2dm' \cdot \frac{d_F\left(\widehat{P}_i, \widehat{Q}_i\right)}{\delta}\right) \geq 1 - \left(2d\left\lceil\frac{M}{K}\right\rceil \cdot \frac{d_F(P,Q)}{\delta}\right) \geq \frac{1}{2}.$$

Note that we can assume $m' \leq \lceil \frac{M}{K} \rceil$ in the above inequality, since m' is the length of the shorter of the two subsequences in the lemma. By construction, the length of $\widehat{Q}_i$ will be at most $\lceil \frac{M}{K} \rceil$.

Since the values t_i are chosen pairwise independent, we have

$$Pr_{\mathcal{H}^\tau_{\delta,K}}\left(g^{t,s}_{\delta,K}(P) = g^{t,*}_{\delta,K}(Q)\right) \geq \left(\prod_{1\leq i\leq K} Pr(E_i \mid E)\right) \cdot Pr(E) \geq \left(\frac{1}{2}\right)^K \cdot \frac{1}{|\mathcal{P}_K|}.$$

Using $|\mathcal{P}_K| \leq \binom{M+K-1}{K-1}$, the first part of the claim follows.

As for the second part of the claim, we can use Lemma 6 applied to the subsequences. If there exists a partition of P, and there exist $t = t_1, \ldots, t_K$, such that for all $0 \leq i \leq K$ $h^{t_i}_\delta\left(\widehat{P}_i\right) = h^{t_i}_\delta\left(\widehat{Q}_i\right)$, then it holds by Lemma 6 that $d_F\left(\widehat{P}_i, \widehat{Q}_i\right) \leq \sqrt{d} \cdot \delta$. In this case, we can combine the traversals of the subsequences to a traversal of the entire curves. This combined traversal has the same cost, therefore it follows that $d_F(P,Q) \leq \sqrt{d} \cdot \delta$. Consequently, if $d_F(P,Q) > cr = 4d^{\frac{3}{2}} Mr/K = \sqrt{d} \cdot \delta$, then it cannot happen that $g^{t,s}_\delta(P) = g^{t,*}_\delta(Q)$ for any combination of $t = t_1, \ldots, t_K$ and s. ◀

▶ **Corollary 11.** *Let* $P, Q \in \mathcal{S}$ *be two curves with* m_1 *and* m_2 *points, respectively, and let* $M = \max\{m_1, m_2\}$. *Let* $K \geq 1$ *be a given integer and let* $\delta = 4dr \cdot \lceil \frac{M}{K} \rceil$ *and* $c = 4d^{\frac{3}{2}} \cdot \lceil \frac{M}{K} \rceil$. *It holds that*

(i) *if* $d_F(P,Q) < r$, *then* $Pr_{\mathcal{H}^\tau_{\delta,K}}\left(g^{t,s}_{\delta,K}(P) = g^{t,*}_{\delta,K}(Q)\right) > \left(\frac{1}{4}\right)^K \cdot \left(\frac{1}{M}\right)^{K-1}$;

(ii) *if* $d_F(P,Q) > cr$, *then* $Pr_{\mathcal{H}^\tau_{\delta,K}}\left(g^{t,s}_{\delta,K}(P) = g^{t,*}_{\delta,K}(Q)\right) = 0$.

6 Handling constrained alignments

We now focus on LSH for discrete Fréchet distance with constraints on the alignment. We first target the w-anchored distance in Section 6.1, and then the w-speed distance in Section 6.2. As in the previous sections, the schemes are asymmetric and consist of a partitioning of the curve into subsequences and on the application of the basic LSH scheme to each subsequence. However, the partitions are different since they leverage on random processes on both input and query curves, consecutive subsequences do not overlap, and the constraints are exploited. We let $\ell \geq 1$ denote an arbitrary given integer that allows to trade-off the collision probability of near curves with a bi-criteria approximation on the distance and on the anchored alignment.

6.1 LSH for anchored distances

Consider an input curve $P = p_1, \ldots, p_m$ and let $r_P = r_{P,1}, r_{P,2}, \ldots r_{P,m}$ and $t = t_1, t_2, \ldots, t_m$ denote sequences of independent and identically distributed random variables in $[1, w/2]$ and $[0, \delta)^d$ respectively, where δ is a suitable parameter defined later. The partition of P consists of a fixed partitioning into subsequences of length ℓ, followed by a random perturbation of subsequence lengths. Specifically, the following three operations are performed:

- **(i)** Partition P into subsequences $\widehat{P'}_1, \ldots, \widehat{P'}_{K'}$ with $K' = \lceil m/\ell \rceil$ of size ℓ, with the possible exception of the last subsequence. Let $s' \in [m]^{K'+1}$ be the vector denoting the final indexes of each subsequence, that is $\widehat{P}_i = p_{(s'_{i-1}+1)}, \ldots, p_{s'_i}$: we have $s'_0 = 0$, $s'_{K'} = m$ and $s'_i = i\ell$ for each $1 \leq i < K'$.
- **(ii)** Random perturb the final index of each subsequence with the random vector r_P: for each $1 \leq i < K'$, set $s_i = \min\{s_i + r_{p,2}, m\}$.
- **(iii)** Clean the partition by removing overlaps among subsequences: for each $1 \leq i < K'$ and starting from $i = 1$, remove each subsequence where $s'_i \leq s'_j$ for some $j < i$. We let $\Phi^{r_P}(P) = \left(\widehat{P}_1, \ldots, \widehat{P}_K\right)$ denote the resulting partition of P with $K \leq \lceil m/\ell \rceil$ and let $s_P \in [m]^{K+1}$ be the resulting vector denoting the final indexes of each subsequence (note that each subsequence has now length at most $\ell + w$).

Once curve P has been partitioned into K subsequences, we apply the basic LSH in Section 3 to each subsequence using the random shifts given by sequence t. Specifically, we snap the i-th subsequence $\widehat{P}_i$ on a grid of side δ shifted by the random value t_i and remove consecutive duplicates within each subsequence; the remaining values denote the hash value of $\widehat{P}_i$ and we denote them with $h_\delta^{t_i}\left(\widehat{P}_i\right)$. The final hash value $g_{w,\delta,\ell}^{t,r_P}(P)$ of curve P is the array containing the hash of each subsequence, specifically:

$$g_{w,\delta,\ell}^{t,r_P}(P) = \left(h_\delta^{t_1}\left(\widehat{P}_1\right), h_\delta^{t_2}\left(\widehat{P}_2\right), \ldots, h_\delta^{t_K}\left(\widehat{P}_K\right)\right).$$

We observe that the final hash value is not a curve as in previous sections, but an array of curves. Equality between two curves then holds only if the two hash values have the same length and coincide in each position (i.e., the hash values $((a,b),(c))$ and $((a),(b,c))$ do not collide, but they collide if their are considered as a single curve (a,b,c)). This enforces the alignment constraint.

The hash process of a query curve Q is the same: however, a different random sequence r_Q is used to partition the curve, while the same sequence t of random shifts is kept. Due to the different random bits in r_Q the proposed LSH scheme is asymmetric. We let $\mathcal{H}^{\mathtt{A}}_{w,\delta,\ell}$ denote the hash family consisting of all possible pairs of hash functions $\left(g_{w,\delta,\ell}^{t,r_P}, g_{w,\delta,\ell}^{t,r_Q}\right)$.

The next theorem shows that the scheme has a bi-criteria approximation: In addition to the distance approximation c, the scheme has also an approximation on the alignment. As an example, we observe that two curves with a w-anchored distance larger than cr can still collide if they have a $w + 2(\ell - 1)$-anchored distance lower than cr.

▶ **Theorem 12.** *Let $P, Q \in \mathcal{S}$ be two curves with m_1 and m_2 points, respectively and let $m = \min\{m_1, m_2\}$. Let $\ell \geq 1$ be an arbitrary integer, $\delta = 4dr\ell$, and $c = 4d^{\frac{3}{2}}\ell$. Then, it holds that:*

- **(i)** *if $d_{w,aF}(P,Q) < r$, then $Pr_{\mathcal{H}^{\mathtt{A}}_{w,\delta,\ell}}\left(g_{w,\delta,\ell}^{t,r_P}(P) = g_{w,\delta,\ell}^{t,r_Q}(Q)\right) > \left(1/\sqrt{2}w\right)^{2m/\ell}$;*
- **(ii)** *if $d_{(w+2(\ell-1)),aF}(P,Q) > cr$, then $Pr_{\mathcal{H}^{\mathtt{A}}_{w,\delta,\ell}}\left(g_{w,\delta,\ell}^{t,r_P}(P) = g_{w,\delta,\ell}^{t,r_Q}(Q)\right) = 0$.*

6.2 LSH for bounded-speed distances

Consider an input curve $P = p_1, \ldots, p_m$ and let $r_P = r_{P,1}, r_{P,2}, \ldots, r_{P,m}$ and $t = t_1, t_2, \ldots t_m$ denote sequences of independent and identically distributed random variables in $[1, w\ell]$ and $[0, \delta)^d$ respectively. We random partition curve P into non overlapping subsequences of length given by the random sequence r_p. Specifically, let $\Phi^s(P) = \left(\widehat{P}_1, \ldots, \widehat{P}_K\right)$ denote a partition of P with $m/(w\ell) \leq K \leq m$ and let $s \in [m]^{K+1}$ be the vector denoting the initial and final indexes of a subsequence, that is $\widehat{P}_i = p_{s_{i-1}+1}, \ldots, p_{s_i}$. Then, s satisfies the following conditions : (i) $s_0 = 0$ and $s_K = m$, (ii) for $1 \leq i \leq K$, $s_i - s_{i-1} = r_{p,1}$, which implies that $s_i = \sum_{j=1}^{i} r_{p,1}$. Once we have partitioned curve P into K subsequences, we continue as in the w-anchored LSH by applying the basic LSH to each subsequence using the random shifts given by sequence t. For a query curve Q, the hash process is the same, but a different random sequence r_Q is used to partition the curve. We let $\mathcal{H}^{\mathsf{s}}_{w,\delta,\ell}$ denote the hash family consisting of all possible pairs of hash functions $\left(g^{t,r_P}_{w,\delta,\ell}, g^{t,r_Q}_{w,\delta,\ell}\right)$. The next theorem shows that the scheme has a bi-criteria approximation (note that the alignment approximation in point (ii) differs from the one for the anchored distance).

▶ **Theorem 13.** *Let $P, Q \in \mathcal{S}$ two curves with m_1 and m_2 points, respectively and let $m = \min\{m_1, m_2\}$. Let $\ell \geq 1$ be an arbitrary integer, $\delta = 4dr\ell$, and $c = 4d^{\frac{3}{2}}\ell$. Then, it holds that:*

(i) *if $d_{w,sF}(P, Q) < r$, then $Pr_{\mathcal{H}^s_{w,\delta,\ell}}\left(g^{t,r_P}_{w,\delta,\ell}(P) = g^{t,r_Q}_{w,\delta,\ell}(Q)\right) > \left(1/\sqrt{2}w\ell\right)^{2m/\ell}$;*

(ii) *if $d_{w\ell,sF}(P, Q) > cr$, then $Pr_{\mathcal{H}^s_{w,\delta,\ell}}\left(g^{t,r_P}_{w,\delta,\ell}(P) = g^{t,r_Q}_{w,\delta,\ell}(Q)\right) = 0$.*

7 Extensions to dynamic time warping

All our schemes can be applied to DTW without any algorithmic change, and in this section we analyze some of them. We first investigate in Section 7.1 the basic scheme in Section 3.1 for this distance. Then, we provide a few insights on DTW with constrained alignments in Section 7.2. We do not analyze the techniques proposed in Sections 4 and 5 since they have the same linear approximation of the basic LSH, and – in contrast to our previous results for the Fréchet distance – do not provide a sublinear approximation for DTW.

7.1 Analysis of the basic LSH

▶ **Lemma 14.** *Let $P, Q \in \Delta^d$ be two curves with m_1 and m_2 points, respectively. For any $\delta \geq 0$, it holds that*

$$Pr_{\mathcal{H}^t_\delta}\left(h^t_\delta(P) = h^t_\delta(Q)\right) \geq 1 - \left(d \cdot \frac{d_{DTW}(P, Q)}{\delta}\right).$$

▶ **Lemma 15.** *Let $P, Q \in \Delta^d$ be two curves with m_1 and m_2 points, respectively, and let $M = \max\{m_1, m_2\}$ and $\delta \geq 0$. If there exists a value of $t \in [0, \delta)^d$ such that $h^t_\delta(P) = h^t_\delta(Q)$, then it holds that $d_{DTW}(P, Q) \leq 2M\sqrt{d} \cdot \delta$.*

Lemma 14 above can be proven by a similar analysis as in the proof of Lemma 5: let T be an optimal traversal of P and Q with respect to their DTW distance; let $\ell = |T|$ and denote with d_k the distance $\|p_{i_k} - q_{j_k}\|$ for $1 \leq k \leq \ell$; we have that $d_{\mathrm{DTW}}(P, Q) = \sum_{1 \leq k \leq \ell} d_k$; now, we can use a union bound over all pairs in the traversal, instead of components. The proof of Lemma 15 is somewhat technical since DTW does not satisfy the triangle inequality. The following theorem can be obtained by plugging in the bounds of Lemmas 14 and 15.

▶ **Theorem 16.** *Let $P, Q \in \Delta^d$ be two curves with m_1 and m_2 points, respectively, and let $M = \max\{m_1, m_2\}$, $\delta = 2dr$ and let $c = 4d^{\frac{3}{2}}M$.*
(i) *if $d_{DTW}(P,Q) < r$, then $Pr_{\mathcal{H}_\delta^t}(h_\delta^t(P) = h_\delta^t(Q)) > \frac{1}{2}$;*
(ii) *if $d_{DTW}(P,Q) > cr$, then $Pr_{\mathcal{H}_\delta^t}(h_\delta^t(P) = h_\delta^t(Q)) = 0$.*

7.2 Handling constrained alignments

The schemes in Section 6 for w-anchored/speed traversals automatically apply to DTW distance, with the same collision probabilities stated in Theorems 12 and 13. However, the approximation factor is $4d^{3/2}(m_1 + m_2)$, where m_1 and m_2 are curve lengths. The claim follows by mimicking the proofs for the Fréchet distance and use the bounds in Theorem 16.

8 Conclusion

To the best of our knowledge, this is the first paper providing LSH schemes for curves. When applied to the near neighbor problem, our techniques improve the state of the art for the discrete Fréchet distance [17] under different settings, and provide the first data structure with theoretical guarantees for DTW. The methods presented are simple enough that they may be practical. We do not know if our bounds are tight. It would be interesting to know if lower bounds can be obtained for the studied problem and/or to improve the upper bounds. All of the presented LSH schemes exhibit the property that no collisions happen between far points (i.e., $\alpha_2 = 0$). An open question is to understand if it is possible to slightly increase this collision probability (say $\alpha_2 = 1/n$) to get a better approximation factor. Another interesting direction would be to reduce space by exploiting the independence in the approach described in Section 4.1, or by using a multiprobe approach [24]. Finally, we remark that our results only partially extend to DTW. As such, it is still open to get a sublinear approximation for DTW. We hope that our work inspires further work in one of these directions.

Acknowledgments. The authors would like to thank Rasmus Pagh and the anonymous reviewers for useful comments. This research was initiated at the Dagstuhl Seminar 16101 "Data Structures and Advanced Models of Computation on Big Data, 2016".

References

1 S. Arya, D. Mount, A. Vigneron, and J. Xia. Space-time tradeoffs for proximity searching in doubling spaces. In *Proc. 16th European Symp. Algorithms (ESA)*, pages 112–123, 2008.
2 A. Backurs and A. Sidiropoulos. Constant-distortion embeddings of Hausdorff metrics into constant-dimensional l_p spaces. In *Proc. 19th Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, volume 60, pages 1:1–1:15, 2016.
3 Y. Bartal, L. A. Gottlieb, and O. Neiman. On the impossibility of dimension reduction for doubling subsets of ℓ_p. In *Proc. 13th Symp. on Computational Geometry (SOCG)*, pages 60:60–60:66, 2014.
4 K. Bringmann. Why Walking the Dog Takes Time: Fréchet Distance Has No Strongly Subquadratic Algorithms Unless SETH Fails. In *Proc. 55th Symp. on Foundations of Computer Science (FOCS)*, pages 661–670, 2014.
5 J. C. Brown and P. J. O. Miller. Automatic classification of killer whale vocalizations using dynamic time warping. *J. of the Acoustical Society of America*, 122(2):1201–1207, 2007.
6 J. Campbell, J. Tremblay, and C. Verbrugge. Clustering player paths. In *Proc. 10th Int'l Conf. on the Foundations of Digital Games (FDG)*, 2015.

7 M. de Berg, A. F. Cook, and J. Gudmundsson. Fast Fréchet queries. *Comput. Geom.*, 46(6):747–755, 2013.

8 A. Driemel and S. Har-Peled. Jaywalking your dog: Computing the Fréchet distance with shortcuts. *SIAM J. Computing*, 42(5):1830–1866, 2013.

9 A. Driemel, A. Krivošija, and C. Sohler. Clustering time series under the Fréchet distance. In *Proc. 27th Symp. on Discrete Algorithms (SODA)*, pages 766–785, 2016.

10 A. Driemel and F. Silvesstri. Locality-sensitive hashing of curves. Arxiv:1703.04040, 2017.

11 G. Forestier, F. Lalys, L. Riffaud, B. Trelhu, and P. Jannin. Classification of surgical processes using dynamic time warping. *J. Biomedical Informatics*, 45(2):255–264, 2012.

12 J. Gudmundsson and N. Valladares. A GPU approach to subtrajectory clustering using the Fréchet distance. *IEEE Trans. on Parallel and Distributed Systems*, 26(4):924–937, 2015.

13 Anupam Gupta, Robert Krauthgamer, and James R Lee. Bounded geometries, fractals, and low-distortion embeddings. In *Proc. 44th Symp. Found. Comp. Science (FOCS)*, pages 534–543, 2003.

14 S. Har-Peled, P. Indyk, and R. Motwani. Approximate nearest neighbor: Towards removing the curse of dimensionality. *Theory of Computing*, 8(1):321–350, 2012.

15 B. Huang and W. Kinsner. ECG frame classification using dynamic time warping. In *Proc. Canadian Conf. on Electrical and Computer Engineering*, volume 2, pages 1105–1110, 2002.

16 P. Indyk. On approximate nearest neighbors in non-euclidean spaces. In *Proc. 39th Symp. on Foundations of Computer Science*, pages 148–155, 1998.

17 P. Indyk. Approximate nearest neighbor algorithms for Fréchet distance via product metrics. In *Proc. 18th Symp. on Computational Geometry (SOCG)*, pages 102–106, 2002.

18 P. Indyk and J. Matoušek. Low-distortion embeddings of finite metric spaces. In *Handbook of Discrete and Computational Geometry*, pages 177–196. CRC Press, 2004.

19 P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proc. 30th Symp. Theory of Computing (STOC)*, pages 604–613, 1998.

20 R. J. Kenefic. Track clustering using Fréchet distance and minimum description length. *J. of Aerospace Information Systems*, 11(8):512–524, 2014.

21 E. Keogh and C. A. Ratanamahatana. Exact indexing of dynamic time warping. *Knowledge and Information Systems*, 7(3):358–386, 2005.

22 Z. M. Kovacs-Vajna. A fingerprint verification system based on triangular matching and dynamic time warping. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1266–1276, 2000.

23 B. Legrand, C. S. Chang, S. H. Ong, S. Y. Neo, and N. Palanisamy. Chromosome classification using dynamic time warping. *Pattern Recognition Letters*, 29(3):215–222, 2008.

24 Q. Lv, W. Josephson, Z. Wang, M. Charikar, and K. Li. Multi-probe lsh: Efficient indexing for high-dimensional similarity search. In *Proc. 33rd Int'l Conf. on Very Large Data Bases*, VLDB'07, pages 950–961. VLDB Endowment, 2007.

25 J. Matoušek. Embedding finite metric spaces into euclidean spaces. In *Lectures on Discrete Geometry*, chapter 15. Springer, 2002.

26 T. Rakthanmanon, B. Campana, A. Mueen, G. Batista, B. Westover, Q. Zhu, J. Zakaria, and E. Keogh. Searching and mining trillions of time series subsequences under dynamic time warping. In *Proc. 18th Conf. Knowl. Disc. and Data Mining*, pages 262–270, 2012.

27 C. A. Ratanamahatana and E. Keogh. Three myths about dynamic time warping data mining. In *Proc. SIAM Conf. on Data Mining (SDM)*, pages 506–510, 2005.

28 G. Shakhnarovich, T. Darrell, and P. Indyk, editors. *Nearest-Neighbor Methods in Learning and Vision: Theory and Practice*. MIT Press, 2006.

29 A. Shrivastava and P. Li. Asymmetric LSH (ALSH) for sublinear time maximum inner product search (MIPS). In *Proc. 27th Conf. on Neural Information Processing Systems (NIPS)*, pages 2321–2329, 2014.

30 G. K. D. Vries. *Kernel methods for vessel trajectories.* PhD thesis, Univ. Amsterdam, 2012.
31 H. Zhu, J. Luo, H. Yin, X. Zhou, J. Z. Huang, and F. B. Zhan. Mining trajectory corridors using Fréchet distance and meshing grids. In *Proc. 14th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, pages 228–237, 2010.

Shallow Packings, Semialgebraic Set Systems, Macbeath Regions, and Polynomial Partitioning*

Kunal Dutta[1], Arijit Ghosh[2], Bruno Jartoux[3], and Nabil H. Mustafa[4]

1 **DataShape, INRIA Sophia Antipolis – Méditerranée, Sophia Antipolis, France**
kunal.dutta@inria.fr
2 **ACM Unit, Indian Statistical Institute, Kolkata, India**
agosh@mpi-inf.mpg.de
3 **Université Paris-Est, Laboratoire d'Informatique Gaspard-Monge, ESIEE Paris, Paris, France**
bruno.jartoux@esiee.fr
4 **Université Paris-Est, Laboratoire d'Informatique Gaspard-Monge, ESIEE Paris, Paris, France**
mustafan@esiee.fr

Abstract

The packing lemma of Haussler states that given a set system $(X, \mathcal{R})$ with bounded VC dimension, if every pair of sets in $\mathcal{R}$ have large symmetric difference, then $\mathcal{R}$ cannot contain too many sets. Recently it was generalized to the shallow packing lemma, applying to set systems as a function of their shallow-cell complexity. In this paper we present several new results and applications related to packings:

1. an optimal lower bound for shallow packings,
2. improved bounds on Mnets, providing a combinatorial analogue to Macbeath regions in convex geometry,
3. we observe that Mnets provide a general, more powerful framework from which the state-of-the-art unweighted ϵ-net results follow immediately, and
4. simplifying and generalizing one of the main technical tools in Fox *et al.* (*J. of the EMS*, to appear).

1998 ACM Subject Classification F.2.2 Nonnumerical algorithms and problems, G.2.1 Combinatorics

Keywords and phrases Epsilon-nets, Haussler's packing lemma, Mnets, shallow-cell complexity, shallow packing lemma

Digital Object Identifier 10.4230/LIPIcs.SoCG.2017.38

1 Introduction

Given a set system $(X, \mathcal{R})$ consisting of base elements X together with a set $\mathcal{R}$ of subsets of X, a classical and influential way to capture its 'complexity' has been through the concept of *VC dimension*. First define the *projection* of $\mathcal{R}$ onto any $Y \subseteq X$ to be the system

$$\mathcal{R}|_Y = \big\{Y \cap R \colon R \in \mathcal{R}\big\}.$$

* Bruno Jartoux and Nabil H. Mustafa's research in this paper is supported by the grant ANR SAGA (JCJC-14-CE25-0016-01). Kunal Dutta and Arijit Ghosh are supported by the European Research Council under the Advanced Grant 339025 GUDHI (Algorithmic Foundations of Geometric Understanding in Higher Dimensions) and the Ramanujan Fellowship (No. SB/S2/RJN-064/2015) respectively.

Also, for any positive integer r, define $\mathcal{R}|_{Y,\leq r}$ to be the sets in $\mathcal{R}|_Y$ of size at most r. The VC dimension of a set system $(X, \mathcal{R})$, henceforth denoted by VC-DIM$(\mathcal{R})$, is the size of any largest subset $Y \subseteq X$ for which $|\mathcal{R}|_Y| = 2^{|Y|}$; such a set Y is said to be *shattered* by $\mathcal{R}$.

The importance of VC dimension derives from the fact that it is bounded for most natural geometric set systems, where X is a set of geometric objects in $\mathbb{R}^d$ and $\mathcal{R}$ is defined by geometric constraints. For example, consider the case when X is a set of points in $\mathbb{R}^d$ and the sets in $\mathcal{R}$ are defined by containment by half-spaces, i.e., $\mathcal{R} = \big\{H \cap X \colon H \text{ is a half-space in } \mathbb{R}^d\big\}$. It is not hard to see that the VC dimension of this set system is $d+1$. This forms the basis for bounding the VC dimension of many geometric set systems via *linearization* [21].

Set systems derived from geometric configurations can be categorized into two types. When X is a set of points and sets in $\mathcal{R}$ are defined by containment by members of a family of geometric objects $\mathcal{O}$, we say that $(X, \mathcal{R})$ is a *primal set system induced by* $\mathcal{O}$. The second type is when the base set X is a finite subset of $\mathcal{O}$, and $\mathcal{R}$ is defined to be $\mathcal{R} = \big\{\mathcal{R}_p \colon p \in \mathbb{R}^d\big\}$, where $\mathcal{R}_p = \big\{O \in X \colon\ p \in O\big\}$ is the set of objects containing p. Then we say that $(X, \mathcal{R})$ is the *dual set system induced by* $\mathcal{O}$. For most natural families of geometric objects, these primal and dual set systems can be shown to have bounded VC dimension [21, Section 10.3].

1.1 Shallow-cell Complexity of Set Systems

It turns out that for nearly all results on set systems with bounded VC dimension, the key technical property required is a consequence of bounded VC dimension, the primal shatter lemma of Sauer and Shelah [29, 30].

▶ **Theorem A** (Primal shatter lemma). *Let $(X, \mathcal{R})$ be a set system with* VC-DIM$(\mathcal{R}) = d$. *Then for any $Y \subseteq X$, we have $|\mathcal{R}|_Y| = O\big(|Y|^d\big)$.*

While most set systems derived from geometry have bounded VC dimension and thus satisfy the primal shatter lemma, in fact they often satisfy a finer property—not only is the size of $\mathcal{R}|_Y$ polynomially bounded, but also the number of sets in $\mathcal{R}|_Y$ of *any fixed size* r is bounded by an even smaller function. For example, let X be a set of n points in $\mathbb{R}^2$, and $\mathcal{R}$ the primal set system induced by disks. Then it is well-known that for any set $Y \subseteq X$, the number of sets in $\mathcal{R}|_Y$ of size at most r is $|\mathcal{R}|_{Y,\leq r}| = O\big(|Y| \cdot r^2\big)$. For small values of r, this contrasts sharply with the total size of $\mathcal{R}|_Y$, which can be $\Theta\big(|Y|^3\big)$.

This has motivated a finer classification of set systems. In [11, 9], a set system $(X, \mathcal{R})$ was said to have the (d, d_1) *Clarkson–Shor property* if for any $Y \subseteq X$, the number of sets in $\mathcal{R}|_Y$ of size r was $O\big(|Y|^{d_1} r^{d-d_1}\big)$. More generally, given $(X, \mathcal{R})$, define $f_{\mathcal{R}}(m, r)$ as the maximum number of sets of cardinality at most r in the projection on any set of m points:

$$\forall m, r \in \mathbb{N}, \quad f_{\mathcal{R}}(m, r) = \max_{Y \subseteq X, |Y|=m} |\mathcal{R}|_{Y,\leq r}|.$$

We now define the key property used in this paper.

▶ **Definition 1.** The *shallow-cell complexity*, denoted by $\varphi_{\mathcal{R}}(\cdot, \cdot)$[1], of a set system $(X, \mathcal{R})$ is defined as $\varphi_{\mathcal{R}}(m, r) = \frac{f_{\mathcal{R}}(m,r)}{m}$.

In earlier literature, sometimes this was defined simply as $f_{\mathcal{R}}(m, r)$; however, as usually there is at least a linear factor of m in the function $f_{\mathcal{R}}(m, r)$, we prefer to normalize by m,

[1] The subscript will be dropped when it is clear from the context.

Table 1 Some geometric set systems.

Objects	P/D	$\varphi(m)$	VCdim
Intervals	P/D	$O(1)$	2
Lines in $\mathbb{R}^2$	P/D	$O(m)$	2
Pseudo-disk	P	$O(1)$	3
Pseudo-disk	D	$O(1)$	$O(1)$
Half-spaces	P/D	$O\big(m^{\lfloor d/2\rfloor-1}\big)$	$d+1$
Balls	P/D	$O\big(m^{\lceil d/2\rceil-1}\big)$	$d+1$
Triangles	D	$O(m)$	7
Convex sets	P	$O\big(2^m/m\big)$	∞

which will make later results simpler to state. Often the dependency on r is less important: we say that $(X,\mathcal{R})$ has shallow-cell complexity $\varphi_{\mathcal{R}}(\cdot)$ if $f_{\mathcal{R}}(m,r) = O\big(m\cdot\varphi_{\mathcal{R}}(m)\cdot r^{c_{\mathcal{R}}}\big)$, where $c_{\mathcal{R}}\geq 0$ is a fixed constant independent of m and r.

Note that the shallow-cell complexity of set systems with the (d,d_1) Clarkson–Shor property is $\varphi(m,r) = O\big(m^{d_1-1}r^{d-d_1}\big)$. For a family $\mathcal{O}$ of geometric objects [2], define its *union complexity* $\kappa_{\mathcal{O}}(\cdot)$ by letting $\kappa_{\mathcal{O}}(m)$ be the maximum number of faces of all dimensions in the union of any m of its members. It can be shown that the dual set system $(\mathcal{O},\mathcal{R})$ induced by $\mathcal{O}$ has shallow-cell complexity $\varphi(m) = O\big(\frac{\kappa_{\mathcal{R}}(m)}{m}\big)$.

See Table 1 for the VC dimension and the shallow-cell complexity of many of the commonly studied geometric set systems (**P**rimal and **D**ual).

1.2 Macbeath regions and Mnets

Given a convex object C in $\mathbb{R}^d$ with volume $\mathrm{vol}(C)$, Macbeath's theorem [19] states the existence of a collection of smaller convex regions $\{C_1,\ldots,C_l\}$, each $C_i\subseteq C$ is called a *Macbeath region* of C, and where $l = O\left(\left(\frac{1}{\epsilon}\right)^{1-\frac{2}{d+1}}\right)$, such that

(i) $\mathrm{vol}(C_i) = \Theta(\epsilon\,\mathrm{vol}(C))$ for each i, and
(ii) for any half-space H with $\mathrm{vol}(H\cap C)\geq\epsilon\,\mathrm{vol}(C)$, there exists a j such that $C_j\subseteq H$.

Mnets (or *combinatorial Macbeath regions*), introduced by Mustafa *et al.* [26], are the combinatorial analogue of Macbeath regions for set systems, replacing the Lebesgue measure with the counting measure.

▶ **Definition 2.** Given a set system $(X,\mathcal{R})$ on n elements and a parameter $\epsilon>0$, a collection $\mathcal{M} = \{M_1,\ldots,M_l\}$ of subsets of X is an *ϵ-Mnet* for $\mathcal{R}$ of size l if

(i) $|M_i| = \Theta(\epsilon n)$ for each i, and
(ii) for any $R\in\mathcal{R}$ with $|R|\geq\epsilon n$, there exists an index j such that $M_j\subseteq R$.

Beginning with the breakthrough, and beautiful, result of Haussler and Welzl [16], *epsilon-nets* have been one of the most fundamental structures in combinatorial geometry with many applications in areas such as approximation algorithms, discrete and computational geometry, combinatorial discrepancy theory and learning theory [8, 20, 21, 28].

▶ **Definition 3.** For a given set system $(X,\mathcal{R})$ and a parameter $\epsilon>0$, an (unweighted) ϵ-net for $\mathcal{R}$ is a set $N\subseteq X$ such that for any $R\in\mathcal{R}$, $|R|\geq\epsilon|X| \implies N\cap R\neq\emptyset$.

[2] These objects are usually semialgebraic; see [1] for a discussion of the definition of faces and cells induced by arrangements of geometric objects.

Haussler and Welzl [16] in their paper showed that there exists ϵ-nets of size independent from the size of the ground set X, i.e., the size of the smallest ϵ-net is $O\big(\frac{d}{\epsilon}\log\frac{d}{\epsilon}\big)$, where d is the VC dimension of the set system. Chan *et al.* [6], improving on an earlier result of Varadarajan [31] that stated a slightly weaker result for dual set systems induced by geometric objects, proved the following generalization of the epsilon-net result of Haussler and Welzl. See also [25] for a simpler proof of this theorem.

▶ **Theorem B.** *Let $(X,\mathcal{R})$ be a set system with shallow-cell complexity $\varphi_{\mathcal{R}}(\cdot)$, where $\varphi_{\mathcal{R}}(n) = O(n^d)$ for some constant d. Let $\epsilon > 0$ be a given parameter. Then there exists an ϵ-net for $\mathcal{R}$ of size $O\big(\frac{1}{\epsilon}\log\varphi_{\mathcal{R}}(\frac{1}{\epsilon})\big)$. Furthermore, such an ϵ-net can be computed in deterministic polynomial time.*

Recently Theorem B has been shown to be tight by Kupavskii *et al.* [17]. For a state-of-the-art on ϵ-nets, we refer the reader to [27].

1.3 Packing Lemma for Geometric Set Systems

A set system $(X,\mathcal{R})$ is said to be a *δ-packing* if for all distinct $R,S\in\mathcal{R}$, $|R\Delta S|\geq\delta$, where Δ is the symmetric difference. In 1995 Haussler [15] proved the following key statement.

▶ **Theorem C** (Packing Lemma). *Let $(X,\mathcal{R})$ be a set system with* VC-DIM$(\mathcal{R})\leq d$ *and $|X|=n$. Let δ, $1\leq\delta\leq n$ be such that $(X,\mathcal{R})$ is a δ-packing. Then $|\mathcal{R}| = O\left(\left(\frac{n}{\delta}\right)^d\right)$, where the constant in the asymptotic notation depends on d*[3]*.*

Haussler's seminal proof of Theorem C, later simplified by Chazelle [7], is an elegant application of the probabilistic method, and has since been applied to diverse areas ranging from computational geometry and machine learning to Bayesian inference—see e.g. [15, 20, 18]. It was further shown in [15] that this bound is tight:

▶ **Theorem D** (Optimality of Packing Lemma). *Given any positive integers d, n and $\delta\in\{1,\dots,n\}$, there exists a set system $(X,\mathcal{R})$ such that $|X|=n$,* VC-DIM$(\mathcal{R})\leq d$*, $\mathcal{R}$ is a δ-packing and $|\mathcal{R}| = \Omega\left(\left(\frac{n}{\delta}\right)^d\right)$.*

Recent efforts have been devoted to extending the packing lemma to these finer classifications of set systems. For $k,\delta\in\mathbb{N}^*$, call $(X,\mathcal{R})$ a *k-shallow δ-packing* if $\mathcal{R}$ is a δ-packing and $|S|\leq k$ for all $S\in\mathcal{R}$. After some earlier bounds [26, 11], the following lemma has been recently established in [9, 24].

▶ **Theorem E** (Shallow Packing Lemma). *Let $(X,\mathcal{R})$ be a set system on n elements, and let d_0, d, d_1, k, $\delta>0$ be integers. Assume* VC-DIM$(\mathcal{R})\leq d_0$*. If $(X,\mathcal{R})$ is a k-shallow δ-packing,*

1. $|\mathcal{R}| = O\left(\dfrac{n^{d_1}k^{d-d_1}}{\delta^d}\right)$ *if $\mathcal{R}$ satisfies the (d,d_1) Clarkson–Shor property.*
2. $|\mathcal{R}| \leq \dfrac{24d_0n}{\delta}\cdot\varphi\left(\dfrac{4d_0n}{\delta},\dfrac{12d_0k}{\delta}\right)$ *if $\mathcal{R}$ has shallow-cell complexity $\varphi(\cdot,\cdot)$.*

The constant in the asymptotic notation of 1 depends on d_0, d and d_1.

▶ Remark. Note that 2 implies 1 in Theorem E.

[3] The same bound also holds with bounded *primal shatter dimension* replacing VC dimension, see e.g. Chapter 5.3 [20].

2 Our Contributions

We present three main results: a tight lower bound for shallow packings, a construction of Mnets using the shallow packing lemma, and a generalization of the shallow packing lemma to l-wise packings. A key ingredient which makes the Mnets bound possible is merging the polynomial partitioning technique with the shallow packing lemma.

2.1 Optimality of Shallow Packings (Proof in Section 3)

While Haussler [15] gave a matching lower bound to his packing lemma, the optimality of the *shallow* packing lemma was an open question in previous work [11, 26, 9, 24]. In earlier work [9], a matching lower bound was presented for one particular case, when $\varphi(m) = m$. We show that the shallow packing lemma is tight up to a constant factor for the most common case of shallow-cell complexity, when $\varphi(m, r) = O\big(m^{d_1-1} r^{d-d_1}\big)$ for some integers d, d_1.

▶ **Theorem 4** (Optimality of Shallow Packings). *For any positive integers $d \geq d_1$ and for any positive integer n, there exists a set system $(X, \mathcal{R})$ on n elements such that*

1. *$(X, \mathcal{R})$ has shallow-cell complexity $\varphi(m, r) = O(m^{d_1-1} r^{d-d_1})$, and*
2. *for any δ and $k \geq 4d\delta$, $(X, \mathcal{R})$ has a k-shallow δ-packing of size $\Omega\left(\frac{n^{d_1} k^{d-d_1}}{\delta^d}\right)$.*

Our proof is via an explicit construction of a semialgebraic set system.

2.2 Mnets for Semialgebraic Set Systems (Proof in Section 4)

Semialgebraic sets are subsets of $\mathbb{R}^d$ obtained by taking Boolean operations such as unions, intersections, and complements of sets of the form $\{x \in \mathbb{R}^d : g(x) \geq 0\}$, where g is a d-variate polynomial in $\mathbb{R}[x_1, \ldots, x_d]$. Denote by $\Gamma_{d,\Delta,s}$ the family of all semialgebraic sets in $\mathbb{R}^d$ obtained by taking Boolean operations on at most s polynomial inequalities, each of degree at most Δ. In this paper d, Δ, s are all regarded as constants and therefore the sets in $\Gamma_{d,\Delta,s}$ have constant description complexity[4]. For a set X of points in $\mathbb{R}^d$ and a set system $\mathcal{R}$ on X, we say that $(X, \mathcal{R})$ is a *semialgebraic set system* generated by $\Gamma_{d,\Delta,s}$ if for all $S \in \mathcal{R}$ there exists a $\gamma \in \Gamma_{d,\Delta,s}$ such that $S = X \cap \gamma$.

▶ **Theorem 5** (Mnets). *Let d, d_0, Δ and s be integers and $(X, \mathcal{R})$ a semialgebraic set system generated by $\Gamma_{d,\Delta,s}$ with $|X| = n$ and $\text{VC-DIM}(\mathcal{R}) \leq d_0$. If $\mathcal{R}$ has shallow-cell complexity $\varphi(\cdot,\cdot)$, with $\varphi(\cdot,\cdot)$ a non-decreasing function in the first argument, then $(X, \mathcal{R})$ has an ϵ-Mnet of size*

$$l = O\left(\frac{d_0}{\epsilon} \cdot \varphi\left(\frac{8d_0}{\epsilon}, 48d_0\right)\right).$$

In particular, if $(X, \mathcal{R})$ has shallow-cell complexity $\varphi(\cdot)$, then $l = O\left(\frac{1}{\epsilon} \cdot \varphi\left(\frac{8d_0}{\epsilon}\right)\right)$. Constants depend on d, Δ, and s; the second one also depends on d_0.

Most of the time this bound simplifies to $O\left(\frac{1}{\epsilon} \cdot \varphi\left(\frac{1}{\epsilon}\right)\right)$ The proof of Theorem 5 uses the shallow packing lemma (Theorem E), as well as the *polynomial partitioning* method of Guth and Katz [14], specifically a multilevel refinement due to Matoušek and Patáková [23].

First we point out that Theorem 5 immediately implies the best known bounds on unweighted ϵ-nets, though with the additional restriction that the set system is semialgebraic.

[4] For a detailed introduction to this topic, see [5].

Table 2 Many known results follow from Theorem 5 via their shallow-cell complexity. Polylogarithmic improvements are in bold.

Set System	Primal/Dual	Size of ϵ-Mnets
Objects with union complexity $\kappa(\cdot)$	D	$O(\kappa(\frac{1}{\epsilon}))$
α-fat triangles	D	$O(\frac{1}{\epsilon}\log^*\frac{1}{\epsilon})$
Locally γ-fat objects	D	$\frac{1}{\epsilon}\cdot 2^{O(\log^*\frac{1}{\epsilon})}$
Triangles of approximately same size	D	$O(\frac{1}{\epsilon})$
α-fat triangles	P	$O(\frac{1}{\epsilon}\log^2\frac{1}{\epsilon})$
Rectangles in $\mathbb{R}^2$	P	$O(\frac{1}{\epsilon}\log\frac{1}{\epsilon})$
Lines in $\mathbb{R}^2$	**P**	$O(\frac{1}{\epsilon^2})$
Strips in $\mathbb{R}^2$	**P**	$O(\frac{1}{\epsilon^2})$
Cones in $\mathbb{R}^2$	**P**	$O(\frac{1}{\epsilon^2})$
Pseudo-disks in $\mathbb{R}^2$	P/D	$O(\frac{1}{\epsilon})$
Half-spaces in $\mathbb{R}^d$	P/D	$O(\frac{1}{\epsilon^{\lfloor d/2\rfloor}})$

▶ **Corollary 6.** *Set systems with ϵ-Mnets of size M have ϵ-nets of size $O\big(\frac{1}{\epsilon}\log(\epsilon M)\big)$. In particular, a set system $(X,\mathcal{R})$ with $\text{VC-DIM}(\mathcal{R})\le d_0$ has ϵ-nets of size*
1. *$O\big(\frac{1}{\epsilon}\log\varphi\big(\frac{8d_0}{\epsilon}, 48d_0\big)\big)$ if it has shallow-cell complexity $\varphi(\cdot,\cdot)$, and*
2. *$O\big(\frac{1}{\epsilon}\log\varphi\big(\frac{8d_0}{\epsilon}\big)\big)$ if it has shallow-cell complexity $\varphi(\cdot)$.*

Proof. Let $\mathcal{M}$ be an ϵ-Mnet for $(X,\mathcal{R})$ whose sets have size at least $C\epsilon n$. Pick each point of X into a random sample R independently with probability $p=\frac{1}{C\epsilon n}\log(\epsilon|\mathcal{M}|)$.

R is disjoint from any fixed $M_i\in\mathcal{M}$ with probability at most $(1-p)^{C\epsilon n}\le e^{-pC\epsilon n}=\frac{1}{\epsilon|\mathcal{M}|}$. Therefore the expected number of sets of $\mathcal{M}$ not hit by R is at most $\frac{1}{\epsilon}$; let S be a set consisting of an arbitrary point from each such set. As $\mathbb{E}[|S|]\le\frac{1}{\epsilon}$, we have that $S\cup R$ is an ϵ-net of expected size $\le\frac{1}{\epsilon}+\frac{1}{C\epsilon}\log(\epsilon|\mathcal{M}|)$). ◀

Second, Theorem 5 unifies and generalizes a number of previous statements. In [26], a collection of results on Mnets were presented using different techniques: for the dual set system induced by regions of union complexity $\kappa(\cdot)$ using cuttings, for rectangles using divide-and-conquer constructions, and for triangles using ϵ-nets. All these and more results follow as immediate corollaries of Theorem 5.

▶ **Corollary 7** (See Table 2). *There exist ϵ-Mnets of size*
1. *$O\big(\kappa(\frac{1}{\epsilon})\big)$ for the dual set system induced by objects in $\mathbb{R}^2$ with union complexity $\kappa(\cdot)$. In particular, $O\left(\frac{1}{\epsilon}\log^*\frac{1}{\epsilon}\right)$ for the dual set systems induced by α-fat triangles[5], $O\left(\frac{1}{\epsilon}2^{O(\log^*\frac{1}{\epsilon})}\right)$ for the dual set system induced by locally γ-fat semialgebraic objects[6] in the plane, and $O\left(\frac{1}{\epsilon}\right)$ for the dual set systems induced by triangles of approximately same size [22].*
2. *$O\left(\frac{1}{\epsilon}\log^2\frac{1}{\epsilon}\right)$ for the primal set system induced by α-fat triangles.*
3. *$O\left(\frac{1}{\epsilon}\log\frac{1}{\epsilon}\right)$ for the primal set system induced by rectangles in the plane.*
4. *$O(\frac{1}{\epsilon^2})$ for the primal system induced by lines, strips and cones in the plane, improving the previous-best results by polylogarithmic factors. They were $O(\frac{1}{\epsilon^2}\log^2\frac{1}{\epsilon})$, $O(\frac{1}{\epsilon^2}\log^3\frac{1}{\epsilon^2})$ and $O(\frac{1}{\epsilon^2}\log^4\frac{1}{\epsilon})$ respectively.*
5. *$O\left(\frac{1}{\epsilon}\right)$ for the primal set system of semialgebraic pseudo-disks and $O(\frac{1}{\epsilon^{\lfloor d/2\rfloor}})$ for the primal set system of half-spaces.*

[5] For a fixed parameter α with $0<\alpha\le\pi/3$, a triangle is α-fat if all three of its angles are at least α.
[6] For a fixed parameter γ with $0<\gamma\le 1/4$, a planar semialgebraic object o is called locally γ-fat if, for any disk D centered in o and that does not fully contain o in its interior, we have $\text{area}(D\sqcap o)\ge\gamma\cdot\text{area}(D)$, where $D\sqcap o$ is the connected component of $D\cap o$ that contains the center of D.

The main open question in [26] was the following interesting pattern that was observed: for all the cases studied, a set system that had an ϵ-net of size $O\big(\frac{1}{\epsilon}\log\varphi(\frac{1}{\epsilon})\big)$ had Mnets of size $O\big(\frac{1}{\epsilon}\varphi(\frac{1}{\epsilon})\big)$. Theorem 5 now shows that this was not a coincidence. By Theorem B, a set system with shallow-cell complexity $\varphi(\cdot)$ has ϵ-nets of size $O\big(\frac{1}{\epsilon}\log\varphi(\frac{1}{\epsilon})\big)$. And now, from Theorem 5, it follows that it has Mnets of size $O\big(\frac{1}{\epsilon}\varphi(\frac{1}{\epsilon})\big)$.

2.3 l-Wise k-Shallow δ-Packings (Proof in Section 5)

Call a set system $(X,\mathcal{R})$ an l-wise δ-packing if for all distinct $A_1,\ldots,A_l\in\mathcal{R}$, we have $\big|\,(A_1\cup\cdots\cup A_l)\setminus(A_1\cap\cdots\cap A_l)\,\big|\geq\delta$.
Building on Chazelle's [7] proof of the packing lemma together with Turán's theorem on independent sets in graphs [28], Fox *et al.* [13, Lemma 2.5] proved the following:

▶ **Theorem F** (l-Wise δ-Packing Lemma). *Let $(X,\mathcal{R})$ be a set system such that $|X|=n$ and where for all $Y\subseteq X$ we have $|\mathcal{R}|_Y|=O(|Y|^d)$. If $\mathcal{R}$ is an l-wise δ-packing, for a positive integer l and $\delta\in\{1,\ldots,n\}$, then $|\mathcal{R}|=O\left(\left(\frac{n}{\delta}\right)^d\right)$, where the constant in the asymptotic notation depends on l and d.*

A set system $(X,\mathcal{R})$ is an *l-wise k-shallow δ-packing* if it is an l-wise δ-packing and furthermore, $|S|\leq k,\ \forall S\in\mathcal{R}$. Building on the proof in [20] and [24], we prove the following, which simultaneously generalizes three theorems: that of Haussler [15] (Theorem C), Fox *et al.* [13] (Theorem F) and Ezra *et al.* [9] (Theorem E).

▶ **Theorem 8** (l-Wise k-Shallow δ-Packing Lemma). *Let $(X,\mathcal{R})$ be a set system with $|X|=n$. Let $d,k,l,\delta>0$ be four integers such that $\text{VC-DIM}(\mathcal{R})\leq d$, and $\mathcal{R}$ is an l-wise k-shallow δ-packing. If $\mathcal{R}$ has shallow-cell complexity $\varphi(\cdot,\cdot)$, then*

$$|\mathcal{R}|=O\left(\frac{l^3n}{\delta}\cdot\varphi\Big(s,4l\cdot\frac{ks}{n}\Big)\right),\ \text{where } s=\frac{8l(l-1)dn}{\delta}-1.$$

▶ **Corollary 9.** *Theorems C, E (up to a constant factor) and F.*

Proof. Theorem E is Theorem 8 with l set to 2. To obtain Theorem F, set $k=n$ in Theorem 8. Theorem C is the special case of F when $l=2$. ◀

3 Proof of Theorem 4

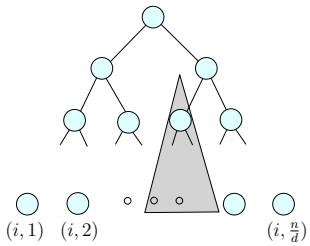
In this section we will build a set system with the desired shallow-cell complexity and then show that it contains a large shallow packing.

Proof of Theorem 4. Without loss of generality we assume that n is an integer multiple of d. The ground set X will be a subset of $\mathbb{N}\times\mathbb{N}$.

For each $1\leq i\leq d_1$, set $X_i=\{i\}\times\{1,\ldots,\frac{n}{d}\}$. Note that we are simply considering d_1 disjoint copies of $\{1,\ldots,\frac{n}{d}\}$; the singleton $\{i\}$ is here to distinguish X_i from X_j.

Define the following set system $\mathcal{P}_i$ on each X_i: $\mathcal{P}_i=\Big\{\big\{i\big\}\times\big\{2^\alpha\beta+1,\ldots,2^\alpha(\beta+1)\big\}\ \Big|\ 0\leq\alpha\leq\log_2\left(\frac{n}{d}\right),\ 0\leq\beta<2^{-\alpha}\frac{n}{d}\Big\}$.

Intuitively, consider a balanced binary tree $\mathcal{T}_i$ on X_i, with its leaves labeled $(i,1)$, $(i,2)$, $\ldots$, $(i,\frac{n}{d})$ (see figure).



Then for each node $v \in \mathcal{T}_i$, $\mathcal{P}_i$ contains a set consisting of the leaves of the subtree rooted at v. Here α is the height of the node (so 2^α is the number of elements in the corresponding subset), while β identifies one of the nodes of that height (among the $2^{\log(\frac{n}{d})-\alpha} = 2^{-\alpha} \cdot \frac{n}{d}$ choices).

▶ **Claim 10.** *For any $Y \subseteq X_i$ and [7] $r \in \mathbb{N}$, $|\mathcal{P}_i|_{Y,\leq r}| = O(|Y|)$. Specifically, $f_{\mathcal{P}_i}(m,r) \leq 2m$.*

Proof. For any $Y \subseteq X_i$, the sets in $\mathcal{P}_i|_Y$ are in a one-to-one correspondence with the nodes of $\mathcal{T}_i$ whose left and right subtrees, if they exist, both contain leaves labeled by Y. It is easy to see that if the nodes of $\mathcal{T}_i$ corresponding to Y form a connected sub-tree, then these nodes define a new binary tree whose leaves are still labeled by Y, and thus their number is at most $2|Y|-1$. Otherwise, the statement holds by induction on the number of connected components of Y in $\mathcal{T}_i$. ◀

Next, for each $d_1 + 1 \leq i \leq d$, let $Y_i = \{i\} \times \{1, \ldots, \frac{n}{d}\}$. For each Y_i, define $\mathcal{Q}_i = \left\{ \{i\} \times \{1, \ldots, \gamma\} \;\middle|\; 1 \leq \gamma \leq \frac{n}{d}, \gamma \in \mathbb{N} \right\}$, which can be seen as prefix sets of the sequence $\langle (i,1), \ldots, (i, \frac{n}{d}) \rangle$.

▶ **Claim 11.** *For any $Y \subseteq Y_i$ and $l \in \mathbb{N}$, $|\mathcal{Q}_i|_{Y,\leq l}| = O(l)$. Specifically, $f_{\mathcal{Q}_i}(m,l) \leq l$.*

Proof. The number of sets of size at most l in $\mathcal{Q}_i|_Y$ is $|\mathcal{Q}_i|_{Y,\leq l}| = \min\{l, |Y|\} \leq l$. ◀

Finally, the required base set will be $X = \left(\bigcup_{i=1}^{d_1} X_i \right) \cup \left(\bigcup_{i=d_1+1}^{d} Y_i \right)$. Its size is $|X| = d_1 \cdot \frac{n}{d} + (d - d_1) \cdot \frac{n}{d} = n$. The set system $\mathcal{R}^0$ is defined on X by taking d-wise union of the sets in $\mathcal{P}_i$'s and $\mathcal{Q}_i$'s: $\mathcal{R}^0 = \left\{ \bigcup_{i=1}^d r_i \;\middle|\; (r_1, r_2, \cdots, r_d) \in \mathcal{P}_1 \times \cdots \times \mathcal{P}_{d_1} \times \mathcal{Q}_{d_1+1} \times \cdots \times \mathcal{Q}_d \right\}$.

We will bound the shallow-cell complexity of $\mathcal{R}^0$ then construct a subset of $\mathcal{R}^0$ which is a large packing.

▶ **Claim 12.** *$\forall Y \subseteq X$, $\forall l \in \mathbb{N}$, $|\mathcal{R}^0|_{Y,\leq l}| = O\big(|Y|^{d_1} l^{d-d_1}\big)$. Specifically, $f_{\mathcal{R}^0}(m,l) \leq (2m)^{d_1} l^{d-d_1}$.*

Proof. Let $Y \subseteq X$, $|Y| = m$. Any set $S \in \mathcal{R}^0|_{Y,\leq l}$ can be uniquely written as the disjoint union $S = p_1 \cup \cdots \cup p_{d_1} \cup q_{d_1+1} \cup \cdots \cup q_d$, where $p_i \in \mathcal{P}_i|_{Y \cap X_i, \leq l}$ and $q_i \in \mathcal{Q}_i|_{Y \cap Y_i, \leq l}$. This yields an injection $\mathcal{R}^0|_{Y,\leq l} \mapsto \left(\prod_{1 \leq i \leq d_1} \mathcal{P}_i|_{Y \cap X_i, \leq l} \right) \times \left(\prod_{d_1+1 \leq i \leq d} \mathcal{Q}_i|_{Y \cap Y_i, \leq l} \right)$.

Thus by Claims 10 and 11, we have the required bound:

$$f_{\mathcal{R}^0}(m,l) = \max_{Y \subseteq X, |Y|=m} |\mathcal{R}^0|_{Y,\leq l}| \leq \Big(f_{\mathcal{P}_1}(m,l) \Big)^{d_1} \cdot \Big(f_{\mathcal{Q}_1}(m,l) \Big)^{d-d_1} \leq (2m)^{d_1} l^{d-d_1}. \quad \blacktriangleleft$$

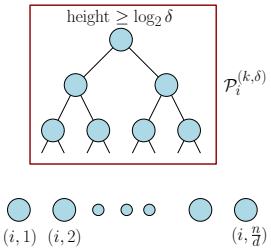
[7] Crucially, the bound is actually independent of r.

It remains to show that some subset of $\mathcal{R}^0$ is a large k-shallow δ-packing. For the given parameters k, δ and for all $1 \le i \le d_1$ and $d_1 + 1 \le j \le d$, define:

$$\mathcal{P}_i^{(k,\delta)} = \left\{ \{i\} \times \{2^\alpha\beta + 1, \dots, 2^\alpha(\beta+1)\} \;\middle|\; \begin{array}{l} \alpha, \beta \in \mathbb{N} \\ \log_2 \delta \le \alpha \le \log_2(\frac{k}{d}) \\ 0 \le \beta < 2^{-\alpha}(\frac{n}{d}) \end{array} \right\} \subseteq \mathcal{P}_i,$$

$$\mathcal{Q}_j^{(k,\delta)} = \left\{ \{j\} \times \{1, 2, \dots, \gamma\delta\} \mid 1 \le \gamma \le \frac{k}{d\delta} \right\} \subseteq \mathcal{Q}_j.$$

The intuition here is that we pick only the nodes in our binary trees $\mathcal{T}_i$ which have height at least $\log_2 \delta$ (and thus a symmetric difference of at least δ elements).



Similarly in $\mathcal{Q}_j$ we only pick every δ-th set. All these sets have size at most $\frac{k}{d}$. This is straightforward for $\mathcal{Q}_i^{(k,\delta)}$; on the other hand, a set in $\mathcal{P}_i^{(k,\delta)}$ defined by the pair (α, β) has size $2^\alpha \le \frac{k}{d}$.

All those sets also are integer intervals of the form $\{\lambda\delta + 1, \dots, \mu\delta\}$ for some $\lambda, \mu \in \mathbb{N}$ and thus pairwise δ-separated (for the $P_i^{(k,\delta)}$, notice that 2^α is a multiple of δ). Hence

$$\mathcal{R} = \left\{ p_1 \cup \dots \cup p_{d_1} \cup q_{d_1+1} \cup \dots \cup q_d \;\middle|\; (p,q) \in \prod_{1 \le i \le d_1} \mathcal{P}_i^{(k,\delta)} \times \prod_{d_1+1 \le i \le d} \mathcal{Q}_i^{(k,\delta)} \right\} \subseteq \mathcal{R}^0$$

is a δ-packing which is k-shallow. We bound its size:

$$\begin{aligned} |\mathcal{R}| &= \prod_{i=1}^{d_1} \left|\mathcal{P}_i^{(k,\delta)}\right| \cdot \prod_{i=d_1+1}^{d} \left|\mathcal{Q}_i^{(k,\delta)}\right| = \left(\frac{n}{d} \sum_{\alpha=\lceil \log_2 \delta \rceil}^{\lfloor \log_2(\frac{k}{d}) \rfloor} 2^{-\alpha} \right)^{d_1} \left(\frac{k}{d\delta}\right)^{d-d_1} \\ &\ge d^{-d} \left(2^{1-\lceil \log_2 \delta \rceil} - 2^{\lfloor \log_2(\frac{k}{d}) \rfloor} \right)^{d_1} n^{d_1} \left(\frac{k}{\delta}\right)^{d-d_1} \\ &\ge d^{-d} \left(\frac{1}{\delta} - \frac{2d}{k}\right)^{d_1} n^{d_1} \left(\frac{k}{\delta}\right)^{d-d_1} \ge d^{-d} (2\delta)^{-d_1} n^{d_1} \left(\frac{k}{\delta}\right)^{d-d_1} = \Omega\left(\frac{n^{d_1} k^{d-d_1}}{\delta^d} \right). \end{aligned}$$

◀

The gist of Haussler's probabilistic lower bound construction for Theorem D was to consider $\mathcal{R}^0$ with $d_1 = 0$ and randomly build a packing [15].

4 Proof of Theorem 5, Corollary 7

We first give a brief overview of a technical tool that is used in the proof of Theorem 5.

4.1 Preliminaries

We will use the following theorem of Matoušek and Patáková [23]. For $\gamma, \omega \subset \mathbb{R}^d$, we say that γ *crosses* ω if $\omega \cap \gamma \notin \{\emptyset, \omega\}$.

▶ **Theorem G** (Multilevel polynomial partitioning). *For every integer $d > 1$, there exist constants $K = K(d)$ and $C = C(d)$ such that the following holds. Given an n-point set $P \subset \mathbb{R}^d$ and a parameter $r > 1$, there exist a set $\Sigma^* \subseteq P$ with $|\Sigma^*| \leq r^K$ and d families of sets, Σ_k, $1 \leq k \leq d$, that form a partition of P*

$$P = \Sigma^* \cup \bigcup_{k=1}^{d} \bigcup_{S \in \Sigma_k} S$$

where the following properties hold for each $1 \leq k \leq d$:

1. *$\Sigma_k = \{\Sigma_{k1}, \ldots, \Sigma_{kt_k}\}$, $t_k \leq Cr^C$, and for $1 \leq l \leq t_k$, $|\Sigma_{kl}| \leq \frac{n}{r_k}$ with $r_k \in [r, r^K]$.*
2. *there exist a family of semialgebraic regions $\mathcal{S}_k = \{S_{k1}, \ldots, S_{kt_k}\}$ such that for each $1 \leq l \leq t_k$,*
   - **(a)** *S_{kl} is connected, defined by $O(r^C)$ polynomial inequalities of degree $O(r^C)$,*
   - **(b)** *$\Sigma_{kl} \subseteq S_{kl}$, and*
   - **(c)** *every set $\gamma \in \Gamma_{d,\Delta,s}$ crosses at most $C_{d,\Delta,s} \cdot r_k^{1-1/d}$ of the sets in $\mathcal{S}_k$, where the constant $C_{d,\Delta,s}$ depends only on d, Δ and s.*

Theorem G extends the Guth–Katz [14] *polynomial partitioning technique*, a partition of $\mathbb{R}^d$ by an algebraic variety which is balanced with respect to the set P. Here partitioning is applied not once but recursively on varieties of decreasing dimension. This will allow us to dispense with assumptions of genericity.

4.2 Proofs

We now give proofs of Theorem 5 and Corollary 7.

Proof of Theorem 5. Note that if $\epsilon = O(\frac{1}{n})$, then the trivial collection of singleton sets $\big\{\{p\} \colon p \in X\big\}$ will be an ϵ-Mnet for $(X, \mathcal{R})$, of size $n = O(\frac{1}{\epsilon})$. Therefore we may restrict ourselves to the case when

$$\epsilon > \frac{4\left(16 \cdot d \cdot C_{d,\Delta,s}\right)^{Kd}}{n}. \tag{1}$$

For $i = 0, \ldots, \lceil \log \frac{1}{\epsilon} \rceil$, let $\mathcal{R}_i \subseteq \mathcal{R}$ be an inclusion-maximal $\big(2^{i-1}\epsilon\, n\big)$-packing, with the additional constraint that each set in $\mathcal{R}_i$ has cardinality in $[2^i \epsilon n,\, 2^{i+1}\epsilon n)$. From Theorem E, we have

$$|\mathcal{R}_i| \leq \frac{C' d_0}{2^i \epsilon} \cdot \varphi\left(\frac{8d_0}{2^i \epsilon}, 48 d_0\right), \text{ where } C' \text{ is an absolute constant.} \tag{2}$$

Say $\mathcal{R}_i = \big\{\mathcal{R}_{i1}, \ldots, \mathcal{R}_{im_i}\big\}$, where $m_i = |\mathcal{R}_i|$. For a parameter r to be fixed later, consider the multilevel polynomial partitioning of $\mathcal{R}_{ij}$ as in Theorem G. We will write

$$\mathcal{R}_{ij} = \Sigma_{ij}^* \cup \bigcup_{k=1}^{d} \bigcup_{l=1}^{t_{ijk}} \Sigma_{ijkl},$$

where

1. We will denote by S_{ijkl} the corresponding connected semialgebraic region in $\mathbb{R}^d$ containing the set Σ_{ijkl}; see Theorem G.
2. $r_{ij1}, r_{ij2}, \ldots, r_{ijd} \in [r, r^K]$ where the constant K depends on d as defined in Theorem G.
3. For all $k = 1, 2, \ldots, d$, $t_{ijk} \leq Cr^C$, where the constant C depends on d and it is defined in Theorem G. This implies $\sum_{i=1}^{d} t_{ijk} \leq Cdr^C$.

4. $|\Sigma_{ijkl}| \leq \frac{|\mathcal{R}_{ij}|}{r_{ijk}}$ for all k and l.
5. $|\Sigma^*_{ij}| \leq r^K$.
6. For all $\gamma \in \Gamma_{d,\Delta,s}$ and every $k = 1, 2, \ldots, d$, the number of S_{ijkl} crossed by γ is at most $C_{d,\Delta,s} r_{ijk}^{1-1/d}$, where the constant $C_{d,\Delta,s}$ is defined in Theorem G.

The ϵ-Mnet $\mathcal{M}$ will be the union of a family $(\mathcal{M}_i)$ of set collections. For each $\mathcal{R}_{ij}$, we do the following: for all $k \in \{1, \ldots, d\}$ and $l \in \{1, \ldots, t_{ijk}\}$, if $|\Sigma_{ijkl}| \geq \frac{2^i \epsilon n}{8Cdr^C}$ then add Σ_{ijkl} to $\mathcal{M}_i$. Finally let $\mathcal{M} = \bigcup_{i=0}^{\lceil \log \frac{1}{\epsilon} \rceil} \mathcal{M}_i$.

It remains to show that $\mathcal{M}$ is the required ϵ-Mnet for an appropriate value of r. Namely,

(i) the required bound on $|\mathcal{M}|$ holds,
(ii) each set in $\mathcal{M}$ has size $\Omega(\epsilon n)$, and
(iii) for any $R \in \mathcal{R}$ with $|R| \geq \epsilon n$, there exists a set $Y \in \mathcal{M}$ where $Y \subseteq R$.

Let $r = (16dC_{d,\Delta,s})^d$, ensuring that $r^K < \frac{1}{4}\epsilon n$.

To see i), observe that $|\mathcal{M}_i| = O\big(dr^C \cdot |\mathcal{R}_i|\big) = O\left(\frac{d_0}{2^i\epsilon}\varphi\left(\frac{8d_0}{2^i\epsilon}, 48d_0\right)\right)$. Therefore, as $\varphi(\cdot,\cdot)$ is a non-decreasing function in the first variable, we have

$$|\mathcal{M}| = \sum_{i=0}^{\lceil \log(d_0/\epsilon)\rceil} |\mathcal{M}_i| = O\left(\sum_{i=0}^{\lceil \log(d_0/\epsilon)\rceil} \frac{d_0}{2^i\epsilon} \cdot \varphi\left(\frac{8d_0}{2^i\epsilon}, 48d_0\right)\right) = O\left(\frac{d_0}{\epsilon} \cdot \varphi\left(\frac{8d_0}{\epsilon}, 48d_0\right)\right).$$

To see ii), observe that each set added to $\mathcal{M}$ satisfies $|\Sigma_{ijkl}| \geq \frac{2^i\epsilon n}{8Cdr^C} = \Omega\big(\epsilon n\big)$.

To see iii), let $R \in \mathcal{R}$ be any set such that $|R| \geq \epsilon n$, and let i be the index such that $|R| \in [2^i\epsilon n, 2^{i+1}\epsilon n)$. There are two cases.

Case 1: $R \in \mathcal{R}_i$

Say $R = \mathcal{R}_{ij}$, then R contains all the sets Σ_{ijkl} (for all values of k and l), and it remains to argue that at least one was added to $\mathcal{M}$. So assume that it is not the case. Observe that

$$|\mathcal{R}_{ij}| = \sum_{k,l} |\Sigma_{ijkl}| + |\Sigma^*_{ij}| \leq Cdr^C \cdot \frac{2^i\epsilon n}{8Cdr^C} + r^K = 2^{i-3}\epsilon n + r^K < 2^i\epsilon n.$$

The last inequality follows from the fact that $r^K < 2^{i-2}\epsilon n$. We have reached a contradiction, as by construction we had $|\mathcal{R}_{ij}| \geq 2^i\epsilon n$.

Case 2: $R \notin \mathcal{R}_i$

By the maximality of $\mathcal{R}_i$, there exists an index j such that $\mathcal{R}_{ij} \in \mathcal{R}_i$ and $|R \cap \mathcal{R}_{ij}| \geq 2^{i-1}\epsilon n$. Note that the above bound on $|R \cap \mathcal{R}_{ij}|$ follows from the fact that $|R \cap \mathcal{R}_{ij}| \geq |\mathcal{R}_{ij}| - |R \Delta \mathcal{R}_{ij}|$. If R contains a set Σ_{ijkl} included in $\mathcal{M}_i$, then we are done. So assume it does not. Then consider the contribution to the points in the set $R \cap \mathcal{R}_{ij} = \left(\bigcup_{k,l} (R \cap \Sigma_{ijkl})\right) \cup \big(R \cap \Sigma^*_{ij}\big)$.

(a) All indices k, l such that $|\Sigma_{ijkl}| < \frac{2^i\epsilon n}{8Cdr^C}$. The total number of points contained in R from all such sets is at most $Cdr^C \cdot \frac{2^i\epsilon n}{8Cdr^C} = \frac{2^i\epsilon n}{8}$.
(b) All k such that the semialgebraic set γ defining R crosses the connected component S_{ijkl} corresponding to Σ_{ijkl}. By Theorem G, there are at most $C_{d,\Delta,s}\, r_{ijk}^{1-1/d}$ such sets, and by the property of multilevel partitioning, each such region contains at most $\frac{2^{i+1}\epsilon n}{r_{ijk}}$ points of X.
(c) The points of R contained in Σ^*_{ij}.

Using the fact that r is sufficiently large in terms of d, Δ and s, we have

$$|R\cap\mathcal{R}_{ij}| \leq 2^{i-3}\epsilon n+\sum_{k=1}^{d}\frac{2^{i+1}\epsilon n}{r_{ijk}}\cdot C_{d,\Delta,s}r_{ijk}^{1-\frac{1}{d}}+r^K < 2^{i-3}\epsilon n+\frac{dC_{d,\Delta,s}2^{i+1}\epsilon n}{r^{1/d}}+r^K < 2^{i-1}\epsilon n.$$

The last inequality follows from the fact that $r_{ijk}\geq r$, $r=(16dC_{d,\Delta,s})^d$ and $r^K<2^{i-2}\epsilon n$. We get a contradiction to the fact that $|R\cap\mathcal{R}_{ij}|\geq 2^{i-1}\epsilon n$, which completes the proof. ◀

Proof of Corollary 7.

1. The shallow-cell complexity of the dual set system induced by objects with union complexity $\kappa(\cdot)$ is $\varphi(m)=O(\frac{\kappa(m)}{m})$, which together with Theorem 5 implies the stated bound. The remaining bounds follow from the facts that $\kappa(m)$ for triangles with approximately same size [22] is $O(m)$, for α-fat triangles [12] is $O(m\log^* m)$ (where the constant of proportionality depends only on α), and for locally γ-fat objects [2] is $O(m2^{\log^* m})$, where the constant of proportionality in the linear term depends only on γ.
2. Ene *et al.* [10] proved the following: given a set X of n points in $\mathbb{R}^2$ and a parameter $r>0$, there exists a collection $\mathcal{O}_r$ of $O(r^3 n\log n)$ regions, such that for every α-fat triangle Δ, $|\Delta\cap X|\leq r$, there exists a subset $\mathcal{M}\subseteq\mathcal{O}_r$, $|\mathcal{M}|\leq 9$, such that $\left(\bigcup_{M\in\mathcal{M}}M\right)\cap X=\Delta\cap X$. This result together with Theorem 5 will give us the bound.
3. Ene *et al.* [10] proved the following: given a set X of n points in the plane and a parameter $r>0$, there exists a collection $\mathcal{O}_r$ of rectangles, with $|\mathcal{O}_r|=O(r^2 n\log n)$, such that for any rectangle R with $|R\cap X|\leq r$ there exists R_1, $R_2\in\mathcal{O}_k$ such that $(R_1\cup R_2)\cap X=R\cap X$. This result together with Theorem 5 will give us the bound.
4. Shallow-cell complexity $\varphi(m,r)$ is $O(m^2)$ for lines, $O(m^2 r)$ for strips, and $O(m^2r^2)$ for cones [26]. ◀

5 Proof of Theorem 8

The proof will use the following technical lemma, combining the ideas in [20, 24, 13].

▶ **Lemma 13.** *Let $(X,\mathcal{R})$ be a set system with $|X|=n$. Let d, l, δ be three integers such that* VC-DIM$(\mathcal{R})\leq d$, *and $\mathcal{R}$ is an l-wise δ-packing. If $A\subseteq X$ is a uniformly selected random sample of size $\frac{8l(l-1)dn}{\delta}-1$, then $|\mathcal{R}|\leq 2l\cdot\mathbb{E}\left[|\mathcal{R}|_A|\right]$.*

Proof. Pick a random sample R of size $s=\frac{8l(l-1)dn}{\delta}$ from X. Let $G_R=(\mathcal{R}|_R,E_{\mathcal{R}})$ be the unit distance graph on $\mathcal{R}|_R$, with an edge between any two sets whose symmetric difference is a singleton. Define the weight of a set $S'\in\mathcal{R}|_R$ to be the number of sets of $\mathcal{R}$ whose projection in $\mathcal{R}|_R$ is S', i.e. $w(S')=|\{r\in\mathcal{R}\mid r\cap R=S'\}|$. Define the weight of an edge $\{S'_i,S'_j\}\in E_{\mathcal{R}}$ as $w(S'_i,S'_j)=\min\{w(S'_i),w(S'_j)\}$. Let $W=\sum_{e\in E_{\mathcal{R}}}w(e)$.

We will use the following result from [20, Chapter 5, Proof 5.14].

▶ **Claim 14.** $W\leq 2d\cdot|\mathcal{R}|$.

Pick R by first picking a set A of $s-1$ elements and then selecting the remaining element a uniformly from $X\setminus A$. Let W_1 be the weight of the edges in G_R for which the element a is the symmetric difference. By symmetry, we have $\mathbb{E}[W]=s\cdot\mathbb{E}[W_1]$.

To compute $\mathbb{E}[W_1]$, first fix a set Y of $s-1$ vertices. Conditioned on this fixed choice of A, one shows (the interested reader will find a proof in the extended version of this paper):

▶ **Claim 15.** $\mathbb{E}\big[W_1|A=Y\big]\geq\frac{\delta/n}{2l(l-1)}\Big(|\mathcal{R}|-l\,|\mathcal{R}|_Y|\Big)$.

Using the fact that $\mathbb{E}[W] = s \cdot \mathbb{E}[W_1]$, one can compute an upper bound on $\mathbb{E}[W]$:

$$\begin{aligned}\mathbb{E}[W] = s \cdot \mathbb{E}[W_1] &= s \cdot \sum_{\substack{Y \subseteq X \\ |Y|=s-1}} \mathbb{E}[W_1 | A = Y] \cdot \Pr[A = Y] \\ &\geq s \cdot \sum_{\substack{Y \subseteq X \\ |Y|=s-1}} \frac{\delta}{2l(l-1)n} \Big(|\mathcal{R}| - l \cdot |\mathcal{R}|_Y| \Big) \cdot \Pr[A = Y] \quad \text{(by Claim 15)} \\ &\geq 4d \left(|\mathcal{R}| \sum_{\substack{Y \subseteq X \\ |Y|=s-1}} \Pr[A = Y] - l \sum_{\substack{Y \subseteq X \\ |Y|=s-1}} |\mathcal{R}|_Y| \cdot \Pr[A = Y] \right) = 4d(|\mathcal{R}| - l\mathbb{E}[|\mathcal{R}|_A|]).\end{aligned}$$

Combining Claim 14 and the above lower bound on $\mathbb{E}[W]$, we get $2d|\mathcal{R}| \geq \mathbb{E}[W] \geq 4d|\mathcal{R}| - 4dl \cdot \mathbb{E}[|\mathcal{R}|_A|]$. This implies $|\mathcal{R}| \leq 2l \cdot \mathbb{E}[|\mathcal{R}|_A|]$. ◀

Proof of Theorem 8. Let $A \subseteq X$ be a random sample of size $s := \frac{8l(l-1)dn}{\delta} - 1$. Let $\mathcal{R}_1 = \big\{ S \in \mathcal{R} \text{ s.t. } |S \cap A| \geq 4l \cdot \frac{ks}{n} \big\}$. Each element $x \in X$ belongs to A with probability $\frac{s}{n}$, and thus the expected number of elements in A from a fixed set of t elements is $\frac{ts}{n}$. This implies that $\mathbb{E}[|S \cap A|] \leq \frac{ks}{n}$ as $|S| \leq k$ for all $S \in \mathcal{R}$. Markov's inequality then bounds the probability of a set of $\mathcal{R}$ belonging to $\mathcal{R}_1$: $\Pr[S \in \mathcal{R}_1] = \Pr\big[|S \cap A| > 4l \cdot \frac{ks}{n}\big] \leq \frac{1}{4l}$. Thus

$$\mathbb{E}[|\mathcal{R}|_A|] \leq \mathbb{E}[|\mathcal{R}_1|] + \mathbb{E}[|(\mathcal{R} \setminus \mathcal{R}_1)|_A|] \leq \sum_{S \in \mathcal{R}} \Pr[S \in \mathcal{R}_1] + s \cdot \varphi\Big(s, 4l \cdot \frac{ks}{n}\Big) \leq \frac{|\mathcal{R}|}{4l} + s \cdot \varphi\Big(s, 4l \cdot \frac{ks}{n}\Big),$$

where we used the fact that $|(\mathcal{R} \setminus \mathcal{R}_1)|_A| = O\big(|A| \cdot \varphi(|A|, t)\big)$, where $t = \max_{S \in \mathcal{R} \setminus \mathcal{R}_1} |S| \leq 4l\frac{ks}{n}$. Now the bound follows from Lemma 13. ◀

6 Conclusion

Lower bound for the Shallow Packing Lemma

The lower bound construction given in the proof of Theorem 4, showing the optimality of the Shallow Packing Lemma (Theorem E), is constructive. Also observe that it can be realized in a number of simple ways, for example with points on a square grid and sets induced by some specific $(2d)$-gons, i.e., a semialgebraic set system with constant description complexity.

Applications of Mnets

Corollary 6 shows that the existence of small ϵ-nets follows immediately from the more general structure of Mnets. Macbeath regions for convex bodies have recently found algorithmic applications such as volume estimation of convex bodies [4, 3]. We believe that Mnets will also find important applications and connections to various aspects of set systems with bounded VC dimension.

Computing Mnets

In the real RAM model of computation one can compute exactly with arbitrary real numbers and each arithmetic operation takes unit time. Matoušek and Patáková [23] gave the following algorithmic counterpart of Theorem G.

▶ **Theorem H** (Algorithmic Multilevel Polynomial Partitioning). *The sets Σ^*, Σ_{ij}, S_{ij} from Theorem G can be computed in time $O(nr^C)$* [8].

Using this result and the construction in the proof of Theorem 5, we can get a randomized algorithm with time complexity poly $\left(n, \frac{1}{\epsilon}\right)$ that computes Mnets for semialgebraic set systems matching the upper bound on the size of Mnets from Theorem 5.

Acknowledgements. Part of this work was done when Kunal Dutta and Arijit Ghosh were researchers in D1: Algorithms & Complexity, Max-Planck-Institute for Informatics, Germany, supported by the Indo-German Max Planck Center for Computer Science (IMPECS).

References

1 P. K. Agarwal, J. Pach, and M. Sharir. State of the Union (of Geometric Objects): A Review. In J. Goodman, J. Pach, and R. Pollack, editors, *Computational Geometry: Twenty Years Later*, pages 9–48. American Mathematical Society, 2008.

2 B. Aronov, M. de Berg, E. Ezra, and M. Sharir. Improved Bounds for the Union of Locally Fat Objects in the Plane. *SIAM J. Comput.*, 43(2):543–572, 2014.

3 S. Arya, G. D. da Fonseca, and D. M. Mount. Optimal Area-Sensitive Bounds for Polytope Approximation. In *Proc. 28th Annual Symposium on Computational Geometry (SoCG)*, pages 363–372, 2012.

4 S. Arya, G. D. da Fonseca, and D. M. Mount. On the Combinatorial Complexity of Approximating Polytopes. In *Proc. 32nd International Symposium on Computational Geometry (SoCG)*, volume 51, pages 11:1–11:15, 2016.

5 S. Basu, R. Pollack, and M. F. Roy. *Algorithms in Real Algebraic Geometry.* Springer-Verlag, 2003.

6 T. M. Chan, E. Grant, J. Könemann, and M. Sharpe. Weighted Capacitated, Priority, and Geometric Set Cover via Improved Quasi-Uniform Sampling. In Proc. 23rd ACM-SIAM Symposium on Discrete Algorithms (SODA), pages 1576–1585, 2012.

7 B. Chazelle. A note on Haussler's packing lemma. See Section 5.3 from *Geometric Discrepancy: An Illustrated Guide* by J. Matoušek, 1992.

8 B. Chazelle. *The Discrepancy Method: Randomness and Complexity.* Cambridge University Press, Cambridge, New York, 2000.

9 K. Dutta, E. Ezra, and A. Ghosh. Two Proofs for Shallow Packings. *Discrete & Computational Geometry*, 56(4):910–939, 2016. Extended abstract appeared in *Proc. 31st International Symposium on Computational Geometry (SoCG)*, pages 96–110, 2015.

10 A. Ene, S. Har-Peled, and B. Raichel. Geometric Packing under Non-uniform Constraints. In *Proc. 28th Annual Symposium on Computational Geometry (SoCG)*, pages 11–20, 2012.

11 E. Ezra. A Size-Sensitive Discrepancy Bound for Set Systems of Bounded Primal Shatter Dimension. *SIAM J. Comput.*, 45(1):84–101, 2016. Extended abstract appeared in *Proc. 25th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1378–1388, 2014.

12 E. Ezra, B. Aronov, and S. Sharir. Improved Bound for the Union of Fat Triangles. In Proc. 22nd ACM-SIAM Symposium on Discrete Algorithms (SODA), pages 1778–1785, 2011.

13 J. Fox, J. Pach, A. Sheffer, A. Suk, and J. Zahl. A Semi-Algebraic Version of Zarankiewicz's Problem. *J. of the European Mathematical Society*, to appear.

14 L. Guth and N. H. Katz. On the Erdös distinct distances problem in the plane. *Annals of Math.*, 181(1):155–190, 2015.

[8] The constant C is the same as in Theorem G.

15 D. Haussler. Sphere Packing Numbers for Subsets of the Boolean n-Cube with Bounded Vapnik-Chervonenkis Dimension. *J. Comb. Theory, Ser. A*, 69(2):217–232, 1995.

16 D. Haussler and E. Welzl. Epsilon-nets and simplex range queries. *Discrete & Computational Geometry*, 2:127–151, 1987.

17 A. Kupavskii, N. H. Mustafa, and J. Pach. Near-Optimal Lower Bounds for ϵ-nets for Half-spaces and Low Complexity Set Systems. In M. Loebl, J. Nešetřil, and R. Thomas, editors, *A Journey Through Discrete Mathematics: A Tribute to Jiří Matoušek*. Springer, 2017. Extended abstract with the title "New Lower Bounds for epsilon-Nets" appeared in *Proc. 32nd International Symposium on Computational Geometry (SoCG)*, 54:1–54:16, 2016.

18 Yi Li, Philip M. Long, and Aravind Srinivasan. Improved bounds on the sample complexity of learning. *J. of Computer and System Sciences*, 62(3):516–527, 2001. doi:10.1006/jcss.2000.1741.

19 A. M. Macbeath. A theorem on non-homogeneous lattices. *Annals of Math.*, 56:269–293, 1952.

20 J. Matoušek. *Geometric Discrepancy: An Illustrated Guide*. Algorithms and Combinatorics. Springer, Berlin, New York, 1999.

21 J. Matoušek. *Lectures in Discrete Geometry*. Springer-Verlag, New York, NY, 2002.

22 J. Matoušek, J. Pach, M. Sharir, S. Sifrony, and E. Welzl. Fat Triangles Determine Linearly Many Holes. *SIAM J. Comput.*, 23(1):154–169, 1994.

23 J. Matoušek and Z. Patáková. Multilevel Polynomial Partitions and Simplified Range Searching. *Discrete & Computational Geometry*, 54(1):22–41, 2015.

24 N. H. Mustafa. A Simple Proof of the Shallow Packing Lemma. *Discrete & Computational Geometry*, 55(3):739–743, 2016.

25 N. H. Mustafa, K. Dutta, and A. Ghosh. A Simple Proof of Optimal Epsilon-nets. *Combinatorica*, to appear.

26 N. H. Mustafa and S. Ray. ϵ -Mnets: Hitting Geometric Set Systems with Subsets. *Discrete & Computational Geometry*, 57(3):625–640, 2017. Extended abstract with the title "Near-Optimal Generalisations of a Theorem of Macbeath" appeared in *Proc. 31st Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 578–589, 2014.

27 N. H. Mustafa and K. Varadarajan. Epsilon-approximations and Epsilon-nets. In J. E. Goodman, J. O'Rourke, and C. D. Tóth, editors, *Handbook of Discrete and Computational Geometry*. CRC Press LLC, 2017.

28 J. Pach and P. K. Agarwal. *Combinatorial Geometry*. John Wiley & Sons, New York, NY, 1995.

29 N. Sauer. On the Density of Families of Sets. *J. Comb. Theory, Ser. A*, 13(1):145–147, 1972.

30 S. Shelah. A Combinatorial Problem, Stability and Order for Models and Theories in Infinitary Languages. *Pacific J. of Mathematics*, 41:247–261, 1972.

31 K. R. Varadarajan. Weighted Geometric Set Cover via Quasi-Uniform Sampling. In *Proc. 42nd Symposium on Theory of Computing (STOC)*, pages 641–648, 2010.

Topological Data Analysis with Bregman Divergences*†

Herbert Edelsbrunner[1] and Hubert Wagner[2]

1 **IST Austria, Klosterneuburg, Austria**
edels@ist.ac.at
2 **IST Austria, Klosterneuburg, Austria**
hub.wag@gmail.com

Abstract

We show that the framework of topological data analysis can be extended from metrics to general Bregman divergences, widening the scope of possible applications. Examples are the Kullback – Leibler divergence, which is commonly used for comparing text and images, and the Itakura – Saito divergence, popular for speech and sound. In particular, we prove that appropriately generalized Čech and Delaunay (alpha) complexes capture the correct homotopy type, namely that of the corresponding union of Bregman balls. Consequently, their filtrations give the correct persistence diagram, namely the one generated by the uniformly growing Bregman balls. Moreover, we show that unlike the metric setting, the filtration of Vietoris-Rips complexes may fail to approximate the persistence diagram. We propose algorithms to compute the thus generalized Čech, Vietoris-Rips and Delaunay complexes and experimentally test their efficiency. Lastly, we explain their surprisingly good performance by making a connection with discrete Morse theory.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Topological data analysis, Bregman divergences, persistent homology, proximity complexes, algorithms

Digital Object Identifier 10.4230/LIPIcs.SoCG.2017.39

1 Introduction

The starting point for the work reported in this paper is the desire to extend the basic topological data analysis (TDA) paradigm to data measured with dissimilarities. In particular for high-dimensional data, such as discrete probability distributions, notions of dissimilarity inspired by information theory behave strikingly different from the Euclidean distance, which is the usual setting for TDA. On the practical side, the Euclidean distance is particularly ill-suited for many types of high-dimensional data; see for example [21], which provides evidence that the Euclidean distance consistently performs the worst among several dissimilarity measures across a range of text-retrieval tasks. A broad class of dissimilarities are the *Bregman divergences* [8]. Its most prominent members are the *Kullback-Leibler divergence* [23], which is commonly used both for text documents [5, 21] and for images [14], and the *Itakura-Saito divergence* [22], which is popular for speech and sound data [18]. We propose a TDA framework in the setting of Bregman divergences. Since TDA and more generally computational topology are young and emerging fields, we provide some context for the reader. For more a comprehensive introduction, see the recent textbook [16].

* A full version of the paper is available at https://arxiv.org/abs/1607.06274.
† This research is partially supported by the Toposys project FP7-ICT-318493-STREP.

Computational topology. Computational topology is an algorithmic approach to describing shape in a coarser sense than computational geometry does. TDA utilizes such algorithms within data analysis. One usually works with a finite set of points, possibly embedded in a high-dimensional space. Such data may be viewed as a collection of balls of a radius that depends on the scale of interest. Intersections reveal the connectivity of the data. For example, the components of the intersection graph correspond to the components of the union of balls.

Homology groups. These are studied in the area of algebraic topology, where they are used to describe and analyze topological spaces; see e.g. [20]. The *connected components* of a space or, dually, the *gaps* between them are encoded in its zero-dimensional homology group. There is a group for each dimension. For example, the one-dimensional group encodes *loops* or, dually, the *tunnels*, and the two-dimensional group encodes *closed shells* or, dually, the *voids*. Importantly, homology provides a formalism to talk about different kinds of connectivity and holes of a space that allows for fast algorithms.

Nerves and simplicial complexes. The *nerve* of a collection of balls generalizes the intersection graph and contains a k-dimensional simplex for every $k+1$ balls that have a non-empty common intersection. It is a hypergraph that is closed under taking subsets, a structure known as a *simplicial complex* in topology. If the balls are convex, then the Nerve Theorem states that this combinatorial construction captures the topology of the union of balls. More precisely, the nerve and the union have the same homotopy type and therefore isomorphic homology groups [24]. This result generalizes to the case in which the balls are not necessarily convex but their common intersections of all orders are contractible. In the context in which we center a ball of some radius at each point of a given set, the nerve is referred to as the *Čech complex* of the points for the given radius. Its k-skeleton is obtained by discarding simplices of dimension greater than k. The practice-oriented reader will spot a flaw in this setup: fixing the radius is a serious drawback that limits data analysis applications.

Persistent homology. To remedy this deficiency, we study the evolution of the topology *across all scales*, thus developing what we refer to as *persistent homology*. For graphs and connected components, this idea is natural but more difficult to flesh out in full generality. In essence, one varies the radius of balls from 0 to ∞, giving rise to a nested sequence of spaces, called a *filtration*. Topological features, namely homology classes of different dimensions, are created and destroyed along the way. In practice, one computes the *persistence diagram* of a filtration, which discriminates topological features based on their lifetime, or *persistence*. The persistence diagram serves as a compact *topological descriptor* of a dataset, which is provably robust against noise. Owing to its algebraic and topological foundations, the theory is very general. Importantly, the Nerve Theorem extends to filtrations [11, Lemma 3.4], therefore, for simplicity, we often restrict our proofs to balls of fixed radius.

TDA in the Bregman setting. In the light of the above, there are only two obstacles to applying topological data analysis to data measured with Bregman divergences. We need to prove that the Nerve Theorem applies also when the balls are induced by Bregman divergences, and we need to provide efficient algorithms to construct the relevant complexes, so that the existing algorithms for persistence diagrams can be used without modification. The main complication is that the balls may be nonconvex, which we overcome by combining results from convex analysis and topology.

Applications. Persistence is an important method within TDA, which has been successfully used in a variety of applications. In low dimensions, it was for example used to shed light on the distribution of matter in the Universe [31] and to characterize the structure of atomic configurations in silica glass [25]. As for high-dimensional data, Chan *et al.* analyze viral DNA and relate persistent cycles with recombinations [10], and Port *et al.* study languages leaving the interpretation of a persistent cycle in the Indo-Germanic family open [28].

Related work. This paper is the first work at the intersection of topology and Bregman divergences. We list related papers in relevant fields. In machine learning, Banerjee *et al.* use the family of Bregman divergences as the unifying framework for clustering algorithms [2]. The field of information geometry deals with selected Bregman divergences and related concepts from a geometric perspective [1]. Building on the classical work of Rockafellar [30] in convex analysis, Bauschke and Borwein are the first to use the Legendre transform for analyzing Bregman divergences [4]. Boissonnat, Nielsen and Nock [6] use similar methods to make significant contributions at the intersection of computational geometry and Bregman divergences. In particular, they study the geometry of Bregman balls and Delaunay triangulations, but not the topologically more interesting Delaunay, or *alpha*, complexes. In the Euclidean setting, the basic constructions are well understood [3, 33], including approximations, which are interesting and useful, but beyond the scope of this paper.

Results. This paper provides the first general TDA framework that applies to high dimensional data measured with non-metric dissimilarities. Indeed, prior high dimensional applications of TDA were restricted to low dimensional homology, required custom-made topological results, or used common metrics such as the Euclidean and the Hamming distances, which are often not good choices for such data. We list the main technical contributions:

1. We show that the balls under any Bregman divergence have common intersections that are either empty of contractible.
2. We show that the persistence diagram of the Vietoris-Rips complex can be arbitrarily far from that of the filtration of the union on Bregman balls.
3. We show that the radius functions that correspond to the Čech and Delaunay complexes for Bregman divergences are generalized discrete Morse functions.
4. We develop algorithms for computing Čech and Delaunay radius functions for Bregman divergences, which owe their speed to non-trivial structural properties implied by Result 3.

Most fundamental of the four is Result 1, which forms the theoretical foundation of TDA in the Bregman setting. It implies that the Čech and Delaunay complexes for a given radius have the same homotopy type as the union of Bregman balls. Combined with the Nerve Theorem for filtrations, it also implies that the filtration of Čech and Delaunay complexes have the same persistence diagram as the filtration of the unions. In the practice of TDA, the filtration of Vietoris-Rips complexes is often substituted for the filtration of Čech or Delaunay complexes. For metrics, this is justified by the small bottleneck distance between the persistence diagrams if drawn in log-log scale. Result 2 shows that such a substitution is not generally justified for Bregman divergences. In other words, for some Bregman divergences higher order interactions have to be taken into account explicitly as they are not approximated by implications of pairwise interactions. To appreciate Results 3 and 4, we note that the *Čech radius function* maps every simplex to the smallest radius, r, such that the simplex belongs to the Čech complex for radius r, and similarly for Delaunay. Being a generalized discrete Morse function has important structural consequences that make it possible to construct Čech and Delaunay complexes in an output-sensitive manner. We support this claim with experiments.

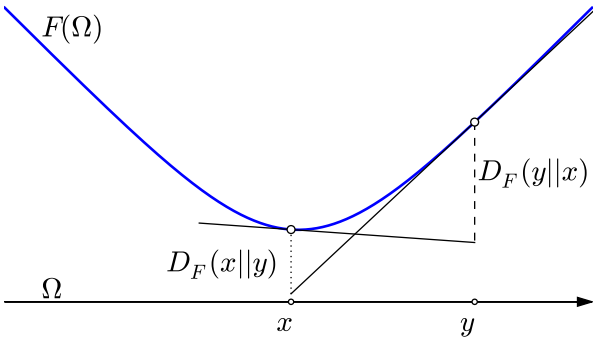


Figure 1 Geometric interpretation of the Bregman divergence associated with the function F on Ω.

2 Bregman Divergences

Bregman divergences are sometimes called *distances* because they measure dissimilarity. As we will see shortly, they are generally not symmetric, and they always violate the triangle inequality. So really they satisfy only the first axiom of a metric, mapping ordered pairs to non-negative numbers and to zero iff the two elements are equal.

We begin with a formal introduction of the concept, which originated in the paper by Bregman [8]. Their basic properties are well known; see the recent paper by Boissonnat, Nielsen and Nock [6]. We stress that our setting is slightly different: following Bauschke and Borwein [4], we define the divergences in terms of *functions of Legendre type*. The crucial benefit of this additional requirement is that the conjugate of a function of Legendre type is again a function of Legendre type, even if the domain is bounded as in the important case of the standard simplex. In contract, the conjugate of a differentiable and strictly convex function that is not of Legendre type is not necessarily again a convex function.

Functions of Legendre type. Let $\Omega \subseteq \mathbb{R}^n$ be an nonempty open convex set and $F\colon \Omega \to \mathbb{R}$ a strictly convex differentiable function. In addition, we require that the length of the gradient of F goes to infinity whenever we approach the boundary of Ω. Following [30, page 259], we say that $F\colon \Omega \to \mathbb{R}$ is a function of *Legendre type*. As suggested by the naming convention, these conditions are crucial when we apply the Legendre transform to F. The last condition prevents us from arbitrarily restricting the domain and is vacuous whenever Ω does not have a boundary, for example when $\Omega = \mathbb{R}^n$. For points $x, y \in \Omega$, the *Bregman divergence* from x to y associated with F is the difference between F and the best linear approximation of F at y, both evaluated at x:

$$D_F(x\|y) = F(x) - [F(y) + \langle \nabla F(y), x - y \rangle] \,. \tag{1}$$

As illustrated in Figure 1, we get $D_F(x\|y)$ by first drawing the hyperplane that touches the graph of F at the point $(y, F(y))$. We then intersect the vertical line that passes through x with the graph of F and the said hyperplane: the Bregman divergence is the height difference between the two intersections. Note that it is not necessarily symmetric: $D_F(x\|y) \neq D_F(y\|x)$ for most F, x, y.

Accordingly, we introduce two balls of radius $r \geq 0$ centered at a point $x \in \Omega$: the *primal Bregman ball* containing all points y so that the divergence from x to y is at most r, and the

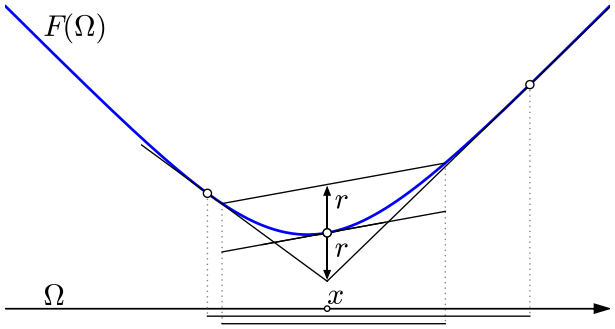


Figure 2 The primal Bregman ball with center x is obtained by illuminating the graph of F from below. In contrast, the dual Bregman ball is constructed by cutting the graph with the elevated line.

dual Bregman ball containing all points y so that the divergence from y to x as at most r:

$$B_F(x;r) = \{y \in \Omega \mid D_F(x\|y) \leq r\}; \tag{2}$$
$$B'_F(x;r) = \{y \in \Omega \mid D_F(y\|x) \leq r\}. \tag{3}$$

To construct the primal ball geometrically, we take the point $(x, F(x) - r)$ at height r below the graph of F and shine light along straight half-lines emanating from this point onto the graph. The ball is the vertical projection of the illuminated portion onto $\mathbb{R}^n$; see Figure 2. To construct the dual ball geometrically, we start with the hyperplane that touches the graph of F at $(x, F(x))$, translating it to height r above the initial position. The ball is the vertical projection of the portion of the graph below the translated hyperplane onto $\mathbb{R}^n$; see again Figure 2.

Since D_F is not necessarily symmetric, the two Bregman balls are not necessarily the same. Indeed, the dual ball is necessarily convex while the primal ball is not.

▶ **Result 1** (Convexity Property). *$D_F : \Omega \times \Omega \to \mathbb{R}$ is strictly convex in the first argument but not necessarily convex in the second argument.*

Proof. Fixing y, set $f(x) = D_F(x\|y)$. According to (1), f is the difference between F and an affine function; compare with the geometric interpretation of the dual Bregman ball. The strict convexity of F implies the strict convexity of f. This argument does not apply to $g(y) = D_F(x\|y)$, which we obtain by fixing x, and it is easy to find an example in which g is non-convex; see Figure 4. ◀

Legendre transform and conjugate function. In a nutshell, the Legendre transform applies elementary polarity to the graph of F, giving rise to the graph of another, conjugate function, $F^* : \Omega^* \to \mathbb{R}$, that relates to F in interesting ways. If F is of Legendre type then so is F^*; see [30, Theorem 26.5].

The notion of polarity we use in this paper relates points in $\mathbb{R}^n \times \mathbb{R}$ with affine functions $\mathbb{R}^n \to \mathbb{R}$. Specifically, it maps a point $C = (c, \gamma)$ to the function defined by $C^*(x) = \langle c, x\rangle - \gamma$, and it maps C^* back to $(C^*)^* = C$. We refer to Figure 3 for an illustration.

As a first step in constructing the conjugate function, we get Ω^* as the set of points $e = c^* = \nabla F(c)$ with $c \in \Omega$. We define $h : \Omega \to \Omega^*$ by mapping c to $h(c) = c^*$. Note that differentiability of strictly convex functions implies *continuous* differentiability [13, Theorem 2.86], hence h is a homeomorphism between the two domains.

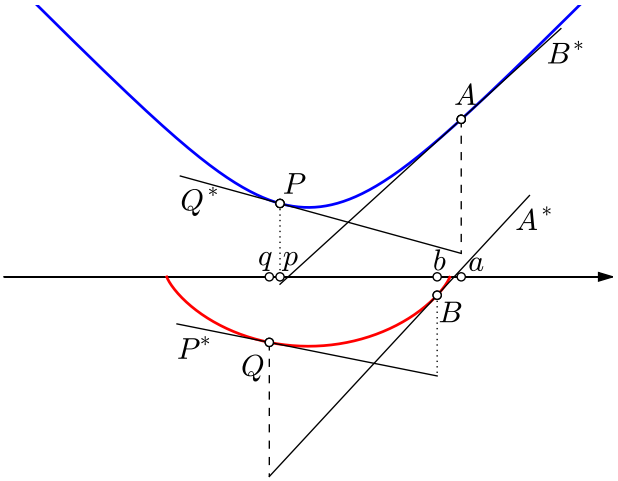


Figure 3 *Top*: the graph of F and the tangent lines that illustrate the two Bregman divergences between a and p associated with F. *Bottom*: the graph of F^* and the tangent lines that illustrate the two Bregman divergences between $b = a^*$ and $q = p^*$ associated with F^*. Note that A^*, B^*, P^*, Q^* are the affine functions corresponding to points A, B, P, Q.

The *conjugate function*, $F^*\colon \Omega^* \to \mathbb{R}$, is then defined by mapping e to $F^*(e) = \epsilon$ such that (e, ϵ) is the polar point of the affine function whose graph touches the graph of F in the point $(c, F(c))$. Writing $b = a^*$ and $q = p^*$, we eventually get

$$D_{F^*}(b\|q) = F^*(b) - P^*(b) \geq 0, \tag{4}$$
$$D_{F^*}(q\|b) = F^*(q) - A^*(q) \geq 0. \tag{5}$$

For more details see the Appendix present in the full version of this paper. For explanation, see again Figure 3. The left-hand sides of (4) and (5) are both non-negative and vanish iff $b = q$. Since this is true for all points $b, q \in \Omega^*$, F^* is strictly convex, provided Ω^* is convex. Proving that this assumption is always fulfilled is more involved. We therefore resort to a classical theorem [30, Theorem 26.5], which states that F^* is again of Legendre type and, in particular, Ω^* is convex. Hence, F^* defines a Bregman divergence and, importantly, this divergence is symmetric to the one defined by F.

▶ **Result 2** (Duality Property). *Let $F\colon \Omega \to \mathbb{R}$ and $F^*\colon \Omega^* \to \mathbb{R}$ be conjugate functions of Legendre type. Then $D_F(a\|p) = D_{F^*}(p^*\|a^*)$ for all $a, p \in \Omega$.*

In words, the Legendre transform preserves the divergences, but it does so by exchanging the arguments. This is interesting because D_F is strictly convex in the first argument and so is D_{F^*}, only that its first argument corresponds to the second argument of D_F. To avoid potential confusion, we thus consider the primal and dual Bregman balls of F^*:

$$B_{F^*}(u; r) = \{v \in \Omega^* \mid D_{F^*}(u\|v) \leq r\}, \tag{6}$$
$$B'_{F^*}(u; r) = \{v \in \Omega^* \mid D_{F^*}(v\|u) \leq r\}, \tag{7}$$

where we write $u = x^*$ and $v = y^*$ so we can compare the two balls with the ones defined in (2) and (3). As mentioned earlier, both dual balls are necessarily convex while both primal balls are possibly non-convex. Recall the homeomorphism $h\colon \Omega \to \Omega^*$ that maps x to x^*. It also maps $B_F(x; r)$ to $B'_{F^*}(u; r)$ and $B'_F(x; r)$ to $B_{F^*}(u; r)$. In words, it makes the

non-convex ball convex and the convex ball non-convex, and it does this while preserving the divergences. We use this property to explain the necessity on using functions of Legendre type; it also plays a crucial role later. Consider a dual Bregman ball with a non-convex conjugate image, namely the corresponding primal ball. Then the restriction of F to this dual ball is strictly convex and differentiable. However, it is not of Legendre type and its conjugate has a non-convex domain.

Examples. We close this section with a short list of functions, their conjugates, and the corresponding Bregman divergences. *Half the squared Euclidean norm* maps a point $x \in \mathbb{R}^n$ to $F(x) = \frac{1}{2}\|x\|^2$. The gradient is $\nabla F(x) = x$, and the conjugate is defined by $F^*(x) = F(x)$. The divergence associated with F is *half the squared Euclidean distance*:

$$D_F(x\|y) = \tfrac{1}{2}\|x - y\|^2. \tag{8}$$

This Bregman divergence is special because it is symmetric in the two arguments.

The *Shannon entropy* of a discrete probability distribution is $-\sum_{i=1}^n x_i \ln x_i$. To turn this into a convex function, we change the sign, and to simplify the computations, we subtract the sum of the x_i, defining $F(x) = \sum_{i=1}^n [x_i \ln x_i - x_i]$ over the positive orthant, which we denote as $\mathbb{R}^n_+$. The gradient is $\nabla F(x) = [\ln x_1, \ln x_2, \ldots, \ln x_n]^T$, and the conjugate is the *exponential function*, $F^*(u) = \sum_{i=1}^n e^{u_i}$, with $u = x^*$, defined on $\mathbb{R}^n$. Associated with F is the *Kullback-Leibler divergence* and with F^* is the *exponential loss*:

$$D_F(x\|y) = \sum_{i=1}^n \left[x_i \ln \tfrac{x_i}{y_i} - x_i + y_i\right], \tag{9}$$

$$D_{F^*}(u\|v) = \sum_{i=1}^n \left[e^{u_i} - (u_i - v_i + 1)e^{v_i}\right]. \tag{10}$$

The Kullback-Leibler is perhaps the best known Bregman divergence; it is also referred to as the *information divergence*, *information gain*, *relative entropy*; see [1, page 57]. If applied to finite distributions, F would be defined on the standard $(n-1)$-simplex, where it measures the expected number of extra bits required to code samples from x using a code that is optimized for y instead of for x. Since the $(n-1)$-simplex is the intersection of $\mathbb{R}^n_+$ with a hyperplane, this restriction of F is again of Legendre type. In this particular case, we can extend the function to the *closed* $(n-1)$-simplex, so that some coordinates may be zero, provided we accept infinite divergences for some pairs. Importantly, our constructions will not use the divergence directly, circumventing the problem with infinite divergences. As explained in Section 4, we will use the radius of first intersection of balls, which is always finite for the Kullback-Leibler divergence. Consequently, the framework is suitable also for sparse data, pervasive for example in text-retrieval applications.

The *Burg entropy* maps a point $x \in \mathbb{R}^n_+$ to $F(x) = \sum_{i=1}^n [1 - \ln x_i]$. The components of the gradient are $-1/x_i$, for $1 \le i \le n$. The conjugate is the function $F^* \colon \mathbb{R}^n_- \to \mathbb{R}$ defined by $F^*(u) = \sum_{i=1}^n [1 - \ln |u_i|]$. Associated with F is the *Itakura-Saito divergence*:

$$D_F(x\|y) = \sum_{i=1}^n \left[\tfrac{x_i}{y_i} - \ln \tfrac{x_i}{y_i} - 1\right]. \tag{11}$$

We note that F and F^* are very similar, but their domains are diagonally opposite orthants. Indeed, the Itakura-Saito distance is not symmetric and generates non-convex primal balls; see Figure 4.

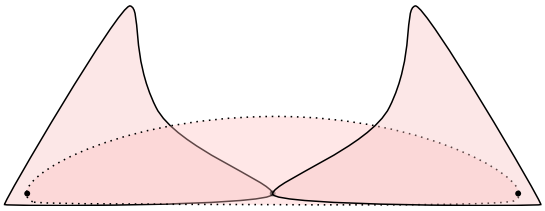


Figure 4 Two primal Itakura-Saito balls and the dual Itakura-Saito ball centered at the point where the primal balls touch. Its boundary passes through the centers of the primal balls.

3 Proximity Complexes for Bregman divergences

In this section, we extend the standard constructions of topological data analysis (Čech, Vietoris-Rips, Delaunay complexes) to the setting of Bregman divergences. Importantly, we prove the contractibility of non-empty common intersections of Bregman balls and Voronoi domains. This property guarantees that the Čech and Delaunay complexes capture the correct homotopy type of the data.

Contractibility for balls. Every non-empty convex set is contractible, which means it has the homotopy type of a point. The common intersection of two or more convex sets is either empty or again convex and therefore contractible. While primal Bregman balls are not necessarily convex, we show that their common intersections are contractible unless empty. The reason for our interest in this property is the Nerve Theorem [7, 24], which asserts that the nerve of a cover with said property has the same homotopy type as the union of this cover.

▶ **Result 3** (Contractibility Lemma for Balls). *Let $F\colon \Omega \to \mathbb{R}$ be of Legendre type, $X \subseteq \Omega$, and $r \geq 0$. Then $\bigcap_{x\in X} B_F(x;r)$ is either empty or contractible.*

Proof. Recall the homeomorphism $h\colon \Omega \to \Omega^*$ obtained as a side-effect of applying the Legendre transform to F. It maps every primal Bregman ball in Ω homeomorphically to a dual Bregman ball in Ω^*, which is convex. Similarly, it maps the common intersection of primal Bregman balls in Ω homeomorphically to the common intersection of dual Bregman balls in Ω^*: $h(\mathbb{X}) = \mathbb{Y}$ in which $\mathbb{X} = \bigcap_{x\in X} B_F(x;r)$ and $\mathbb{Y} = \bigcap_{x\in X} B'_{F^*}(x^*;r)$. Since $\mathbb{X}$ and $\mathbb{Y}$ are homeomorphic, they have the same homotopy type. Hence, either $\mathbb{X} = \mathbb{Y} = \emptyset$ or $\mathbb{Y}$ is convex and $\mathbb{X}$ is contractible. ◀

Čech and Vietoris-Rips constructions for Bregman divergences. The contractibility of the common intersection suggests we take the nerve of the Bregman balls. Given a finite set $X \subseteq \Omega$ and $r \geq 0$, we call the resulting simplicial complex the *Čech complex* of X and r associated with F. Related to it is the *Vietoris-Rips complex*, which is the clique complex of the 1-skeleton of the Čech complex:

$$\check{\mathrm{C}}\mathrm{ech}_F(X;r) = \{P \subseteq X \mid \bigcap_{p\in P} B_F(p;r) \neq \emptyset\}, \tag{12}$$

$$\mathrm{Rips}_F(X;r) = \{Q \subseteq X \mid \tbinom{Q}{2} \subseteq \check{\mathrm{C}}\mathrm{ech}_F(X;r)\}. \tag{13}$$

In words, the Vietoris-Rips complex contains a simplex iff all its edges belong to the Čech complex. We note that for $F(x) = \|x\|^2$, (13) translates to the usual Euclidean definition of the Vietoris-Rips complex. Increasing the radius from 0 to ∞, we get a filtration of Čech

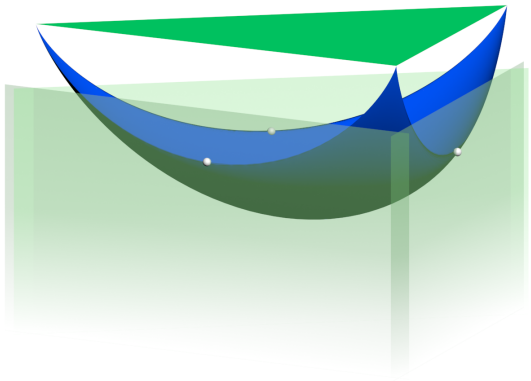


Figure 5 Three points for which pairwise intersecting dual Kullback-Leibler balls centered at these points can be small, but triplewise intersecting such balls are necessarily large.

complexes and a filtration of Vietoris-Rips complexes. By construction, the Čech complex is contained in the Vietoris-Rips complex for the same radius. If we measure distance with the Euclidean metric, this relation extends to

$$\check{\mathrm{C}}\mathrm{ech}(X;r) \subseteq \mathrm{Rips}(X;r) \subseteq \check{\mathrm{C}}\mathrm{ech}(X;\sqrt{2}r). \tag{14}$$

Indeed, if all pairs in a set of $k+1$ balls of radius r have a non-empty common intersection, then increasing the radius to $\sqrt{2}r$ guarantees that the $k+1$ balls have a non-empty intersection. This fact is often expressed by saying that the two filtrations have a small interleaving distance if indexed logarithmically.

No interleaving. The interleaving property expressed in (14) extends to general metrics – except that the constant factor is 2 rather than $\sqrt{2}$ – but not to general Bregman divergences. To see that (14) does not extend, we give an example of 3 points whose Bregman balls overlap pairwise for a small radius but not triplewise until the radius is very large.

The example uses the exponential function defined on the standard triangle, which we parametrize using barycentric coordinates. For convenience, the explanation uses the conjugate function, which is the Shannon entropy; that is: we look at dual balls in which distance is measured with the Kullback-Leibler divergence. Specifically, we use $F(x) = \sum_{i=1}^{3} x_i \ln x_i$. The barycentric coordinates are non-negative and satisfy $\sum_{i=1}^{3} x_i = 1$. We therefore get the maximum value of 0 at the three corners, and the minimum of $-\ln 3$ at the center of the triangle; see Figure 5. After some calculations, we get the squared length of the gradient at x as $\frac{1}{3}[(\ln x_1 - \ln x_2)^2 + (\ln x_1 - \ln x_3)^2 + (\ln x_2 - \ln x_3)^2]$. It goes to infinity when x approaches the boundary of the triangle.

We construct the example using points near the midpoints of the edges. Choosing them in the interior of the triangle but close to the boundary, the corresponding three tangent planes are as steep as we like. Moving the planes upward, we get the dual balls as the vertical projections of the parts of the graph of F on or below the planes. Moving the planes continuously, we let r be the height above the initial positions, and note that r is also the radius of the dual balls. Pairwise overlap between the balls starts when the three lines at which the planes meet intersect the graph of F. This happens at $r < \ln 3$. Triplewise overlap starts when the point common to all three planes passes through the graph of F. This happens at a value of r that we can make arbitrarily large.

Contractibility for Voronoi domains. Čech and Vietoris-Rips complexes can be high dimensional and of exponential size, even if the data lives in low dimensions. To remedy this shortcoming, we use the Delaunay (or alpha) complex; see [16, 17]. It is obtained by clipping the balls before taking the nerve. We explain this by introducing the Voronoi domains of the generating points as the clipping agents.

Letting $X \subseteq \Omega$ be finite, we define the *primal* and *dual Voronoi domains* of $x \in X$ associated with F as the sets of points for which x minimizes the Bregman divergence to or from the point:

$$V_F(x) = \{a \in \Omega \mid D_F(x\|a) \leq D_F(y\|a), \forall y \in X\}; \tag{15}$$
$$V'_F(x) = \{a \in \Omega \mid D_F(a\|x) \leq D_F(a\|y), \forall y \in X\}. \tag{16}$$

An intuitive construction of the primal domains grows the primal Bregman balls around the points, stopping the growth at places where the balls meet. Similarly, we get the dual Voronoi domains by growing dual Bregman balls. Not surprisingly, the primal Voronoi domains are not necessarily convex, and the dual Voronoi cells are convex. To see the latter property, we recall that the dual ball centered at x is constructed by translating the hyperplane that touches the graph of F above x. Specifically, $D_F(a\|x)$ is the height at which the hyperplane passes through the point $(a, F(a))$. This implies that we can construct the dual Voronoi domains as follows:

- For each $x \in X$, consider the half-space of points in $\mathbb{R}^n \times \mathbb{R}$ on or above the hyperplane that touches the graph of F at $(x, F(x))$.
- Form the intersection of these half-spaces, which is a convex polyhedron. We call its boundary the *upper envelope* of the hyperplanes, noting that it is the graph of a piecewise linear function from $\mathbb{R}^n$ to $\mathbb{R}$.
- Project the upper envelope vertically onto $\mathbb{R}^n$. Each dual Voronoi domain is the intersection of Ω with the image of an n-dimensional face of the upper envelope.

We conclude that the dual Voronoi domains are convex and use this property to show that the primal Voronoi domains intersect contractibly.

▶ **Result 4** (Contractibility Lemma for Voronoi Domains). *Let $F\colon \Omega \to \mathbb{R}$ be of Legendre type, and $X \subseteq \Omega$ finite. Then $\bigcap_{x\in X} V_F(x)$ is either empty or contractible.*

The proof is similar to that of the Contractibility Lemma for Balls and therefore omitted.

Delaunay construction for Bregman divergences. Taking the nerve of the primal Voronoi domains, we get the *Delaunay triangulation* of X associated with F, which we denote as $\mathrm{Del}_F(X)$. Further restricting the primal Voronoi domains by primal Bregman balls of radius r, we get the *Delaunay complex* of X and r associated with F:

$$\mathrm{Del}_F(X;r) = \{P \subseteq X \mid \bigcap_{p\in P} [B_F(p;r) \cap V_F(p)] \neq \emptyset\}. \tag{17}$$

Assuming general position of the points in X, the Delaunay triangulation is a simplicial complex of dimension at most n. We will be explicit about what we mean by general position shortly. Combining the proofs of the two Contractibility Lemmas, we see that the common intersection of any set of clipped primary balls is either empty or contractible. This together with the Nerve Theorem implies that $\mathrm{Del}_F(X;r)$ has the same homotopy type as $\check{\mathrm{C}}\mathrm{ech}_F(X;r)$, namely the homotopy type of the union of the Bregman balls that define the two complexes.

4 Algorithms

Recall that all three proximity complexes defined in Section 3 depend on a radius parameter. In this section, we give algorithms that compute the values of this parameter beyond which the simplices belong to the complexes. By focusing on the resulting radius functions, we decouple the computation of the radius for each simplex from the technicalities of constructing the actual simplicial complex. In particular, we show that the Čech complexes can be efficiently reconstructed from the Vietoris-Rips complexes, and the Delaunay complexes from the Delaunay triangulations. We exploit a connection with discrete Morse theory to develop efficient algorithms.

Radius functions. Let $X \subseteq \Omega$ be finite, write $\Delta(X)$ for the simplex whose vertices are the points in X, and recall that $\mathrm{Del}_F(X)$ is the Delaunay triangulation of X associated with F. The *Čech*, *Vietoris-Rips*, and *Delaunay radius functions* associated with F,

$$\varrho_F^{\check{\mathrm{C}}\mathrm{ech}} : \Delta(X) \to \mathbb{R}, \tag{18}$$

$$\varrho_F^{\mathrm{Rips}} : \Delta(X) \to \mathbb{R}, \tag{19}$$

$$\varrho_F^{\mathrm{Del}} : \mathrm{Del}_F(X) \to \mathbb{R}, \tag{20}$$

are defined such that $P \in \check{\mathrm{C}}\mathrm{ech}_F(X;r)$ iff $\varrho_F^{\check{\mathrm{C}}\mathrm{ech}}(P) \leq r$, and similarly for Vietoris-Rips and for Delaunay. By definition of the Čech complex, $\varrho_F^{\check{\mathrm{C}}\mathrm{ech}}(P)$ is the minimum radius at which the primal Bregman balls centered at the points of P have a non-empty common intersection. We are interested in an equivalent characterization using dual Bregman balls. To this end, we say that a dual Bregman ball, B', *includes* P if $P \subseteq B'$, and we call B' the *smallest including dual ball* if there is no other dual ball that includes P and has a smaller radius. Because F is strictly convex, the smallest including dual ball of P is unique; see Figure 4, which shows the smallest including dual Itakura-Saito ball of a pair of points. We call B' *empty* if no point of X lies in its interior, and we call it a *circumball* of P if all points of P lie on its boundary. We observe that a simplex $P \in \Delta(X)$ belongs to $\mathrm{Del}_F(X)$ iff it has an empty dual circumball. Because F is strictly convex, the smallest empty dual circumball of a simplex is either unique or does not exist. The characterization of the radius functions in terms of dual balls is strictly analogous to the Euclidean case studied in [3].

▶ **Result 5** (Radius Function Lemma). *Let $F : \Omega \to \mathbb{R}$ be of Legendre type, $X \subseteq \Omega$ finite, and $\emptyset \neq P \subseteq X$.*

I. *$\varrho_F^{\check{\mathrm{C}}\mathrm{ech}}(P)$ is the radius of the smallest including dual ball of P, and $\varrho_F^{\mathrm{Rips}}(P)$ is the maximum radius of the smallest including dual balls of the pairs in P.*

II. *If $P \in \mathrm{Del}_F(X)$, $\varrho_F^{\mathrm{Del}}(P)$ is the radius of the smallest empty dual circumball of P.*

We omit the proof, which is not difficult. Every circumball also includes, which implies that $\varrho_F^{\mathrm{Rips}}(P) \leq \varrho_F^{\check{\mathrm{C}}\mathrm{ech}}(P) \leq \varrho_F^{\mathrm{Del}}(P)$ whenever the radius functions are defined. Correspondingly, $\mathrm{Del}_F(X;r) \subseteq \check{\mathrm{C}}\mathrm{ech}_F(X;r) \subseteq \mathrm{Rips}_F(X;r)$ for every value of r.

General position. It is often convenient and sometimes necessary to assume that the points in $X \subseteq \Omega$ are in general position, for example when we require the Delaunay triangulation be a simplicial complex in $\mathbb{R}^n$. Here is a notion that suffices for the purposes of this paper.

▶ **Result 6** (Definition of General Position). *Let $\Omega \subseteq \mathbb{R}^n$ and $F : \Omega \to \mathbb{R}$ of Legendre type. A finite set $X \subseteq \Omega$ is in* general position *with respect to F if, for every $P \subseteq X$ of cardinality at most $n+1$,*

I. *the points in P are affinely independent,*

II. *no point of $X \setminus P$ lies on the boundary of the smallest dual circumball of P.*

Let $k = \dim P$. Property I implies that P has an $(n-k)$-parameter family of circumballs. In particular, there is at least one circumball as long as $k \leq n$. Property II implies that no two different simplices have the same smallest dual circumball. In particular, no two n-simplices in the Delaunay triangulation have the same circumball.

Discrete Morse theory. For points in general position, two of the radius functions exhibit a structural property that arises in the translation of Morse theoretic ideas from the smooth category to the simplicial category. Following [3], we extend the original formulation of discrete Morse theory given by Forman [19]. Letting K be a simplicial complex, and $P, R \in K$ two simplices, we write $P \leq R$ if P is a face of R. The *interval* of simplices between P and R is $[P, R] = \{Q \in K \mid P \leq Q \leq R\}$. We call P the *lower bound* and R the *upper bound* of the interval. A *generalized discrete vector field* is a partition of K into intervals. We call it a *generalized discrete gradient* if there exists a function $f\colon K \to \mathbb{R}$ such that $f(P) \leq f(Q)$ whenever P is a face of Q, with equality iff P and Q belong to a common interval. A function with this property is called a *generalized discrete Morse function.* To get an intuitive feeling for this concept, consider the sequence of sublevel sets of f. Any two contiguous sublevel sets differ by one or more intervals, and any two of these intervals are independent in the sense that neither interval contains a face of a simplex in the other interval. Indeed, this property characterizes generalized discrete Morse functions.

▶ **Result 7** (GDMF Theorem). *Let $F\colon \Omega \to \mathbb{R}$ be of Legendre type and let $X \subseteq \Omega$ be finite and in general position. Then $\varrho_F^{\check{\text{C}}\text{ech}}\colon \Delta(X) \to \mathbb{R}$ and $\varrho_F^{\text{Del}}\colon \text{Del}_F(X) \to \mathbb{R}$ are generalized discrete Morse functions. (We give a full proof in the Appendix of the full version of this paper.)*

Observe that the Vietoris-Rips radius function is not a generalized discrete Morse function. The structural properties implied by the GDMF Theorem will be useful in the design of algorithms that compute the radius functions. The theorem should be compared with the analogous result in the Euclidean case [3]. The arguments used there can be translated almost verbatim to prove additional structural results for Bregman divergences. Perhaps most importantly, they imply that the Wrap complex of F and X is well defined – see [15] for the original paper on these complexes defined in 3-dimensional Euclidean space – and that the Čech complex collapses to the Delaunay complex and further to the Wrap complex, all defined for the same radius.

Bregman circumball algorithm. Depending on how the function F is represented, there may be a numerical component to the algorithms needed to find smallest including dual balls. Consider a k-simplex $Q \subseteq X$ with $0 \leq k \leq n$. Assuming general position, the affine hull of the points $A = (a, F(a))$ with $a \in Q$ is a k-dimensional plane, which we denote as $\mathcal{Q}$. We are interested in the point $(q, \psi) \in \mathcal{Q}$ that maximizes $\psi - F(q)$, the height above the graph of F. The point q is the center of the smallest dual circumball of Q, and $\psi - F(q)$ is the radius. Interestingly, this observation implies that the point of first intersection of two primal Bregman balls lies on a line joining their centers. For later reference, we assume a routine that computes this point, possibly using a standard numerical optimization method.

dualball `routine` CircumBall (Q):
  let $\mathcal{Q}$ be the affine hull of the points $(a, F(a))$, $a \in Q$;
  find $(q, \psi) \in \mathcal{Q}$ maximizing $\psi - F(q)$;
  `return` $(q, \psi - F(q))$.

This is an unconstrained k-dimensional convex optimization, and k is much smaller than n for high dimensional data. Indeed, the optimization can be performed in the space of affine coordinates of the plane $\mathcal{Q}$. Importantly, the Hessian is of dimension $k \times k$ and not $n \times n$, which would be prohibitive. This allows us to use second-order quasi-Newton methods, such as the fast BFGS algorithm [27].

Note that the smallest dual circumball of Q includes Q but is not necessarily the smallest including dual ball. However, the latter is necessarily the smallest dual circumball of a face of Q. Next, we show how the CircumBall routine is used to efficiently compute the radius functions.

Čech radius function algorithm. According to the Radius Function Lemma (i), the value of a simplex, $Q \in \Delta(X)$, under the Čech radius function is the radius of the smallest including dual ball of Q. To compute this value, we visit the simplices in a particular sequence. Recalling the GDMF Theorem, we note that the smallest including dual ball of a simplex Q is the smallest dual circumball of the minimum face $P \subseteq Q$ in the same interval. It is therefore opportune to traverse the simplices in the order of increasing dimension. Whenever the smallest dual circumball of a simplex Q is not the smallest including dual ball, we get $\varrho_F^{\check{\text{C}}\text{ech}}(Q)$ from one of its codimension 1 faces. We identify such a simplex Q when we come across a face whose smallest dual circumball includes Q, and we mark Q with the center and radius of this ball. The following pseudocode computes the radius function of the Čech complex restricted to the k-skeleton of $\Delta(X)$ for some nonnegative integer k:

```
for i = 0 to k do
  forall P ⊆ X with dim P = i do
    if P unmarked then (p, r) = CircumBall(P);
    forall a ∈ X with D_F(a‖p) < r do mark P ∪ {a} with (p, r).
```

As in the Euclidean setting, the size of $\Delta(X)$ is exponential in the size of X so that the computations are feasible only for reasonably small values of k or small radius cut-offs. In practice, we would run the algorithm with a radius cut-off, or use an approximation strategy yielding a similar persistence diagram.

Observe the similarity to the standard algorithm for constructing the k-skeleton of the Vietoris-Rips complex: after adding all edges of length at most $2r$, we add simplices of dimension 2 and higher whenever possible. Geometric considerations are thus restricted to edges and the rest of the construction is combinatorial; see [33] for a fast implementation. Our algorithm can be interpreted as constructing the Čech complex from the Vietoris-Rips complex at the cost of at most one call to CircumBall per simplex. This is more efficient than explicitly computing the smallest *including* dual ball for each simplex, even if we use fast randomized algorithms as described in [26, 32]. Furthermore, the CircumBall routine is only called for the lower bounds of the intervals of the Čech radius function or, equivalently, for each subcomplex in the resulting filtration. The number of such intervals depends on the relative position of the points in X and not only on the cardinality. Notwithstanding, the number of intervals is significantly smaller than the number of simplices in the Čech complex. This suggests that only a small overhead is needed to compute the Čech from the Vietoris-Rips complexes. Our preliminary experiments for the Kullback-Leibler divergence support this claim; see Table 1. Note that the number of calls to the CircumBall routine is between $\frac{1}{10}$ and $\frac{1}{3}$ of the number of simplices, with an average between 6 and 15 function evaluations per call.

Table 1 Experimental evaluation on three synthetic datasets: (**A**) Full Čech complex with 20 points in $\mathbb{R}^{20}$; (**B**) 3-skeleton with 256 points in $\mathbb{R}^4$ and radius cutoff $r = 0.1$; (**C**) 4-skeleton with 4,000 points in $\mathbb{R}^4$ and radius cutoff $r = 0.01$.

	A (20 pts)	**B** (256 pts)	**C** (4,000 pts)
#edges	190	7,715	36,937
#simplices	1,048,575	1,155,301	1,222,688
#calls to CircumBall	104,030	346,475	283,622
#function evaluations in CircumBall	1,523,295	2,904,603	1,783,474

Delaunay radius function algorithm. According to the Radius Function Lemma (ii), the value of a simplex $Q \in \mathrm{Del}_F(X)$ under the Delaunay radius function is the radius of the smallest empty dual Bregman circumball of Q.

```
real routine DelaunayRadius (Q):
  (q, r) = CircumBall(Q);
  forall a ∈ X \ Q do
    if D_F(a‖q) < r then return none;
  return r.
```

The CircumBall routine gives only the smallest dual circumball of Q, and if it is not empty, then we have to get the value of the Delaunay radius function from somewhere else. According to the GDMF Theorem, we get the value from the maximum simplex in the interval that contains Q. It is therefore opportune to traverse the simplices of the Delaunay triangulation in the order of decreasing dimension. Whenever the smallest dual circumball of a simplex Q is non-empty, we get $\varrho_F^{\mathrm{Del}}(Q)$ from one of the simplices that contain Q as a codimension 1 face.

As already observed in [6], we can construct the full Delaunay triangulation, $\mathrm{Del}_F(X)$, using existing algorithms for the Euclidean case. We get the Delaunay complexes as sublevel sets of the radius function. Specifically, we first use the polarity transform to map the points $(x, F(x))$ to the corresponding affine functions; see Section 2. We then get a geometric realization of $\mathrm{Del}_F(X)$ from the vertical projection of the upper envelope of the affine functions onto $\mathbb{R}^n$, which is a *Euclidean weighted Voronoi diagram*, also known as *power diagram* or *Dirichlet tessellation*. Its dual is the *Euclidean weighted Delaunay triangulation*, also known as *regular* or *coherent triangulation*. The data that defines these Euclidean diagrams are the points $x \in X$ with weights $\xi = F(x) - \|x\|^2$. Finally, after computing the radius function on all simplices in $\mathrm{Del}_F(X)$, we get the Delaunay complexes as a filtration of this weighted Delaunay triangulation. Interestingly, this is not necessarily the filtration we obtain by simultaneously and uniformly increasing the weights of the points.

5 Discussion

The main contribution of this paper is the extension of the mathematical and computational machinery of topological data analysis (TDA) to applications in which distance is measured with a Bregman divergence. This includes text and image data often compared with the Kullback-Leibler divergence, and speech and sound data often studied with the Itakura-Saito divergence. It is our hope that the combination of Bregman divergences and TDA technology will bring light into the generally difficult study of high-dimensional data. In support of this optimism, Rieck and Leitte [29] provide experimental evidence that good dimension

reduction methods preserve the persistent homology of the data. With our extension to Bregman divergences, such experiments can now be performed for a much wider spectrum of applications. There are specific mathematical questions whose incomplete understanding is currently an obstacle to progress in the direction suggested by this paper:

- A cornerstone of TDA is the stability of its persistence diagrams, as originally proved in [12]. How does the use of Bregman divergences affect the stability of the diagrams?
- Related to the question of stability is the existence of sparse complexes and filtrations for data in Bregman spaces whose persistence diagrams are close to the ones we get for the Čech and Delaunay complexes.
- What about effective approximations of the introduced radius functions? In other words, are there simpler constructions yielding similar results, preferably using existing computational packages?

Acknowledgements The authors thank Žiga Virk for discussions on the material presented in this paper.

References

1. S. Amari and H. Nagaoka. *Methods of Information Geometry.* Amer. Math. Soc., Providence, Rhode Island, 2000.
2. A. Banerjee, S. Merugu, I.S. Dhillon and J. Ghosh. Clustering with Bregman divergences. *J. Mach. Learn. Res.* 6 (2005), 1705–1749.
3. U. Bauer and H. Edelsbrunner. The Morse theory of Čech and Delaunay complexes. *Trans. Amer. Math. Soc.*, 369 (2017), 3741–3762.
4. H. H. Bauschke and J. M. Borwein. Legendre functions and the method of random Bregman projections. *J. Convex Analysis* 4 (1997), 27–67.
5. B. Bigi. Using Kullback-Leibler distance for text categorization. *In* "Proc. 25th European Conf. Inform. Retrieval, 2003", LNCS 2633, 305–319.
6. J.-D. Boissonnat, F. Nielsen and R. Nock. Bregman Voronoi diagrams. *Discrete Comput. Geom.* 44 (2010), 281–307.
7. K. Borsuk. On the imbedding of systems of compacta in simplicial complexes. *Fund. Math.* 35 (1948), 217–234.
8. L. M. Bregman. The relaxation method of finding the common point of convex sets and its applications to the solution of problems in convex programming. *USSR Comput. Math. Math. Phys.* 7 (1967), 200–217.
9. G. Carlsson. Topology and data. *Bull. Amer. Math. Soc.* 46 (2009), 255–308.
10. J. M. Chan, G. Carlsson and R. Rabadan. Topology of viral evolution. *Proc. Natl. Acad. Sci.* 110 (2013), 18566–18571.
11. F. Chazal and S. Y. Oudot. Towards persistence-based reconstruction in Euclidean spaces. *In* "Proc. 24th Ann. Sympos. Comput. Geom., 2008", 232–241.
12. D. Cohen-Steiner, H. Edelsbrunner and J. L. Harer. Stability of persistence diagrams. *Discrete Comput. Geom.* 37 (2007), 103–120.
13. A. Dhara and J. Dutta. *Optimality Conditions in Convex Optimization: a Finite-dimensional View.* CRC Press, Taylor & Francis Group, Boca Raton, Florida, 2012.
14. M. N. Do and M. Vetterli. Wavelet-based texture retrieval using generalized Gaussian density and Kullback-Leibler distance. *IEEE Trans. Image Proc.* 11 (2002), 146–158.
15. H. Edelsbrunner. Surface reconstruction by wrapping finite point sets in space. In *Discrete and Computational Geometry. The Goodman-Pollack Festschrift*, 379–404, eds. B. Aronov, S. Basu, J. Pach and M. Sharir, Springer-Verlag, 2003.

16 H. Edelsbrunner and J. L. Harer. *Computational Topology. An Introduction.* Amer. Math. Soc., Providence, Rhode Island, 2010.

17 H. Edelsbrunner, E. P. Mücke. Three-dimensional alpha shapes. *ACM Trans. Graphics* 13 (1994), 43–72.

18 C. Févotte, N. Bertin and J.-L. Durrieu. Nonnegative matrix factorization with the Itakura-Saito divergence: with application to music analysis. *Neural Comput.* 21 (2009), 793–830.

19 R. Forman. Morse theory for cell complexes. *Adv. Math.* 134 (1998), 90–145.

20 A. Hatcher. *Algebraic Topology.* Cambridge Univ. Press, Cambridge, England, 2002.

21 A. Huang. Similarity measures for text document clustering. Proc. 6th New Zealand Computer Science Research Student Conference, 49–56, 2008.

22 F. Itakura and S. Saito. An analysis-synthesis telephony based on the maximum likelihood method. *In* "Proc. 6th Internat. Congress Acoustics, 1968", Tokyo, Japan, c17–c20.

23 S. Kullback and R. A. Leibler. On information and sufficiency. *Ann. Math. Stat.* 22 (1951), 79–86.

24 J. Leray. Sur la forme des espaces topologiques et sur les points fixes des représentations. *J. Math. Pures Appl.* 24 (1945), 95–167.

25 T. Nakamura, Y. Hiraoka, A. Hirata, E.G. Escolar and Y. Nishiura. Persistent homology and many-body atomic structure for medium-range order in the glass. *Nanotechnology*, 26 (2015), 304001.

26 F. Nielsen and R. Nock. On the smallest enclosing information disk. *Proc. 18th Canad. Conf. Comput. Geom.*, 2006.

27 J. Nocedal and S. Wright. *Numerical Optimization.* Springer Science and Business Media, 2006.

28 A. Port, I. Gheorghita, D. Guth, J. M. Clark, C. Liang, S. Dasu and M. Marcolli. Persistent topology of syntax. arXiv:1507.05134, 2015.

29 B. Rieck, H. Leitte. Persistent homology for the evaluation of dimensionality reduction schemes. *Computer Graphics Forum* 34 (2015), 431–440.

30 R. T. Rockafellar. *Convex Analysis.* Princeton University Press, Princeton, New Jersey, 1970.

31 T. Sousbie. The persistent cosic web and its filamentary structure–I. Theory and implementation. *Monthly Notices Royal Astro. Soc.* 414 (2011), 350–383.

32 E. Welzl. Smallest enclosing disks (balls and ellipsoids). In *New Results and New Trends in Computer Science*, H.À. Maurer (ed.), Springer, LNCS 555 (1991), 359–370.

33 A. Zomorodian. Fast construction of Vietoris-Rips complex. *Computer & Graphics* 34 (2010), 263–271.

Finding Small Hitting Sets in Infinite Range Spaces of Bounded VC-Dimension

Khaled Elbassioni

Masdar Institute of Science and Technology, Abu Dhabi, UAE
kelbassioni@masdar.ac.ae

Abstract

We consider the problem of finding a small hitting set in an *infinite* range space $\mathcal{F} = (Q, \mathcal{R})$ of bounded VC-dimension. We show that, under reasonably general assumptions, the infinite-dimensional convex relaxation can be solved (approximately) efficiently by multiplicative weight updates. As a consequence, we get an algorithm that finds, for any $\delta > 0$, a set of size $O(s_{\mathcal{F}}(z^*_{\mathcal{F}}))$ that hits $(1-\delta)$-fraction of $\mathcal{R}$ (with respect to a given measure) in time proportional to $\log(\frac{1}{\delta})$, where $s_{\mathcal{F}}(\frac{1}{\epsilon})$ is the size of the smallest ϵ-net the range space admits, and $z^*_{\mathcal{F}}$ is the value of the *fractional* optimal solution. This *exponentially* improves upon previous results which achieve the same approximation guarantees with running time proportional to $\text{poly}(\frac{1}{\delta})$. Our assumptions hold, for instance, in the case when the range space represents the *visibility* regions of a polygon in $\mathbb{R}^2$, giving thus a deterministic polynomial-time $O(\log z^*_{\mathcal{F}})$-approximation algorithm for guarding $(1-\delta)$-fraction of the area of any given simple polygon, with running time proportional to $\text{polylog}(\frac{1}{\delta})$.

1998 ACM Subject Classification G.1.6 Convex Programming, I.3.5 Geometric Algorithms, Languages, and Systems

Keywords and phrases VC-dimension, approximation algorithms, fractional covering, multiplicative weights update, art gallery problem, polyhedral separators, geometric covering

Digital Object Identifier 10.4230/LIPIcs.SoCG.2017.40

1 Introduction

Let $\mathcal{F} = (Q, \mathcal{R})$ be a range space defined by a (possibly) *infinite* set of ranges $\mathcal{R} \subseteq 2^Q$ over a (possibly) *infinite* set Q. A *hitting* set of $\mathcal{R}$ is a subset $H \subseteq Q$ such that $H \cap R \neq \emptyset$ for all $R \in \mathcal{R}$. Finding a hitting set of minimum size for a given range space is a fundamental problem in computational geometry. For finite range spaces (that is, when Q is finite), standard algorithms for SetCover [29, 33, 13] yield $(\log |Q|+1)$-approximation in polynomial time, and this is essentially the best possible guarantee assuming $P \neq NP$ [17]. Better approximation algorithms exist for special cases, such as range spaces of *bounded VC-dimension* [8], of *bounded union complexity* [15, 47], of *bounded shallow cell complexity* [9], as well as several classes of geometric range spaces [3, 40, 30]. Many of these results are based on showing the existence of a small-size *ϵ-net* for the range space $\mathcal{F}$ and then using the multiplicative weights update algorithm of Brönnimann and Goodrich [8]. For instance, if a range space $\mathcal{F}$ has VC-dimension d, then it admits an ϵ-net of size $O(\frac{d}{\epsilon}\log\frac{1}{\epsilon})$ [27, 31], which by the above mentioned method implies an $O(d \cdot \log \text{Opt}_{\mathcal{F}})$-approximation algorithm for the hitting set problem for $\mathcal{F}$, where $\text{Opt}_{\mathcal{F}}$ denotes the size of a minimum-size hitting set. Even et al. [19] observed that this can be improved to $O(d \cdot \log z^*_{\mathcal{F}})$-approximation by first solving the LP-relaxation of the problem to obtain the value of the *fractional* optimal solution $z^*_{\mathcal{F}}$, and then finding an ϵ-net, with $\epsilon := 1/z^*_{\mathcal{F}}$.

The multiplicative weights update algorithm in [8] works by maintaining weights on the *points*. The straightforward extension to infinite (or continuous) range spaces (that is, the case when Q is infinite) does not seem to work, since the bound on the number of iterations depends on the measure of the regions created during the course of the algorithm, which can be arbitrarily small. In this paper we take a different approach, which can be thought of as a combination of the methods in [8] and [19] (with LP replaced by an *infinite-dimensional convex relaxation*[1]):

- We maintain weights on the *ranges* (in contrast to the method of Brönnimann and Goodrich [8] which maintains weights on the points, and the second method suggested by Agarwal and Pan [1] which maintains weights on both points and ranges);
- We first solve the covering convex relaxation within a factor of $1+\varepsilon$ using the multiplicative weights update (MWU) methoid, extending the approach in [21] to infinite-dimensional covering LP's (under reasonable assumptions);
- We finally use the rounding idea of [19] to get a small integral hitting set from the obtained fractional solution.

Informal main theorem. There is an algorithm that, given a range space $\mathcal{F} = (Q, \mathcal{R})$ of VC-dimension d and $\delta > 0$, (under mild assumptions) finds a subset of Q of size $O(d \cdot z^*_{\mathcal{F}} \log z^*_{\mathcal{F}})$ that hits $(1-\delta)$-fraction of $\mathcal{R}$ (with respect to a given measure) in time polynomial in the input description of $\mathcal{F}$ and $\log(\frac{1}{\delta})$.

This *exponentially* improves upon previous results[2] which achieve the same approximation guarantees, but with running time depending *polynomially* on $\frac{1}{\delta}$. It should be noted that the main contribution of this paper is an efficient algorithm for approximately solving the *infinite-dimensional* covering linear programming relaxation with an *infinite* number of constraints. Even though such relaxation is convex, none of the known polynomial-time methods for convex programming (such as the ellipsoid method and interior point methods) can be used since their running time depends polynomially on the dimension. For the class of infinite-dimensional covering linear programs with infinitely many constraints corresponding to the ranges of a range space, we observe an interesting connection between the convergence of the MWU method and the fact that the range space has bounded VC-dimension.

We apply this result to a number of problems:

- The art gallery problem: given a simple polygon H, our main theorem implies that there is a *deterministic* polytime $O(\log z^*_{\mathcal{F}})$-approximation algorithm (with running time proportional to polylog$(\frac{1}{\delta})$) for guarding $(1-\delta)$-fraction of the area of H. When δ is (exponentially) small, this improves upon a previous result [12] which gives a polytime algorithm that finds a set of size $O(\textsc{Opt}_{\mathcal{F}} \cdot \log \frac{1}{\delta})$, guarding $(1-\delta)$-fraction of the area of H. Other (*randomized*) $O(\log \textsc{Opt}_{\mathcal{F}})$-approximation results which provide full guarding (i.e., $\delta = 0$) also exist, but they either run in pseudo-polynomial time [16], restrict the set of candidate guard locations [18, 22], or make some general position assumptions [6].
- Covering a polygonal region by translates of a convex polygon: Given a collection $\mathcal{H}$ of polygons in the plane and a convex polygon H_0, our main theorem implies that there is a randomized polytime $O(1)$-approximation algorithm for covering $(1-\delta)$ of the total area

[1] More precisely, an *infinite-dimensional* covering linear programming relaxation with an *infinite* number of constraints

[2] More precisely (as pointed to us by an anonymous reviewer), using relative approximation results (see, e.g., [26]), one can obtain the same approximation guarantees as our main Theorem by solving the problem on the set system induced on samples of size $O((d \cdot \textsc{Opt}_{\mathcal{F}}/\delta)\log(1/\delta))$

of the polygons in $\mathcal{H}$ by the minimum number of translates of H_0. Previous results with proved approximation guarantees mostly consider only the case when $\mathcal{H}$ is a set of points [15, 28, 32].

- Polyhedral separation in fixed dimension: Given two convex polytopes $\mathcal{P}_1, \mathcal{P}_2 \subseteq \mathbb{R}^d$ such that $\mathcal{P}_1 \subset \mathcal{P}_2$, our main theorem implies that there is a randomized polytime $O(d \cdot \log z^*_{\mathcal{F}})$-approximation algorithm for finding a polytope $\mathcal{P}_3$ with the minimum number of facets separating $\mathcal{P}_1$ from $(1-\delta)$-fraction of the volume of $\partial\mathcal{P}_2$. This improves the approximation ratio by a factor of d over the previous (deterministic) result [8] (but which gives a complete separation).

The paper is organized as follows. In the next section we define our notation, recall some preliminaries, and describe the infinite-dimensional convex relaxation. In Section 3, we state our main result, followed by the algorithm for solving the fractional problem in Section 4 and its analysis in Section 5. The success of the whole algorithm relies crucially on being able to efficiently implement the so-called *maximization oracle*, which essentially calls for finding, for a given measure on the ranges, a point that is contained in the heaviest subset of ranges (with respect to the given measure). We utilize the fact that the dual range space has bounded VC-dimension in Section 6 to give an efficient randomized implementation of the maximization oracle in the *Real RAM* model of computation. With more work, we show in fact that, in the case of the art gallery problem, the maximization oracle can be implemented in *deterministic* polynomial time in the *bit model*. Sections 7.2 and 7.3 describe the two other applications.

2 Preliminaries

2.1 Notation

Let $\mathcal{F} = (Q, \mathcal{R})$ be a range space. For a point $q \in Q$ and a subset of ranges $\mathcal{R}' \subseteq \mathcal{R}$, let $\mathcal{R}'[q] := \{R \in \mathcal{R}' : \ q \in R\}$. The *dual* range space $\mathcal{F}^* = (Q^*, \mathcal{R}^*)$ is defined as the range space with $Q^* := \mathcal{R}$ and $\mathcal{R}^* := \{\mathcal{R}[q] : \ q \in Q\}$. For a set of points $P \subseteq Q$, let $\mathcal{R}_{|P} := \{R \cap P \ : \ R \in \mathcal{R}\}$ be the *projection* of $\mathcal{R}$ onto P. Similarly, for a set of ranges $\mathcal{R}' \subseteq \mathcal{R}$, let $\mathcal{R}^*_{|\mathcal{R}'} := \{\mathcal{R}'[q] : \ q \in Q\}$. For a positive integer r, we denote by $g_{\mathcal{F}}(r) \leq 2^r$ the smallest integer such that for every finite set $P \subseteq Q$ of size r, we have $|\mathcal{R}_{|P}| \leq g_{\mathcal{F}}(r)$. For $p \in Q$ and $R \in \mathcal{R}$, we denote by $\mathbb{1}_{p \in R} \in \{0, 1\}$ the indicator variable that takes value 1 if and only if $p \in R$.

2.2 Assumptions

We shall make the following assumptions:

- **(A1)** $g_{\mathcal{F}}(r) \leq cr^{\gamma}$, for all integers $r \geq 0$ and some constants $\gamma \geq 1$ and $c > 0$ (known to the algorithm).
- **(A2)** There exists a finite integral optimum whose value $\textsc{Opt}_{\mathcal{F}}$ is bounded by a parameter n (that is not necessarily part of the input description).
- **(A3)** There exists an integrable function $w_0 : \mathcal{R} \to \mathbb{R}_+$. We assume that the integration of w_0 over any subset of $\mathcal{R}$ of input description of size k can be computed in time poly(k).

Note that if assumption (A3) holds then w_0 naturally defines a finite measure on $\mathcal{R}$ where the measure for a (measurable) set $\mathcal{R}' \subseteq \mathcal{R}$ is $w_0(\mathcal{R}') := \int_{R \in \mathcal{R}'} w_0(R) dR$. [3]

[3] One may also consider a general measure on $\mathcal{R}$. For simplicity of presentation, we assume here the more restrictive condition (A3), as all the applications we consider have this restriction. However, the

2.3 Range spaces of bounded VC-dimension

We consider range spaces $\mathcal{F} = (Q, \mathcal{R})$ of bounded *VC-dimension* defined as follows. A finite set $P \subseteq Q$ is said to be *shattered* by $\mathcal{F}$ if $\mathcal{R}_{|P} = 2^P$. The VC-dimension of $\mathcal{F}$, denoted VC-dim($\mathcal{F}$), is the cardinality of the largest subset of Q shattered by $\mathcal{F}$. If arbitrarily large subsets of Q can be shattered then VC-dim($\mathcal{F}$) $= +\infty$. It is well-known [42, 43] that if VC-dim($\mathcal{F}$) $= d$ then $g_{\mathcal{F}}(r) \leq g(r,d) := \sum_{i=0}^{d} \binom{r}{i} = O(r^d)$, and that VC-dim($\mathcal{F}^*$) $< 2^{d+1}$. Thus, if VC-dim($\mathcal{F}$) $= d$ then Assumption (A1) is satisfied with $\gamma = d$.

2.4 ϵ-nets

Given a range space $(Q, \mathcal{R})$, an integrable function $\mu : Q \to \mathbb{R}_+$, and a parameter $\epsilon > 0$, an *ϵ-net* for $\mathcal{R}$ (w.r.t. μ) is a set $P \subseteq Q$ such that $P \cap R \neq \emptyset$ for all $R \in \mathcal{R}$ that satisfy $\mu(R) \geq \epsilon \cdot \mu(Q)$, where for $Q' \subseteq Q$, we write $\mu(Q') := \int_{q \in Q'} \mu(q) dq$. We say that a range space $\mathcal{F}$ admits an ϵ-net of size $s_{\mathcal{F}}(\cdot)$, if for any $\epsilon > 0$, there is an ϵ-net of size $s_{\mathcal{F}}(\frac{1}{\epsilon})$. For range spaces of VC-dimension d, it is known [27, 31] that a random sample (w.r.t. to the probability density function $\frac{\mu}{\mu(Q)}$) of size $s_{\mathcal{F}}(\frac{1}{\epsilon}) = O(\frac{d}{\epsilon} \log \frac{1}{\epsilon})$ is an ϵ-net with (high) probability $\Omega(1)$.

We say that $\mu : Q \to \mathbb{R}_+$ has (finite) support of size K if μ can be written as a conic combination of K *Dirac delta functions*[4]: for any $q \in Q$, $\mu(q) := \sum_{p \in P} \mu(p) \boldsymbol{\delta}_p(q)$, for some finite $P \subseteq Q$ of cardinality K and non-negative multipliers $\mu(p)$, for $p \in P$. If this is the case, an ϵ-net for $\mathcal{R}$ of size $s_{\mathcal{F}}(\frac{1}{\epsilon}) = O(\frac{d}{\epsilon} \log \frac{d}{\epsilon})$ can be computed deterministically in time $O(d)^{3d} \frac{1}{\epsilon^{2d}} \log^d(\frac{d}{\epsilon}) K$ under the following assumption [7, 11, 34]:

(A1′) The range space is given by a *subsystem oracle* SUBSYS($\mathcal{F}, P$) that, given any finite $P \subseteq Q$, returns the set of ranges $\mathcal{R}_{|P}$ in time $O(|P|)^{d+1}$.

It should also be noted that some special range spaces may admit a smaller size ϵ-net, e.g., $s_{\mathcal{F}}(\frac{1}{\epsilon}) = O(\frac{1}{\epsilon})$ for half-spaces in $\mathbb{R}^3$ [35, 36]; see also [9, 30, 15, 47].

2.5 ϵ-approximations

Given the dual range space $\mathcal{F}^*$, a probability density function w on $\mathcal{R}$, and an $\epsilon > 0$, an ϵ-approximation is a finite subset of ranges $\mathcal{R}' \subseteq \mathcal{R}$ such that, for all $q \in Q$,

$$\left| \frac{|\mathcal{R}'[q]|}{|\mathcal{R}'|} - \frac{w(\mathcal{R}[q])}{w(\mathcal{R})} \right| \leq \epsilon; \tag{1}$$

see, e.g., [10]. It is known [2, 10, 46] that if $\mathcal{F} = (Q, \mathcal{R})$ is a range space of VC-dimension d, and w is an arbitrary probability density function on $\mathcal{R}$, then for any $\epsilon > 0$, a random sample (w.r.t. w) of size $O(\frac{d2^d}{\epsilon^2} \log \frac{1}{\epsilon\sigma})$ is an ϵ-approximation for $\mathcal{F}^*$, with probability $1 - \sigma$.

2.6 The fractional problem

Given a range space $\mathcal{F} = (Q, \mathcal{R})$, satisfying assumptions (A1)–(A3), the fractional covering problem for $\mathcal{F}$ seeks to find an integrable function $\mu : Q \to \mathbb{R}_+$, such that $\mu(R) \geq 1$ for all

extension to general measurable sets should be straightforward.

[4] A Dirac delta function satisfies $\int_{Q'} \boldsymbol{\delta}_p(q) dq = 1$ if $p \in Q'$, and $\int_{Q'} \boldsymbol{\delta}_p(q) dq = 0$, for any $Q' \subseteq Q$.

$R \in \mathcal{R}$ and $\mu(Q)$ is minimized[5]:

$$z^*_{\mathcal{F}} := \inf_{\text{integrable } \mu} \int_{q\in Q} \mu(q)dq \qquad \text{(F-HITTING)}$$

$$\text{s.t.} \int_{q\in R} \mu(q)dq \geq 1, \qquad \forall R \in \mathcal{R}, \qquad (2)$$

$$\mu(q) \geq 0, \qquad \forall q \in Q.$$

Equivalently, it is required to find a *probability* measure μ on Q that solves the maximin problem: $\sup_\mu \inf_{R\in\mathcal{R}} \mu(R)$.

▶ **Proposition 1.** *For a range space $\mathcal{F}$ satisfying (A2), we have $\text{OPT}_{\mathcal{F}} \geq z^*_{\mathcal{F}}$.*

Proof. Given a finite integral optimal solution P^*, we define an integrable function $\mu : Q \to \mathbb{R}_+$ of support size $\text{OPT}_{\mathcal{F}}$ by $\mu(q) := \sum_{p\in P^*} \boldsymbol{\delta}_p(q)$, for $q \in Q$. Then $\mu(Q) = \int_{q\in Q} \sum_{p\in P^*} \boldsymbol{\delta}_p(q)dq = \sum_{p\in P^*} \int_{q\in Q} \boldsymbol{\delta}_p(q)dq = \sum_{p\in P^*} 1 = |P^*| = \text{OPT}_{\mathcal{F}}$ and $\mu(R) = \int_{q\in R} \sum_{p\in P^*} \boldsymbol{\delta}_p(q)dq = \sum_{p\in P^*} \int_{q\in R} \boldsymbol{\delta}_p(q)dq = \sum_{p\in P^*} \mathbb{1}_{p\in R} = |\{p \in P^* : p \in R\}| \geq 1$, for all $R \in \mathcal{R}$, since P^* is a hitting set. Since μ is feasible for (F-HITTING), the claim follows. ◀

Assume $\mathcal{F}$ satisfies (A3). For $\alpha \geq 1$, we say that $\mu : Q \to \mathbb{R}_+$ is an *α-approximate* solution for (F-HITTING) if μ is feasible for (F-HITTING) and $\mu(Q) \leq \alpha \cdot z^*_{\mathcal{F}}$. For $\beta \in [0, 1]$, we say that μ is β-feasible if $\mu(R) \geq 1$ for all $R \in \mathcal{R}'$, where $\mathcal{R}' \subseteq \mathcal{R}$ satisfies $w_0(\mathcal{R}') \geq \beta \cdot w_0(\mathcal{R})$. Finally, we say that μ is an (α, β)-approximate solution for (F-HITTING) if μ is both α-approximate and β-feasible.

2.7 Rounding the fractional solution

Brönnimann and Goodrich [8] gave a multiplicative weights update algorithm for approximating the minimum hitting set for a *finite* range space satisfying (A1′) and admitting an ϵ-net of size $s_{\mathcal{F}}(\frac{1}{\epsilon})$. Their algorithm works as follows. It first guesses the value of the optimal solution (within a factor of 2), and initializes the weights of all *points* to 1. It then finds an $\epsilon = \frac{1}{2\text{OPT}_{\mathcal{F}}}$-net of size $s_{\mathcal{F}}(\frac{1}{\epsilon})$. If there is a range R that is not hit by the net (which can be checked by the subsystem oracle), the weights of all the points in R are doubled. The process is shown to terminate in $O(\text{OPT}_{\mathcal{F}} \log \frac{|Q|}{\text{OPT}_{\mathcal{F}}})$ iterations, giving an $s_{\mathcal{F}}(2\text{OPT}_{\mathcal{F}})/\text{OPT}_{\mathcal{F}}$-approximation. Even et al. [19] strengthen this result by using the linear programming relaxation to get $s_{\mathcal{F}}(z^*_{\mathcal{F}})/z^*_{\mathcal{F}}$-approximation. We can restate this result as follows.

▶ **Lemma 2.** *Let $\mathcal{F} = (Q, \mathcal{R})$ be a range space admitting an ϵ-net of size $s_{\mathcal{F}}(\frac{1}{\epsilon})$ and μ be a measure on Q satisfying (2). Then there is a hitting set for $\mathcal{R}$ of size $s_{\mathcal{F}}(\mu(Q))$.*

Proof. Let $\epsilon := \frac{1}{\mu(Q)}$. Then for all $R \in \mathcal{R}$ we have $\mu(R) \geq 1 = \epsilon \cdot \mu(Q)$, and hence an ϵ-net for $\mathcal{R}$ is actually a hitting set. ◀

▶ **Corollary 3.** *Let $\mathcal{F} = (Q, \mathcal{R})$ be a range space of VC-dimension d and $\mu : Q \to \mathbb{R}_+$ be an integrable function satisfying (2). Then a random sample of size $O(d \cdot \mu(Q) \log(\mu(Q)))$, w.r.t. the probability density function $\mu' := \frac{\mu}{\mu(Q)}$, is a hitting set for $\mathcal{R}$ with probability $\Omega(1)$. Furthermore, if μ has support size K then there is a deterministic algorithm that computes a hitting set for $\mathcal{R}$ of size $O(d \cdot \mu(Q) \log(d \cdot \mu(Q)))$ in time $O(d)^{3d}\mu(Q)^{2d} \log^d(d \cdot \mu(Q))K$.*

Further improvements on the Brönnimann-Goodrich algorithm can be found in [1].

[5] We may as well restrict μ to have finite support and replace the integrals over Q by summations.

3 Solving the fractional problem – Main result

We shall make the following further assumption:

(A4) There is a deterministic (resp., randomized) oracle $\textsc{Max}(\mathcal{F}, w, \nu)$ (resp., $\textsc{Max}(\mathcal{F}, w, \sigma, \nu)$), that given a range space $\mathcal{F} = (Q, \mathcal{R})$, an integrable function $w : \mathcal{R} \to \mathbb{R}_+$, and $\nu > 0$, returns (resp., with probability $1-\sigma$) a point $p \in Q$ such that $\xi_w(p) \geq (1-\nu)\max_{q \in Q} \xi_w(q)$, where $\xi_w(p) := w(\mathcal{R}[p]) = \int_{R \in \mathcal{R}} w(R)\mathbb{1}_{p \in R} dR$.

The following is the main result of the paper.

▶ **Theorem 4.** *Given a range space $\mathcal{F}$ satisfying (A1)–(A4), and $\varepsilon, \delta, \nu \in (0,1)$, there is a deterministic (resp., randomized) algorithm that finds (resp., with probability $\Omega(1)$) a function $\mu : Q \to \mathbb{R}_+$ of support size $K := O(\frac{\gamma}{\varepsilon^3(1-\nu)} \log \frac{\gamma}{\varepsilon} \cdot \textsc{Opt}_{\mathcal{F}} \log \frac{\textsc{Opt}_{\mathcal{F}}}{\varepsilon\delta(1-\nu)})$ that is a $(\frac{1+5\varepsilon}{1-\nu}, 1-\delta)$-approximate solution for (*F-HITTING*), using K calls to the oracle $\textsc{Max}(\mathcal{F}, w, \nu)$ (resp., $\textsc{Max}(\mathcal{F}, w, \sigma, \nu)$).*

In view of Corollary 3, we get the following theorem as an immediate consequence of Theorem 4.

▶ **Theorem 5** (Main Theorem)**.** *Let $\mathcal{F} = (Q, \mathcal{R})$ be a range space satisfying (A1)–(A4) and admitting a hitting set of size $s_{\mathcal{F}}(\frac{1}{\epsilon})$, and $\varepsilon, \delta, \nu \in (0,1)$ be given parameters. Then there is a (deterministic) algorithm that computes a set of size $s_{\mathcal{F}}(z^*_{\mathcal{F}})$, hitting a subset of $\mathcal{R}$ of measure at least $(1-\delta)w_0(\mathcal{R})$, using $O(\frac{\gamma}{\varepsilon^3(1-\nu)} \log \frac{\gamma}{\varepsilon} \cdot \textsc{Opt}_{\mathcal{F}} \log \frac{\textsc{Opt}_{\mathcal{F}}}{\varepsilon\delta(1-\nu)})$ calls to the oracle $\textsc{Max}(\ldots, \nu)$ and a single call to an ϵ-net finder.*

In Section 6, we observe that the maximization oracle can be implemented in randomized polynomial time. As a consequence, we obtain the following corollary of Theorem 5, under the assumption of the availability of the following oracles:

- $\textsc{Subsys}(\mathcal{F}^*, \mathcal{R}')$: this is the dual subsystem oracle; given a finite subset of ranges $\mathcal{R}' \subseteq \mathcal{R}$, it returns the set of ranges $\mathcal{R}^*_{|\mathcal{R}'}$. Note that $|\mathcal{R}^*_{|\mathcal{R}'}| \leq g(|\mathcal{R}'|, 2^{d+1})$.
- $\textsc{PointIn}(\mathcal{F}, \mathcal{R}')$: Given $\mathcal{F}$ and a finite subset of ranges $\mathcal{R}' \subseteq \mathcal{R}$, the oracle returns a point $p \in Q$ that lies in $\bigcap_{R \in \mathcal{R}'} R$ (if one exists).
- $\textsc{Sample}(\mathcal{F}, \widehat{w})$: Given $\mathcal{F} = (Q, \mathcal{R})$ and a probability density function $\hat{w} : \mathcal{R} \to \mathbb{R}_+$, it samples a range $R \in \mathcal{R}$ according to $\widehat{w}$.

▶ **Corollary 6.** *Let $\mathcal{F} = (Q, \mathcal{R})$ be a range space of VC-dimension d satisfying (A2) and (A3) and $\varepsilon, \delta \in (0,1)$ be given parameters. Then there is a randomized algorithm that computes a set of size $O(d \cdot z^*_{\mathcal{F}} \log z^*_{\mathcal{F}})$, hitting a subset of $\mathcal{R}$ of measure at least $(1-\delta)w_0(\mathcal{R})$, in time $O(K \cdot (\tau_1 \cdot N + \tau_2(N) + \tau_3(N) + g_{\mathcal{F}^*}(N)))$, where $K := O(\frac{d}{\varepsilon^3} \log \frac{d}{\varepsilon} \cdot \textsc{Opt}_{\mathcal{F}} \log \frac{\textsc{Opt}_{\mathcal{F}}}{\varepsilon\delta})$, and τ_1, $\tau_2(N)$ and $\tau_3(N)$, are respectively the maximum times taken by the oracle $\textsc{Sample}(\mathcal{F}, \widehat{w})$, and the oracles $\textsc{Subsys}(\mathcal{F}^*, \mathcal{R}')$, $\textsc{PointIn}(\mathcal{F}, \mathcal{R}')$ on a set $\mathcal{R}'$ of size $N := \frac{d2^d \textsc{Opt}^2_{\mathcal{F}}}{\varepsilon^2} \log \frac{\textsc{Opt}_{\mathcal{F}}}{\varepsilon}$.*

Note that $g_{\mathcal{F}^*}(r) \leq r^{2^{d+1}}$, but stronger bounds can be obtained for special cases.

4 The algorithm

The algorithm is shown in Algorithm 1 below. For any iteration t, let us define the *active* range-subspace $\mathcal{F}_t = (Q, \mathcal{R}_t)$ of $\mathcal{F}$, where

$$\mathcal{R}_t := \{R \in \mathcal{R} : |P_t \cap R| < T\}.$$

Data: A range space $\mathcal{F} = (Q, \mathcal{R})$ satsfying (A1)–(A4), and approximation accuracies $\varepsilon, \delta, \nu \in (0, 1)$.
Result: A $(\frac{1+5\varepsilon}{1-\nu}, 1-\delta)$-approximate solution μ for (F-HITTING).
1 $t \leftarrow 0$; $P_0 \leftarrow \emptyset$; set T as in (3)
2 **while** $w_0(\mathcal{R}_t) \geq \delta \cdot w_0(\mathcal{R})$ **do**
// `define`[7] $w_t : \mathcal{R}_t \to \mathbb{R}_+$ `by` $w_t(R) \leftarrow (1-\varepsilon)^{|P_t \cap R|} w_0(R)$, `for` $R \in \mathcal{R}_t$
3 $p_{t+1} \leftarrow \textsc{Max}(\mathcal{F}_t, w_t, \nu)$
4 $P_{t+1} \leftarrow P_t \cup \{p_{t+1}\}$
5 $t \leftarrow t+1$
6 **end**
7 **return** the measure $\widehat{\mu} : Q \to \mathbb{R}_+$ defined by $\widehat{\mu}(q) \leftarrow \frac{1}{T} \sum_{p \in P_t} \boldsymbol{\delta}_p(q)$

Algorithm 1: The fractional covering algorithm

Clearly, (since these properties are hereditary) VC-dim$(\mathcal{F}_t) \leq$ VC-dim$(\mathcal{F})$, and $\mathcal{F}_t$ admits an ϵ-net of size $s_{\mathcal{F}}(\frac{1}{\epsilon})$ whenever $\mathcal{F}$ does. For convenience, we assume below that P_t is (possibly) a multi-set (repetitions allowed).

Define[6] $a := \dfrac{\gamma}{\varepsilon^2}$, $b := \max\{\ln T_0, 1\}$, $T_0 := \dfrac{\textsc{Opt}_{\mathcal{F}} \delta^{1/\gamma} c^{1/\gamma}}{\varepsilon(1-\nu)\delta^{1/\gamma}} \left(\ln \dfrac{1}{1-\varepsilon} + \ln \dfrac{1}{\varepsilon\delta} \right)$, and

$$T := e^2 ab(\ln(a+e-1)+1) = \Theta\left(\frac{\gamma}{\varepsilon^2} \log \frac{\gamma}{\varepsilon} \log \frac{\textsc{Opt}_{\mathcal{F}}}{\varepsilon\delta(1-\nu)} \right). \tag{3}$$

For simplicity of presentation, we will assume in what follows that the maximization oracle is deterministic; the extension to the probabilistic case is straightforward.

5 Analysis

Define the potential function $\Phi(t) := w_t(\mathcal{R}_t)$, where $w_t(R) := (1-\varepsilon)^{|P_t \cap R|} w_0(R)$, and $P_t = \{p_{t'} : t' = 1, \ldots, t\}$ is the set of points selected by the algorithm in Step 3 up to time t. We can also write $w_{t+1}(R) = w_t(R)(1 - \varepsilon \cdot \mathbb{1}_{p_{t+1} \in R})$.

The analysis is done in three steps: the first one (Section 5.1), which is typical for MWU methods, is to bound the potential function, at each iteration, in terms of the ratio between the current solution obtained by the algorithm at that iteration and the optimum fractional solution. The second step (Section 5.2) is to bound the number of iterations until the desired fraction of the ranges is hit. Finally, the third step (Section 5.3) uses the previous two steps to show that the algorithm reaches the required accuracy after a polynomial number of iterations.

5.1 Bounding the potential

The following three lemmas are obtained by the standard analysis of MWU methods with "$\sum$"'s replaced by "$\int$"'s.

▶ **Lemma 7.** *For all $t = 0, 1, \ldots$, it holds that* $\Phi(t+1) \leq \Phi(t) \exp\left(-\frac{\varepsilon}{\Phi(t)} \cdot w_t(\mathcal{R}_t[p_{t+1}]) \right)$.

[6] When a polynomial upper bound n on $\textsc{Opt}_{\mathcal{F}}$ is known, as per Assumption (A2), we can use it instead of $\textsc{Opt}_{\mathcal{F}}$ in the formula for T_0; otherwise, exponential search can be used to estimate $\textsc{Opt}_{\mathcal{F}}$.

Proof.

$$\begin{aligned}\Phi(t+1) &= \int_{R\in\mathcal{R}_{t+1}} w_{t+1}(R)dR = \int_{R\in\mathcal{R}_{t+1}} w_t(R)(1-\varepsilon\cdot\mathbb{1}_{p_{t+1}\in R})dR \\ &\le \int_{R\in\mathcal{R}_t} w_t(R)(1-\varepsilon\cdot\mathbb{1}_{p_{t+1}\in R})dR = \Phi(t)\left(1-\varepsilon\int_{R\in\mathcal{R}_t}\mathbb{1}_{p_{t+1}\in R}\frac{w_t(R)}{\Phi(t)}dR\right) \\ &\le \Phi(t)\exp\left(-\varepsilon\int_{R\in\mathcal{R}_t}\mathbb{1}_{p_{t+1}\in R}\frac{w_t(R)}{\Phi(t)}dR\right),\end{aligned}$$

where the first inequality is because $\mathcal{R}_{t+1}\subseteq\mathcal{R}_t$ since $|P_t\cap R|$ is non-decreasing in t, and the last inequality is because $1-z\le e^{-z}$ for all z. ◀

▶ **Lemma 8.** *Let* $\kappa(t) := \sum_{t'=0}^{t-1}\frac{w_{t'}(\mathcal{R}_{t'}[p_{t'+1}])}{\Phi(t')}$. *Then* $z^*_{\mathcal{F}}\cdot\kappa(t)\ge\frac{1-\nu}{1+\varepsilon}|P_t|$.

Proof. Due to the choice of $p_{t'+1}$, we have that

$$\xi_{t'}(p_{t'+1}) := w_{t'}(\mathcal{R}_{t'}[p_{t'+1}]) \ge (1-\nu)\max_{q\in Q} w_{t'}(\mathcal{R}_{t'}[q]). \tag{4}$$

Consequently, for a $(1+\varepsilon)$-approximate solution μ^*,

$$\begin{aligned}z^*_{\mathcal{F}}\cdot\kappa(t) &= \sum_{t'=0}^{t-1} z^*_{\mathcal{F}}\frac{w_{t'}(\mathcal{R}_{t'}[p_{t'+1}])}{\Phi(t')} \ge \frac{1}{1+\varepsilon}\sum_{t'=0}^{t-1}\left(\int_{q\in Q}\mu^*(q)dq\right)\int_{R\in\mathcal{R}_{t'}}\mathbb{1}_{p_{t'+1}\in R}\frac{w_{t'}(R)}{\Phi(t')}dR \\ &\ge \frac{1-\nu}{1+\varepsilon}\sum_{t'=0}^{t-1}\int_{q\in Q}\mu^*(q)\int_{R\in\mathcal{R}_{t'}}\mathbb{1}_{q\in R}\frac{w_{t'}(R)}{\Phi(t')}dRdq \\ &= \frac{1-\nu}{1+\varepsilon}\sum_{t'=0}^{t-1}\int_{R\in\mathcal{R}_{t'}}\left(\int_{q\in Q}\mu^*(q)\mathbb{1}_{q\in R}dq\right)\frac{w_{t'}(R)}{\Phi(t')}dR \\ &= \frac{1-\nu}{1+\varepsilon}\sum_{t'=0}^{t-1}\int_{R\in\mathcal{R}_{t'}}\mu^*(R)\frac{w_{t'}(R)}{\Phi(t')}dR \ge \frac{1-\nu}{1+\varepsilon}\sum_{t'=0}^{t-1}\int_{R\in\mathcal{R}_{t'}}\frac{w_{t'}(R)}{\Phi(t')}dR \\ &= \frac{1-\nu}{1+\varepsilon}\sum_{t'=0}^{t-1}1 = \frac{1-\nu}{1+\varepsilon}|P_t|.\end{aligned}$$

where the first inequality is due to the $(1+\varepsilon)$-approximability of μ^*, the second inequality is due to (4), and the last inequality is due to the feasibility of μ^* for (F-HITTING). ◀

▶ **Lemma 9.** *For all* $t=0,1,\ldots,$ *we have*

$$\Phi(t)\le\Phi(0)\exp\left(-\varepsilon\cdot\frac{1-\nu}{1+\varepsilon}\cdot\frac{|P_t|}{z^*_{\mathcal{F}}}\right). \tag{5}$$

Proof. By repeated application of Lemma 7, and using the result in Lemma 8, we can deduce that

$$\begin{aligned}\Phi(t) &\le \Phi(0)\exp\left(-\sum_{t'=0}^{t-1}\frac{\varepsilon}{\Phi(t')}\cdot w_{t'}(\mathcal{R}_{t'}[p_{t'+1}])\right) = \Phi(0)\exp\left(-\varepsilon\kappa(t)\right) \\ &\le \Phi(0)\exp\left(-\varepsilon\cdot\frac{1-\nu}{1+\varepsilon}\cdot\frac{|P_t|}{z^*_{\mathcal{F}}}\right).\end{aligned}$$

◀

5.2 Bounding the number of iterations

▶ **Lemma 10.** *After at most* $t_{\max} := \frac{\text{OPT}_{\mathcal{F}}}{\varepsilon(1-\nu)}\left(T\ln\frac{1}{1-\varepsilon}+\ln\frac{1}{\varepsilon\delta}\right)$ *iterations, we have* $w_0(\mathcal{R}_{t_{\max}}) < \delta\cdot w_0(\mathcal{R})$.

Proof. For a range $R\in\mathcal{R}$, let us denote by $\mathcal{T}_t(R) := \{0\le t'\le t-1 :\ p_{t'+1}\in R\in\mathcal{R}_{t'}\}$ the set of time steps, up to t, at which R was hit by the selected point $p_{t'+1}$, when it was still active. Initialize $w'_0(R) := w_0(R)+\sum_{t'\in\mathcal{T}_t(R)} w_{t'+1}(R)$. For the purpose of the analysis, we will think of the following update step during the algorithm: upon choosing p_{t+1}, set $w'_{t+1}(R) := w'_t(R) - w_t(R)\mathbb{1}_{p_{t+1}\in R}$ for all $R\in\mathcal{R}_t$. Note that the above definition implies that $w'_t(R)\ge(1-\epsilon)^{|\mathcal{T}_t(R)|}w_0(R)$ for all $R\in\mathcal{R}$ and for all t.

▶ **Claim 11.** *For all* t, $w'_{t+1}(\mathcal{R}_{t+1})\le\left(1-\frac{\varepsilon(1-\nu)}{\text{OPT}_{\mathcal{F}}}\right)w'_t(\mathcal{R}_t)$.

Proof. Consider an integral optimal solution $P^*\subseteq Q$ (which is guaranteed to exist by (A2)). Then

$$w_t(\mathcal{R}_t)=\int_{R\in\mathcal{R}_t} w_t(R)dR = w_t\left(\bigcup_{q\in P^*}\mathcal{R}_t[q]\right)\le\sum_{q\in P^*} w_t(\mathcal{R}_t[q]). \tag{6}$$

From (6) it follows that there is a $q\in P^*$ such that $w_t(\mathcal{R}_t[q])\ge\frac{w_t(\mathcal{R}_t)}{\text{OPT}_{\mathcal{F}}}$. Note that for such q we have $\xi_t(q) := w_t(\mathcal{R}_t[q])\ge\frac{w_t(\mathcal{R}_t)}{\text{OPT}_{\mathcal{F}}}$, and thus by the choice of p_{t+1}, $\xi_t(p_{t+1})\ge(1-\nu)\xi_t(q)\ge\frac{(1-\nu)w_t(\mathcal{R}_t)}{\text{OPT}_{\mathcal{F}}}$. It follows that

$$\begin{aligned} w'_{t+1}(\mathcal{R}_{t+1})\le w'_{t+1}(\mathcal{R}_t) &= \int_{R\in\mathcal{R}_t}(w'_t(R)-w_t(R)\mathbb{1}_{p_{t+1}\in R})dR\\ &= \int_{R\in\mathcal{R}_t} w'_t(R)dR-\int_{R\in\mathcal{R}_t} w_t(R)\mathbb{1}_{p_{t+1}\in R}dR = w'_t(\mathcal{R}_t)-\xi_t(p_{t+1})\\ &\le w'_t(\mathcal{R}_t)-\frac{(1-\nu)w_t(\mathcal{R}_t)}{\text{OPT}_{\mathcal{F}}}. \end{aligned}\tag{7}$$

Note that, for all t,

$$w'_t(R)<w_t(R)\sum_{t'\ge0}(1-\varepsilon)^{t'}=\frac{w_t(R)}{\varepsilon}. \tag{8}$$

Thus, $w_t(\mathcal{R}_t)>\varepsilon\cdot w'_t(\mathcal{R}_t)$. Using this in (7), we get the claim. ◀

Claim 11 implies that, for $t=t_{\max}$, $w'_t(\mathcal{R}_t)\le\left(1-\frac{\varepsilon(1-\nu)}{\text{OPT}_{\mathcal{F}}}\right)^t w'_0(\mathcal{R})<e^{-\frac{\varepsilon(1-\nu)}{\text{OPT}_{\mathcal{F}}}t}w'_0(\mathcal{R})$. Since $|R\cap P_t|<T$ for all $R\in\mathcal{R}_t$, we have $|\mathcal{T}_t(R)|\le T$ and hence $w'_t(\mathcal{R}_t)=\int_{R\in\mathcal{R}_t}w'_t(R)dR\ge(1-\varepsilon)^T w_0(\mathcal{R}_t)$. On the other hand, (8) implies that $w'_0(\mathcal{R})<\frac{w_0(\mathcal{R})}{\varepsilon}$. Thus, if $w_0(\mathcal{R}_t)\ge\delta\cdot w_0(\mathcal{R})$, we get $(1-\varepsilon)^T\delta<\frac{1}{\varepsilon}\cdot e^{-\frac{\varepsilon(1-\nu)}{\text{OPT}_{\mathcal{F}}}t}$, giving $t<\frac{\text{OPT}_{\mathcal{F}}}{\varepsilon(1-\nu)}\left(T\ln\frac{1}{1-\varepsilon}+\ln\frac{1}{\varepsilon\delta}\right)=t_{\max}$, in contradiction to $t=t_{\max}$. ◀

5.3 Convergence to an $(\frac{1+5\varepsilon}{1-\nu}, 1-\delta)$-approximate solution

▶ **Lemma 12.** *Suppose that* $T\ge\frac{\max\{1,\ln(g_{\mathcal{F}}(t_{\max})/\delta)\}}{\varepsilon^2}$ *and* $\varepsilon\le 0.67$. *Then Algorithm 1 terminates with a* $(\frac{1+5\varepsilon}{1-\nu}, 1-\delta)$*-approximate solution* $\widehat{\mu}$ *for* (F-HITTING).

Proof. Suppose that Algorithm 1 (the while-loop) terminates in iteration $t_f \le t_{\max}$.
$(1-\delta)$-*Feasibility:* By the stopping criterion, $w_0(\mathcal{R}_{t_f}) < \delta \cdot w_0(\mathcal{R})$. Then for $t = t_f$ and any $R \in \mathcal{R} \setminus \mathcal{R}_t$, we have $\widehat{\mu}(R) = \frac{1}{T}\int_{q\in R}\sum_{p\in P_t}\boldsymbol{\delta}_p(q)dq = \frac{1}{T}\sum_{p\in P_t}\int_{q\in R}\boldsymbol{\delta}_p(q)dq = \frac{1}{T}\sum_{p\in P_t}\mathbb{1}_{p\in R} = \frac{1}{T}|P_t \cap R| \ge 1$, since $|P_t \cap R| \ge T$, for all $R \in \mathcal{R}\setminus\mathcal{R}_t$.
Quality of the solution $\widehat{\mu}$*:* We can write

$$\Phi(t) = \sum_{P\in(\mathcal{R}_t)_{|P_t}} (1-\varepsilon)^{|P|} w_0(\mathcal{R}_t[P]), \tag{9}$$

where $\mathcal{R}_t[P] := \{R \in \mathcal{R}_t : \ R \cap P_t = P\}$. Since $\Phi(t)$ satisfies (5), we get by (9) that

$$(1-\varepsilon)^{|P|} w_0(\mathcal{R}_t[P]) \le \Phi(0)\exp\left(-\varepsilon\cdot\frac{1-\nu}{1+\varepsilon}\cdot\frac{|P_t|}{z^*_{\mathcal{F}}}\right), \quad \text{for all } P \in (\mathcal{R}_t)_{|P_t}$$

$$\therefore |P|\ln(1-\varepsilon) + \ln(w_0(\mathcal{R}_t[P])) \le \ln\Phi(0) - \varepsilon\cdot\frac{1-\nu}{1+\varepsilon}\cdot\frac{|P_t|}{z^*_{\mathcal{F}}}, \quad \text{for all } P \in (\mathcal{R}_t)_{|P_t}.$$

Dividing by $\varepsilon\cdot\frac{1-\nu}{1+\varepsilon}\cdot T$ and rearranging, we get

$$\frac{|P_t|}{z^*_{\mathcal{F}}T} \le \frac{(1+\varepsilon)\left(\ln\Phi(0) - \ln(w_0(\mathcal{R}_t[P]))\right)}{\varepsilon(1-\nu)T} + \frac{(1+\varepsilon)|P|}{\varepsilon(1-\nu)T}\cdot\ln\frac{1}{1-\varepsilon}, \quad \text{for all } P \in (\mathcal{R}_t)_{|P_t}. \tag{10}$$

Since $w_0(\mathcal{R}_t) = w_0\left(\bigcup_{P\in(\mathcal{R}_t)_{|P_t}}\mathcal{R}_t[P]\right) = \sum_{P\in(\mathcal{R}_t)_{|P_t}} w_0\left(\mathcal{R}_t[P]\right)$, there is a set $\widehat{P} \in (\mathcal{R}_t)_{|P_t}$ such that $w_0(\mathcal{R}_t[\widehat{P}]) \ge \frac{w_0(\mathcal{R}_t)}{|(\mathcal{R}_t)_{|P_t}|}$.

We apply (10) for $t = t_f - 1$ and $\widehat{P} \in (\mathcal{R}_t)_{|P_t}$. By the definition of $g_{\mathcal{F}}(\cdot)$, we have $|(\mathcal{R}_t)_{|P_t}| \le g_{\mathcal{F}}(|P_t|) \le g_{\mathcal{F}}(t_{\max})$. Using $\Phi(0) = w_0(\mathcal{R}) \le \frac{w_0(\mathcal{R}_t)}{\delta}$, , $\widehat{\mu}(Q) = \frac{|P_t|+1}{T}$, $|\widehat{P}| < T$ (as $\widehat{P} = R \cap P_t$ for some $R \in \mathcal{R}_t$), $T \ge \frac{\ln(g_{\mathcal{F}}(t_{\max})/\delta)}{\varepsilon^2}$ and $T \ge \frac{1}{\varepsilon^2}$ (by assumption), and $z_{\mathcal{F}^*} \ge 1$, we get (for $\varepsilon \le 0.67$)

$$\begin{aligned}\frac{\widehat{\mu}(Q)}{z^*_{\mathcal{F}}} &\le \frac{(1+\varepsilon)\ln(g_{\mathcal{F}}(t_{\max})/\delta)}{\varepsilon(1-\nu)T} + \frac{(1+\varepsilon)}{\varepsilon(1-\nu)}\cdot\ln\frac{1}{1-\varepsilon} + \frac{1}{T\cdot z^*_{\mathcal{F}}}\\ &\le \frac{\varepsilon(1+\varepsilon)}{(1-\nu)} + \frac{(1+\varepsilon)}{\varepsilon(1-\nu)}\cdot\ln\frac{1}{1-\varepsilon} + \varepsilon^2 < \frac{1+5\varepsilon}{1-\nu}.\end{aligned}$$

◀

Finally one can verify that the choice of T in (3) satisfies the precondition in Lemma 12.

6 Implementation of the maximization oracle

Let $\mathcal{F} = (Q, \mathcal{R})$ be a range space with VC-dim$(\mathcal{F}) = d$. Recall that the maximization oracle needs to find, for a given $\nu > 0$ and function $w : \mathcal{R} \to \mathbb{R}_+$, a point $p \in Q$ such that $\xi_w(p) \ge (1-\nu)\max_{q\in Q}\xi_w(q)$, where $\xi_w(p) := w(\mathcal{R}[p])$.

To implement the maximization oracle, we follow the approach in [12], based on ϵ-approximations. Recall that an ϵ-approximation for $\mathcal{F}^*$ is a finite subset of ranges $\mathcal{R}' \subseteq \mathcal{R}$, such that (1) holds for all $q \in Q$. We use $\epsilon := \frac{\nu}{2\mathrm{OPT}_{\mathcal{F}}}$ and $\sigma = o(1)$, and take a random sample $\mathcal{R}'$ of size $N = O(\frac{d2^d}{\epsilon^2}\log\frac{1}{\epsilon\sigma}) = O(\frac{d2^d\mathrm{OPT}^2_{\mathcal{F}}}{\nu^2}\log\frac{\mathrm{OPT}_{\mathcal{F}}}{\nu\sigma})$ from $\mathcal{R}$ according to the probability density function $\widehat{w} := w/w(\mathcal{R})$ (for this, we use the sampling oracle $\textsc{Sample}(\mathcal{F}, \widehat{w})$). Then $\mathcal{R}'$ is an ϵ-approximation with high probability. We call $\textsc{Subsys}(\mathcal{F}^*, \mathcal{R}')$ to obtain the set $\mathcal{R}^*_{|\mathcal{R}'}$, then return the subset of ranges $\mathcal{R}'' \in \mathrm{argmax}_{\mathcal{R}'''\in\mathcal{R}^*_{|\mathcal{R}'}}|\mathcal{R}'''|$. Finally, we call the oracle $\textsc{PointIn}(\mathcal{F}, \mathcal{R}'')$ to obtain a point $p \in \bigcap_{R\in\mathcal{R}''} R$.

▶ **Lemma 13.** *With probability* $\Omega(1)$, $\xi(p) \geq (1-\nu)\max_{q\in Q}\xi(q)$.

▶ Remark. The above implementation of the maximization oracle assumes the *unit-cost* model of computation and *infinite* precision arithmetic (real RAM). In some of the applications in the next section, we note that, in fact, *deterministic* algorithms exist for the maximization oracle, which can be implemented in the *bit-model* with *finite precision*.

7 Applications

7.1 Art gallery problem

In the art gallery problem we are given a (non-simple) polygon H with n vertices and h holes. Two points $p, q \in H$ are said to see each other, denoted by $p \sim q$, if the line segment joining them lies inside H (say, including the boundary ∂H). The objective is to find a subset $G \subseteq H$ such that for every point $q \in H$, there is a point $p \in G$ such that $p \sim q$.

Let $Q = H$, $\mathcal{R} = \{V_H(q) : \ q \in H\}$, where $V_H(q) := \{p \in H : \ p \sim q\}$ is the *visibility region* of $q \in H$. For convenience, we shall consider $\mathcal{R}$ as a *multi-set* and hence assume that ranges in $\mathcal{R}$ are in *one-to-one correspondence* with points in H. We shall see below that the range space $\mathcal{F} = (Q, \mathcal{R})$ satisfies (A1)–(A4).

Note that (A1) is satisfied with $\gamma = \text{VC-dim}(\mathcal{F}) \leq 14$ by the result of [23] for simple polygons, while $\gamma = O(\log h)$ for polygons with h holes [45]. (A2) follows immediately from the fact that each point in the polygon is seen from some vertex. (A3) is satisfied if we use $w_0(R) = 1$ for all $R \in \mathcal{R}$ and note that it is integrable over $\mathcal{R}$ as $\int_{R\in\mathcal{R}} w_0(R)dR = \text{area}(H)$ (recall that ranges in $\mathcal{R}$ are in one-to-one correspondence with points in H). Now we show that (A4) is also satisfied. Consider the randomized implementation of the maximization oracle in Section 6. We need to show that the oracles $\textsc{Subsys}(\mathcal{F}^*, \mathcal{R}')$, $\textsc{PointIn}(\mathcal{F}, \mathcal{R}')$ and $\textsc{Sample}(\mathcal{F}, w)$ can be implemented in polynomial time. This is more or less standard; we sketch it here for completeness.

It is known (see, e.g., [18]) that the subsystem oracle $\textsc{Subsys}(\mathcal{F}^*, \mathcal{R}')$ can be computed efficiently, for any (finite) $\mathcal{R}' \subset \mathcal{R}$, as follows. Observe that $\mathcal{R}'$ is a finite set of polygons which induces an arrangement of lines (in $\mathbb{R}^2$) of total complexity $O(nh|\mathcal{R}'|^2)$. We can construct the set of cells of this arrangement, call it $\text{cells}(\mathcal{R}')$, in time $O(nh|\mathcal{R}'|^2 \log(nh|\mathcal{R}'|))$, and label each cell of the arrangement by the set of visibility polygons from $\mathcal{R}'$ it is contained in. Then $\mathcal{R}^*_{|\mathcal{R}'}$ is the set of different cell labels which can be obtained, for e.g., by a sweep algorithm in time $O(nh|\mathcal{R}'|^2 \log(nh|\mathcal{R}'|))$. This argument also implies that $g_{\mathcal{F}^*}(r) \leq O(nhr^2)$, and that we can implement $\textsc{PointIn}(\mathcal{F}, \mathcal{R}')$ in $O(nh|\mathcal{R}'|^2 \log(nh|\mathcal{R}'|))$ time. Finally, we can implement $\textsc{Sample}(\mathcal{F}, \widehat{w}_t)$ given the probability density function $\widehat{w}_t : \mathcal{R} \to \mathbb{R}_+$ defined by the subset $P_t \subseteq Q$ as follows. We construct the cell arrangement $\text{cells}(\mathcal{R})$, induced by the current set P_t as described above. We first sample a cell $\mathcal{R}'$ (which corresponds to an infinite set of ranges with the same weight) with probability $\frac{\widehat{w}_t(\mathcal{R}')}{\sum_{\mathcal{R}'\in\text{cells}(\mathcal{R})}\widehat{w}_t(\mathcal{R}')}$, then we sample a point (corresponding to a range) R uniformly at random from $\mathcal{R}'$. Thus we obtain the following result from Corollary 6, in the *unit-cost* model.

▶ **Corollary 14.** *Given a polygon H with n vertices and h holes and $\delta > 0$, there is a randomized algorithm that finds in* $\text{poly}(n, h, \log\frac{1}{\delta})$ *time a set of points in H of size $O(z^*_{\mathcal{F}} \log z^*_{\mathcal{F}} \cdot \log(h+2))$ guarding at least $(1-\delta)$ of the area of H, where $z^*_{\mathcal{F}}$ is the value of the optimal fractional solution.*

We can also obtain a deterministic version of Corollary 14 in the *bit model* of computation. The idea, following [38], is to express the function $\xi_t(q) := w_t(\mathcal{R}_t[q])$ as a sum of continuous

functions, each of which is a ratio of two polynomials of two variables, namely, the x and y-coordinates of q. Then maximizing over q amounts to solving a system of two polynomial equations of degree $\text{poly}(n, h, \log \frac{1}{\delta})$ in two variables, which an be solved using *quantifier elimination techniques*, e.g., [5, 25, 41]. However, a technical hurdle that we need to overcome is that the required bit length may grow from one iteration to the next, resulting in an exponential blow-up in the bit length needed for the computation. To deal with this issue, we need to round the set $\mathcal{R}_t$ in each iteration so that the total bit length in all iterations remains bounded by a polynomial in the input size.

▶ **Corollary 15.** *Given a polygon H with n vertices and h holes with rational representation of maximum bit-length L and $\delta > 0$, there is a deterministic algorithm that finds in $\text{poly}(L, n, h, \log \frac{1}{\delta})$ time a set of points in H of size $O(z^*_{\mathcal{F}} \log(h+2) \cdot \log(z^*_{\mathcal{F}} \cdot \log(h+2)))$ and bit complexity $\text{poly}(L, n, h, \log \frac{1}{\delta})$ guarding at least $(1-\delta)$ of the area of H, where $z^*_{\mathcal{F}}$ is the value of the optimal fractional solution.*

7.2 Covering a polygonal region by translates of a convex polygon

Let $\mathcal{H}$ be a collection of (non-simple) polygons in the plane and H_0 be a given *full-dimensional convex* polygon. The problem is to minimally cover all the points of the polygons in $\mathcal{H}$ by translates of H_0, that is to find the minimum number of translates $H_0^1, \ldots, H_0^k$ of H_0 such that each point $p \in \bigcup_{H \in \mathcal{H}} H$ is contained in some H_0^i. The discrete case when $\mathcal{H}$ is a set of points has been considered extensively, e.g., covering points with unit disks/squares [28] and generalizations in 3D [15, 32]. Fewer results are known for the continuous case, e.g., [24] which considers the covering of simple polygons by translates of a rectangle[8] and only provides an exact (exponential-time) algorithm; see also [20] for another example, where it is required to hit every polygon in $\mathcal{H}$ by a copy of H_0 (but with rotations allowed).

This problem can be modeled as a hitting set problem in a range space $\mathcal{F} = (Q, \mathcal{R})$, where Q is the set of translates of H_0 and $\mathcal{R} := \left\{\{H_0^i \in Q : \ R \in H_0^i\} : \ R \in \bigcup_{H \in \mathcal{H}} H\right\}$. Again considering $\mathcal{R}$ as a multi-set, we have $\mathcal{R} \leftrightarrow \bigcup_{H \in \mathcal{H}} H$, and we shall refer to elements of $\mathcal{R}$ as sets of translates of H_0 as well as points in $\bigcup_{H \in \mathcal{H}} H$. It was shown by Pach and Woeginger [39] that VC-dim$(\mathcal{F}^*) \leq 3$ and also that $\mathcal{F}^*$ admits an ϵ-net of size $s_{\mathcal{F}^*} = O(\frac{1}{\epsilon})$. As observed in [32], this would also imply that VC-dim$(\mathcal{F}) \leq 3$ and $s_{\mathcal{F}} = O(\frac{1}{\epsilon})$. Thus (A1) is satisfied with $\gamma = 3$. Moreover, assuming that $\mathcal{H}$ is contained in a box of size D and that H_0 contains a box of size d, then (A2) is satisfied as $\text{OPT}_{\mathcal{F}} \leq \frac{D}{d}$. (A3) is satisfied if we use $w_0(R) = 1$ for all $R \in \mathcal{R}$ (which defines the area measure over $\mathcal{R}$). Now we show that (A4) is also satisfied.

Consider the randomized implementation of the maximization oracle in Section 6. We need to show that the oracles $\text{SUBSYS}(\mathcal{F}^*, \mathcal{R}')$, $\text{POINTIN}(\mathcal{F}, \mathcal{R}')$ and $\text{SAMPLE}(\mathcal{F}, w)$ can be implemented in polynomial time. Let m be the total number of vertices of the polygons in $\mathcal{H}$ and H_0. Note that for a given finite $\mathcal{R}' \subseteq \mathcal{R}$, the set $\mathcal{R}^*_{|\mathcal{R}'}$ is the set of all subsets of points in $\mathcal{R}'$ that are contained in the same copy of H_0. Observe that each such subset is determined by at most two points from $\mathcal{R}'$ that lie on the boundary of a copy of H_0. It follows that $\text{SUBSYS}(\mathcal{F}^*, \mathcal{R}')$ can be implemented in $O(m^2|\mathcal{R}'|^2 \log m)$ time. This argument also shows that $\text{POINTIN}(\mathcal{F}, \mathcal{R}')$ can be implemented in the same time $O(m^2|\mathcal{R}'|^3)$ and that $g_{\mathcal{F}^*}(r) \leq r^2$. Finally, we can implement $\text{SAMPLE}(\mathcal{F}, \widehat{w}_t)$ given the probability density function $\widehat{w}_t : \mathcal{R} \to \mathbb{R}_+$ defined by the subset $P_t \subseteq Q$ as follows. Given the current subset $P_t \subseteq Q$ of translates of H_0, we can find (e.g. by a sweep line algorithm) in $O(m \log m)$ time the cells of the arrangement defined by $\mathcal{H} \cup P_t$ (where a cell is naturally defined to be a

[8] Note that in [24], each polygon has to be covered *completely* by a rectangle.

maximal set of points in $\mathcal{R}$ that all belong exactly to the same polygons in the arrangement). Let us call this set cells($\mathcal{R}$) and note that it has size $O(m)$. We first sample a cell $\mathcal{R}'$ with probability $\frac{\widehat{w}_t(\mathcal{R}')}{\sum_{\mathcal{R}'\in \text{cells}(\mathcal{R})} \widehat{w}_t(\mathcal{R}')}$, then we sample a point R uniformly at random from $\mathcal{R}'$.

▶ **Corollary 16.** *Given a collection of polygons in the plane $\mathcal{H}$ and a convex polygon H_0, with m total vertices and $\delta > 0$, there is a randomized algorithm that finds in* $\text{poly}(n, m, \log \frac{1}{\delta})$ *time a set of $O(z^*_{\mathcal{F}})$ translates of H_0 covering at least $(1-\delta)$ of the total area of the polygons in $\mathcal{H}$, where $z^*_{\mathcal{F}}$ is the value of the optimal fractional solution.*

7.3 Polyhedral separation in $\mathbb{R}^d$

Given two (full-dimensional) convex polytopes $\mathcal{P}_1, \mathcal{P}_2 \subseteq \mathbb{R}^d$ such that $\mathcal{P}_1 \subset \mathcal{P}_2$, it is required to find a (separator) polytope $\mathcal{P}_3 \subseteq \mathbb{R}^d$ such that $\mathcal{P}_1 \subseteq \mathcal{P}_3 \subseteq \mathcal{P}_2$, with as few facets as possible. This problem can be modeled as a hitting set problem in a range space $\mathcal{F} = (Q, \mathcal{R})$, where Q is the set of supporting hyperplanes for $\mathcal{P}_1$ and $\mathcal{R} := \{\{p \in Q : \ p \text{ separates } R \text{ from } \mathcal{P}_1\} : \ R \in \partial\mathcal{P}_2\}$ (thus, we may assume that $\mathcal{R} \leftrightarrow \partial\mathcal{P}_2$). Note that VC-dim$(\mathcal{F}) = d$ (and VC-dim$(\mathcal{F}^*) = d+1$). In their paper [8], Brönnimann and Goodrich gave a deterministic $O(d^2 \log \text{OPT}_{\mathcal{F}})$-approximation algorithm, improving on earlier results by Mitchell and Suri [37], and Clarkson [14]. It was shown in [37] that, at the cost of losing a factor of d in the approximation ratio, one can consider a finite set Q, consisting of the hyperplances passing through the facets of $\mathcal{P}_1$. We can save this factor of d by showing that $\mathcal{F}$ satisfies (A1)–(A4).

Let n and m be the number of facets of $\mathcal{P}_1$ and $\mathcal{P}_2$, respectively. Then (A1) is satisfied with $\gamma = d$ as explained above. Also, (A2) is obviously satisfied since $\mathcal{P}_3 = \mathcal{P}_2$ is a separator with n facets. For (A3), we use w_0 as the *surface area* measure, i.e., $w_0(\mathcal{R}') = \text{vol}_{d-1}(\mathcal{R}')$ for $\mathcal{R}' \subseteq \mathcal{R}$. Now we show that (A4) also holds.

Consider the randomized implementation of the maximization oracle in Section 6. We need to show that the oracles SUBSYS$(\mathcal{F}^*, \mathcal{R}')$, POINTIN$(\mathcal{F}, \mathcal{R}')$ and SAMPLE$(\mathcal{F}, w)$ can be implemented in polynomial time. Note that for a given finite $\mathcal{R}' \subseteq \mathcal{R}$, the set $\mathcal{R}^*_{|\mathcal{R}'}$ has size at most $g(|\mathcal{R}'|, d+1)$, and furthermore, for any hyperplane $q \in Q$, $\mathcal{R}'[q]$ is the set of points in $\mathcal{R}'$ separated from $\mathcal{P}_1$ by q. Thus, $\mathcal{R}'[q]$ is determined by exactly d points chosen from $\mathcal{R}'$ and the vertices of $\mathcal{P}_1$. It follows that the set $\mathcal{R}^*_{|\mathcal{R}'}$ can be found (and hence SUBSYS$(\mathcal{F}^*, \mathcal{R}')$ can be implemented) in time $\text{poly}((n^{\frac{d}{2}} + |\mathcal{R}'|)^d)$. This argument also shows that POINTIN$(\mathcal{F}, \mathcal{R}')$ can be implemented in the time $\text{poly}((n^{\frac{d}{2}} + |\mathcal{R}'|)^d)$. Finally, we can implement SAMPLE$(\mathcal{F}, \widehat{w}_t)$ given the probability density function $\widehat{w}_t : \mathcal{R} \to \mathbb{R}_+$ defined by the current subset $P_t \subseteq Q$ as follows. We first construct the set of cells of the hyperplane arrangement of P_t, which has complexity $O(|P_t|^d)$, in time $O(|P_t|^{d+1})$; see, e.g., [4, 44]. Next, we intersect every facet of $\mathcal{P}_2$ with every cell in the arrangement. This allows us to identify the partition of $\partial\mathcal{P}_2$ induced by the cell arrangement; let us call it cells($\mathcal{R}$). The running time for this is $\text{poly}(|P_t|^d, m^d)$. We first sample $\mathcal{R}'$ with probability $\frac{\widehat{w}_t(\mathcal{R}')}{\sum_{\mathcal{R}'\in \text{cells}(\mathcal{R})} \widehat{w}_t(\mathcal{R}')}$, then we sample a point R uniformly at random from $\mathcal{R}'$ (note that both volume computation and uniform sampling can be done in polynomial time in fixed dimension).

▶ **Corollary 17.** *Given two convex polytopes $\mathcal{P}_1, \mathcal{P}_2 \subseteq \mathbb{R}^d$ such that $\mathcal{P}_1 \subset \mathcal{P}_2$, with n and m facets respectively and $\delta > 0$, there is a randomized algorithm that finds in* $\text{poly}((nm)^d, \log \frac{1}{\delta})$ *time a polytope $\mathcal{P}_3$ with $O(z^*_{\mathcal{F}} d \cdot \log z^*_{\mathcal{F}})$ facets separating $\mathcal{P}_1$ from a subset of $\partial\mathcal{P}_2$ of volume at least $(1-\delta)$ of the volume of $\partial\mathcal{P}_2$, where $z^*_{\mathcal{F}}$ is the value of the optimal fractional solution.*

Acknowledgement. The author is grateful to Waleed Najy for his help in the proof of Lemma 12 and for many useful discussions, and to the anonymous reviewers for the careful reading and the helpful remarks.

References

1. P. K. Agarwal and J. Pan. Near-linear algorithms for geometric hitting sets and set covers. In *SoCG'14*, pages 271–279, 2014.
2. N. Alon and J. H. Spencer. *The Probabilistic Method*. Wiley Series in Discrete Mathematics and Optimization. Wiley, 2008.
3. B. Aronov, E. Ezra, and M. Sharir. Small-size ϵ-nets for axis-parallel rectangles and boxes. *SIAM Journal on Computing*, 39(7):3248–3282, 2010.
4. D. Avis and K. Fukuda. A pivoting algorithm for convex hulls and vertex enumeration of arrangements and polyhedra. *Discrete & Computational Geometry*, 8(3):295–313, 1992.
5. S. Basu, R. Pollack, and M. Roy. On the combinatorial and algebraic complexity of quantifier elimination. *J. ACM*, 43(6):1002–1045, 1996.
6. É. Bonnet and T. Miltzow. An approximation algorithm for the art gallery problem. In *EuroCG'16, also available online as: https://arxiv.org/abs/1607.05527*, 2016.
7. H. Brönnimann, B. Chazelle, and J. Matoušek. Product range spaces, sensitive sampling, and derandomization. *SIAM Journal on Computing*, 28(5):1552–1575, 1999.
8. H. Brönnimann and M. T. Goodrich. Almost optimal set covers in finite VC-dimension. *Discrete & Computational Geometry*, 14(4):463–479, 1995.
9. T. M. Chan, E. Grant, J. Könemann, and M. Sharpe. Weighted capacitated, priority, and geometric set cover via improved quasi-uniform sampling. In *SODA'12*, pages 1576–1585, 2012.
10. B. Chazelle. *The Discrepancy Method: Randomness and Complexity*. Cambridge University Press, New York, NY, USA, 2000.
11. B. Chazelle and J.Matoušek. On linear-time deterministic algorithms for optimization problems in fixed dimension. *Journal of Algorithms*, 21(3):579–597, 1996.
12. O. Cheong, A. Efrat, and S. Har-Peled. Finding a guard that sees most and a shop that sells most. *Discrete & Computational Geometry*, 37(4):545–563, 2007.
13. V. Chvatal. A greedy heuristic for the set-covering problem. *Mathematics of Operations Research*, 4(3):233–235, 1979.
14. K. L. Clarkson. Algorithms for polytope covering and approximation. In *WADS'93*, pages 246–252, 1993.
15. K. L. Clarkson and K. Varadarajan. Improved approximation algorithms for geometric set cover. *Discrete & Computational Geometry*, 37(1):43–58, 2006.
16. A. Deshpande, T. Kim, E. D. Demaine, and S. E. Sarma. A pseudopolynomial time $O(\log n)$-approximation algorithm for art gallery problems. In *WADS'07*, pages 163–174, 2007.
17. I. Dinur and D. Steurer. Analytical approach to parallel repetition. In *STOC'14*, pages 624–633, 2014.
18. A. Efrat and S. Har-Peled. Guarding galleries and terrains. *Inf. Process. Lett.*, 100(6):238–245, 2006.
19. G. Even, D. Rawitz, and S. (M.) Shahar. Hitting sets when the VC-dimension is small. *Inf. Process. Lett.*, 95(2):358–362, 2005.
20. S. K. Ganjugunte. *Geometric Hitting Sets and Their Variants*. PhD thesis, Duke University, USA, 2011.
21. N. Garg and J. Könemann. Faster and simpler algorithms for multicommodity flow and other fractional packing problems. *SIAM J. Comput.*, 37(2):630–652, 2007.
22. S. K. Ghosh. Approximation algorithms for art gallery problems in polygons. *Discrete Applied Mathematics*, 158(6):718–722, 2010.

23 A. Gilbers and R. Klein. A new upper bound for the VC-dimension of visibility regions. *Computational Geometry*, 47(1):61–74, 2014.

24 R. Glück. Covering polygons with rectangles. In *EuroCG'16*, 2016.

25 D. Grigoriev and N. Vorobjov. Solving systems of polynomial inequalities in subexponential time. *J. Symb. Comput.*, 5(1/2):37–64, 1988.

26 S. Har-Peled and M. Sharir. Relative (p, ϵ)-approximations in geometry. *Discrete & Computational Geometry*, 45(3):462–496, 2011.

27 D. Haussler and E. Welzl. Epsilon-nets and simplex range queries. *Discrete & Computational Geometry*, 2:127–151, 1987.

28 D. S. Hochbaum and W. Maass. Approximation schemes for covering and packing problems in image processing and VLSI. *J. ACM*, 32(1):130–136, 1985.

29 D. S. Johnson. Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences*, 9(3):256–278, 1974.

30 J. King and D. G. Kirkpatrick. Improved approximation for guarding simple galleries from the perimeter. *Discrete & Computational Geometry*, 46(2):252–269, 2011.

31 J. Komlós, J. Pach, and G. Woeginger. Almost tight bounds for ϵ-nets. *Discrete & Computational Geometry*, 7(2):163–173, March 1992.

32 S. Laue. Geometric set cover and hitting sets for polytopes in $\mathbb{R}^3$. In *STACS'08*, pages 479–490, 2008.

33 L. Lovász. On the ratio of optimal integral and fractional covers. *Discrete Mathematics*, 13(4):383–390, 1975.

34 J. Matoušek. Cutting hyperplane arrangements. *Discrete & Computational Geometry*, 6(3):385–406, 1991.

35 J. Matoušek. Reporting points in halfspaces. *Computational Geometry*, 2(3):169–186, 1992.

36 J. Matoušek, R. Seidel, and E. Welzl. How to net a lot with little: Small ϵ-nets for disks and halfspaces. In *SoCG'90*, pages 16–22, 1990.

37 J. S. B. Mitchell and S. Suri. Separation and approximation of polyhedral objects. *Computational Geometry*, 5(2):95–114, 1995.

38 S. Ntafos and M. Tsoukalas. Optimum placement of guards. *Information Sciences*, 76(1–2):141–150, 1994.

39 J. Pach and G. Woeginger. Some new bounds for Epsilon-nets. In *SoCG'90*, pages 10–15, 1990.

40 E. Pyrga and S. Ray. New existence proofs ϵ-nets. In *SoCG'08*, pages 199–207, 2008.

41 J. Renegar. On the computational complexity and geometry of the first-order theory of the reals. *J. Symb. Comput.*, 13(3):255–352, 1992.

42 N. Sauer. On the density of families of sets. *Journal of Combinatorial Theory, Series A*, 13(1):145–147, 1972.

43 S. Shelah. A combinatorial problem; stability and order for models and theories in infinitary languages. *Pacific J. Math.*, 41(1):247–261, 1972.

44 N. H. Sleumer. Output-sensitive cell enumeration in hyperplane arrangements. *Nordic J. of Computing*, 6(2):137–147, 1999.

45 P. Valtr. Guarding galleries where no point sees a small area. *Israel Journal of Mathematics*, 104(1):1–16, 1998.

46 V. N. Vapnik and A. Ya. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability & Its Applications*, 16(2):264–280, 1971.

47 K. Varadarajan. Epsilon nets and union complexity. In *SoCG'09*, pages 11–16, 2009.

A Nearly Quadratic Bound for the Decision Tree Complexity of k-SUM*

Esther Ezra[1] **and Micha Sharir**[2]

1 **Georgia Institute of Technology, Atlanta, GA, USA**
eezra3@math.gatech.edu
2 **Blavatnik School of Computer Science, Tel Aviv University, Tel Aviv, Israel**
michas@post.tau.ac.il

Abstract

We show that the k-SUM problem can be solved by a linear decision tree of depth $O(n^2 \log^2 n)$, improving the recent bound $O(n^3 \log^3 n)$ of Cardinal *et al.* [7]. Our bound depends linearly on k, and allows us to conclude that the number of linear queries required to decide the n-dimensional KNAPSACK or SUBSETSUM problems is only $O(n^3 \log n)$, improving the currently best known bounds by a factor of n [28, 29]. Our algorithm extends to the RAM model, showing that the k-SUM problem can be solved in expected polynomial time, for any fixed k, with the above bound on the number of linear queries. Our approach relies on a new point-location mechanism, exploiting "ε-cuttings" that are based on *vertical decompositions* in hyperplane arrangements in high dimensions. A major side result of the analysis in this paper is a sharper bound on the complexity of the vertical decomposition of such an arrangement (in terms of its dependence on the dimension). We hope that this study will reveal further structural properties of vertical decompositions in hyperplane arrangements.

1998 ACM Subject Classification F.2.2 [Nonnumerical Algorithms and Problems] Computations on Discrete Structures, Geometrical Problems and Computations

Keywords and phrases k-SUM and k-LDT, linear decision tree, hyperplane arrangements, point-location, vertical decompositions, ε-cuttings

Digital Object Identifier 10.4230/LIPIcs.SoCG.2017.41

1 Introduction

Problem definition and the model. In this paper we study the k-SUM problem, and the more general *k-linear degeneracy testing* (k-LDT) problem. We define them formally:

▶ **Definition 1** (k-SUM)**.** Given a point $\mathbf{x} := (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$, decide whether there exist k indices $i_1, i_2, \ldots, i_k$ such that $x_{i_1} + x_{i_2} + \ldots + x_{i_k} = 0$.

In what follows, we assume that we are looking for a k-tuple of *distinct* coordinates. The case where some coordinates can be repeated more than once is also easy to handle, by a straightforward extension of the technique presented here, which we omit, for the sake of simplicity of presentation.

* Work on this paper by Esther Ezra has been supported by NSF CAREER under grant CCF:AF 1553354. Work on this paper by Micha Sharir was supported by Grant 892/13 from the Israel Science Foundation, by Grant 2012/229 from the U.S.–Israel Binational Science Foundation, by the Blavatnik Research Fund in Computer Science at Tel Aviv University, by the Israeli Centers of Research Excellence (I-CORE) program (Center No. 4/11), and by the Hermann Minkowski-MINERVA Center for Geometry at Tel Aviv University.

▶ **Definition 2** (k-LDT)**.** Given a fixed k-variate linear function $f(y_1, \ldots, y_k) = a_0 + \sum_{i=1}^{k} a_i y_i$, where $a_0, a_1, \ldots, a_k$ are real coefficients, and a point $\mathbf{x} := (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$, decide whether there exist k indices $i_1, i_2, \ldots, i_k$ such that $f(x_{i_1}, x_{i_2}, \ldots, x_{i_k}) = 0$.[1]

By definition, k-SUM is a special case of k-LDT when we set $f(y_1, \ldots, y_k) = \sum_{i=1}^{k} y_i$. We note that the special case $k = 3$, the so-called 3-SUM problem, received considerable attention in the past two decades, due to its implications to conditional lower bounds on the complexity of many fundamental geometric problems; see [18] and below for a list of such problems. From now on we focus only on the k-SUM problem; the algorithm that we present applies more or less verbatim to k-LDT too.

Following the approach in Cardinal *et al.* [7] for k-SUM (see also [2, 15]), let H be the collection of the $\binom{n}{k}$ hyperplanes h in $\mathbb{R}^n$ of the form $x_{i_1} + x_{i_2} + \cdots + x_{i_k} = 0$, over all k-tuples $1 \le i_1 < i_2 < \cdots < i_k \le n$. Then the k-SUM problem can be reformulated as asking, for a query point $\mathbf{x}$, whether $\mathbf{x}$ lies on any hyperplane of H. This can be determined by locating $\mathbf{x}$ in the *arrangement* $\mathcal{A}(H)$ formed by those hyperplanes.

The model in which we consider this problem is the *s-linear decision tree*: Solving an instance of the problem with input $\mathbf{x} = (x_1, \ldots, x_n)$ is implemented as a search with $\mathbf{x}$ in some tree T. Each internal node v of T is assigned a linear function in the n variables $x_1, \ldots, x_n$, with at most s non-zero coefficients. The outgoing edges from v are labeled $<$, $>$, or $=$, indicating the branch to follow depending on the sign of the expression at v evaluated at $\mathbf{x}$. Leaves are labeled "YES" or "NO", where "YES" means that we have managed to locate $\mathbf{x}$ on a hyperplane of H, and "NO" means that $\mathbf{x}$ does not lie on any hyperplane. Each "YES" leaf has an edge labeled "=" leading to it (but not necessarily vice versa, because some of the tests may involve auxiliary hyperplanes that are not part of the input). To solve an instance of the problem, we begin at the root of T. At each node v that we visit, we test the sign at $\mathbf{x}$ of the linear function at v, and proceed along the outgoing edge labeled by the result of the test. We conduct this search until we reach a leaf, and output its label "YES" or "NO". At each internal node, the test (which we also refer to as a *linear query*) is assumed to cost one unit. All other operations are assumed (or rather required) not to depend on the specific coordinates of $\mathbf{x}$ (although they might depend on discrete data that has been obtained from the preceding queries with $\mathbf{x}$), and incur no cost in this model. Thus the length of the search path from the root to a leaf is the overall number of linear queries performed by the algorithm on the given input, and is thus our measure for its cost. In other words, the worst-case complexity of the algorithm, in this model, is the maximum depth of its corresponding tree. As in [7], when $s = n$ (the maximum possible value for s), we refer to the model just as a "linear decision tree". The study in this paper will only consider this unconstrained case.

We also note that, although we could in principle construct the whole tree T in a preprocessing stage, the algorithm that we present only constructs, on the fly, the search path that $\mathbf{x}$ traces in T.

To recap, solving an instance of k-SUM, with input $\mathbf{x}$, in this model amounts to processing a sequence of *linear queries* of the form "Does $\mathbf{x}$ lie on some hyperplane h, or else on which side of h does it lie?". Each such query is a sign test, asking for the sign of $h(\mathbf{x})$, where $h(\cdot)$ is the linear expression defining h. Some of the hyperplanes h that participate in these queries will be original hyperplanes of H, but others will be auxiliary hyperplanes that the algorithm constructs. The algorithm succeeds if at least one of the linear queries that involves

[1] We emphasize that in the k-LDT problem, the coefficients $a_0, a_1, \ldots, a_k$ are fixed and the input is the point $\mathbf{x}$.

an original hyperplane (or, during the recursion, a lower-dimensional hyperplane that is contained in an original hyperplane) results in an equality, determining that $\mathbf{x}$ does lie on a hyperplane of H.

Previous work. The k-SUM problem is a variant of the SUBSETSUM problem,[2] and is therefore NP-complete (when k is part of the input); however, its behavior as a function of k (say, in the standard RAM model) has not yet been fully resolved. Specifically, Erickson [15] showed that the k-SUM problem can be solved (in the RAM model) in time $O((2n/k)^{\lceil k/2\rceil})$ for k odd, and $O((2n/k)^{k/2}\log(n/k))$ for k even. Moreover, he presented a nearly-tight lower bound of $\Omega((n/k^k)^{\lceil k/2\rceil})$ for k-SUM in the k-linear decision tree model (see [2] for a more comprehensive overview of Erickson's result). Ailon and Chazelle [2] slightly improved Erickson's lower bound, and extended it to the s-linear decision tree model, where $s > k$, showing a lower bound of $\Omega\left((nk^{-3})^{\frac{2k-s}{2\lceil(s-k+1)/2\rceil}(1-\varepsilon_k)}\right)$, where $\varepsilon_k > 0$ tends to 0 as k goes to ∞. As stated in [2], in spite of the strength of this latter lower bound, it is not very informative for $s \geq 2k$. In particular, when s is arbitrarily large (the case studied in this paper), one can no longer derive a lower bound of the form $n^{\Omega(k)}$. Indeed, Meyer auf der Heide [29] showed an upper bound of $O(n^4 \log n)$ on the number of linear queries for the n-dimensional KNAPSACK problem[3] (and thus, in particular, for k-SUM). Meiser [28] presented an efficient point-location mechanism for high-dimensional hyperplane arrangements, in the standard real RAM model. When interpreted in the linear decision tree model, and applied to the instances at hand, it yields a linear decision tree for k-SUM (as well as for the more general problem k-LDT) of depth that is only polynomial[4] in k and n. Cardinal *et al.* [7] improved this bound[5] to $O(n^3 \log^3 n)$. Concerning lower bounds in this model of computation, Dobkin and Lipton [14] showed a lower bound of $\Omega(n \log n)$ on the depth of the linear decision tree for k-LDT, and, in another paper [13], a lower bound of $\Omega(n^2)$ for the n-dimensional KNAPSACK problem. See also [5, 31] for more general non-linear decision tree models of computation.

The case $k = 3$, i.e., the 3SUM-problem, is related to various geometric problems to which it can be reduced. These problems are known as 3SUM-*hard*. These include problems such as testing whether there exist three collinear points in a given planar set of n points, testing whether the union of n given triangles in the plane covers the unit square, or just testing whether the union has any holes (i.e., is not simply connected), checking for polygon containment under translation, visibility among triangles in 3-space, planar motion planning (under translations and rotations), translational motion planning in 3-space, maximum depth in an arrangement of disks, and more. The study of 3SUM-hard problems was pioneered in the seminal work of Gajentaan and Overmars [18], who showed subquadratic reductions from 3-SUM to many of these problems; see also Barequet and Har-Peled [4] and Aronov and Har-Peled [3] for several additional reductions. During the last two decades, the prevailing conjecture was that any algorithm for 3-SUM requires $\Omega(n^2)$ time.[6] In a recent dramatic

[2] In this problem, given n real numbers, we want to determine whether there is a subset of them that sums to 0.

[3] This problem is an extension of SUBSETSUM, and asks, given n real numbers, whether there is a subset of them that sums to 1.

[4] The original analysis of Meiser was sketchy and relied on a suboptimal choice of parameters. Meiser's analysis has been somewhat tightened by Liu [26]. A careful and meticulous study of Meiser's algorithm, with improved performance bounds, is given in the full version (written with Har-Peled and Kaplan) [16].

[5] We note however that the bound in [7] only applies to instances of k-LDT where all the coefficients are *rational*.

[6] In fact, before the term "3SUM-hard" was coined, these problems were referred to as "n^2-hard" problems [18].

development, this has been refuted by Grønlund and Pettie [20], who presented a (slightly) subquadratic algorithm for 3-SUM (see also the more recent work of Chan and Lewenstein [8], as well as those of Gold and Sharir [19] and of Freund [17]). Furthermore, Grønlund and Pettie showed that in the $(2k-2)$-linear decision tree model, only $O(n^{k/2}\sqrt{\log n})$ queries are required for k odd. In particular, for 3SUM, this bound is $O(n^{3/2}\sqrt{\log n})$. More recently, this bound has further been improved by Gold and Sharir [19] to $O(n^{3/2})$, or, more generally, to $O(n^{k/2})$ for arbitrary k, under a *randomized* $(2k-2)$-linear decision tree model. Note that in all these cases, the best known lower bound is just the standard $\Omega(n \log n)$ bound, and closing the gap between this bound and the aforementioned upper bounds still remains elusive.

Our result. Our main result is an improvement by (more than) a factor of n over the recent bound of Cardinal *et al.* [7] on the complexity of a linear decision tree for k-SUM and k-LDT. Specifically, we show:

▶ **Theorem 3.** *For any fixed k, the complexity of k-SUM and k-LDT in the linear decision-tree model is $O(n^2 \log^2 n)$, where the constant of proportionality is linear in k.*

In fact, the actual bound in Theorem 3 is $O(n^2 \log n \log |H|)$. We can apply this bound to the n-dimensional KNAPSACK problem, in which the relevant hyperplanes are of the form $x_{i_1} + \cdots + x_{i_k} = 1$, for all the $2^n - 1$ possible nonempty subsets $\{i_1, \ldots, i_k\}$ of $\{1, \ldots, n\}$. Taking then $|H| = 2^n - 1$, we obtain the following corollary of Theorem 3, which improves the previous bounds in [28, 29].

▶ **Corollary 4.** *The complexity of the n-dimensional* SUBSETSUM *and* KNAPSACK *problems in the linear decision-tree model is $O(n^3 \log n)$.*

We note that the two bounds that we obtain in Theorem 3 and Corollary 4 are larger by only a factor of $O(n \log n)$ from the respective lower bounds of Dobkin and Lipton [13, 14].

On the "real" algorithmic front, we show that in the RAM model the k-SUM and k-LDT problems can be solved in expected polynomial time, with the same number, $O(n^2 \log^2 n)$, of linear queries (and with all other operations independent of the actual coordinates of $\mathbf{x}$), as in Theorem 3 (the description of the algorithm is deferred to the full version of this paper).

Our analysis uses a variant of the approach in [7], inspired by the point-location mechanism of Meiser [28], where we locate the input point $\mathbf{x}$ in $\mathcal{A}(H)$ using a recursive algorithm that exploits and locally simulates the construction of (a specific kind of) an *ε-cutting* of $\mathcal{A}(H)$. While this framework is not new, a major difference between the construction of [7] and ours is that the former construction applies *bottom-vertex triangulation* to the cells in arrangements of suitable subsets of H, which partitions each cell into simplices. Since the ambient dimension is n, each simplex is defined (in general) by $\Theta(n^2)$ hyperplanes of H; see, e.g., [1, 12] and below. In contrast, in our construction we partition the cells of such arrangements using the *vertical decomposition* technique [1, 9], where each cell of the arrangement is partitioned into a special kind of *vertical prisms*, each of which is defined by only at most $2n$ hyperplanes of H. In both studies, ours and that of Cardinal *et al.* [7], the local construction of the cell containing $\mathbf{x}$ (in an arrangement of some subsample of H) is carried out through n recursive steps (reducing the dimension by 1 at each step). The difference is that the algorithm in [7] needs to perform roughly quadratically many queries at each such recursive step, whereas our algorithm performs only nearly linearly many queries. With a few additional observations about the structure of vertical decompositions (see below for a detailed discussion), this will eventually decrease the overall depth of the linear decision tree by (slightly more than)

a factor of n, with respect to the bound in [7]. We note that, although the combinatorial bound on the overall complexity of bottom vertex triangulations in hyperplane arrangements is in general smaller than the currently best known bound on the complexity of vertical decompositions (in dimensions $d \geq 5$), this is not an issue in the decision tree model. In other words, for the purpose of locating the cell containing $\mathbf{x}$, in the (linear) decision tree model, using vertical decompositions is the decisive winning strategy.

A note on vertical decomposition. As a by-product of this study, our analysis leads to some new insights concerning the structure and complexity of vertical decompositions of arrangements of hyperplanes, including a sharper bound on the complexity of such a decomposition in high dimensions. Specifically, it follows from the study of Chazelle *et al.* [9], or rather from its extension by Koltun [25], that this bound is $O(n^{2d-4})$, where the coefficient of proportionality is $2^{O(d^2)}$. We improve this coefficient to $2^{O(d)}$ (Theorem 5), using a simple but crucial observation about the structure of vertical decompositions, given in Lemma 8. This property is also used by our algorithm to efficiently construct the (prism-like) cell containing the query point $\mathbf{x}$. This improvement is significant when d is not assumed to be a constant (as in the cases, studied here, of the k-SUM and k-LDT problems, and in the cases of the SubsetSum and Knapsack problems). Moreover, this improvement (from $2^{O(d^2)}$ to $2^{O(d)}$) is crucial for obtaining ε-cuttings with samples of size that is only (nearly) *linear* in d. More details are given later in the paper. We believe these results to be of independent interest, and we hope that these insights will lead to further improved bounds on the complexity of vertical decompositions and for additional useful structural properties and further applications of this construct. A more thorough and detailed analysis of these issues is given in the full version [16].

2 Preliminaries: Arrangements and Vertical Decomposition

Let H be a collection of n hyperplanes in $\mathbb{R}^d$ (observe that the notation in this section is different, as it caters to any collection of hyperplanes in any dimension). We emphasize that H is not necessarily in general position, and that it may contain vertical hyperplanes (as it does in the case of k-SUM). The *vertical decomposition* $\mathcal{V}(H)$ of the arrangement $\mathcal{A}(H)$ is defined in the following recursive manner (see [1, 9] for the general setup, and [21, 25] for the case of hyperplanes in four dimensions). Let the coordinate system be $x_1, x_2, \ldots, x_d$, and let C be a cell in $\mathcal{A}(H)$. For each $(d-2)$-face g on ∂C, we erect a $(d-1)$-dimensional *vertical wall* passing through g and confined to C; this is the union of all the maximal x_d-vertical line-segments that have one endpoint on g and are contained in C. The walls extend downwards (resp., upwards) from faces g on the top boundary (resp., bottom boundary) of C (faces on the "equator" of C, i.e., faces that have a vertical supporting hyperplane, have no wall (within C) erected from them). Note that if g lies on a vertical hyperplane $h \in H$, the vertical wall is contained in h. This collection of walls subdivides C into convex vertical prisms, each of which is bounded by (potentially many) vertical walls, and by two hyperplanes of H, one appearing on the bottom portion and one on the top portion of ∂C, referred to as the *floor* and the *ceiling* of the prism, respectively; in case C is unbounded, a prism may be bounded by just a single (floor or ceiling) hyperplane of H. In rare situations, where all the hyperplanes of H are vertical, prisms have neither a floor nor a ceiling. (Note that, by construction, a floor (resp., a ceiling) of a prism cannot be contained in a vertical hyperplane of H.) More formally, this step is accomplished by projecting the bottom and the top portions of ∂C onto the hyperplane $x_d = 0$, and by constructing the *overlay* of these

two convex subdivisions. Each full-dimensional (i.e., $(d-1)$-dimensional) cell in the overlay, when lifted vertically back to $\mathbb{R}^d$ and intersected with C, becomes one of the above prisms.

Note that after this step, the two bases (or the single base, in case the prism is unbounded) of a prism may have arbitrarily large complexity, or, more precisely, be bounded by arbitrarily many hyperplanes. Each base, say the floor base, is a convex polyhedron in $\mathbb{R}^{d-1}$, namely in the hyperplane h^- containing it, bounded by at most $2n-1$ hyperplanes (of dimension $d-2$), where each such hyperplane is either an intersection of h^- with another original hyperplane h, or the vertical projection onto h^- of an intersection of the corresponding ceiling hyperplane h^+ with some other h (for h vertical, the two cases coincide; that is, they yield the same $(d-2)$-hyperplane within h^-); this collection might also include $h^- \cap h^+$. In what follows we refer to these prisms as *undecomposed prisms*, or *first-stage prisms*. Our goal is to decimate the dependence of the complexity of the prisms on n, and to construct a decomposition of this kind so that each of its prisms is bounded by no more than $2d$ hyperplanes. To do so, we recurse with the construction at each base of each prism, or rather, for simplicity, within the common projection of the bases onto $x_d = 0$. Each recursive subproblem is now $(d-1)$-dimensional.

Specifically, after the first decomposition step described above, we project each of the first-stage prisms just obtained onto the hyperplane $x_d = 0$, obtaining a $(d-1)$-dimensional convex polyhedron C', which we vertically decompose using the same procedure described above, only in one lower dimension. That is, we now erect vertical walls within C' from each $(d-3)$-face of $\partial C'$, in the x_{d-1}-direction. These walls subdivide C' into x_{d-1}-vertical (undecomposed) prisms, each of which is bounded by (at most) two facets of C', which form its floor and ceiling (in the x_{d-1}-direction), and by some of the vertical walls. We keep projecting these prisms onto hyperplanes of lower dimensions, and produce the appropriate vertical walls. We stop the recursion as soon as we reach a one-dimensional instance, in which case all prisms projected from previous steps become line-segments, requiring no further decomposition.[7] We now backtrack, and lift the vertical walls (constructed in lower dimensions, over all iterations), one dimension at a time, ending up with $(d-1)$-dimensional walls within the original cell C; that is, a $(d-i)$-dimensional wall is "stretched" in directions $x_{d-i+2}, \ldots, x_d$ (applied in that order), for every $i = d, \ldots, 2$.

Each of the final cells is a "box-like" prism, bounded by at most $2d$ hyperplanes. Of these, two are original hyperplanes, two are hyperplanes supporting two x_d-vertical walls erected from some $(d-2)$-faces, two are hyperplanes supporting two $x_{d-1}x_d$-vertical walls erected from some $(d-3)$-faces (within the appropriate lower-dimensional subspaces), and so on. Note that since we do not assume general position, some of these vertical walls may be original hyperplanes of H (this issue is discussed in more detail later on).

We note that each final prism is *defined* in terms of at most $2d$ original hyperplanes of H, in a sense made precise in the ensuing description. We establish this property using backward induction on the dimension of the recursive instance. Initially, we have two original hyperplanes h^-, h^+, which contain the floor and ceiling of the prism, respectively. We intersect each of them with the remaining hyperplanes of H (including the intersection $h^- \cap h^+$), and project all these intersections onto the $(d-1)$-hyperplane $x_d = 0$. Suppose inductively that, when we are at dimension j, we already have a set D_j of (at most) $2(d-j)$ original defining hyperplanes (namely, original hyperplanes defining the walls erected so

[7] If we care about the complexity of the resulting decomposition, in terms of its dependence on n, which is not a crucial issue in our approach, it is better to stop the recursion earlier. The terminal dimension is $d = 2$ or $d = 3$ in [9], and $d = 4$ in [25].

far), and that each (lower-dimensional) hyperplane in the current collection H_j of $(j-1)$-hyperplanes is obtained by an interleaved sequence of intersections and projections, which are expressed in terms of some subset of the $\leq 2(d-j)$ defining hyperplanes and (at most) one additional original hyperplane. Clearly, all this holds trivially in the initial step $j = d$. We now choose a new floor and a new ceiling from among the hyperplanes in H_j, gaining two new defining hyperplanes (the unique ones that define the new floor and ceiling and are the ones not in D_j). We add them to D_j to form D_{j-1}, intersect each of them with the other hyperplanes in H_j, and project all the resulting $(j-2)$-intersections onto the $(j-1)$-hyperplane $x_j = 0$, to obtain a new collection H_{j-1} of $(j-2)$-hyperplanes. Clearly, the inductive properties that we assume carry over to the new sets D_{j-1} and H_{j-1}, so this holds for the final setup in $d = 1$ dimensions. Since each step adds at most two new defining hyperplanes, the claim follows.

We apply this recursive decomposition for each cell C of $\mathcal{A}(H)$, and thereby obtain the entire vertical decomposition $\mathcal{V}(H)$. We remark though that our algorithm does not explicitly construct $\mathcal{V}(H)$. In fact, it does not even construct the (full discrete representation of the) prism of $\mathcal{V}(H)$ containing the query point $\mathbf{x}$. It will be clear shortly from the presentation what the algorithm actually constructs. The description given above, while being constructive, is made only to define the relevant notions, and to set the infrastructure within which our algorithm will operate.

3 ε-Cuttings from Vertical Decompositions

Given a finite collection H of hyperplanes in $\mathbb{R}^d$, by an *ε-cutting* for H we mean a subdivision of space into prism-like cells, of the form just defined, that we simply refer to as prisms[8], such that every cell is *crossed* by (i.e., the interior of the cell is intersected by) at most $\varepsilon|H|$ hyperplanes of H, where $0 < \varepsilon < 1$ is the parameter of the cutting. ε-cuttings are a major tool for a variety of applications, including our own; they have been established and developed in several fundamental studies [10, 11, 27].

Roughly speaking, when d is a (small) constant, the random sampling theory of Clarkson [11] (see also Clarkson and Shor [12]) produces an ε-cutting as follows. We draw[9] a random sample R of $\frac{c_d}{\varepsilon}\log\frac{d}{\varepsilon}$ hyperplanes from H, where c_d is a parameter that depends only on d; its actual dependence on d becomes a major issue in the analysis when d is large. We then construct the arrangement $\mathcal{A}(R)$ of R and its vertical decomposition $\mathcal{V}(R)$. With a suitable choice of c_d, the random sampling technique of Clarkson [11] then guarantees, with constant (high) probability, that each prism of $\mathcal{V}(R)$ is crossed by at most $\varepsilon|H|$ hyperplanes of H. (This also follows from the ε-net theory of Haussler and Welzl [23], but, as it turns out, the coefficient c_d has to be much larger when d is large; see below and [16] for more details.)

3.1 The Clarkson Framework

We keep denoting by H a set of n hyperplanes in $\mathbb{R}^d$. Following the definitions and notations in [22, Chapter 8], put $\mathcal{T} = \mathcal{T}(H) := \bigcup_{S\subseteq H} \mathcal{V}(S)$; that is, $\mathcal{T}$ is the set of all possible prisms

[8] In the original studies (see, e.g., [10]), these subcells were taken to be simplices, although both forms have been used in the literature by now.

[9] We use an alternative drawing mode, in which, to get a sample of size r, we sample each element of H independently with probability $p = r/|H|$. The size of the sample is r only in expectation, but this does not affect (in fact, it simplifies) the overall analysis; see, e.g., [30].

defined by the subsets of H. For each prism $\tau \in \mathcal{T}$, we associate with τ its *defining set* $D(\tau)$ and its *conflict set* $K(\tau)$. The former is the smallest subset $D \subseteq H$ such that τ is a prism in $\mathcal{V}(D)$, and the latter is the set of all hyperplanes $h \in H$ for which τ does not appear in $\mathcal{V}(D \cup \{h\})$; they are precisely the hyperplanes in $H \setminus D$ that cross τ. By our discussion in Section 2 we have $|D(\tau)| \leq 2d$, for each $\tau \in \mathcal{T}$.

We have the following two axioms, which hold for any subset $S \subseteq H$:

(i) For any $\tau \in \mathcal{V}(S)$, we have $D(\tau) \subseteq S$ and $K(\tau) \cap S = \emptyset$.

(ii) If $D(\tau) \subseteq S$ and $K(\tau) \cap S = \emptyset$, then $\tau \in \mathcal{V}(S)$.

A key novel property of vertical decompositions, which we establish in this paper, is the following result (some highlights of whose proof are given at the end of this section):

▶ **Theorem 5.** *Let H be a set of n hyperplanes in $\mathbb{R}^d$. Then the cardinality of $\mathcal{T}(H)$ is at most $O\left(\frac{2^{2d}}{d^{7/2}} n^{2d}\right)$.*

In particular, we get a sharper bound on the complexity of vertical decompositions:

▶ **Corollary 6.** *Let H be a set of n hyperplanes in $\mathbb{R}^d$. Then the number of prisms in $\mathcal{V}(H)$ is at most $O\left(\frac{2^{2d}}{d^{7/2}} n^{2d}\right)$.*

▶ Remark. As already mentioned in the introduction, the bound in Corollary 6 significantly improves the previous upper bound of [9] in terms of its dependence on d, in that its "constant" of proportionality drops from $2^{O(d^2)}$ to less than 4^d. We pay a small price (it is small unless n is huge relative to d) in terms of the dependence on n, which is n^{2d} in the new bound, instead of n^{2d-4} in [25] (and only $O(n^d)$ if one uses instead bottom-vertex triangulation). See below for an additional discussion of this issue.

Equipped with Theorem 5, we obtain (we omit the standard proof, which follows the analysis in [11, 12] [10]):

▶ **Theorem 7.** *Given a set H of n hyperplanes in d-space, and a parameter $\varepsilon \in (0, 1)$, a random sample R of $O\left(\frac{d}{\varepsilon} \log \frac{d}{\varepsilon}\right)$ hyperplanes of H (with an appropriate absolute constant of proportionality) satisfies, with constant probability, the property that each prism in the vertical decomposition $\mathcal{V}(R)$ of $\mathcal{A}(R)$ is crossed by at most $\varepsilon|H|$ hyperplanes of H.*

▶ Remark.

1. To turn this random sampling into a procedure that generates an ε-cutting almost surely, we draw a random sample R of the aforementioned (expected) size, and test whether it satisfies the property asserted in Theorem 7. If not, we simply discard this sample and repeat the construction with a new sample. Clearly, since we fail with constant probability (which we can make rather small by increasing the constant of proportionality), the expected number of trials till a successful sample is drawn is constant (close to 1).
2. Our construction uses vertical decomposition. Expanding upon an earlier made comment, we note that an alternative construction, for arrangements of hyperplanes, is the *bottom-vertex triangulation* (see [1]). It has the advantage, over vertical decomposition, that the (bound on the) number of cells (simplices) that it produces is significantly smaller (at most $|R|^d$), but its major disadvantage for the analysis in this paper is that the typical size of a defining set of a simplex in this decomposition is $d_0 = d(d+3)/2$, as opposed to the much smaller value $d_0 = 2d$ for vertical decomposition; see above, [1], and the

[10] It is important to notice that we can follow this analysis because the coefficient is only singly exponential in d.

full version [16], for more details. The fact that prisms in the vertical decomposition have such a smaller bound on the size of their defining sets, combined with our improved bound on the complexity of vertical decomposition, is what makes vertical decomposition a superior technique for the (decision-tree complexity of the) k-SUM problem.

3. We also remark that the method that we use here is not optimal, from a general perspective, in several aspects: First, it does not involve the refining second resampling stage of Chazelle and Friedman [10] (and of others), which leads to a slight improvement in the number of cells (or, alternatively, to a smaller required sample size). More significantly, in $d \geq 5$ dimensions there are no sharp known bounds on the complexity of the vertical decomposition, even for arrangements of hyperplanes (see [9, 24] for the general case, and [21, 25] for the case of hyperplanes). Nevertheless, these issues are irrelevant for the technique employed here (mainly because, as already mentioned, we will not construct the entire vertical decomposition), and the coarser method reviewed above serves our purposes just fine.
4. Finally, we note that, in principle, we could have also used the ε-net theory of [23] to ensure the ε-cutting property of the resulting decomposition. However, the VC-dimension of the suitably defined corresponding range space is much larger than $2d$. Concretely, it follows from the analysis in the full version [16] that the VC-dimension is $O(d^3)$ and $\Omega(d^2)$. Since the size of the random sample in the theory in [23] has to be (slightly more than) proportional to the VC-dimension, this approach results in much poorer bounds, which will cause our algorithm to be at least as slow as the one in [7].

3.2 Key Properties in the Proof of Theorem 5

Following the presentation in Section 2, we first analyze the complexity of the vertical decomposition of a single cell of $\mathcal{A}(H)$, and then derive a global bound for the entire arrangement. Due to lack of space, we only present here a key property of the analysis, which is also crucial for the analysis of our k-SUM algorithm presented in Section 4.

Let C be a fixed cell of $\mathcal{A}(H)$. With a slight abuse of notation, denote by n the number of its facets (that is, the number of hyperplanes of H that actually appear on its boundary), and consider the procedure of constructing its vertical decomposition, as described in Section 2. As we recall, the first stage produces vertical prisms, each having a fixed floor and a fixed ceiling. We take each such prism, whose ceiling and floor are contained in two respective hyperplanes h_1, h_2 of H, project it onto the hyperplane $x_d = 0$, and decompose the projection C_{d-1} recursively.

The $(d-2)$-hyperplanes that bound C_{d-1} are projections of intersections of the form $h \cap h_1$, $h \cap h_2$, for $h \in H \setminus \{h_1, h_2\}$, including also $h_1 \cap h_2$, if it arises. In principle, the number of such hyperplanes is at most $2n-1$, but, as shown in the following lemma the actual number is smaller:

▶ **Lemma 8.** *Let τ be a first-stage prism, whose ceiling and floor are contained in two respective hyperplanes h_1, h_2. Then for each hyperplane $h \in H$, $h \neq h_1, h_2$, the following holds.*

(a) *If h is nonvertical then only one of $g_1 := h_1 \cap h$ or $g_2 := h_2 \cap h$ can appear on $\partial\tau$. It is g_1 if C lies below h, and g_2 if C lies above h.*

(b) *If h is vertical, both g_1 and g_2 can appear on $\partial\tau$, but their projections onto $x_d = 0$ coincide.*

Proof.

(a) Assume that h is nonvertical. Then either C lies fully above h or it lies fully below h. Without loss of generality, assume that the former case holds. Since C lies below h_1, the intersection $g_1 = h \cap h_1$, if it shows up on ∂C at all, must bound an equator facet φ of C. If φ appears on $\partial\tau$, then the interior of τ must contain a vertical segment whose endpoints lie on h (bottom) and on h_1 (top), contradicting the fact that the floor of τ lies on h_2. Hence only g_2 can appear on $\partial\tau$. The case where C lies below h is handled symmetrically.

The proof of (b) is straightforward; it follows from the fact that h_1 and h_2 are nonvertical, and in fact both projections of g_1 and g_2 coincide with that of the entire h. ◀

▶ Remark. An important feature of the proof is that it also holds when τ is *any* convex vertical prism, obtained at any recursive step of the decomposition, and, in particular, when τ is the vertical prism obtained at the final step.

It is straightforward to verify that Lemma 8 implies that the projection of τ onto $x_d = 0$ has at most $n-1$ facets. Using this property we derive a recurrence relation to bound the complexity of the vertical decomposition of a single cell C, and then, using axioms (i)–(ii), we obtain a bound for the entire arrangement. These details appear in the full paper [16].

4 The Algorithm

4.1 Algorithm outline

The high-level approach of our algorithm can be regarded as an optimized variant of the algorithm of Cardinal *et al.* [7], which is inspired by the point-location mechanism of Meiser [28]. We choose $\varepsilon > 0$ to be a constant, smaller than, say, $1/2$, and apply the ε-cutting machinery, as reviewed in Theorem 7. For a given input point $\mathbf{x}$, the algorithm proceeds as follows.

- **(i)** Construct a random sample R of $r := O\left(\frac{n}{\varepsilon}\log\frac{n}{\varepsilon}\right) = O(n\log n)$ hyperplanes of H, with a suitable absolute constant of proportionality (recall that in our application, the dimension of the underlying space is n). If R violates the ε-cutting property asserted in Theorem 7, discard R and repeat the process with a new sample.
- **(ii)** Construct the prism $\tau = \tau_{\mathbf{x}}$ of $\mathcal{V}(R)$ that contains the input point $\mathbf{x}$. If at that step we detect an original hyperplane of H that contains $\mathbf{x}$, we stop and return "YES".
- **(iii)** Construct the conflict list $CL(\tau)$ of τ (the subset of hyperplanes of H that cross τ), and recurse on it.
- **(iv)** Stop as soon as $|CL(\tau)|$ is smaller than the sample size $r = O\left(\frac{n}{\varepsilon}\log\frac{n}{\varepsilon}\right)$ (we use the same sample size in all recursive steps). When that happens, test $\mathbf{x}$, in brute force, against each original hyperplane of H in $CL(\tau)$; return "YES" if one of the tests results in an equality, and "NO" otherwise.

We note that those parts of the algorithm that do not depend[11] on $\mathbf{x}$, which are costly in the RAM model, are performed here for free. That is, our goal at this point is only to bound the number of linear queries performed by the algorithm. We also note that although the construction of the prism containing $\mathbf{x}$ (described below) is conceptually simple, it involves several technical details, which mainly follow from the fact that H may contain vertical hyperplanes (vs. the simpler scenario where all hyperplanes are in general position).

[11] By this we mean that they do not compute any explicit expression that depends on the coordinates of $\mathbf{x}$. They might (and in general, will) depend on previously computed discrete data that does depend on $\mathbf{x}$, but accessing this data in our model, once computed, is for free.

We next describe the details of implementing step (ii). We comment that step (i) costs nothing in our model, so we do not bother with its implementation details. The tests in step (iii), although being very costly in the "full" standard RAM model of computation, are independent of the specific coordinates of $\mathbf{x}$, and thus cost nothing in our model. We present this step in detail in the full version of this paper.

Constructing the prism containing $\mathbf{x}$. Since the overall complexity of a prism (the number of its faces of all dimensions) is exponential in the dimension n, we do not construct it explicitly. Instead we only construct explicitly its at most $2n$ bounding hyperplanes, consisting of a floor and a ceiling (or only one of them in case the cell $C_{\mathbf{x}}$ in $\mathcal{A}(R)$ containing $\mathbf{x}$ is unbounded), and at most $2n-2$ vertical walls (we have strictly fewer than $2n-2$ walls in cases when either the floor of τ intersects its ceiling (on $\partial\tau$), or when this happens in any of the projections τ^* of τ in lower dimensions, or when the current subcell becomes unbounded at any of the recursive steps). The prism τ, as defined in step (ii), is then implicitly represented as the intersection of the halfspaces bounded by these hyperplanes and containing $\mathbf{x}$. Let $H_{\mathbf{x}}$ denote this set of at most $2n$ hyperplanes. From now on we assume, to simplify the presentation but without loss of generality, that $C_{\mathbf{x}}$ is bounded, and that τ has exactly $2n-2$ vertical walls (and thus exactly $2n$ bounding hyperplanes).

The following recursive algorithm constructs $H_{\mathbf{x}}$, and also detects whether $\mathbf{x}$ lies on one of the bounding hyperplanes of τ. Let $r = O\left(\frac{n}{\varepsilon}\log\frac{n}{\varepsilon}\right)$ denote the (expected) size of our sample R. Initially, we set $H_{\mathbf{x}} := \emptyset$. We first perform r linear queries with $\mathbf{x}$ and each of the hyperplanes of R, resulting in a sequence of r output labels "above" / "below" / "sideways" / "on". At the top level of recursion (before reducing the dimension), encountering a label "on" means that $\mathbf{x}$ lies on an original hyperplane of H, and thus there is a positive solution to our instance of k-SUM, and we stop the entire procedure and output "YES". At deeper recursive levels (in lower-dimensional spaces), when we encounter "on", we need to check that the relevant (now lower-dimensional) hyperplane is fully contained in an original hyperplane of H, in order to output "YES" (the full containment condition is addressed later on). As will be discussed below, such a hyperplane does not have to belong to R, so the procedure for performing this test, and in particular its analysis, is rather elaborate.

We thus assume, without loss of generality, that all labels are "above", "below", or "sideways". We next partition the set of the hyperplanes in R according to their labels, letting R_1 denote the set of hyperplanes lying above $\mathbf{x}$, R_2 the set of hyperplanes below it, and R_0 the set of vertical hyperplanes to the side of $\mathbf{x}$. We then identify the upper hyperplane $h_1 \in R_1$ and the lower hyperplane $h_2 \in R_2$ with shortest vertical distances from $\mathbf{x}$. We do this by computing the minimum of these vertical distances, each of which is a linear expression in $\mathbf{x}$, using $(|R_1|-1)+(|R_2|-1) < r$ additional comparisons. The hyperplanes h_1 and h_2 contain the ceiling and the floor of τ, respectively, and we thus insert them into $H_{\mathbf{x}}$.

In order to produce the hyperplanes containing the vertical walls of τ, we recurse on the dimension n. This process somewhat imitates the one producing the entire vertical decomposition of $C_{\mathbf{x}}$ described above. However, the challenges in the current construction are to build only the single prism containing $\mathbf{x}$, to keep the representation implicit, and to do this efficiently.

We generate all pairwise intersections $h_1 \cap h$ and $h_2 \cap h$, for $h \in R$, $h \neq h_1$, $h \neq h_2$, and obtain two collections G_1, G_2 of $(n-2)$-dimensional flats, each of size at most $r-2$, which we project onto the hyperplane $x_n = 0$.

By Lemma 8 (when the input set is now R) and the remark after it, the following holds for each $h \in R \setminus \{h_1, h_2\}$. Put $g_1 := h_1 \cap h$ and $g_2 := h_2 \cap h$. (a) If h is nonvertical then

at most one of g_1, g_2 can appear on $\partial\tau$. (b) If h is vertical, the projections of g_1 and g_2 coincide, and are in fact equal to the projection of the entire h. We can therefore discard one of g_1, g_2 when h is nonvertical (using the simple rule in Lemma 8), and replace both by the single projection of h, when h is vertical. Hence, the subset $G \subseteq G_1 \cup G_2$ of the surviving intersections consists of at most $|R| - 2$ flats (of dimension $n-2$). We denote by $R^{(1)}$ the set of their projections onto the hyperplane $x_n = 0$. (If $h_1 \cap h_2$ is also relevant, we add its projection to $R^{(1)}$, making its size go up to $|R| - 1$.)

We continue the construction recursively on $R^{(1)}$ in $n-1$ dimensions. That is, at the second iteration, we project $\mathbf{x}$ onto the subspace $x_n = 0$; let $\mathbf{x}^{(1)}$ be the resulting point. We first perform at most r linear tests with $\mathbf{x}^{(1)}$ and each of the hyperplanes in $R^{(1)}$. If we encounter "on" for some $h^{(1)} \in R^{(1)}$ then $\mathbf{x}$ lies on a vertical wall of τ passing through $h^{(1)}$. If $h^{(1)}$ is fully contained in a (vertical) hyperplane $h' \in H$, we output "YES". We emphasize that h' does not have to belong to R (see a discussion of this issue in the proof of correctness, given below), so, to determine whether such an h' exists, we simply test $h^{(1)}$ against all hyperplanes of H (which costs nothing in our model). Otherwise, if we encountered "on" for some $h^{(1)} \in R^{(1)}$ in the above test, but $h^{(1)}$ is not contained in any vertical hyperplane of H, then the prism τ containing $\mathbf{x}$ (in the original n-space) is of one lower dimension (or, alternatively, $\mathbf{x}$ lies on a facet of a full-dimensional prism, which projects to a portion of $h^{(1)}$). In this case, we can intersect all the remaining hyperplanes in $R^{(1)}$ with $h^{(1)}$, projecting the whole setting to the hyperplane $x_{n-1} = 0$, and continue the construction recursively within that hyperplane.

The general flow of the recursive procedure is as follows. At each step i, for $i = 1, 2, \ldots, n$, we have a collection $R^{(i-1)}$ of at most $|R|$ hyperplanes of dimension $n-i$, and a point $\mathbf{x}^{(i-1)}$, in the $x_1 \cdots x_{n-i+1}$-hyperplane (for $i = 1$ we have $R^{(0)} = R$ and $\mathbf{x}^{(0)} = \mathbf{x}$). We first test whether $\mathbf{x}^{(i-1)}$ lies on any of the hyperplanes $h^{(i-1)}$ in $R^{(i-1)}$. If so, we test whether $h^{(i-1)}$ is contained in an original hyperplane of H (essentially[12] a hyperplane that is parallel to all the coordinates $x_{n-i+2}, \ldots, x_n$ that we have already processed, that is, vertical in all of them), and, if so, we output "YES". If $\mathbf{x}^{(i-1)}$ lies on some $h^{(i-1)}$ (but no original hyperplane of H contains $h^{(i-1)}$), we recurse in one lower dimension, as described for the case $i = 2$. Otherwise, we assume, without loss of generality, that no "on" label is produced. We find the pair of hyperplanes that lie respectively above and below $\mathbf{x}^{(i-1)}$ in the x_{n-i+1}-direction, and are closest to $\mathbf{x}^{(i-1)}$ in that direction (they support the "ceiling" and "floor" of the recursive prism, in the x_{n-i}-direction), and then produce a set $R^{(i)}$ of fewer than $|R|$ hyperplanes of dimension $(n-i-1)$ in the $x_1 \cdots x_{n-i}$-hyperplane. We also project $\mathbf{x}^{(i-1)}$ onto this hyperplane, thereby obtaining the next point $\mathbf{x}^{(i)}$. The construction of $R^{(i)}$ is performed similarly to the way it is done in case $i = 1$, described above, and Lemma 8 (and the remark following it) continues to apply, so as to ensure that indeed $|R^{(i)}|$ continues to be (progressively) smaller than $|R|$, and that its members are easy to construct.

We stop when we reach $i = n$, in which case we are given a set of at most $|R|$ points on the real line, and we locate the two closest points to the final projected point $\mathbf{x}^{(n)}$.

To complete the construction, we take each of the hyperplanes $h_1^{(i-1)}$, $h_2^{(i-1)}$, obtained at each of the iterations $i = 2, \ldots, n$, and lift it "vertically" in all the remaining directions $x_{n-i+2}, \ldots, x_n$, and add the resulting $(n-1)$-hyperplanes in $\mathbb{R}^n$ to $H_{\mathbf{x}}$. We comment that in case $\mathbf{x}$ lies on a facet (or, more generally, a lower dimensional face) of τ, we need to confine these liftings to the appropriate flat h containing this face(t).

[12] To be precise, it could also be that, accidentally, some other original hyperplane contains $h^{(i-1)}$.

▶ Remark. The importance of Lemma 8 is that it controls the sizes of the sets $R^{(i)}$, $i \geq 1$. Without the filtering that it provides, the size of each $R^{(i)}$ would be roughly twice the size of $R^{(i-1)}$, and the query would then require exponentially many linear tests. This doubling effect shows up in the original analysis of the complexity of vertical decompositions [9].

Algorithm correctness. Let h be a hyperplane of H that contains $\mathbf{x}$. Clearly, h must intersect the interior or the boundary of τ. In the former case, h belongs to $CL(\tau)$, and will be passed down the recursion. In the latter case, either h is the floor or ceiling of τ, and then the first stage of constructing τ will detect that $\mathbf{x} \in h$. Otherwise h must be an x_d-vertical hyperplane. Indeed, no other original hyperplane can meet τ unless it intersects its floor or ceiling; since $\mathbf{x}$ was found not to lie on the floor or the ceiling, it cannot lie on any nonvertical h. If h is in R, then the algorithm outputs "YES". It is possible, though, that $h \notin R$, in which case, since h does not intersect the interior of τ, it does not belong to $CL(\tau)$, and we risk missing h altogether. (Clarkson's theory, in the context used in this paper, does not control the number of hyperplanes that touch τ without crossing it.) However, if $\mathbf{x}$ does lie on h then $\mathbf{x}$ must lie on $\partial\tau$, and one of the recursive steps in the construction of τ will detect this fact. In the full version of this paper we describe a procedure, already alluded to several times earlier, that tests for this property, and establish its correctness.

We emphasize that at each recursive step we construct the prism τ only with respect to the conflict list of its parent cell τ_0 (initially, $\tau_0 = \mathbb{R}^d$ and $CL(\tau_0) = H$), implying that τ is not necessarily contained in τ_0. In other words, the sequence of cells τ constructed in our algorithm are spatially in no particular relation to one another (except that all of them contain $\mathbf{x}$). Still, this does not harm the correctness of the search process, a claim that is argued as follows. The fact that $\mathbf{x}$ lies in τ_0 and in τ implies that it can only lie either on one of the hyperplanes in $CL(\tau_0)$ or on one of the (at most $2d$) bounding hyperplanes of τ. The latter situation will be detected during the non-recursive processing of τ, during which the algorithm (step (ii)) will test $\mathbf{x}$ against each of the bounding hyperplanes of τ (once again, we describe this in more detail in the full paper). If it finds that $\mathbf{x}$ lies on such a hyperplane, it then determines, as mentioned above, whether that hyperplane is an original hyperplane of H (or, more precisely, of $CL(\tau_0)$). Hence, if $\mathbf{x}$ lies on some hyperplane $h \in H$, and this fact has not yet been detected, h will be passed to the recursion as an element of $CL(\tau)$ at step (iii). The case where τ is lower-dimensional is handled in a similar manner.

We next claim that the algorithm terminates (almost surely). Indeed, at each recursive step, the sample R is drawn from the corresponding subset $CL(\tau_0)$. Applying Theorem 7 to $CL(\tau_0)$, we obtain that the conflict list of the next prism τ contains (with certainty, due to the test applied at step (i)) only at most $\varepsilon|CL(\tau_0)|$ hyperplanes of $CL(\tau_0)$. Hence, with probability 1, after a logarithmic number of steps (see below for the concrete analysis) we will reach step (iv), and then the algorithm will correctly determine whether $\mathbf{x}$ lies on a hyperplane of H (by the invariant that we maintain, any such hyperplane belongs to the final conflict list $CL(\tau)$).

(The termination is guaranteed only almost surely, because of the possibility of the event (that has probability 0) of repeatedly failing to choose a good sample at some recursive application of step (i).)

The query complexity. Due to lack of space, we describe the analysis of the query complexity very briefly, and postpone the remaining details to the full paper. Roughly speaking, at each recursive step in the construction of τ we need to perform $O(r)$ linear queries in order to determine, for each hyperplane $h \in R$ whether it lies above, below, on, or sideways from $\mathbf{x}$,

and then find the ceiling and floor hyperplanes h_1 and h_2. In order to test, for a nonvertical hyperplane $h \in R \setminus \{h_1, h_2\}$, which of $g_1 := h_1 \cap h$ or $g_2 := h_2 \cap h$ can appear on $\partial\tau$ (or, more specifically, on the boundary of the *undecomposed* convex prism $\bar{\tau}$ containing τ), we use the simple rule provided in Lemma 8 (which holds in any dimension $i \leq n$). This eventually implies that we spend a total of $O(n^2 \log n)$ linear queries over all n steps of the recursion (on the dimension), for a grand total of $O(n^2 \log n \log |H|) = O(kn^2 \log^2 n)$ linear queries, over all $O(\log |H|)$ steps of the algorithm.

This completes the proof of Theorem 3 for the k-SUM problem. The analysis proceeds more or less verbatim to the more general case of k-LDT with the same performance bound, and generalizes, also trivially, to the cases of SubsetSum and Knapsack. We omit the easy details in this version. ◀

Concluding remarks and open problems. It looks likely that the number of queries can be brought down to $O(n^2 \log n)$. To fit into the general theory of Clarkson, we have drawn a sample R of size $O(n \log n)$. The logarithmic factor is needed if we want to ensure (with constant, high probability) that the ε-cutting property holds for *all* prisms that arise in $\mathcal{V}(R)$, but we only need this property to hold for the single prism that contains $\mathbf{x}$. With a smaller sample size $O(n)$, and with some extra care, we seem to obtain an expected number of $O(n^2 \log n)$ queries. We plan to present this improvement in the full version [16], where this under-sampling technique is referred to as *optimal sampling*. Applying this improvement to the Knapsack or the SubsetSum problems, the number of queries goes down to $O(n^3)$.

We show in the full version that our algorithm, when cast into the RAM model, has an implementation whose expected running time is $n^{k+O(1)}$ (but still using only $O(n^2 \log^2 n)$ linear queries on $\mathbf{x}$). An interesting open problem is whether the running time can be improved to roughly $O(n^{\lceil k/2 \rceil})$, while still using only nearly-quadratically many linear queries. A similar result was obtained in the previous work of Cardinal *et al.* [7] (but with a nearly-cubic number of linear queries). We hope to obtain a similar improvement for the approach used in this paper.

Acknowledgments. The authors would like to thank Shachar Lovett, Sariel Har-Peled, and Haim Kaplan for many useful discussions.

References

1 P. K. Agarwal and M. Sharir, Arrangements and their applications, In *Handbook of Computational Geometry*, (J. Sack and J. Urrutia, eds.), Elsevier, Amsterdam, pp. 973–1027, 2000.

2 N. Ailon and B. Chazelle, Lower bounds for linear degeneracy testing, *J. ACM*, 52(2):157–171, 2005.

3 B. Aronov and S. Har-Peled, On approximating the depth and related problems, *SIAM J. Comput.* 38:899–921, 2008.

4 G. Barequet and S. Har-Peled, Polygon-containment and translational min-Hausdorff-distance between segment sets are 3SUM-hard, *Int'l J. Comput. Geometry Appl.*, 11(4):465–474, 2001.

5 M. Ben-Or, Lower bounds for algebraic computation trees, In *Proc. 16th Annual ACM Symp. Theory Comput. (STOC)*, pp. 80–86, 1983.

6 R. G. Bland, D. Goldfarb, and M. J. Todd, The ellipsoid method: A survey, *Operations Research*, 29(6):1039–1091, 1981.

7 J. Cardinal, J. Iacono, and A. Ooms, Solving k-SUM using few linear queries, In *Proc. 24th European Symp. Alg. (ESA)*, 2016, 25:1–25:17.

8 T. M. Chan and M. Lewenstein, Clustered integer 3SUM via additive combinatorics, In *Proc. 47th Annual ACM Symp. Theory Comput. (STOC)*, pp. 31–40, 2015.

9 B. Chazelle, H. Edelsbrunner, L. Guibas and M. Sharir, A singly exponential stratification scheme for real semi–algebraic varieties and its applications, *Theoret. Comput. Sci.*, 84:77–105, 1991. Also in *Proc. 16th Int'l Colloq. on Automata, Languages and Programming*, 1989, pp. 179–193.

10 B. Chazelle and J. Friedman, A deterministic view of random sampling and its use in geometry, *Combinatorica*, 10:229–249, 1990.

11 K. L. Clarkson, New applications of random sampling in computational geometry, *Discrete Comput. Geom.*, 2:195–222, 1987.

12 K. L. Clarkson and P. W. Shor, Applications of random sampling in computational geometry, II, *Discrete Comput. Geom.*, 4:387–421, 1989.

13 D. Dobkin and R. Lipton, A lower bound of $n^2/2$ on linear search programs for the Knapsack problem, *J. Comput. Syst. Sci.*, 16(3):413–417, 1978.

14 D. Dobkin and R. Lipton, On the complexity of computations under varying set of primitives, *J. Comput. Syst. Sci.*, 18:86–91, 1979.

15 J. Erickson, Lower bounds for linear satisfiability problems, *Chicago. J. Theoret. Comput. Sci.*, 8:388–395, 1997.

16 E. Ezra, S. Har-Peled, H. Kaplan and M. Sharir, Decomposing arrangements of hyperplanes: VC-dimension, combinatorial dimension, and point location, Manuscript, 2017.

17 A. Freund, Improved Subquadratic 3SUM, *Algorithmica*, 77(2):440–458, 2017.

18 A. Gajentaan and M. H. Overmars, On a class of $O(n^2)$ problems in computational geometry, *Comput. Geom. Theory Appl.*, 5:165–185, 1995.

19 O. Gold and M. Sharir, Improved bounds on 3SUM, k-SUM, and linear degeneracy, *CoRR abs*/1512.05279v2, 2017.

20 A. Grønlund and S. Pettie, Threesomes, degenerates, and love triangles, In *Proc. 55th Annual Symp. Found. Comput. Sci.*, pp. 621–630, 2014.

21 L. J. Guibas, D. Halperin, J. Matoušek, and M. Sharir, On vertical decomposition of arrangements of hyperplanes in four dimensions, *Discrete Comput. Geom.*, 14:113–122, 1995.

22 S. Har-Peled, *Geometric Approximation Algorithms*, Mathematical Surveys and Monographs, Vol. 173, AMS Press, Providence, RI, 2011.

23 D. Haussler and E. Welzl, ε-nets and simplex range queries, *Discrete Comput. Geom.*, 2:127–151, 1987.

24 V. Koltun, Almost tight upper bounds for vertical decompositions in four dimensions, *J. ACM* 51(5):699–730, 2004.

25 V. Koltun, Sharp bounds for vertical decompositions of linear arrangements in four dimensions, *Discrete Comput. Geom.*, 31(3):435–460, 2004.

26 D. Liu, A note on point location in arrangements of hyperplanes, *Inform. Process. Letts.*, 90(2):93–95, 2004.

27 J. Matoušek, Cutting hyperplane arrangements, *Discrete Comput. Geom.*, 6:385–406, 1991.

28 S. Meiser, Point location in arrangements of hyperplanes, *Information Comput.*, 106(2):286–303, 1993.

29 F. Meyer auf der Heide, A polynomial linear search algorithm for the n-dimensional knapsack problem, *J. ACM*, 31:668–676, 1984.

30 M. Sharir, The Clarkson–Shor technique revisited and extended, *Combinat. Probab. Comput.*, 12:191–201, 2003.

31 M. Steele and A. Yao, Lower bounds for algebraic decision trees, *J. Alg.*, 3:1–8, 1982.

Computing the Fréchet Gap Distance*†

Chenglin Fan[1] and Benjamin Raichel[2]

1 **Dept. of Computer Science, University of Texas at Dallas, Dallas, TX, USA**
`cxf160130@utdallas.edu`
2 **Dept. of Computer Science, University of Texas at Dallas, Dallas, TX, USA**
`benjamin.raichel@utdallas.edu`

Abstract

Measuring the similarity of two polygonal curves is a fundamental computational task. Among alternatives, the Fréchet distance is one of the most well studied similarity measures. Informally, the Fréchet distance is described as the minimum leash length required for a man on one of the curves to walk a dog on the other curve continuously from the starting to the ending points. In this paper we study a variant called the Fréchet gap distance. In the man and dog analogy, the Fréchet gap distance minimizes the difference of the longest and smallest leash lengths used over the entire walk. This measure in some ways better captures our intuitive notions of curve similarity, for example giving distance zero to translated copies of the same curve.

The Fréchet gap distance was originally introduced by Filtser and Katz [19] in the context of the discrete Fréchet distance. Here we study the continuous version, which presents a number of additional challenges not present in discrete case. In particular, the continuous nature makes bounding and searching over the critical events a rather difficult task.

For this problem we give an $O(n^5 \log n)$ time exact algorithm and a more efficient $O(n^2 \log n + \frac{n^2}{\varepsilon} \log \frac{1}{\varepsilon})$ time $(1+\varepsilon)$-approximation algorithm, where n is the total number of vertices of the input curves. Note that for (small enough) constant ε and ignoring logarithmic factors, our approximation has quadratic running time, matching the lower bound, assuming SETH [10], for approximating the standard Fréchet distance for general curves.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems, I.1.2 Algorithms, I.3.5 Computational Geometry and Object Modeling

Keywords and phrases Fréchet Distance, Approximation, Polygonal Curves

Digital Object Identifier 10.4230/LIPIcs.SoCG.2017.42

1 Introduction

Polygonal curves arise naturally in the modeling of a number computational problems, and for such problems assessing the similarity of two curves is one of the most fundamental tasks. There are several competing measures for defining curve similarity. Among these, there has been strong interest in the Fréchet distance, particularly from the computational geometry community, as the Fréchet distance takes into account the continuous "shape" of the curves rather than just the set of points in space they occupy. The Fréchet distance and related measures have been used for a variety of applications [21, 9, 24, 23, 11], and it is typically illustrated as follows. Let the two polygonal curves be denoted π and σ, with n vertices in total. Imagine a man and a dog are respectively placed at the starting vertices of π and σ, and they must each move continuously along their curves to their respective ending points.

* See [18] for the full version (`http://www.utdallas.edu/~bar150630/gap.pdf`).
† Work on this paper was partially supported by NSF CRII Award 1566137.

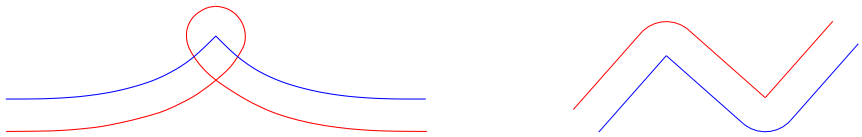


Figure 1 Left: A 2D "airplane roll". Right: Turning in 2D by pivoting on one side at a time.

The man and dog are connected by a leash, and the Fréchet distance is the minimum leash length required over all possible walks of the man and dog, where the man and dog can independently control their speed but cannot backtrack.

In this paper we consider a variant called the *Fréchet gap distance*, originally introduced by Filtser and Katz in the context of the discrete Fréchet distance [19]. In the man and dog analogy, this variant minimizes the difference of the lengths of the longest and shortest leashes used over the entire walk. As discussed in [19], since this measure considers both the closest and farthest relative positions of the man and dog, in many cases it is closer to our intuitive notion of curve similarity. Notably, two translated copies of the *same* curve have Fréchet gap distance zero, as opposed to the magnitude of the translation under the standard Fréchet distance. Though this is not to say that it is the same as minimizing the standard Fréchet distance under translation. For instance, fix any two points on a rigid body in two or three dimensions. The pair of curves traced out by these points as we arbitrarily rotate and translate the rigid body will always have Fréchet gap distance zero (see Figure 1).

A natural scenario for the gap distance is planning the movement of military units, where one wants them to be sufficiently close to support each other in case of need, but sufficiently far from each other to avoid unintended interaction (i.e., friendly fire). Such units might move on two major roads that are roughly parallel to each other, thus matching our setup.

Previous Work. Alt and Godau [4] presented an $O(n^2 \log(n))$ time algorithm to compute the standard Fréchet distance. More recently Buchin et al. [12] improved the logarithmic factor in the running time (building on [1]), however Bringmann [10] showed that assuming the Strong Exponential Time Hypothesis (SETH), no strongly subquadratic time algorithm is possible. Moreover, Bringmann showed that assuming SETH there is no strongly subquadratic 1.001-approximation algorithm, thus ruling out the possibility of a strongly subquadratic PTAS for general curves. On the other hand, there are fast approximation algorithms for several families of nicely behaved curves, for example Driemel et al. [16] gave an $O(cn/\varepsilon + cn \log n)$ time algorithm for the case of c-packed curves.

Many variants of the Fréchet distance between polygonal curves have been considered. Alt and Godau [4] gave a quadratic time algorithm for the weak Fréchet distance, where backtracking on the curves is allowed. Driemel and Har-Peled [15] considered allowing shortcuts between vertices, and for this more challenging variant, they give a near linear time 3-approximation for c-packed curves. Later Buchin et al. [14] proved the general version, where shortcutting is also allowed on edge interiors, is NP-hard (and gave an approximation for the general and an exact algorithm for the vertex case). The *discrete* Fréchet distance only considers distances at the vertices of polygonal curves, i.e. rather than a continuously walking man and dog, there is a pair of frogs hopping along the vertices. This somewhat simpler variant can be solved in $O(n^2)$ time using dynamic programming [17]. Interestingly, Agarwal et al. [1] showed the discrete variant can be solved in weakly subquadratic $O(n^2 \log\log n/\log n)$ time, however the above results of Bringmann [10] also imply there is no strongly subquadratic algorithm for the discrete case, assuming SETH. Avraham et al. [6] considered shortcuts in the discrete case, providing a strongly subquadratic running time, showing shortcuts make it more tractable, which was the reverse for the continuous case.

Minimizing Fréchet distance under translation (and other transformations) was previously considered, though running times are typically large. For example, Alt et al. [5] gave a roughly $O(n^8)$ time algorithm, though they also gave a $O(n^2/\varepsilon^2)$ time $(1+\varepsilon)$-approximation. Avraham et al. [7] consider the discrete case, and provide a nice summary of other previous work. The Fréchet distance has also been extended to more general inputs, such as graphs [3], piecewise smooth curves [22], simple polygons [13], surfaces [2], and complexes [20]. In general there are too many Fréchet distance results to cover, and the above is just a sampling.

The most relevant previous work is that of Filtser and Katz [19], who first proposed the Fréchet gap distance. The technical content of the two papers differs significantly however, as [19] considers the discrete case, avoiding many of the difficulties faced in our continuous setting. In particular, a solution to the gap problem is a distance interval. In the continuous case the challenge is bounding the number of possible intervals, while in the discrete case a bound of $O(n^4)$ holds, as each interval endpoint is a vertex to vertex distance. Using a result of Avraham et al. [7], Filtser and Katz improve this to an $O(n^3)$ time algorithm to compute the minimum discrete Fréchet gap. They also provide $O(n^2 \log^2 n)$ time algorithms for one-sided discrete Fréchet gap with shortcuts and the weak discrete Fréchet gap distance.

Contributions and Overview. Here we consider the continuous Fréchet gap distance problem (defined informally above, and formally below). This is the first paper to consider the more challenging continuous version of this problem. For this problem we provide an $O(n^5 \log n)$ time exact algorithm and a more efficient $O(n^2 \log n + \frac{n^2}{\varepsilon} \log \frac{1}{\varepsilon})$ time $(1+\varepsilon)$-approximation algorithm, and we now outline our approach and main contributions.

The standard approach for computing the Fréchet distance starts by solving the decision version for a given query distance $\delta \geq 0$, by using the free space diagram, which describes the pairs of points (one from each curve) which are within distance δ. The convexity of the free space cells allows one to efficiently propagate reachibility information, leading to a quadratic time proceedure overall. For the Fréchet gap problem the free space cells are no longer convex, but despite this we show that they have sufficient structure to allow efficient reachability propagation, again leading to a quadratic time decider, which in our case determines whether a given query interval $[s,t]$ is feasible.

The next step in computing the Fréchet distance is to find a polynomially sized set of critical events, determined by the input curves, to search over. For the standard Fréchet distance this set has $O(n^3)$ size. For the Fréchet gap case however the number of critical events can be much larger as they are determined by two rather than one distance value. As mentioned above, for the discrete case only pairs of vertex distances are relevant and so there are $O(n^4)$ events. On the other hand, for the continuous case there can now be "floating" monotonicity events where increasing (or decreasing) the gap interval endpoint values simultaneously may lead to an entire continuum of optimum intervals. Despite this we show there is an $O(n^6)$ sized set of canonical intervals containing an optimum solution.

The last step is efficiently searching over the critical events. For the standard Fréchet distance this can be done via parametric search [4] or sampling [20], yielding an $O(n^2 \log n)$ running time. Searching in the gap case however is more challenging, as there is no longer a natural linear ordering of events. Specifically, the set of feasible intervals may not appear contiguously when ordering candidate intervals by width. Despite this, we similarly get a near linear factor speed up, by using a more advanced version of the basic approach in [20].

Our approximation uses the observation that all feasible intervals share a common value. Roughly speaking, at the cost of a 2-approximation, this allows us to consider the radius of intervals centered at this common value, rather than two independent interval endpoints, reducing the number of critical events. This is improved to a $(1+\varepsilon)$-approximation, and finally the running time is reduced by a linear factor, again using a modified version of [20].

2 Preliminaries

Throughout, given points $p, q \in \mathbb{R}^d$, $||p-q||$ denotes their Euclidean distance. Moreover, given two (closed) sets $P, Q \subseteq \mathbb{R}^d$, $dist(P,Q) = \min_{p\in P, q\in Q} ||p-q||$ denotes their distance.

2.1 Fréchet Distance and Fréchet Gap Distance

A *polygonal curve* π of length n is a continuous mapping from $[0,n]$ to $\mathbb{R}^d$, such that for any integer $1 \leq i \leq n$, the restriction of π to the interval $[i-1,i]$ is defined by $\pi((i-1)+\alpha) = (1-\alpha)\pi(i-1) + \alpha\pi(i)$ for any $\alpha \in [0,1]$, i.e. a straight line segment. When it is clear from the context, we often use π to denote the image $\pi([0,n])$. The set of vertices of π is defined as $V(\pi) = \{\pi_0, \pi_1 \ldots, \pi_n\}$, where $\pi_i = \pi(i)$, and the set of edges is $E(\pi) = \{\pi_0\pi_1, \ldots, \pi_{n-1}\pi_n\}$, where $\pi_{i-1}\pi_i$ is the line segment connecting π_{i-1} and π_i.

A reparameterization for a curve π of length n is a continuous non-decreasing bijection $f : [0,1] \to [0,n]$ such that $f(0) = 0, f(1) = n$. Given reparameterizations f, g of an n length curve π and an m length curve σ, respectively, the *width* between f and g is defined as

$$width_{f,g}(\pi,\sigma) = \max_{\alpha\in[0,1]} ||\pi(f(\alpha)) - \sigma(g(\alpha))||\,.$$

The (standard) *Fréchet distance* between π and σ is then defined as

$$d_{\mathcal{F}}(\pi,\sigma) = \min_{f,g} width_{f,g}(\pi,\sigma)$$

where f, g range over all possible reparameterizations of π and σ.

A *gap* is an interval $[s,t]$ where $0 \leq s \leq t$ are real numbers, and the *gap width* is $t-s$. Similarly, given reparameterizations f, g for curves π, σ, define their gap and gap width as

$$gap_{f,g}(\pi,\sigma) = \left[\min_{\alpha\in[0,1]} ||\pi(f(\alpha)) - \sigma(g(\alpha))||,\ \max_{\alpha\in[0,1]} ||\pi(f(\alpha)) - \sigma(g(\alpha))||\right],$$

$$gapwidth_{f,g}(\pi,\sigma) = \max_{\alpha\in[0,1]} ||\pi(f(\alpha)) - \sigma(g(\alpha))|| - \min_{\alpha\in[0,1]} ||\pi(f(\alpha)) - \sigma(g(\alpha))||\,.$$

The *Fréchet gap distance* between two curves π and σ is then defined as

$$d_{\mathcal{G}}(\pi,\sigma) = \min_{f,g} gapwidth_{f,g}(\pi,\sigma)$$

where f, g range over all possible reparameterizations of π and σ.

If there exist reparameterizations f and g for curves π and σ satisfying the inequalities,

$$\max_{\alpha\in[0,1]} ||\pi(f(\alpha)) - \sigma(g(\alpha))|| \leq t \qquad \min_{\alpha\in[0,1]} ||\pi(f(\alpha)) - \sigma(g(\alpha))|| \geq s,$$

we say $[s,t]$ is a *feasible gap* between curves π and σ. Throughout the paper $[s^*,t^*]$ denotes an arbitrary optimal gap, that is $t^* - s^* = d_{\mathcal{G}}(\pi,\sigma)$. (Note there may be more than one such optimal gap, and moreover a feasible gap does not necessarily contain an optimal gap.)

Note that in the later sections of the paper we refer to gaps or intervals $[s,t]$ instead as parametric points or pairs (s,t), in which case feasibility is defined analogously.

2.2 Free Space

To compute the standard Fréchet distance one normally looks at the so called *free space*. The t free space between curves π and σ, with n and m edges respectively, is defined as

$$F_t = \{(\alpha,\beta) \in [0,n] \times [0,m] \ \mid\ ||\pi(\alpha) - \sigma(\beta)|| \leq t\}.$$

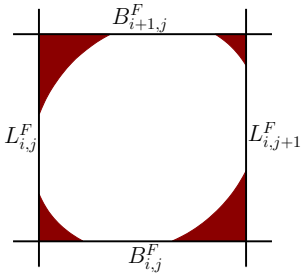


Figure 2 Free space cell.

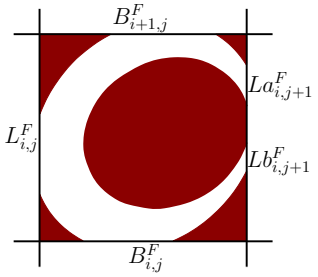


Figure 3 Relative free space cell.

Similarly define $F_t^< = \{(\alpha,\beta) \in [0,n] \times [0,m] \mid ||\pi(\alpha) - \sigma(\beta)|| < t\}$ to be F_t without its boundary. $C(i,j) = [i-1,i] \times [j-1,j]$ is referred to as the cell of the free space diagram determined by edges $\pi_{i-1}\pi_i$ and $\sigma_{j-1}\sigma_j$, and the free space within this cell is

$$F_t(i,j) = \{(\alpha,\beta) \in [i-1,i] \times [j-1,j] \mid ||\pi(\alpha) - \sigma(\beta)|| \leq t\}.$$

Alt and Godau [4] showed that the free space within a cell is always a convex set (specifically, the clipping of an affine transformation of a disk to the cell). Moreover, any x,y monotone path in the free space from $(0,0)$ to (n,m) corresponds to a pair of reparameterizations f, g of π, σ such that $width_{f,g}(\pi,\sigma) \leq t$. The converse also holds and hence $d_{\mathcal{F}}(\pi,\sigma) \leq t$ if and only if such a monotone path exists. These two statements together imply that in order to determine if $d_{\mathcal{F}}(\pi,\sigma) \leq t$, it suffices to restrict attention to the free space intervals on the boundaries of the cells. Specifically, let $L^F_{i,j}$ (resp. $B^F_{i,j}$) denote the left (resp. bottom) free space interval of $C(i,j)$, i.e. $L^F_{i,j} = F_t(i,j) \cap (\{i-1\} \times [j-1,j])$ (resp. $B^F_{i,j} = F_t(i,j) \cap ([i-1,i] \times \{j-1\})$). See Figure 2.

2.3 Relative Free Space

We extend the standard free space definitions of the previous section to the Fréchet gap distance problem. First we define the s,t relative free space between π and σ to be

$$F_{[s,t]} = \{(\alpha,\beta) \in [0,n] \times [0,m]|| \mid s \leq ||\pi(\alpha) - \sigma(\beta)|| \leq t\} = F_t \setminus F_s^<,$$

describing all pairs of points, one on π and one on σ, whose distance is contained in $[s,t]$. For a point (α,β) in a cell of $F_{[s,t]}$ or F_t, throughout we use the colloquial terms higher or lower (resp. right or left) to refer larger or smaller value of α (resp. β).

Again we seek an x,y monotone path in the relative free space from $(0,0)$ to (n,m), since such a path corresponds to a pair of reparameterizations f, g of π, σ such that $gapwidth_{f,g}(\pi,\sigma) \leq t-s$, and hence $d_{\mathcal{G}}(\pi,\sigma) \leq t-s$. Conversely, if no such path exists then $[s,t]$ is not a feasible gap for π and σ, implying that $[s^*,t^*] \not\subseteq [s,t]$, but note however that unlike the standard Fréchet distance, it may still hold that $t^* - s^* \leq t-s$.

The relative free space in the cell $C(i,j)$ determined by edges $\pi_{i-1}\pi_i$ and $\sigma_{j-1}\sigma_j$ is,

$$F_{[s,t]}(i,j) = \{(\alpha,\beta) \in [i-1,i] \times [j-1,j] \mid s \leq ||\pi(\alpha) - \sigma(\beta)|| \leq t\} = F_t(i,j) \setminus F_s^<(i,j).$$

Another technical challenge with the Fréchet gap problem arises from the fact that relative free space in a cell may not be convex (see Figure 3). However, there is some structure. Observe that $F_{[s,t]}(i,j) = F_t(i,j) \setminus F_s^<(i,j)$, and hence is the set difference of two convex sets, where one is contained in the other. In other words, it looks like a standard free space cell with a hole removed. In particular, we can again look at the free space intervals on the cell

boundaries. As $F_t(i,j)$ is convex, it still determines a single interval on each cell boundary, however, this interval may be broken into two subintervals by the removal of $F_s(i,j)$ (whose convexity implies it is at most two subintervals). Let $L^F_{i,j} = Lb^F_{i,j} \cup La^F_{i,j}$ denote the relative free space on the left boundary of $C(i,j)$, where $Lb^F_{i,j}$ denotes the bottom and $La^F_{i,j}$ the top interval (note if $F_s(i,j)$ does not intersect the boundary then $Lb^F_{i,j} = La^F_{i,j} = L^F_{i,j}$). Similarly, let $B^F_{i,j} = Bl^F_{i,j} \cup Br^F_{i,j}$ denote the relative free space on the bottom boundary of $C(i,j)$, where $Bl^F_{i,j}$ denotes the left and $Br^F_{i,j}$ the right interval.

3 The Fréchet Gap Decision Problem

The Fréchet gap decision problem is defined as follows.

▶ **Problem 1.** *Given polygonal curves π, σ, is a given interval $[s,t]$ a feasible gap for π, σ?*

As discussed in Section 2.3, $[s,t]$ is a feasible gap for π and σ if and only if there exists an x,y monotone path from $(0,0)$ to (n,m) in the $[s,t]$ relative free space $F_{[s,t]}$. This motivates the definition of the reachable relative free space,

$$RF_{[s,t]} = \{(\alpha,\beta) \in [0,n]\times[0,m] \mid \text{there exists an } x,y \text{ monotone path from} (0,0) \text{ to } (\alpha,\beta)\}.$$

Hence the answer to Problem 1 is 'yes' if and only if $(n,m) \in RF_{[s,t]}$. As was the case with the relative free space, the relevant information for the reachable relative free space is contained on the cell boundaries. We now describe how to propagate the reachable information from the left and bottom boundary to the right and top boundary of a cell, which ultimately will allow us to propagate the reachable information from $(0,0)$ to (n,m). (Note this is the typical approach to solving the standard Fréchet distance decision problem.)

Let $L^R_{i,j}$ and $B^R_{i,j}$ denote the reachable subsets of the left and bottom boundaries of $C(i,j)$. First we argue that like $L^F_{i,j}$, $L^R_{i,j}$ is composed of at most two disjoint intervals. Let $Lx^F_{i,j}$ be either $La^F_{i,j}$ or $Lb^F_{i,j}$. The reachable subset of $Lx^F_{i,j}$ is a single connected interval. To see this, observe that wherever the lowest reachable point in $Lx^F_{i,j}$ lies, all points above it in $Lx^F_{i,j}$ are reachable by a monotone path. As $L^R_{i,j}$ is a subset of $L^F_{i,j}$, this implies it is composed of at most two intervals denoted $La^R_{i,j}$ and $Lb^R_{i,j}$ (if $L^F_{i,j}$ is single interval then $L^R_{i,j} = La^R_{i,j} = Lb^R_{i,j}$). $Bl^R_{i,j}$ and $Br^R_{i,j}$ are defined similarly.

Propagating in a cell: Given $L^R_{i,j}$ and $B^R_{i,j}$, we now describe how to compute $L^R_{i,j+1}$ ($B^R_{i+1,j}$ is handled similarly). There are four cases, determined by whether we are propagating $L^R_{i,j}$ or $B^R_{i,j}$, and whether we are going above or below the hole $F_s(i,j)$. First, some notation.

▶ **Definition 2.** Label the leftmost and rightmost vertical lines tangent to the hole $F_s(i,j)$ as $v\ell^l_{i,j}$ and $v\ell^r_{i,j}$, and label the topmost and bottommost horizontal tangent lines as $h\ell^a_{i,j}$ and $h\ell^b_{i,j}$ (see Figure 4). Similarly define the leftmost point $\mathcal{H}^l_{i,j}$, the rightmost point $\mathcal{H}^r_{i,j}$, the topmost point $\mathcal{H}^a_{i,j}$, and the bottommost point $\mathcal{H}^b_{i,j}$, of $F_s(i,j)$ (Note any one of these points may be undefined if $F_s(i,j)$ intersects the boundary in more than a single point, as is the case for $\mathcal{H}^r_{i,j}$ in Figure 4.) Finally, let $\mathcal{I}^a_{i,j}$ be the highest and $\mathcal{I}^b_{i,j}$ the lowest point of $L^F_{i,j+1}$. When i,j is fixed, the subscript is often dropped.

Propagation of the reachable relative free space is done similarly to that for the standard free space [4]. Namely points from $L^R_{i,j}$ and $B^R_{i,j}$ are projected onto $L^F_{i,j+1}$ by paths which locally stay as low as possible. The only difference is that now the $L^R_{i,j}$ and $B^R_{i,j}$ cases are each broken into two subcases based on whether the path must go above or below the hole $F_s(i,j)$ (see Figure 5 and Figure 6). Note this stays a constant time operation per cell, since

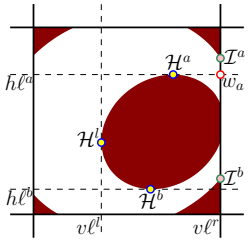


Figure 4 Free space cell.

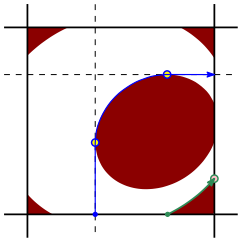


Figure 5 $B^R_{i,j}$ to $L^R_{i,j+1}$.

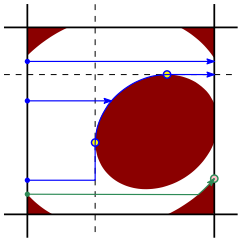


Figure 6 $L^R_{i,j}$ to $L^R_{i,j+1}$.

as proved above $L^R_{i,j}$ and $B^R_{i,j}$ are each always composed of at most two disjoint subintervals. Due to space limitations, the straightforward but tedious details of propagation are left to the full version [18] (the above definition was kept as it is needed later).

▶ **Theorem 3** (For proof see [18]). *Given polygonal curves π of length n, σ of length m, and an interval $[s,t]$, the Fréchet gap decision problem, Problem 1, can be solved in $O(nm)$ time.*

4 Finding the Relative Free Space Critical Events

In this section we describe the relative free space critical events, that is a polynomially sized subset of possible intervals, which must contain an optimal interval $[s^*,t^*]$. The relative free space events are significantly more complicated than the free space events for the standard Fréchet distance. The following definitions will be used throughout this section.

▶ **Definition 4.** Two free space cells $C(i,j)$ and $C(k,l)$ are *adjacent* if they share a horizontal or vertical boundary, i.e. $k=i$ and $|l-j|=1$, or $l=j$ and $|k-i|=1$. Call any monotone path from $(0,0)$ to (n,m) in the relative free space a *valid* path. Given any valid path p, the *cell sequence* of p, denoted $\mathsf{cp}(p)=(C_1,\ldots,C_{n+m-1})$, is the ordered sequence of cells p intersects (so $C_1=C(1,1)$, $C_{n+m-1}=C(n,m)$). For horizontally adjacent cells $C(i,j)$ and $C(i,j+1)$ in the cell sequence, p either passes above or below $F_s(i,j)$, specifically if p intersects the vertical segment connecting $\mathcal{H}^a$ to the top boundary of $C(i,j)$ then p *passes above* $F_s(i,j)$, and otherwise p *passes below*. (Similarly define passing left or right for vertically adjacent cells.) This defines the *passing sequence* of p, denoted $\mathsf{pass}(p)=(h_1,\ldots,h_{n+m-1})$, where $h_i\in\{\text{above, below, left, right}\}$.

For the standard Fréchet distance, Alt and Godau [4] specified the following set of distance values, called the critical events, which must contain the optimal Fréchet distance.

- *Initialization* event: The minimum value ε such that $(0,0)\in F_\varepsilon$ and $(n,m)\in F_\varepsilon$.
- *Connectivity* events: For any cell C_i, the minimum ε such that L^F_i or B^F_i is non-empty, corresponding to the distance between a vertex of one curve and an edge of the other.
- *Monotonicity* events: Let I_j and I_k be two non-empty vertical free space boundary intervals in the same row with I_j left of I_k (or horizontal intervals in the same column). The minimum ε such that $\mathcal{I}^b_j\leq\mathcal{I}^a_k$, that is there is a monotone path between I_j and I_k.

Since any valid path can be decomposed into a set of row and column subpaths, proving that $d_{\mathcal{F}}(\pi,\sigma)$ is one the above defined critical events is a straightforward task.

For the Fréchet gap distance, the critical events will be a super-set of the standard Fréchet events. As an optimal gap is defined by an interval $[s,t]$, the events below can either be a value of s or a value of t. A *critical interval* is then any valid $s\leq t$ pair from the first three critical event types defined below. Additionally, there is now a fourth type called a floating

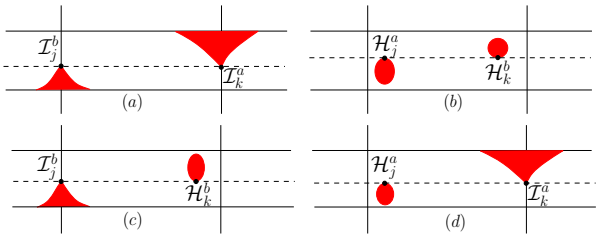


Figure 7 Opening of a horizontal passage.

monotonicity event. These events directly specify the s, t pair (i.e. these "events" are also "critical intervals"), and there are potentially an infinite number of such events.

1. *Initialization* events: The values $s = \min\{||\pi_0 - \sigma_0||, ||\pi_n - \sigma_m||\}$ and $t = \max\{||\pi_0 - \sigma_0||, ||\pi_n - \sigma_m||\}$. That is, the supremum of values for s such that $(0,0) \notin F_s$ and $(n,m) \notin F_s$, and the minimum value of t such that $(0,0) \in F_t$ and $(n,m) \in F_t$.
2. *Connectivity* events: For any row i and column j, the values $dist(\pi_{i-1}, \sigma_{j-1}\sigma_j)$, $dist(\pi_i, \sigma_{j-1}\sigma_j)$, $dist(\pi_{i-1}\pi_i, \sigma_{j-1})$, $dist(\pi_{i-1}\pi_i, \sigma_j)$, for either s or t. In other words for cell $C_{i,j}$, the maximum value s such that $\mathcal{H}^a$, $\mathcal{H}^b$, $\mathcal{H}^l$, or $\mathcal{H}^r$ are defined, or minimum value t such that $\mathcal{I}^a, \mathcal{I}^b$ (or similarly any of the other three cell boundary intervals) are defined. Note $\mathcal{I}^a, \mathcal{I}^b$ are first defined at the same location/value where $\mathcal{H}^r$ is last defined, yet we still regard these as separate events, one for s and the other for t. (For s this is when the free space intervals may break into two, and for t it is when the interval is first non-empty.)
3. *Standard Monotonicity* events: For any cells C_j, C_k in the same row with C_j left of C_k:
   - **(a)** The value t such that $height(\mathcal{I}_j^b) = height(\mathcal{I}_k^a)$.
   - **(b)** The value s such that $height(\mathcal{H}_j^a) = height(\mathcal{H}_k^b)$.
3. *Floating Monotonicity* events: For any cells C_j, C_k in the same row with C_j left of C_k:
   - **(a)** Any pair s, t such that $height(\mathcal{I}_j^b) = height(\mathcal{H}_k^b)$.
   - **(b)** Any pair s, t such that $height(\mathcal{H}_j^a) = height(\mathcal{I}_k^a)$.

Here $height()$ denotes the vertical coordinate of a point in the relative free space. Analogous definitions apply to the case when cells are in the same column. Note that depending on the geometry such events may not be defined.

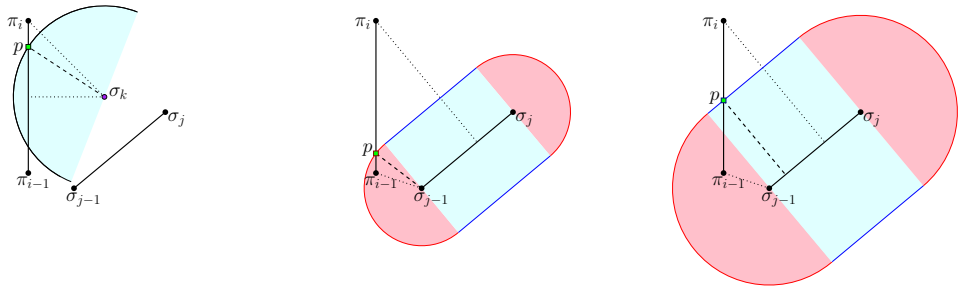
Let S_s and S_t denote the set of values for s and t, respectively, determined by the initialization, connectivity and standard monotonicity critical events, and let $S_s \times S_t$ denote the corresponding set of valid critical intervals determined by these values. Let S_F be the set of s, t intervals determined by floating monotonicity events. The set of all critical intervals is then $S_I = S_F \cup (S_s \times S_t)$. The proof of the following is similar to the standard Fréchet case, except now valid paths are characterized by passing sequences in addition to cell sequences.

▶ **Lemma 5** (For proof see [18]). *S_I contains any optimal Fréchet gap interval $[s^*, t^*]$.*

4.1 Bounding the number of critical intervals

We now bound the number of critical intervals, i.e. $|S_I|$. An interval $[s,t] \in S_I$, is either in $S_s \times S_t$ or S_F. Now S_s (resp. S_t) has size[1] $O(n^3)$ as it contains one initialization event,

[1] For simplicity, from this point onwards we assume without loss of generality that $m \leq n$ and only write sizes and running times with respect to n.



(a) Distance from σ_k **(b)** Distance from σ_{j-1} **(c)** Distance from $\sigma_{j-1}\sigma_j$

Figure 8 How point p determines s and t. In general segments may not lie in a single plane.

$O(n^2)$ connectivity events, and $O(n^3)$ monotonicity events (just like the standard Fréchet case). As we consider all valid pairs from S_s and S_t, this gives an $O(n^6)$ bound on $|S_s \times S_t|$.

Bounding the size of S_F is significantly more complicated. In particular, the floating monotonicity events may give rise to an entire continuum of critical intervals. For example, consider the second type of floating monotonicity event (2), shown in Figure 7. The value of $height(\mathcal{H}_j^a)$ is governed only by a function of s and the value of $height(\mathcal{I}_k^a)$ only by a function of t. These functions might be such that if we increase or decrease s, but keep $t - s$ constant (i.e. the gap value we are optimizing), $height(\mathcal{H}_j^a) = height(\mathcal{I}_k^a)$ remains an invariant. (Hence the term "floating" events.)

In this section we describe the functions which govern how s and t can vary such that $height(\mathcal{H}_j^a) = height(\mathcal{I}_k^a)$ remains an invariant. Ultimately our understanding of these function will yield a polynomially sized set of canonical critical intervals (determined by vertices of the arrangement of these functions), which must contain an optimum gap interval.

4.1.1 Function Description of Floating Monotonicity Events

Consider the floating monotonicity event type (2) (similar statements will hold for type (1)). Such an event is specified by a triple of indices, i, j, k, where i specifies an edge $\pi_{i-1}\pi_i$ (i.e. a row of the relative free space), j specifies an edge $\sigma_{j-1}\sigma_j$ (i.e. a column), and $k \geq j$ specifies a vertex σ_k (i.e. the right boundary of a column). The event occurs when $height(\mathcal{H}_j^a) = height(\mathcal{I}_k^a) = h$.

Geometrically, a fixed height h corresponds to a point p on $\pi_{i-1}\pi_i$. The point $\mathcal{H}_j^a$ is determined by s, and $\mathcal{I}_k^a$ by t. First lets understand $\mathcal{I}_k^a$. In order to have $h = height(\mathcal{I}_k^a)$, t must be such that $t = ||\sigma_k - p||$, and moreover p must be the higher (i.e. closer to π_i) of the possibly two points on $\pi_{i-1}\pi_i$ satisfying this condition (the other point determining $\mathcal{I}_k^b$). Consider the plane determine by π_{i-1}, π_i, and σ_k, and let $\pi_{i-1} = (0, 0)$, $p = (0, h)$, and $\sigma_k = (\chi, \gamma)$ (see Figure 8a). Then as a function of h, t is described by the equation $t = \sqrt{\chi^2 + (\gamma - h)^2}$. Note that $\mathcal{I}_k^a$ is only defined when $t \in [t_1, t_2]$, where $t_1 = dist(\sigma_k, \pi_{i-1}\pi_i)$ and $t_2 = ||\sigma_k - \pi_i||$, and hence this equation is only relevant in this interval.

$height(\mathcal{H}_j^a)$ on the other hand is determined by s, however the relationship is a bit more complicated. Observe that $\mathcal{H}_j^a$ is the only point on the horizontal line $h\ell_j^a$ that is in the set $F_s(i, j)$, meaning the point on $\sigma_{j-1}\sigma_j$ that $\mathcal{H}_j^a$ corresponds to must be the closest point on $\sigma_{j-1}\sigma_j$ to p (see Figure 8b and Figure 8c). If this closest point is either σ_{j-1} or σ_j, then the form of the equation for s in terms of h is the same as it was t, namely $s = \sqrt{\alpha^2 + (\beta - h)^2}$ (where α, β are now the coordinates of either σ_{j-1} or σ_j). Otherwise this closest point is in the interior of $\sigma_{j-1}\sigma_j$ in which case the equation is of the form $s = c \cdot h + d$, for some

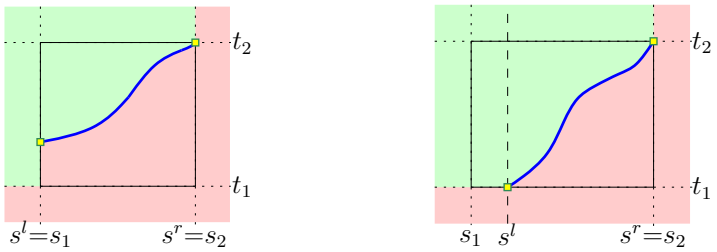


Figure 9 Two cases for curve piece $f_{i,j,k}$, and shaded satisfying points in s,t parametric space.

constants c and d (since as one walks along a line, the distance to another fixed line is given by a linear equation). Similar to $\mathcal{I}^a_k$, $\mathcal{H}^a_j$ is only defined when $s \in [s_1, s_2]$, where $s_1 = dist(\sigma_{j-1}\sigma_j, \pi_{i-1}\pi_i)$ and $s_2 = dist(\sigma_{j-1}\sigma_j, \pi_i)$, and hence this equation is only relevant in this interval.

Now that we have a description of $height(\mathcal{H}^a_j)$ in terms of s and $height(\mathcal{I}^a_k)$ in terms of t, we can describe the function for t in terms of s, denoted $f_{i,j,k}(s)$, which describes when $height(\mathcal{H}^a_j) = height(\mathcal{I}^a_k) = h$. There are two cases based on the form of the function describing s.

Interior of $\sigma_{j-1}\sigma_j$ case:

$$s = c \cdot h + d, \quad t = \sqrt{\chi^2 + (\gamma - h)^2} \quad \Rightarrow \quad f_{i,j,k}(s) = \sqrt{\left(\frac{s-d}{c} - \gamma\right)^2 + \chi^2}\,.$$

Endpoint of $\sigma_{j-1}\sigma_j$ case:

$$s = \sqrt{\alpha^2 + (\beta - h)^2}, \quad t = \sqrt{\chi^2 + (\gamma - h)^2} \quad \Rightarrow \quad f_{i,j,k}(s) = \sqrt{(\sqrt{s^2 - \alpha^2} + (\beta - \gamma))^2 + \chi^2}\,.$$

To summarize, $f_{i,j,k}(s)$ is composed of at most three hyperbola[2] pieces, and is only (possibly) defined within the region $s \in [s_1, s_2]$ and $t \in [t_1, t_2]$, see Figure 9. Also, the geometry of the problem implies that when $f_{i,j,k}(s)$ is defined it is a monotone increasing function. Hence the intersection of $f_{i,j,k}$ with the bounding box $[s_1, s_2] \times [t_1, t_2]$ is connected, and so rather than using this box to define $f_{i,j,k}$, we instead say $f_{i,j,k}$ is either completely undefined or is defined only in the interval $[s^l, s^r]$ where s^l and s^r are the s coordinate where $f_{i,j,k}$ respectively enters and leaves the bounding box. Note that one can argue if $f_{i,j,k}$ is defined then $s^r = s_2$, however, it may be that $s^l > s_1$ (if the closest point to $\sigma_{j-1}\sigma_j$ is lower on $\pi_{i-1}\pi_i$ than the closest point to σ_k).

The exact form of the equation $f_{i,j,k}(s)$ is not needed in our analysis, however, the above discussion implies the following simple observation which will be used later.

▶ **Observation 6.** *In the s,t parametric space $f_{i,j,k}$ is either undefined or defines a constant complexity monotonically increasing curve piece, with endpoints at values $s^l_{i,j,k} \leq s^r_{i,j,k}$. In particular, $f_{i,j,k}$ has only a constant number of local minima and maxima (i.e. points of tangency) with respect to translations of the line $t = s$.*

Note that for (1), i.e. when $height(\mathcal{I}^b_j) = height(\mathcal{H}^b_k)$, $f_{i,j,k}$ can be defined similarly, and the above observation again holds. One must also define functions for the analogous events in the free space columns. Such functions are again determined by triples i, j, k, however now i, j refer to rows and k to the column. Below we will denote these functions by $g_{i,j,k}$.

[2] Technically, the endpoint case is not a hyperbola, though it is similar.

4.1.2 Events minimizing the gap

As discussed above each $f_{i,j,k}$, if defined, gives an entire continuum of critical intervals. However, ultimately we are only interested in feasible intervals which minimize the gap, and this will allow us to reduce this continuum to a polynomial number of canonical intervals. This polynomially sized set is determined not only by the $f_{i,j,k}$, but also by the other types of critical events. Note that initialization (1), connectivity (2), and standard monotonicity events (3) only define constraints on either just s or t, whereas the $f_{i,j,k}$ and $g_{i,j,k}$ define a continuum of $[s,t]$ intervals. Hence to put them on equal footing we think of all of them as defining constraints in the two dimensional s,t parametric space.

First observe that in the parametric space, for any point (s,t) of interest, $0 \leq s \leq t$, and so we only consider points in the first quadrant that are above the line $t = s$. Initialization, connectivity, and standard monotonicity events are simply defined by horizontal or vertical lines. Specifically, for each such event the points satisfying the corresponding constraint are those above (resp. left of) the corresponding horizontal (resp. vertical) line:

1. Initialization events: $s \leq \alpha_0, \quad t \geq \beta_0$
   Where $\alpha_0 = \min\{||\pi_0 - \sigma_0||, ||\pi_n - \sigma_m||\}$ and $\beta_0 = \max\{||\pi_0 - \sigma_0||, ||\pi_n - \sigma_m||\}$.
2. Connectivity events: $s \leq \alpha^l_{i,j}$ or $s \leq \alpha^b_{i,j}, \quad t \geq \beta^l_{i,j}$ or $t \geq \beta^b_{i,j}$
   Where the $\alpha_{i,j}$ and $\beta_{i,j}$ are *vertex-edge* distances, that is $\alpha^l_{i,j} = \beta^l_{i,j} = dist(\pi_{i-1}\pi_i, \sigma_{j-1})$ or $\alpha^b_{i,j} = \beta^b_{i,j} = dist(\pi_{i-1}, \sigma_{j-1}\sigma_j)$. Note defining both $\alpha_{i,j}$ and $\beta_{i,j}$ is not necessary but useful to distinguish constraints on s from those on t.
3. Standard Monotonicity events: $s \leq \alpha_{i,(j,k)}$ or $s \leq \alpha_{(i,j),k}, \quad t \geq \beta_{i,(j,k)}$ or $t \geq \beta_{(i,j),k}$
   Which happens when the free space is such that $\alpha_{i,(j,k)} = height(\mathcal{H}^a_{i,j}) = height(\mathcal{H}^b_{i,k})$ or $\alpha_{(i,j),k} = height(\mathcal{H}^a_{i,k}) = height(\mathcal{H}^b_{j,k})$, and when $\beta_{i,(j,k)} = height(\mathcal{I}^b_{i,j}) = height(\mathcal{I}^a_{i,k})$ or $\beta_{(i,j),k} = height(\mathcal{I}^b_{i,k}) = height(\mathcal{I}^a_{j,k})$.
4. Floating Monotonicity events: $t \geq f_{i,j,k}(s)$ for $s \in [s^{lf}_{i,j,k}, s^{rf}_{i,j,k}]$, or $t \geq g_{i,j,k}(s)$ for $s \in [s^{lg}_{i,j,k}, s^{rg}_{i,j,k}]$. Note depending on the geometry such constraints may not be defined.

Note that the first three event types each partition the entire parametric space into two connected sets, those which either satisfy or do not satisfy the constraint. The $f_{i,j,k}$ (and $g_{i,j,k}$) can also be thought of in this way, see the shaded regions in Figure 9. Specifically, (s,t) satisfies the constraint if $t \geq t_1$, $s \leq s_2$, and if $s \in [s^l, s^r]$ then (s,t) must lie above the curve $f_{i,j,k}$. Otherwise (s,t) does not satisfy the constraint.

Any valid path in the relative free space must have a well defined cell sequence $(C_1, \ldots, C_{n+m-1})$ and passing sequence $\mathsf{pass}(p) = (h_1, \ldots, h_{n+m-1})$ (see Definition 4). Moreover, such a pair of sequences precisely determine a subset of the constraints defined above, such that there is a valid path with this cell and passing sequence if and only if all constraints in the subset are satisfied (this is implied by Lemma 5). In other words, for a given cell and passing sequence we want to solve a well defined optimization problem, where constraints on s and t are of the form described above, and the objective is to minimize $t - s$.

Clearly the optimal value of this optimization problem must lie on the boundary of at least one constraint. In particular, the optimum lies either at the intersection point of the boundaries of two constraints, or at a local minimum of one of the boundary constraints, with respect to the objective of minimizing $t - s$. By Observation 6, each $f_{i,j,k}$ or $g_{i,j,k}$ has at most a constant number of local minima, and as the boundaries of all other constraints are straight lines, this is also true for every boundary function. Thus we have now determined the set of canonical critical intervals discussed earlier in this section.

▶ **Lemma 7.** *The above defined constraints, determined by all types of critical events, determine an $O(n^6)$ sized set of canonical critical intervals, i.e. (s,t) pairs, that must contain an optimal gap $[s^*, t^*]$.*

Proof. Any optimal gap determines a cell and passing sequence of some valid path in the corresponding relative free space. Above it was discussed how such sequences determine a subset of constraints, where the optimum gap width is determined either at an intersection of the boundaries of two constraints or at a local minimum of an $f_{i,j,k}$ or $g_{i,j,k}$. Now a priori we do not know the cell and passing sequence of a path determining an optimal gap, hence we will consider them all. So consider the arrangement of all planar curves defined by the boundaries of any of the possible constraints defined above. There are a constant number of initialization constraints, $O(n^2)$ possible connectivity constraints, and $O(n^3)$ possible standard or floating monotonicity constraints. Due to the particularly nice form of these curves, each pair intersect at most a constant number of times, and hence there are $O(n^6)$ intersections overall. Moreover, as discussed above, each curve has only a constant number of local minima with respect the the objective of minimizing $t - s$. Hence this arrangement determines as set of $O(n^6)$ points, at least one of which realizes the minimum gap width. ◀

▶ **Observation 8.** *Whether or not a given (s,t)-pair is feasible for the Fréchet gap problem, is solely determined by which constraints the point satisfies or does not satisfy. So consider the arrangement of curves determined by the boundaries of all the constraint types discussed above. Then within the interior of a given cell of the arrangement all (s,t)-pairs are thus either all feasible or all infeasible.*

5 Exact Computation of the Fréchet Gap Distance

The $O(n^6)$ critical intervals given by Lemma 7 together with the $O(n^2)$ decider of Theorem 3, naively give only an $O(n^8)$ algorithm for computing the Fréchet gap distance, as there is no immediate linear ordering to search over the events. However, here we give a much faster $O(n^5 \log n)$ time algorithm to compute the Fréchet gap distance exactly.

The standard Fréchet distance is computed in $O(n^2 \log n)$ time by searching over the $O(n^3)$ critical events with an $O(n^2)$ time decision procedure. This searching originally was done with parametric search [4], though for our purposes the simpler sampling based approach of [20] is more relevant.

Searching is a far more challenging task in the Fréchet gap setting. Specifically, in the standard Fréchet case there is a linear ordering of the critical events, and in this ordering all events are infeasible up until the true Fréchet distance, and then feasible afterwards. However, in our two dimensional parametric space there is no such natural linear ordering. Moreover, recall that even if an interval $[s,t]$ is feasible, it does *not* imply $[s,t]$ contains an optimal gap as a subinterval. Thus the following lemma, while easy to prove, is crucial.

▶ **Lemma 9** (For proof see [18]). *In the parametric space, the set of feasible (s,t) pairs is a connected set.*

The algorithm for exactly computing the Fréchet gap distance uses the following subroutines:

- **deciderPoint**(s,t): Decides whether or not the pair (s,t) is feasible, in $O(n^2)$ time.
- **deciderLine**(c): Given a positive number c, returns "below" if there is any feasible (s,t)-pair with $t - s \leq c$, and returns "above" otherwise. The running time is $O(n^5)$.
- **sample**(r): Samples r (s,t)-pairs, independently and uniformly at random, from the set of $O(n^6)$ canonical critical pairs of Lemma 7. The running time is $O(r)$.
- **sweep**(c_1, c_2): Returns the set of all canonical critical (s,t)-pairs of Lemma 7 such that $c_1 \leq t - s \leq c_2$, in $O((n^3 + k) \log n)$ time, where k is the number of such critical pairs.

First observe the subroutine **deciderPoint**(s,t) is given by Theorem 3. **deciderLine**(c) is computed as follows. First compute the intersection points of the line, $t-s=c$, with the $O(n^3)$ boundaries of all the constraints discussed in Section 4.1.2. Since these constraints are horizontal/vertical lines or $f_{i,j,k}/g_{i,j,k}$, by Observation 6, there are $O(n^3)$ intersection points. Thus calling **deciderPoint** on each of these intersection points, takes $O(n^5)$ time as **deciderPoint** takes $O(n^2)$ time. By Observation 8, if all these point queries return infeasible, then all points on the line $t-s=c$ are infeasible. In this case, since by Lemma 9 the feasible region is connected, any optimal gap pair must lie above the the line $t-s=c$. On the other hand, again by by Lemma 9, if one of the point queries returned true then any optimal gap pair must lie below (or on) the line $t-s=c$.

The subroutine **sample**(r) is also straightforward. Specifically, every canonical critical pair is either a local minima or an intersection of the boundaries of two constraints from Section 4.1.2. Thus in order to sample a canonical critical pair, we sample either one or two constraints[3], where whether we sample one or two is done in proportion to the number of pairs versus single constraints. Each constraint is determined by either a pair or triple of indices (and a few bits, such as whether the side of bottom of a cell, etc.), and hence each can be sampled in $O(1)$ time (again done proportionally to the number of triples versus pairs of indices). Thus r canonical pairs can be sampled in $O(r)$ time.

Thus what remains is to describe the subroutine **sweep**, for which we have the following.

▶ **Lemma 10.** *Given two real values $0 \leq c_1 \leq c_2$, one can compute the set of all canonical critical (s,t)-pairs of Lemma 7 such that $c_1 \leq t-s \leq c_2$, in $O((n^3+k)\log n)$ time, where k is the number of such critical pairs. This algorithm is denoted* **sweep**(c_1,c_2).

Proof. It is well known that one can compute the set of all k intersection points of a set of m x-monotone constant-complexity curves in $O((m+k)\log m)$ time using a horizontal sweep line in the standard sweep line algorithm of Bentley and Ottmann [8]. In our case the curves are given by the $O(n^3)$ constraints of Section 4.1.2, clipped to only be defined in the region bounded by the lines $t-s=c_1$ and $t-s=c_2$. The constraints with straight line boundaries are s-monotone, and by Observation 6 so are the $f_{i,j,k}$ and $g_{i,j,k}$. Thus the claim follows by applying the standard sweep line algorithm to our case. ◀

1 $\mathcal{R} =$ **sample**(αn^4) `// α a sufficiently large constant`
2 Sort $\widehat{\mathcal{R}} = \{c = t-s \mid (s,t) \in \mathcal{R}\}$ in increasing order
3 Binary search over $\widehat{\mathcal{R}}$ using **deciderLine**(c) for the interval $[c_1,c_2]$
  s.t. **deciderLine**(c_1) = above and **deciderLine**(c_2) = below
  `// Set initial values` $c_1 = 0$, $c_2 = \infty$
4 $\mathcal{S} =$ **sweep**(c_1,c_2)
5 Call **deciderPoint**(s,t) on each $(s,t) \in \mathcal{S}$, and
  return the feasible pair with smallest $t-s$ value.

Algorithm 1: Computing the Fréchet gap distance

The algorithm for computing the Fréchet gap distance is shown in Algorithm 1. We need the following lemma to bound the number of critical pairs that we end up searching over.

[3] Note the number of local minima per constraint and the number of times two constraints intersect is a constant, but the constant may be more than one. Thus technically the described sampling is not truly uniform. One can make it uniform, though this distinction is irrelevant for our asymptotic analysis.

▶ **Lemma 11** (For proof see [18]). *Let $[c_1, c_2]$ be the interval described in Algorithm 1. Then with exponentially high probability, this interval contains $O(n^3)$ canonical critical pairs.*

Note instead one could argue that with polynomially high probability the number of canonical critical pairs in $[c_1, c_2]$ is only $O(n^2 \log n)$. Ultimately though this would not change the running time, as the real bottleneck is searching with the $O(n^5)$ time **deciderLine**.

▶ **Theorem 12.** *Given polygonal curves π and σ, each of length at most n, Algorithm 1 computes the Fréchet gap distance in $O(n^5 \log n)$ time.*

Proof. The correctness of Algorithm 1 has essentially already been argued. Specifically, the random sample $\mathcal{R}$ partitions the real line into intervals based on the values in $\widehat{\mathcal{R}}$. One of these intervals contains the optimal gap width, implying the interval $[c_1, c_2]$ found by searching using **deciderLine**(c) is well defined. Moreover, $\mathcal{S}$ contains a canonical critical pair with optimal gap width as **sweep**(c_1, c_2) returns all canonical critical pairs in the region bounded by the lines $t - s = c_1$ and $t - s = c_2$, and by Lemma 7 the set of canonical critical pairs contains a pair with optimal gap width. As **deciderPoint** is called on all pairs in $\mathcal{S}$, the algorithm will find this optimal gap pair.

For the running time, calling **sample**(αn^4) takes $O(n^4)$ time. Sorting $\widehat{\mathcal{R}}$ takes $O(n^4 \log n)$ time, and searching over $\widehat{\mathcal{R}}$ takes $O(n^5 \log n)$ time as **deciderLine** takes $O(n^5)$ time. By Lemma 10, **sweep**(c_1, c_2) takes $O((n^3 + |\mathcal{S}|) \log n)$ time. Calling **deciderPoint** on each pair in $\mathcal{S}$ takes $O(|\mathcal{S}|n^2)$ time, as **deciderPoint** takes $O(n^2)$ time. By Lemma 11, with high probability $|\mathcal{S}| = O(n^3)$, so sweeping and all **deciderPoint** calls combined take $O(n^5)$ time. Thus the overall time is $O(n^5 \log n)$, i.e. dominated by the time to search with **deciderLine**. ◀

6 Approximation

In this section, we propose an efficient algorithm to approximate the Fréchet gap distance, based on the following simple fact. Let d_o be the average of the starting and ending vertex pair distances of π and σ, that is $d_o = \frac{d_b + d_e}{2}$ where $d_b = ||\pi_0 - \sigma_0||$ and $d_e = ||\pi_n - \sigma_m||$.

▶ **Observation 13.** *If a parametric point (s, t) is feasible then $s \leq d_o \leq t$.*

This implies we only need to consider parametric points such that $s \leq d_o \leq t$, which we call *centered* points. Define the *radius* of any such point (s, t) to be $\mathsf{r}_{s,t} = \max\{t - d_o, d_o - s\}$, and define the *projection* to be $proj(s, t) = (d_o - \mathsf{r}_{s,t}, d_o + \mathsf{r}_{s,t})$.

Observe that in order to get a 2-approximation it suffices to restrict our attention to projected points (as $[s, t] \subseteq [d_o - \mathsf{r}_{s,t}, d_o + \mathsf{r}_{s,t}]$ for any centered point (s, t)), and the advantage is that projected points are more nicely behaved. Specifically, projected points define a linear ordering by the parameter r with the nice property that if $(d_o - \mathsf{r}, d_o + \mathsf{r})$ is feasible then for any $\mathsf{r}' \geq \mathsf{r}$ it holds that $(d_o - \mathsf{r}', d_o + \mathsf{r}')$ is also feasible. Moreover, below we show that the $O(n^6)$ critical intervals of Lemma 7, can be reduced to $O(n^3)$ in this setting, intuitively since now there is only a single parameter r, rather than independent s and t parameters.

The details of how the above high level idea is employed are interesting, but are omitted due to space constraints. Here we give a brief outline. First it is shown that considering the minimum radius projected point in each region defined by one of the $O(n^3)$ constraints of Section 4.1.2, gives a set of $O(n^3)$ projected points containing a 2-approximation to the Fréchet gap distance. Then it is observed that sorting these points by radii induces a linear ordering of feasibility, thus already implying a near cubic time algorithm to find a 2-approximation. Next we how to construct an $O(n^2/\varepsilon)$ time $(1 + \varepsilon)$-approximate decider

(again making use of Observation 13), and then show how to use it to efficiently turn any constant factor approximation into a $(1+\varepsilon)$-approximation. Finally, the most challenging part is removing a near linear factor in the running time, and involves sampling and careful sweeping over the functions for s and t discussed in Section 4.1.1, modified for the projected point setting, thus advancing the basic sampling and sweeping approach of [20].

▶ **Theorem 14** (For proof see [18]). *Given polygonal curves π and σ, each of length at most n, one can $(1+\varepsilon)$-approximate the Fréchet gap distance in $O(n^2(\log n + \frac{1}{\varepsilon}\log\frac{1}{\varepsilon}))$ time.*

References

1. P. Agarwal, R. Avraham, H. Kaplan, and M. Sharir. Computing the discrete Fréchet distance in subquadratic time. *SIAM Journal on Computing*, 43(2):429–449, 2014.
2. H. Alt and M. Buchin. Can we compute the similarity between surfaces? *Discrete & Computational Geometry*, 43(1):78–99, 2010.
3. H. Alt, A. Efrat, G. Rote, and C. Wenk. Matching planar maps. In *Proc. of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 589–598, 2003.
4. H. Alt and M. Godau. Computing the Fréchet distance between two polygonal curves. *Int. J. Comput. Geometry Appl.*, 5:75–91, 1995.
5. H. Alt, C. Knauer, and C. Wenk. Matching polygonal curves with respect to the Fréchet distance. In *Annual Symp. on Theo. Aspects of Comp. Sci. (STACS)*, pages 63–74, 2001.
6. R. Avraham, O. Filtser, H. Kaplan, M. Katz, and M. Sharir. The discrete and semicontinuous Fréchet distance with shortcuts via approximate distance counting and selection. *ACM Trans. Algorithms*, 11(4):29, 2015.
7. R. Avraham, H. Kaplan, and M. Sharir. A faster algorithm for the discrete Fréchet distance under translation. *CoRR*, abs/1501.03724, 2015.
8. J. Bentley and T. Ottmann. Algorithms for reporting and counting geometric intersections. *IEEE Trans. Computers*, 28(9):643–647, 1979.
9. S. Brakatsoulas, D. Pfoser, R. Salas, and C. Wenk. On map-matching vehicle tracking data. In *Proc. 31st VLDB Conference*, pages 853–864, 2005.
10. K. Bringmann. Why walking the dog takes time: Fréchet distance has no strongly subquadratic algorithms unless seth fails. In *Symp. on Found. of Comp. Sci. (FOCS)*, pages 661–670. IEEE, 2014.
11. K. Buchin, M. Buchin, J. Gudmundsson, M. Löffler, and J. Luo. Detecting commuting patterns by clustering subtrajectories. *Int. J. Comput. Geom. Appl.*, 21(3):253–282, 2011.
12. K. Buchin, M. Buchin, W. Meulemans, and W. Mulzer. Four soviets walk the dog – with an application to alt's conjecture. In *Proc. of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1399–1413, 2014.
13. K. Buchin, M. Buchin, and C. Wenk. Computing the Fréchet distance between simple polygons in polynomial time. In *22nd Annual Symp. Comput. Geom.*, pages 80–87, 2006.
14. M. Buchin, A. Driemel, and B. Speckmann. Computing the Fréchet distance with shortcuts is np-hard. In *30th Annual Symp. Comput. Geom. (SoCG)*, page 367, 2014.
15. A. Driemel and S. Har-Peled. Jaywalking your dog: computing the Fréchet distance with shortcuts. *SIAM Journal on Computing*, 42(5):1830–1866, 2013.
16. A. Driemel, S. Har-Peled, and C. Wenk. Approximating the Fréchet distance for realistic curves in near linear time. *Discrete & Computational Geometry*, 48(1):94–127, 2012.
17. T. Eiter and H. Mannila. Computing discrete Fréchet distance, 1994.
18. C. Fan and B. Raichel. Computing the Fréchet gap distance. `http://www.utdallas.edu/~bar150630/gap.pdf`.
19. O. Filtser and M. Katz. The discrete Fréchet distance gap. *arXiv:1506.04861*, 2015.

20 S. Har-Peled and B. Raichel. The Fréchet distance revisited and extended. *ACM Transactions on Algorithms (TALG)*, 10(1):3, 2014.

21 M. Kim, S. Kim, and M. Shin. Optimization of subsequence matching under time warping in time-series databases. In *Proc. ACM Symp. on Applied Computing*, pages 581–586, 2005.

22 G. Rote. Computing the Fréchet distance between piecewise smooth curves. *Computational Geometry*, 37(3):162–174, 2007.

23 J. Serrà, E. Gómez, P. Herrera, and X. Serra. Chroma binary similarity and local alignment applied to cover song identifica. *Audio, Speech & Lang. Proc.*, 16(6):1138–1151, 2008.

24 C. Wenk, R. Salas, and D. Pfoser. Addressing the need for map-matching speed: Localizing global curve-matching algorithms. In *Sci. Statis. Database Manag.*, pages 879–888, 2006.

Erdős-Hajnal Conjecture for Graphs with Bounded VC-Dimension

Jacob Fox[*1], **János Pach**[†2], **and Andrew Suk**[‡3]

1 **Stanford University, Stanford, CA, USA**
jacobfox@stanford.edu
2 **EPFL and Rényi Institute, Lausanne, Switzerland**
pach@cims.nyu.edu
3 **University of Illinois at Chicago, Chicago, IL, USA**
suk@uic.edu

Abstract

The *Vapnik-Chervonenkis dimension* (in short, VC-dimension) of a *graph* is defined as the VC-dimension of the set system induced by the neighborhoods of its vertices. We show that every n-vertex graph with bounded VC-dimension contains a clique or an independent set of size at least $e^{(\log n)^{1-o(1)}}$. The dependence on the VC-dimension is hidden in the $o(1)$ term. This improves the general lower bound, $e^{c\sqrt{\log n}}$, due to Erdős and Hajnal, which is valid in the class of graphs satisfying any fixed nontrivial hereditary property. Our result is almost optimal and nearly matches the celebrated Erdős-Hajnal conjecture, according to which one can always find a clique or an independent set of size at least $e^{\Omega(\log n)}$. Our results partially explain why most geometric intersection graphs arising in discrete and computational geometry have exceptionally favorable Ramsey-type properties.

Our main tool is a partitioning result found by Lovász-Szegedy and Alon-Fischer-Newman, which is called the "ultra-strong regularity lemma" for graphs with bounded VC-dimension. We extend this lemma to k-uniform hypergraphs, and prove that the number of parts in the partition can be taken to be $(1/\varepsilon)^{O(d)}$, improving the original bound of $(1/\varepsilon)^{O(d^2)}$ in the graph setting. We show that this bound is tight up to an absolute constant factor in the exponent. Moreover, we give an $O(n^k)$-time algorithm for finding a partition meeting the requirements in the k-uniform setting.

1998 ACM Subject Classification G.2.2. Graph Theory

Keywords and phrases VC-dimension, Ramsey theory, regularity lemma

Digital Object Identifier 10.4230/LIPIcs.SoCG.2017.43

1 Introduction

During the relatively short history of computational geometry, there were many breakthroughs that originated from results in extremal combinatorics [23]. Range searching turned out to be closely related to discrepancy theory [9], linear programming to McMullen's Upper Bound theorem and to properties of the facial structure of simplicial complexes [40], motion planning to the theory of Davenport-Schinzel sequences and to a wide variety of other forbidden

* Supported by a Packard Fellowship, by NSF CAREER award DMS 1352121, and by an Alfred P. Sloan Fellowship.
† Supported by Hungarian Science Foundation EuroGIGA Grant OTKA NN 102029, by Swiss National Science Foundation Grants 200020-162884 and 200021-165977.
‡ Supported by NSF grant DMS-1500153.

configuration results [35], graph drawing and VLSI design to the crossing lemma, to the Szemerédi-Trotter theorem, and to flag algebras [41]. A particularly significant example that found many applications in discrete and computational geometry, was the discovery of Haussler and Welzl [26], according to which many geometrically defined set systems have bounded Vapnik-Chervonenkis dimension. Erdős's "Probabilistic Method" [5] or "Random Sampling" techniques, as they are often referred to in computational context, had been observed to be "unreasonably effective" in discrete geometry and geometric approximation algorithms [24]. Haussler and Welzl offered an explanation and a tool: set systems of bounded Vapnik-Chervonenkis dimension admit much smaller hitting sets and "epsilon-nets" than other set systems with similar parameters.

It was also observed a long time ago that geometrically defined graphs and set systems have unusually strong Ramsey-type properties. According to the quantitative version of Ramsey's theorem, due to Erdős and Szekeres [19], every graph on n vertices contains a clique or an independent set of size at least $\frac{1}{2}\log n$. In [14], Erdős proved that this bound is tight up to a constant factor. However, every intersection graph of n segments in the plane, say, has a much larger clique or an independent set, whose size is at least n^{ε} for some $\varepsilon > 0$ [29]. The proof extends to intersection graphs of many other geometric objects [3]. Interestingly, most classes of graphs and hypergraphs in which a similar phenomenon has been observed turned out to have (again!) bounded Vapnik-Chervonenkis dimension. (We will discuss this fact in a little more detail at the end of the Introduction.)

The problem can be viewed as a special case of a celebrated conjecture of Erdős and Hajnal [15], which is one of the most challenging open problems in Ramsey theory. Let P be a *hereditary* property of finite graphs, that is, if G has property P, then so do all of its induced subgraphs. Erdős and Hajnal conjectured 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 $n^{\varepsilon(P)}$. They proved the weaker lower bound $e^{\varepsilon(P)\sqrt{\log n}}$. According to the discovery of Haussler and Welzl mentioned above, the Vapnik-Chervonenkis dimension of most classes of "naturally" defined graphs arising in geometry is bounded from above by a constant d. The property that the Vapnik-Chervonenkis dimension of a graph is at most d, is hereditary.

The aim of this paper is to investigate whether the observation that the Erdős-Hajnal conjecture tends to hold for geometrically defined graphs can be ascribed to the fact that they have bounded VC-dimension. Our first theorem (Theorem 1 below) shows that the answer to this question is likely to be positive. To continue, we need to agree on the basic definitions and terminology.

Let $\mathcal{F}$ be a set system on a ground set V. The *Vapnik-Chervonenkis dimension* (*VC-dimension*, for short) of $\mathcal{F}$ 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 $A \cap S = B$. Given a graph $G = (V, E)$, for any vertex $v \in V$, let $N(v)$ denote the neighborhood of v in G, that is, the set of vertices in V that are connected to v. We note that v itself is not in $N(v)$. Then we say that G has VC-dimension d, if the set system induced by the neighborhoods in G, i.e. $\mathcal{F} = \{N(v) \subset V : v \in V\}$, has VC-dimension d.

The VC-dimension of a set system is one of the most useful combinatorial parameters that measures its complexity, and, apart from its geometric applications, it has proved to be relevant in many other branches of pure and applied mathematics, such as statistics, logic, learning theory, and real algebraic geometry. The notion was introduced by Vapnik and Chervonenkis [42] in 1971, as a tool in mathematical statistics. Kranakis et al. [28] observed that the VC-dimension of a graph can be determined in quasi-polynomial time and,

for bounded degree graphs, in quadratic time. Schaefer [34], addressing a question of Linial, proved that determining the VC-dimension of a set system is Σ_3^p-complete. For each positive integer d, Anthony, Brightwell, and Cooper [6] determined the threshold for the Erdős-Rényi random graph $G(n,p)$ to have VC-dimension d (see also [27]). Given a bipartite graph F, its *closure* is defined as the set of all graphs that can be obtained from F by adding edges between two vertices in the same part. It is known (see [30]) that a class of graphs has bounded VC-dimension if and only if none of its members contains any induced subgraph that belongs to the closure of some fixed bipartite graph F.

Our first result states that the Erdős-Hajnal conjecture "almost holds" for graphs of bounded VC-dimension.

▶ **Theorem 1.** *Let d be a fixed positive integer. If G is an n-vertex graph with VC-dimension at most d, then G contains a clique or independent set of size $e^{(\log n)^{1-o(1)}}$.*

Note that the dependence of the bound on d is hidden in the $o(1)$-notation.

There has been a long history of studying off-diagonal Ramsey numbers, where one is interested in finding the maximum size of an independent set guaranteed in a K_s-free graph on n vertices with s fixed. An old result of Ajtai, Komlós, and Szemerédi [1] states that all such graphs contain independent sets of size $cn^{\frac{1}{s-1}}(\log n)^{\frac{s-2}{s-1}}$. In the other direction, Spencer [38] used the Lovász Local Lemma to show that there are K_s-free graphs on n vertices and with no independent set of size $c'n^{\frac{2}{s+1}}\log n$. This bound was later improved by Bohman and Keevash [8] to $c'n^{\frac{2}{s+1}}(\log n)^{1-\frac{2}{(s+1)(s-2)}}$. In Section 4, we give a simple proof, extending Spencer's argument, showing that there are K_s-free graphs with bounded VC-dimension and with no large independent sets.

▶ **Theorem 2.** *For fixed $s \geq 3$ and $d \geq 5$ such that $d \geq s+2$, there exists a K_s-free graph on n vertices with VC-dimension at most d and no independent set of size $cn^{\frac{2}{s+1}}\log n$, where $c = c(d)$.*

For large s ($s > d$), a result of Fox and Sudakov (Theorem 1.9 in [22]) implies that all n-vertex K_s-free graphs G with VC-dimension d contain an independent set of size $n^{\frac{1}{c\log s}}$ where $c = c(d)$.

Regularity lemma for hypergraphs with bounded VC-dimension. First, we generalize the definition of VC-dimension for graphs to *hypergraphs*. Given a k-uniform hypergraph $H = (V, E)$, for any $(k-1)$-tuple of distinct vertices $v_1, \ldots, v_{k-1} \in V$, let

$$N(v_1, \ldots, v_{k-1}) = \{u \in V : \{v_1, \ldots, v_{k-1}, u\} \in E(H)\}.$$

Then we say that H *has VC-dimension* d, if the set system

$$\mathcal{F} = \{N(v_1, \ldots, v_{k-1}) \subset V : v_1, \ldots, v_{k-1} \in V\}$$

has VC-dimension d. Of course, the hyperedges of H form a set system, but the VC-dimension of this set system is usually *different* from the VC-dimension of H. The latter one is defined as the VC-dimension of the set system $\mathcal{F}$ induced by the neighborhoods of the vertices of H, rather than by the hyperedges.

The *dual* of the set system $(V, \mathcal{F})$ on the ground set V is the set system obtained by interchanging the roles of V and $\mathcal{F}$. That is, it is the set system $(\mathcal{F}, \mathcal{F}^*)$, where the ground set is $\mathcal{F}$ and

$$\mathcal{F}^* = \{\{A \in \mathcal{F} : v \in A\} : v \in V\}.$$

In other words, $\mathcal{F}^*$ is isomorphic to the set system whose ground set is $\binom{V}{k-1}$, and each set is a maximal collection of $(k-1)$-tuples $\{S_1, \ldots, S_p\}$ such that for all i, $v \cup S_i \in E(H)$ for some fixed v. Hence, we have $(\mathcal{F}^*)^* = \mathcal{F}$, and it is known that if $\mathcal{F}$ has VC-dimension d, then $\mathcal{F}^*$ has VC-dimension at most $2^d + 1$. We say that $H = (V, E)$ has *dual VC-dimension* d if $\mathcal{F}^*$ has VC-dimension d.

The main tool used to prove Theorem 1 is an ultra-strong regularity lemma for graphs with bounded VC-dimension obtained by Lovász and Szegedy [30] and Alon, Fischer, and Newman [2]. Here, we extend the ultra-strong regularity lemma to uniform hypergraphs.

Given k vertex subsets $V_1, \ldots, V_k$ of a k-uniform hypergraph H, we write $E(V_1, \ldots, V_k)$ to be the set of edges going across $V_1, \ldots, V_k$, that is, the set of edges with exactly one vertex in each V_i. The *density* across $V_1, \ldots, V_k$ is defined as $\frac{|E(V_1,\ldots,V_k)|}{|V_1|\cdots|V_k|}$. We say that the k-tuple $(V_1, \ldots, V_k)$ is *ε-homogeneous* if the density across it is less than ε or greater than $1 - \varepsilon$. A partition is called *equitable* if any two parts differ in size by at most one.

In [30], Lovász and Szegedy established an *ultra-strong* regularity lemma for graphs ($k = 2$) with bounded VC-dimension, which states that for any $\varepsilon > 0$, there is a (least) $K = K(\varepsilon)$ such that the vertex set V of a graph with VC-dimension d has a partition into at most $K \leq (1/\varepsilon)^{O(d^2)}$ parts such that all but at most an ε-fraction of the pairs of parts are ε-homogeneous. A better bound was obtained by Alon, Fischer, and Newman [2] for bipartite graphs with bounded VC-dimension, who showed that the number of parts in the partition can be taken to be $(d/\varepsilon)^{O(d)}$. Since the VC-dimension of a graph G is equivalent to the dual VC-dimension of G, we generalize their result to hypergraphs with the following result.

▶ **Theorem 3.** *Let $\varepsilon \in (0, 1/4)$ and let $H = (V, E)$ be an n-vertex k-uniform hypergraph with dual VC-dimension d. Then V has an equitable partition $V = V_1 \cup \cdots \cup V_K$ with $8/\varepsilon \leq K \leq c(1/\varepsilon)^{2d+1}$ parts such that all but an ε-fraction of the k-tuples of parts are ε-homogeneous. Here $c = c(d, k)$ is a constant depending only on d and k. Moreover, there is an $O(n^k)$ time algorithm for computing such a partition.*

Our next result shows that the partition size in the theorem above is tight up to an absolute constant factor in the exponent.

▶ **Theorem 4.** *For $d \geq 16$ and $\varepsilon \in (0, 1/100)$, there is a graph G with VC-dimension d such that any equitable vertex partition on G with the property that all but an ε-fraction of the pairs of parts are ε-homogeneous, requires at least $(5\varepsilon)^{-d/4}$ parts.*

Semi-algebraic graphs vs. graphs with bounded VC-dimension. A *semi-algebraic* graph G, is a graph whose vertices are points in $\mathbb{R}^d$ and edges are pairs of points that satisfy a semi-algebraic relation of constant complexity.[1] In a sequence of recent works [3, 11, 21], several authors have shown that classical Ramsey and Turán-type results in combinatorics can be significantly improved for semi-algebraic graphs.

It follows from the Milnor-Thom theorem (see [31]) that semi-algebraic graphs of bounded complexity have bounded VC-dimension. Therefore, all results in this paper on properties of graphs of bounded VC-dimension apply to semi-algebraic graphs of bounded description complexity. However, a graph being semi-algebraic of bounded complexity is a much more restrictive condition than having bounded VC-dimension. In particular, it is known (it

[1] A binary semi-algebraic relation E on a point set $P \subset \mathbb{R}^d$ is the set of pairs of points (p, q) from P whose $2d$ coordinates satisfy a boolean combination of a fixed number of polynomial inequalities.

follows, e.g., from [6]) that for each $\varepsilon > 0$ there is a positive integer $d = d(\varepsilon)$ such that the number of n-vertex graphs with VC-dimension d is $2^{\Omega(n^{2-\varepsilon})}$, while the Milnor-Thom theorem can be used to deduce that the number of n-vertex semi-algebraic graphs coming from a relation with bounded "description complexity" is only $2^{O(n \log n)}$. Furthermore, it is known [3] that semi-algebraic graphs have the *strong Erdős-Hajnal property*, that is, there exists a constant $\delta > 0$ such that every n-vertex semi-algebraic graph of bounded complexity contains a complete or an empty *bipartite* graph whose parts are of size at least δn. This is not true, in general, for graphs with bounded VC-dimension. In particular, the probabilistic construction in Section 4 shows the following.

▶ **Theorem 5.** *For fixed $d \geq 5$ and for every sufficiently large n, there is an n-vertex graph $G = (V, E)$ with VC-dimension at most d with the property that there are no two disjoint subsets $A, B \subset V(G)$ such that $|A|, |B| \geq 4n^{4/d} \log n$ and (A, B) is homogeneous, that is, either $A \times B \subset E(G)$ or $(A \times B) \cap E(G) = \emptyset$.*

It follows from a result of Alon *et al.* [3] that a stronger regularity lemma holds for semi-algebraic graphs of bounded description complexity, where all but an ε-fraction of the pairs of parts in the equitable partition are complete or empty, instead of just ε-homogeneous as in the bounded VC-dimension case (see [32]). This result was further extended to k-uniform hypergraphs by Fox et al. [20], and the authors [21] recently showed that it holds with a polynomial number of parts.

Organization. In the next section, we prove Theorem 3. In Section 3, we prove Theorem 1, which nearly settles the Erdős-Hajnal conjecture for graphs with bounded VC-dimension. In Section 4, we prove Theorems 2 and 5. We conclude by discussing a number of other results for graphs and hypergraphs with bounded VC-dimension. We systemically omit floors and ceilings whenever they are not crucial for sake of clarity in our presentation. All logarithms are natural logarithms.

2 Regularity partition for hypergraphs with bounded VC-dimension

In this section, we prove Theorem 3. We start by recalling several classic results on set systems with bounded VC-dimension. Let $\mathcal{F}$ be a set system on a ground set V. The *primal shatter function* of $\mathcal{F}$ is defined as

$$\pi_{\mathcal{F}}(z) = \max_{V' \subset V, |V'|=z} |\{A \cap V' : A \in \mathcal{F}\}|.$$

In other words, $\pi_{\mathcal{F}}(z)$ is a function whose value at z is the maximum possible number of distinct intersections of the sets of $\mathcal{F}$ with a z-element subset of V. The *dual shatter function* of $(V, \mathcal{F})$, denoted by $\pi^*_{\mathcal{F}}$, whose value at z is defined as the maximum number of equivalence classes on V defined by a z-element subfamily $\mathcal{Y} \subset \mathcal{F}$, where two points $x, y \in V$ are *equivalent* with respect to $\mathcal{Y}$ if x belongs to the same sets of $\mathcal{Y}$ as y does. In other words, the dual shatter function of $\mathcal{F}$ is the primal shatter function of the dual set system $\mathcal{F}^*$.

The VC-dimension of $\mathcal{F}$ is closely related to its shatter functions. A famous result of Sauer [33], Shelah [36], Perles, and Vapnik-Chervonenkis [42] states the following.

▶ **Lemma 6.** *If $\mathcal{F}$ is a set system with VC-dimension d, then*

$$\pi_{\mathcal{F}}(z) \leq \sum_{i=0}^{d} \binom{z}{i}.$$

On the other hand, suppose that the primal shatter function of $\mathcal{F}$ satisfies $\pi_{\mathcal{F}}(z) \leq cz^d$ for all z. Then, if the VC-dimension of $\mathcal{F}$ is d_0, we have $2^{d_0} \leq c(d_0)^d$, which implies $d_0 \leq 4d\log(cd)$. It is known that if $\mathcal{F}$ has VC-dimension d, then $\mathcal{F}^*$ has VC-dimension at most $2^d + 1$.

Given two sets $A_1, A_2 \in \mathcal{F}$, the *symmetric difference* of A_1 and A_2, denoted by $A_1 \triangle A_2$, is the set $(A_1 \cup A_2) \setminus (A_1 \cap A_2)$. We say that the set system $\mathcal{F}$ is δ-*separated* if for any two sets $A_1, A_2 \in \mathcal{F}$ we have $|A_1 \triangle A_2| \geq \delta$. The following *packing lemma* was proved by Haussler in [25].

▶ **Lemma 7.** *Let $\mathcal{F}$ be a set system on a ground set V such that $|V| = n$ and $\pi_{\mathcal{F}}(z) \leq cz^d$ for all z. If $\mathcal{F}$ is δ-separated, then $|\mathcal{F}| \leq c_1(n/\delta)^d$ where $c_1 = c_1(c, d)$.*

We will use Lemma 7 and the following lemma to prove Theorem 3.

▶ **Lemma 8.** *Let $0 < \varepsilon < 1/2$ and $H = (W_1 \cup \cdots \cup W_k, E)$ be a k-partite k-uniform hypergraph such that $|W_i| = m$ for all i. If $(W_1, \ldots, W_k)$ is not ε-homogeneous, then there are at least εm^{k+1} pairs of k-tuples (e, e'), where $|e \cap e'| = k - 1$, $e \in E(H)$, $e' \notin E(H)$, and $|e \cap W_i| = |e' \cap W_i| = 1$ for all i.*

Proof. Let ε_j be the fraction of pairs of k-tuples (e, e'), each containing one vertex in each W_i and agree on all vertices except in W_j, and e is an edge and e' is not an edge. It suffices to show that $\varepsilon_1 + \varepsilon_2 + \cdots + \varepsilon_k \geq \varepsilon$.

Pick vertices $a_i, b_i \in W_i$ uniformly at random with repetition for $i = 1, 2, \ldots, k$. For $0 \leq i \leq k$, let $e_i = \{a_j : j \leq i\} \cup \{b_j : j > i\}$. In particular, $e_k = (a_1, \ldots, a_k)$ and $e_0 = (b_1, \ldots, b_k)$. Then let X be the event that e_0 and e_k have different adjacency, that is, e_0 is an edge and e_k is not an edge, or e_0 is not an edge and e_k is an edge. Then we have

$$Pr[X] \geq 2\varepsilon(1 - \varepsilon) \geq \varepsilon,$$

since $(W_1, \ldots, W_k)$ is not homogeneous. Let X_i be the event that e_i and e_{i+1} have different adjacency, and let Y be the event that at least one event X_i occurs. Then by the union bound, we have

$$Pr[Y] \leq Pr[X_0] + Pr[X_1] + \cdots + Pr[X_{k-1}] = \varepsilon_1 + \varepsilon_2 + \cdots + \varepsilon_k.$$

On the other hand, if X occurs, then Y occurs. Therefore $\varepsilon_1 + \varepsilon_2 + \cdots + \varepsilon_k \geq Pr[Y] \geq Pr[X] \geq \varepsilon$, which completes the proof. ◀

Proof of Theorem 3. Let $0 < \varepsilon < 1/2$ and $H = (V, E)$ be an n-vertex k-uniform hypergraph with dual VC-dimension d. For every vertex $v \in V$, let $N(v)$ denote the set of $(k-1)$-tuples $S \in \binom{V}{k-1}$ such that $v \cup S \in E(H)$. Let $\mathcal{F}$ be the set-system whose ground set is $\binom{V}{k-1}$, and $A \in \mathcal{F}$ if and only if $A = N(v)$ for some vertex $v \in V$. Hence $\mathcal{F} = \{N(v) : v \in V\}$ has VC-dimension d. Set $\delta = \frac{\varepsilon^2}{4k^2}\binom{n}{k-1}$. By examining each vertex and its neighborhood one by one, we greedily construct a maximal set $S \subset V(H)$ such that $\mathcal{F}' = \{N(s) : s \in S\}$ is δ-separated. By Lemma 7, we have $|S| \leq c_1(4k^2/\varepsilon^2)^d$. Let $S = \{s_1, s_2, \ldots, s_{|S|}\}$.

We define a partition $\mathcal{Q} : V = U_1 \cup \cdots \cup U_{|S|}$ of the vertex set such that $v \in U_i$ if i is the smallest index such that $|N(v) \triangle N(s_i)| < \delta$. Such an i always exists, since S is maximal. By the triangle inequality, for $u, v \in U_i$, we have $|N(u) \triangle N(v)| < 2\delta$. Set $K = 8k|S|/\varepsilon$. Partition each part U_i into parts of size $|V|/K = n/K$ and possibly one additional part of size less than n/K. Collect these additional parts and divide them into parts of size $|V|/K$ to obtain an equitable partition $\mathcal{P} : V = V_1 \cup \cdots \cup V_K$ into K parts. The number of vertices of V belonging to parts V_i that are not fully contained in one part of $\mathcal{Q}$ is at most $|S||V|/K$. Hence, the fraction of (unordered) k-tuples $(V_{i_1}, \ldots, V_{i_k})$ such that at least one of the parts

is not fully contained in some part of $\mathcal{Q}$ is at most $k|S|/K = \varepsilon/8$. Let X denote the set of unordered k-tuples of parts $(V_{i_1}, \ldots, V_{i_k})$ such that each part is fully contained in a part of $\mathcal{Q}$ (though, in not necessarily the same part) and $(V_{i_1}, \ldots, V_{i_k})$ is not ε-homogeneous.

Let T be the set of pairs of k-tuples (e, e'), such that $|e \cap e'| = k-1$, $e \in E(H)$, $e' \notin E(H)$, $|e \cap V_{i_j}| = |e' \cap V_{i_j}| = 1$ for $j = 1, 2, \ldots, k$, and $(V_{i_1}, \ldots, V_{i_k}) \in X$. Notice that for $(e, e') \in T$, such that $e \cap V_{i_j} = b$, $e' \cap V_{i_j} = b'$, $b \neq b'$, and V_{i_j} lies completely inside a part in $\mathcal{Q}$, we have $|N(b) \triangle N(b')| \leq 2\delta$. Therefore

$$|T| \leq K \left(\frac{n}{K}\right)^2 2\delta \leq \frac{\varepsilon^2}{2Kk^2} n^2 \binom{n}{k-1}.$$

On the other hand, by Lemma 8, every k-tuple of parts $(V_{i_1}, \ldots, V_{i_k})$ that is not ε-homogeneous gives rise to at least $\varepsilon(n/K)^{k+1}$ pairs (e, e') in T. Hence $|T| \geq |X|\varepsilon(n/K)^{k+1}$, which implies

$$|X| \leq (\varepsilon/2)\binom{K}{k}.$$

Thus, the fraction of k-tuples of parts in $\mathcal{P}$ that are not ε-homogeneous is at most $\varepsilon/8 + \varepsilon/2 < \varepsilon$, and $K \leq c(1/\varepsilon)^{2d+1}$ where $c = c(k, d)$.

Finally, it remains to show that the partition $\mathcal{P}$ can be computed in $O(n^k)$ time. Given two vertices $s, v, \in V$, we have $|N(s) \triangle N(v)| = |N(s)| + |N(v)| - 2|N(s) \cap N(v)|$. Therefore we can determine if $|N(s) \triangle N(u)| < \delta$ in $O(n^{k-1})$ time. Hence the maximal set $S \subset V$ described above (and therefore the partition $\mathcal{Q}$) can be computed in $O(n^k)$ time since $|S| \leq n$. The final equitable partition $\mathcal{P}$ requires an additional $O(n)$ time, which gives a total running time of $O(n^k)$. ◀

We now establish Theorem 4 which shows that the partition size in Theorem 3 is tight up to an absolute constant factor.

Proof of Theorem 4. Given two vertex subsets X, Y of a graph G, we write $e_G(X, Y)$ for the number of edges between X and Y in G, and write $d_G(X, Y)$ for the density of edges between X and Y, that is, $d_G(X, Y) = \frac{e_G(X,Y)}{|X||Y|}$. The pair (X, Y) is said to be (ε, δ)-*regular* if for all $X' \subset X$ and $Y' \subset Y$ with $|X'| \geq \delta|X|$ and $|Y'| \geq \delta|Y|$, we have $|d_G(X, Y) - d_G(X', Y')| \leq \varepsilon$. In the case that $\varepsilon = \delta$, we just say ε-*regular*. We will make use of the following construction due to Conlon and Fox.

▶ **Lemma 9** ([10]). *For $d \geq 16$ and $\varepsilon \in (0, 1/100)$, there is a graph H on $n = \lceil (5\varepsilon)^{-d/2} \rceil$ vertices such that for every equitable vertex partition of H with at most $\sqrt{n}$ parts, there are at least an ε-fraction of the pairs of parts which are not $(4/5)$-regular.*

Let $H = (V, E)$ be the graph obtained from Lemma 9 on $n = \lceil (5\varepsilon)^{-d/2} \rceil$ vertices, where $\varepsilon \in (0, 1/100)$ and $d \geq 16$, and consider a random subgraph $G \subset H$ by picking each edge in E independently with probability $p = n^{-2/d} = 5\varepsilon$. Then we have the following.

▶ **Lemma 10.** *In the random subgraph G, with probability at least $1 - n^{-2}$, every pair of disjoint subsets $X, Y \subset V$, with $|X| \leq |Y|$, satisfy*

$$|e_G(X, Y) - p \cdot e_H(X, Y)| < \sqrt{g}, \tag{1}$$

where $g = 2|X||Y|^2 \ln(ne/|Y|)$.

Proof. For fixed sets $X, Y \subset V(G)$, where $|X| = u_1$ and $|Y| = u_2$, let $E_H(X,Y) = \{e_1, \ldots, e_m\}$. We define $S_i = 1$ if edge e_i is picked and $S_i = 0$ otherwise, and set $S = S_1 + \cdots + S_m$. A Chernoff-type estimate (see Theorem A.1.4 in [5]) implies that for $a > 0$, $Pr[|S - pm| > a] < 2e^{-2a^2/m}$. Since $m \leq u_1 u_2$, the probability that (1) does not hold is less than $2e^{-2g/(u_1u_2)}$. By the union bound, the probability that there are disjoint sets $X, Y \subset V(G)$ for which (1) does not hold is at most

$$\sum_{u_2=1}^{n}\sum_{u_1=1}^{u_2}\binom{n}{u_2}\binom{n-u_2}{u_1}2e^{-2g/(u_1u_2)} \leq \sum_{u_2=1}^{n}\sum_{u_1=1}^{u_2}\left(\frac{ne}{u_2}\right)^{u_2}\left(\frac{ne}{u_1}\right)^{u_1}2e^{-2g/(u_1u_2)}$$
$$\leq \sum_{u_2=1}^{n}\sum_{u_1=1}^{u_2}2\left(\frac{ne}{u_2}\right)^{-2u_2} \leq n^{-2}. \quad \blacktriangleleft$$

By the analysis in Section 4, the probability that G has VC-dimension at least $d+1$ is at most

$$\binom{n}{d+1}n^{2^{d+1}}p^{(d+1)2^d} \leq n^{d+1}n^{-2^{d+1}/d} < \frac{1}{10},$$

since $d \geq 16$. Therefore, the union bound implies that there is a subgraph $G \subset H$ such that G has VC-dimension at most d, and every pair of disjoint subsets $X, Y \subset V$, with $|X| \leq |Y|$, satisfy

$$|e_G(X,Y) - p \cdot e_H(X,Y)| < \sqrt{2|X||Y|^2 \ln(ne/|Y|)}. \tag{2}$$

We will now show that for every equitable vertex partition of G into fewer than $\sqrt{n} = (5\varepsilon)^{-d/4}$ parts, there are at least an ε-fraction of the pairs of parts which are not ε-homogenous.

Let $\mathcal{P}$ be a equitable partition on V into t parts, where $t < \sqrt{n} = (5\varepsilon)^{-d/4}$. By Lemma 9, there are at least $\varepsilon\binom{t}{2}$ pairs of parts in $\mathcal{P}$ which are not $(4/5)$-regular in H. Let (X,Y) be such a pair. Then there are subsets $X' \subset X$ and $Y' \subset Y$ such that $|X'| \geq 4|X|/5$, $|Y'| \geq 4|Y|/5$, and

$$|d_H(X,Y) - d_H(X',Y')| \geq 4/5.$$

Moreover, by (2), we have

$$|e_G(X,Y) - p \cdot e_H(X,Y)| \leq \sqrt{2}\left(\frac{n}{t}\right)^{3/2}\ln(te) \leq \frac{\sqrt{2}\ln(te)}{n^{1/4}}(n/t)^2.$$

Since $d \geq 16$ and $\varepsilon \in (0, 1/100)$, this implies

$$|e_G(X,Y) - p \cdot e_H(X,Y)| \leq (5\varepsilon)^2\sqrt{2}\ln(te)(n/t)^2 \leq \frac{\varepsilon}{4}(n/t)^2.$$

Hence $|d_G(X,Y) - p \cdot d_H(X,Y)| \leq \varepsilon/4$. Therefore we have

$$|d_G(X',Y') - d_G(X,Y)| \geq p \cdot |d_H(X',Y') - d_H(X,Y)| - 2\frac{\varepsilon}{4} \geq 4\varepsilon - \frac{\varepsilon}{2} > 3\varepsilon.$$

Finally, it is easy to see that (X,Y) is not ε-homogeneous in G. Indeed if (X,Y) were ε-homogeneous, then we have either $d_G(X,Y) < \varepsilon$ or $d_G(X,Y) > 1 - \varepsilon$. In the former case we have $d_G(X',Y') > 3\varepsilon$, which implies

$$e_G(X,Y) \geq e_G(X',Y') > 3\varepsilon\frac{4|X|}{5}\frac{4|Y|}{5} > \varepsilon|X||Y|,$$

contradiction. In the latter case, we have $d(X',Y') < 1 - 3\varepsilon$, and a similar analysis shows that $e_G(X,Y) < (1-\varepsilon)|X||Y|$, contradiction.

Thus, any equitable vertex partition on G such that all but an ε-fraction of the pairs of parts are ε-homogeneous, requires at least $(5\varepsilon)^{-d/4}$ parts. $\blacktriangleleft$

3 Proof of Theorem 1

The family $\mathcal{G}$ of all *complement reducible graphs*, or *cographs*, is defined as follows: The graph with one vertex is in $\mathcal{G}$, and if two graphs $G, H \in \mathcal{G}$, then so does their disjoint union, and the graph obtained by taking their disjoint union and adding all edges between G and H. Clearly, every induced subgraph of a cograph is a cograph, and it is well known that every cograph on n vertices contains a clique or independent set of size $\sqrt{n}$.

Let $f_d(n)$ be the largest integer f such that every graph G with n vertices and VC-dimension at most d has an induced subgraph on f vertices which is a cograph. Cographs are perfect graphs, so that Theorem 1 is an immediate consequence of the following result.

▶ **Theorem 11.** *For any $\delta \in (0, 1/2)$ and for every integer $d \geq 1$, there is a $c = c(d, \delta)$ such that $f_d(n) \geq e^{c(\log n)^{1-\delta}}$ for every n.*

Proof. For simplicity, let $f(n) = f_d(n)$. The proof is by induction on n. The base case $n = 1$ is trivial. For the inductive step, assume that the statement holds for all $n' < n$. Let $\delta > 0$ and let $G = (V, E)$ be an n-vertex graph with VC-dimension d. We will determine $c \in (0, 1)$ later.

Set $\varepsilon = (1/32)e^{-3c(\log n)^{1-\delta}}$. We apply Theorem 3 to obtain an equitable partition $\mathcal{P} : V = V_1 \cup \cdots \cup V_K$ into at most $K \leq \varepsilon^{-c_4}$ parts, where $c_4 = O(d)$, such that all but an ε-fraction of the pairs of parts are ε-homogeneous. We call an unordered pair of distinct vertices (u, v) *bad* if at least one of the following holds:

1. (u, v) lie in the same part, or
2. $u \in V_i$ and $v \in V_j$, $i \neq j$, where (V_i, V_j) is not ε-homogeneous, or
3. $u \in V_i$ and $v \in V_j$, $i \neq j$, $uv \in E(G)$ and $|E(V_i, V_j)| < \varepsilon |V_i||V_j|$, or
4. $u \in V_i$ and $v \in V_j$, $i \neq j$, $uv \notin E(G)$ and $|E(V_i, V_j)| > (1 - \varepsilon)|V_i||V_j|$.

By Theorem 3, the number of bad pairs of vertices in G is at most

$$K\binom{n/K}{2} + \left(\frac{n}{K}\right)^2 \varepsilon \binom{K}{2} + \varepsilon \left(\frac{n}{K}\right)^2 (1 - \varepsilon)\binom{K}{2} \leq 2\varepsilon \binom{n}{2}.$$

By Turán's Theorem, there is a subset $R \subset S$ of at least $\frac{1}{4\varepsilon}$ vertices such that R does not contain any bad pairs. This implies that all vertices of R are in distinct parts of $\mathcal{P}$. Furthermore, if uv are adjacent in R, then the corresponding parts V_i, V_j satisfy $|E(V_i, V_j)| \geq (1 - \varepsilon)|V_i||V_j|$, and if uv are not adjacent, then we have $|E(V_i, V_j)| < \varepsilon |V_i||V_j|$. Since the induced graph $G[R]$ has VC-dimension at most d, $G[R]$ contains a cograph U_0 of size $t = f(1/(4\varepsilon))$, which, by the induction hypothesis, is a set of size at least $e^{c(\log(1/4\varepsilon))^{1-\delta}}$. Without loss of generality, we denote the corresponding parts of U_0 as $V_1, \ldots, V_t$. Each part contains n/K vertices.

For each vertex $u \in V_1$, let $d_b(u)$ denote the number of bad pairs uv, where $v \in V_i$ for $i = 2, \ldots, t$. Then there is a subset $V_1' \subset V_1$ of size $\frac{n}{2K}$, such that each vertex $u \in V_1'$ satisfies $d_b(u) < 8t\varepsilon(n/K)$. Indeed, otherwise at least $n/(2K)$ vertices in V_1 satisfies $d_b(u) \geq 8t\varepsilon(n/K)$, which implies

$$\frac{n}{2K}\frac{8t\varepsilon n}{K} \leq \sum_{u \in V_1'} d_b(u) \leq \sum_{u \in V_1} d_b(u) \leq \varepsilon(t-1)\left(\frac{n}{K}\right)^2,$$

and hence a contradiction. By the induction hypothesis, we can find a subset $U_1 \subset V_1'$ such that the induced subgraph $G[U_1]$ is a cograph of size $f(n/(2K))$. If the inequality

$$f\left(\frac{n}{2K}\right) 8t\varepsilon \frac{n}{K} > \frac{n}{4tK}$$

is satisfied, then we have

$$f^3(n) \geq f\left(\frac{n}{2K}\right) t^2 > \frac{1}{32\varepsilon}.$$

By setting ε such that $\frac{1}{\varepsilon} = 32e^{3c(\log n)^{1-\delta}}$, we have $f(n) \geq e^{c(\log n)^{1-\delta}}$ and we are done.

Therefore, we can assume that

$$f\left(\frac{n}{2K}\right) 8t\varepsilon\frac{n}{K} \leq \frac{n}{4tK}.$$

Hence, by deleting any vertex $v \in V_2 \cup \cdots \cup V_t$ that is in a bad pair with a vertex in U_1, we have deleted at most $\frac{n}{4tK}$ vertices in each V_i for $i = 2, \ldots, t$.

We repeat this entire process on the remaining vertices in $V_2, \ldots, V_t$. At step i, we will find a subset $U_i \subset V_i$ that induces a cograph of size

$$f\left(\frac{n}{2K} - i\frac{n}{4Kt}\right) \geq f\left(\frac{n}{4K}\right),$$

and again, if the inequality

$$f\left(\frac{n}{4K}\right) 8t\varepsilon\frac{n}{K} > \frac{n}{4tK}$$

is satisfied, then we are done by the same argument as above. Therefore we can assume that our cograph $G[U_i]$ has the property that there are at most $n/(4tK)$ bad pairs between U_i and V_j for $j > i$. At the end of this process, we obtain subsets $U_1, \ldots, U_t$ such that the union $U_1 \cup \cdots \cup U_t$ induces a cograph of size at least $tf\left(\frac{n}{4K}\right)$. Therefore we have

$$\begin{aligned} f(n) &\geq f\left(\tfrac{1}{4\varepsilon}\right) f\left(\tfrac{n}{4K}\right) \\ &\geq f\left(e^{3c(\log n)^{1-\delta}}\right) f\left(e^{\log n - c\cdot c_5(\log n)^{1-\delta}}\right) \\ &\geq e^{c\left(3c(\log n)^{1-\delta}\right)^{1-\delta}} e^{c\left(\log n - c\cdot c_5(\log n)^{1-\delta}\right)^{1-\delta}}, \end{aligned} \tag{3}$$

where $c_5 = c_5(d)$. Notice we have the following estimate:

$$\begin{aligned} \left(\log n - c\cdot c_5(\log n)^{1-\delta}\right)^{1-\delta} &= (\log n)^{1-\delta}\left(1 - \tfrac{c\cdot c_5}{\log^{\delta} n}\right)^{1-\delta} \\ &\geq (\log n)^{1-\delta}\left(1 - \tfrac{c\cdot c_5}{(\log n)^{\delta}}\right) \\ &\geq (\log n)^{1-\delta} - c\cdot c_5(\log n)^{1-2\delta}. \end{aligned} \tag{4}$$

Plugging (4) into (3) gives

$$\begin{aligned} f(n) &\geq e^{c\left(3c(\log n)^{1-\delta}\right)^{1-\delta}} \cdot e^{c(\log n)^{1-\delta} - c^2\cdot c_5(\log n)^{1-2\delta}} \\ &= e^{c(\log n)^{1-\delta}} \cdot e^{\left(3^{1-\delta}c^2(\log n)^{1-2\delta+\delta^2} - c^2 c_5(\log n)^{1-2\delta}\right)}. \end{aligned} \tag{5}$$

The last inequality follows from the fact that $c < 1$. Let $n_0 = n_0(d, \delta)$ be the minimum integer such that for all $n \geq n_0$ we have

$$3^{1-\delta}(\log n)^{1-2\delta+\delta^2} - c_5(\log n)^{1-2\delta} \geq 0.$$

We now set $c = c(d, t)$ to be sufficiently small such that the statement is trivial for all $n < n_0$. Hence we have $f(n) \geq e^{c(\log n)^{1-\delta}}$ for all n. ◀

4 Random constructions

Here we prove Theorems 2 and 5. The proof of Theorem 2 uses the Lovász Local Lemma [17] in a similar manner as Spencer [38] to give a lower bound on Ramsey numbers.

▶ **Lemma 12** (Lovász Local Lemma). *Let $\mathcal{A}$ be a finite set of events in a probability space. For $A \in \mathcal{A}$ let $\Gamma(A)$ be a subset of $\mathcal{A}$ such that A is independent of all events in $\mathcal{A} \setminus (\{A\} \cup \Gamma(A))$. If there is a function $x : \mathcal{A} \to (0,1)$ such that for all $A \in \mathcal{A}$,*

$$Pr[A] \leq x(A) \prod_{B \in \Gamma(A)} (1 - x(B)),$$

then $Pr\left[\bigcap_{A \in \mathcal{A}} \overline{A}\right] \geq \prod_{A \in \mathcal{A}} (1 - x(A))$. In particular, with positive probability no event in $\mathcal{A}$ holds.

Proof of Theorem 2. Let s and d be positive integers such that $d > s+2$. Let $G(n,p)$ denote the random graph on n vertices in which each edge appears with probability p independently of all the other edges, where $p = n^{-2/(s+1)}$ and n is a sufficiently large number. For each set S of s vertices, let A_S be the event that S induces a complete graph. For each set T of t vertices, let B_T be the event that T induces an empty graph. Clearly, we have $Pr[A_S] = p^{\binom{s}{2}}$ and $Pr[B_T] = (1-p)^{\binom{t}{2}}$.

For each set D of d vertices, let C_D be the event that D is shattered. Then

$$\begin{aligned} Pr[C_D] &\leq \prod_{W \subset D} Pr[\exists v \in V(G) : N(v) \cap D = W] \\ &= \prod_{W \subset D} \left(1 - \left(1 - p^{|W|}(1-p)^{d-|W|}\right)^n\right) \\ &= \prod_{j=0}^{d} \left(1 - \left(1 - p^j(1-p)^{d-j}\right)^n\right)^{\binom{d}{j}} \\ &\leq \prod_{j=1}^{d} \left(n \cdot p^j(1-p)^{d-j}\right)^{\binom{d}{j}} \leq \prod_{j=1}^{d} n^{\binom{d}{j}} \cdot p^{j\binom{d}{j}} \leq n^{2^d} \cdot p^{d2^{d-1}}. \end{aligned}$$

Next we estimate the number of events dependent on each A_S, B_T and C_D. Let $S \subset V$ such that $|S| = s$. Then the event A_S is dependent on at most $\binom{s}{2}\binom{n}{s-2} \leq s^2 n^{s-2}$ events $A_{S'}$, where $|S'| = s$. Likewise, A_S is dependent on at most $\binom{n}{t}$ events B_T where $|T| = t$. Finally A_S is dependent on at most $\binom{s}{2}\binom{n}{d-2} \leq s^2 n^{d-2}$ events C_D where $|D| = d$.

Let $T \subset V$ be a set of vertices such that $|T| = t$. Then the event B_T is dependent on at most $\binom{t}{2}\binom{n}{s-2} \leq t^2 n^{s-2}$ events A_S where $|S| = s$. Likewise, B_T is dependent on at most $\binom{n}{t}$ events $B_{T'}$ where $|T'| = t$. Finally B_T is dependent on at most $\binom{t}{2}\binom{n}{d-2} \leq t^2 n^{d-2}$ events C_D where $|D| = d$.

Let $D \subset V$ be a set of vertices such that $|D| = d$. Then the event C_D is dependent on at most $\binom{d}{2}\binom{n}{s-2} \leq d^2 n^{s-2}$ events A_S where $|S| = s$. Likewise, C_D is dependent on at most $\binom{n}{t}$ events B_T where $|T| = t$. Finally C_D is dependent on at most $\binom{d}{2}\binom{n}{d-2} \leq d^2 n^{d-2}$ events $C_{D'}$ where $|D'| = d$.

By Lemma 12, it suffices to find three real numbers $x, y, z \in (0,1)$ such that

$$p^{\binom{s}{2}} \leq x(1-x)^{s^2 n^{s-2}}(1-y)^{\binom{n}{t}}(1-z)^{s^2 n^{d-2}}, \tag{6}$$

$$(1-p)^{\binom{t}{2}} \leq y(1-x)^{t^2 n^{s-2}}(1-y)^{\binom{n}{t}}(1-z)^{t^2 n^{d-2}}, \text{ and} \tag{7}$$

$$n^{2^d} \cdot p^{d2^{d-1}} \leq z(1-x)^{d^2 n^{s-2}}(1-y)^{\binom{n}{t}}(1-z)^{d^2 n^{d-2}}. \tag{8}$$

Recall $p = n^{\frac{-2}{s+1}}$, $s \geq 3$, and $d > s+2$. We now set $t = c_1 n^{\frac{2}{s+1}}(\log n)$, $x = c_2 n^{\frac{-2\binom{s}{2}}{s+1}}$, $y = e^{-c_3 n^{\frac{2}{s+1}}(\log n)^2}$, and $z = c_4 n^{2^d - \frac{2}{s+1}d2^{d-1}}$, where c_1, c_2, c_3, c_4 only depend on s and d. By letting $c_1 > 10c_3$, setting c_1, c_2, c_3, c_4 sufficiently large, an easy (but tedious) calculation shows that (6), (7), (8) are satisfied when n is sufficiently large. By Lemma 12, there is an n-vertex K_s-free graph G with VC-dimension at most d and independence number at most $c_1 n^{\frac{2}{s+1}} \log n$. ◀

Proof of Theorem 5. Let $d \geq 5$ and n be a sufficiently large integer. Consider the random n-vertex graph $G = G(n,p)$, where each edge is chosen independently with probability $p = n^{-4/d}$. As n is sufficiently large, the union bound and the analysis above implies that the probability that G has VC-dimension at least d is at most $1/3$.

Let $A, B \subset V(G)$ be vertex subsets, each of size k. The probability that (A,B) is homogenous is at most

$$p^{k^2} + (1-p)^{k^2} \leq n^{-4k^2/d} + e^{-n^{-4/d}k^2}.$$

The probability that G contains a homogeneous pair (A,B), where $|A|, |B| = k$, is at most

$$\binom{n}{k}\binom{n-k}{k}\left(n^{-4k^2/d} + e^{-n^{-4/d}k^2}\right) < 1/3,$$

for $k = 4n^{4/d}\log n$ and n sufficiently large. Thus, again by the union bound, there is a graph with VC-dimension less than d, with no two disjoint subsets $A, B \subset V(G)$ such that (A,B) is homogeneous and $|A|, |B| = 4n^{4/d}\log n$. ◀

5 Concluding remarks

Many interesting results arose in our study of graphs and hypergraphs with bounded VC-dimension. In particular, we strengthen several classical results from extremal hypergraph theory for hypergraphs with bounded VC-dimension. Below, we briefly mention two of them.

Hypergraphs with bounded VC-dimension. Erdős, Hajnal, and Rado [16] showed that every 3-uniform hypergraph on n vertices contains a clique or independent set of size $c\log\log n$. A famous open question of Erdős asks if $\log\log n$ is the correct order of magnitude for Ramsey's theorem for 3-uniform hypergraphs. According to the best known constructions, there are 3-uniform hypergraphs on n vertices with no clique or independent set of size $c'\sqrt{\log n}$. For $k \geq 4$, the best known lower and upper bounds on the size of the largest clique or independent set in every n-vertex k-uniform hypergraph is of the form $c\log^{(k-1)} n$ (the $(k-1)$-times iterated logarithm) and $c'\sqrt{\log^{(k-2)} n}$, respectively (see [12] for more details). By combining Theorem 1 with an argument of Erdős and Rado [18], one can significantly improve these bounds for hypergraphs of bounded (neighborhood) VC-dimension.

▶ **Theorem 13.** *Let $k \geq 3$ and $d \geq 1$. Every k-uniform hypergraph on n vertices with VC-dimension d contains a clique or independent set of size $e^{\left(\log^{(k-1)} n\right)^{1-o(1)}}$.*

Geometric constructions given by Conlon et al. [11] show that Theorem 13 is tight apart from the $o(1)$ term in the second exponent. That is, for fixed $k \geq 3$, there are k-uniform hypergraphs on n vertices with VC-dimension $d = d(k)$ such that the largest clique or independent set is of size $O(\log^{(k-2)} n)$.

The Erdős-Hajnal conjecture for tournaments. A *tournament* $T = (V, E)$ on a set V is an orientation of the edges of the complete graph on the vertex set V, that is, for $u, v \in V$ we have either $(u, v) \in E$ or $(v, u) \in E$, but not both. A tournament with no directed cycle is called *transitive*. If a tournament has no subtournament isomorphic to T, then it is called T-free. An old result due to Entringer, Erdős, and Harner [13] and Spencer [39] states that every tournament on n vertices contains a transitive subtournament of size $c \log n$, which is tight apart from the value of the constant factor. Alon, Pach, and Solymosi [4] showed that the Erdős-Hajnal conjecture is equivalent to the following conjecture.

▶ **Conjecture 14.** *For every tournament T, there is a positive $\delta = \delta(T)$ such that every T-free tournament on n vertices has a transitive subtournament of size n^{δ}.*

In particular, it is known that every T-free tournament on n vertices contains a transitive subtournament of size $e^{c\sqrt{\log n}}$, where $c = c(T)$. Another application of the ultra-strong regularity lemma, Theorem 3, improves this bound in the special case where $T = (V, E)$ is *2-colorable*, that is, there is a 2-coloring of $V(T)$ such that each color class induces a transitive subtournament. This follows from the fact that, if T is 2-colorable, then the set system formed by the out-neighborhoods of the vertices of a T-free tournament has VC-dimension at most $c' = c'(T)$. On the other hand, if the same set system has bounded VC-dimension, then the tournament is T'-free for some bounded size 2-colorable T'.

▶ **Theorem 15.** *For a fixed integer $k > 0$, let T be a 2-colorable tournament on k vertices. Every T-free tournament on n vertices contains a transitive subtournament of size $e^{(\log n)^{1-o(1)}}$.*

References

1 M. Ajtai, J. Komlós, and E. Szemerédi, A note on Ramsey numbers, *J. Combin. Theory Ser. A* **29** (1980), 354–360.

2 N. Alon, E. Fischer, and I. Newman, Efficient testing of bipartite graphs for forbidden induced subgraphs, *SIAM J. Comput.* 37 (2007), 959–976.

3 N. Alon, J. Pach, R. Pinchasi, R. Radoičić, and M. Sharir, Crossing patterns of semi-algebraic sets, *J. Combin. Theory Ser. A* 111 (2005), 310–326.

4 N. Alon, J. Pach, J. Solymosi, Ramsey-type theorems with forbidden subgraphs, Combinatorica 21 (2001), 155–170.

5 N. Alon and J. H. Spencer, *The probabilistic method*, 3rd ed., Wiley, 2008.

6 M. Anthony, G. Brightwell, and C. Cooper, The Vapnik-Chervonenkis dimension of a random graph, 14th British Combinatorial Conference (Keele, 1993). *Discrete Math.* **138** (1995), 43–56.

7 T. Bohman, The triangle-free process, *Adv. Math.* **221** (2009), 1653–1677.

8 T. Bohman and P. Keevash, The early evolution of the H-free process, *Invent. Math.* **181** (2010), 291–336.

9 B. Chazelle, *The Discrepancy Method: Randomness and Complexity,* Cambridge University Press, New York, 2000.

10 D. Conlon and J. Fox, Bounds for graph regularity and removal lemmas, *Geom. Funct. Anal.*, 22 (2012), 1191–1256.

11 D. Conlon, J. Fox, J. Pach, B. Sudakov, and A. Suk, Ramsey-type results for semi-algebraic relations, *Trans. Amer. Math. Soc.* 366 (2014), 5043–5065.

12 D. Conlon, J. Fox, and B. Sudakov, Hypergraph Ramsey numbers, *J. Amer. Math. Soc.* 23 (2010), 247–266.

13 R. C. Entringer, P. Erdős, and C. C. Harner, Some extremal properties concerning transitivity in graphs, *Period. Math. Hungar.* 3 (1973), 275–279.

14 P. Erdős, Some remarks on the theory of graphs, *Bull. Amer. Math. Soc.* 53 (1947), 292–294.
15 P. Erdős and A. Hajnal, Ramsey-type theorems, *Discrete Appl. Math.* 25 (1989), 37–52.
16 P. Erdős, A. Hajnal, and R. Rado, Partition relations for cardinal numbers, *Acta Math. Acad. Sci. Hungar.* 16 (1965), 93–196.
17 P. Erdős and L. Lovász, Problems and results on 3-chromatic hypergraphs and some related questions. *Infinite and finite sets* (Colloq., Keszthely, 1973; dedicated to P. Erdős on his 60th birthday), Vol. II, pp. 609–627. Colloq. Math. Soc. János Bolyai, Vol. 10, North-Holland, Amsterdam, 1975.
18 P. Erdős and R. Rado, Combinatorial theorems on classifications of subsets of a given set, *Proc. London Math. Soc.* 3 (1952), 417–439.
19 P. Erdős and G. Szekeres, A combinatorial problem in geometry, *Compos. Math.* 2 (1935), 463–470.
20 J. Fox, M Gromov, V. Lafforgue, A. Naor, and J. Pach, Overlap properties of geometric expanders, *J. Reine Angew. Math.* **671** (2012), 49–83.
21 J. Fox, J. Pach, and A. Suk, A polynomial regularity lemma for semi-algebraic hypergraphs and its applications in geometry and property testing, to appear in *SIAM J. Comput.*
22 J. Fox and B. Sudakov, Density theorems for bipartite graphs and related Ramsey-type results, *Combinatorica* 29 (2009), 153–196.
23 J. E. Goodman, J. O'Rourke, and C. D. Tóth (eds.), *Handbook of Discrete and Computational Geometry,* Chapman Hill/CRC Press, Boca Raton, 2017.
24 S. Har-Peled, *Geometric Approximation Algorithms,* Mathematical Surveys and Monographs, Vol. 173, Amer. Math. Soc., Providence, 2011.
25 D. Haussler, Sphere packing numbers for subsets of the Boolean n-cube with bounded Vapnik-Chervonenkis dimension, *J. Combin. Theory Ser. A*, 69 (1995), 217–232.
26 D. Haussler and E. Welzl, ε-nets and simplex range queries, *Discrete Comput. Geom.* **2** (1987), 127–151.
27 J. Komlós, J. Pach, and G. Woeginger, Almost tight bounds for ε-nets, *Discrete Comput. Geom.* **7** (1992), 163–173.
28 E. Kranakis, D. Krizanc, B. Ruf, J. Urrutia, and G. Woeginger, The VC-dimension of set systems defined by graphs, *Discrete Appl. Math.* **77** (1997), 237–257.
29 D. Larman, J. Matoušek, J. Pach, and J. Törőcsik, A Ramsey-type result for convex sets, *Bull. London Math. Soc.* **26**(2) (1994), 132–136.
30 L. Lovász and B. Szegedy, Regularity partitions and the topology of graphons, An Irregular Mind, Imre Bárány, József Solymosi, and Gábor Sági editors, *Bolyai Society Mathematical Studies* 21 (2010), 415–446.
31 J. Matoušek, *Lectures on Discrete Geometry*, Springer-Verlag, New York, 2002.
32 J. Pach and J. Solymosi, Structure theorems for systems of segments, *Proceeding of the Japanese Conference on Discrete and Computational Geometry* (2000), 308–317.
33 N. Sauer, On the density of families of sets, *J. Combin. Theory Ser. A* 13 (1972), 145–147.
34 M. Schaefer, Deciding the Vapnik-Cervonenkis dimension is Σ_p^3-complete, *J. Comput. System Sci.* **58** (1999), 177–182.
35 M. Sharir and P. K. Agarwal, *Davenport-Schinzel Sequences and Their Geometric Applications*, Cambridge University Press, Cambridge, 1995.
36 S. Shelah, A combinatorial problem; stability and order for models and theories in infinitary languages, *Pacific J. Math.* (1972) 41, 247–261.
37 M. Simonovits and V. Sós, Ramsey-Turán theory, *Discrete Math.* 229 (2001), 293–340.
38 J. Spencer, Asymptotic lower bounds for Ramsey functions, *Discrete Math.* 20 (1977), 69–76.
39 J. Spencer, Random regular tournaments, *Period. Math. Hungar.* 5 (1974), 105–120.
40 R. Stanley, *Combinatorics and Commutative Algebra,* Birkhäuser, Boston, 1996.

41 R. Tamassia, ed., *Handbook of Graph Drawing and Visualization*, Chapman and Hall/CRC Press, Boca Raton, 2013.

42 V. Vapnik and A. Chervonenkis, On the uniform convergence of relative frequencies of events to their probabilities, *Theory Probab. Appl.* 16 (1971), 264–280.

Implementing Delaunay Triangulations of the Bolza Surface

Iordan Iordanov[1] **and Monique Teillaud**[2]

1 **Loria, Inria Centre de recherche Nancy – Grand Est, Université de Lorraine, CNRS UMR 7503, Villers-lès-Nancy, France**
2 **Loria, Inria Centre de recherche Nancy – Grand Est, Université de Lorraine, CNRS UMR 7503, Villers-lès-Nancy, France**

Abstract

The CGAL library offers software packages to compute Delaunay triangulations of the (flat) torus of genus one in two and three dimensions. To the best of our knowledge, there is no available software for the simplest possible extension, i.e., the Bolza surface, a hyperbolic manifold homeomorphic to a torus of genus two.

In this paper, we present an implementation based on the theoretical results and the incremental algorithm proposed last year [2]. We describe the representation of the triangulation, we detail the different steps of the algorithm, we study predicates, and report experimental results.

1998 ACM Subject Classification I.3.5 Computational Geometry and Object Modeling

Keywords and phrases hyperbolic surface, Fuchsian group, arithmetic issues, Dehn's algorithm, CGAL

Digital Object Identifier 10.4230/LIPIcs.SoCG.2017.44

1 Introduction

Motivated by applications in various fields, some packages to compute periodic Delaunay triangulations in the Euclidean spaces $\mathbb{E}^2$ and $\mathbb{E}^3$ have been introduced in the CGAL library [4, 12] and have attracted a number of users. To the best of our knowledge, no software is available to compute periodic triangulations in a hyperbolic space. This would be a natural extension: periodic triangulations in $\mathbb{E}^2$ can be seen as triangulations of the two-dimensional (flat) torus of genus one; similarly, periodic triangulations in the hyperbolic plane $\mathbb{H}^2$ can be seen as triangulations of hyperbolic surfaces. The Bolza surface is a hyperbolic surface with the simplest possible topology, as it is homeomorphic to a genus-two torus. First steps in computing Delaunay triangulations of hyperbolic surfaces have recently been made [2]. Due to lack of space, we refer the reader to that paper for examples of applications.

All previous work mentioned above is generalizing the well-known incremental algorithm introduced by Bowyer [3], which has proved to be reasonably easy to implement and very efficient in practice. For each new point p, the set of conflicting simplices, i.e., simplices whose circumscribing ball contains p, are removed; then their union is triangulated by simply filling it with new simplices with apex p. This simple update operation relies on the fact that the union of conflicting simplices is always a topological ball. As proved earlier [2], for an input set S on a closed hyperbolic surface M, this property is ensured as soon as

$$\mathrm{sys}(M) > 2\delta_S, \tag{1}$$

where $\mathrm{sys}(M)$ denotes the *systole* of M, i.e., the length of a shortest non-contractible loop on M, and δ_S denotes the diameter of the largest disks that do not contain any point of S

in their interior. This condition ensures that there is no cycle of length one or two in the 1-skeleton of the Delaunay triangulation.

Two ideas have been proposed to fulfill this condition [2]. The first one consists in increasing the systole by using covering spaces of M, but it was shown to require at least 32 sheets for the Bolza surface, so this does not lead to a practical method. A more practical idea was quickly sketched in the last section of the same paper; it consists in initializing the triangulation with a set of "dummy" vertices that ensure that largest empty disks are small enough so that inequality (1) holds. As the diameter of largest empty disks cannot increase when new points are inserted, the condition will still be fulfilled when inserting points. If sufficiently many reasonably well-distributed points are inserted, then the dummy vertices can be removed from the triangulation without violating condition (1). In this paper, we elaborate on this approach and propose a first implementation.

We recall some background for the Bolza surface in Section 2. In Section 3 we propose a representation for Delaunay triangulations of the Bolza surface. Then we present the various steps of the construction in Section 4. We investigate the algebraic complexity of the algorithm in Section 5. Finally, we present some results of our implementation.

The source code, all Maple sheets, and the appendix are publicly available at

`https://members.loria.fr/Monique.Teillaud/DT_Bolza_SoCG17/`.

The software package will be submitted for integration in CGAL as soon as the documentation is completed.

2 The Bolza surface

Details and references for the background given in this section can be found in [2].

As the *Poincaré disk model* of the hyperbolic plane $\mathbb{H}^2$ is conformal, it is often used in applications. The hyperbolic plane is represented as the open unit disk $\mathcal{B}$ of the Euclidean plane $\mathbb{E}^2$. The boundary of $\mathcal{B}$ represents the set of points at infinity, denoted as $\mathcal{H}_\infty$. Hyperbolic lines, or geodesics, are represented as diameters of $\mathcal{B}$ or as arcs of circles orthogonal to $\mathcal{H}_\infty$. A hyperbolic circle is represented as a Euclidean circle contained in $\mathcal{B}$ and its hyperbolic center is the limit point of the pencil of circles that it generates with $\mathcal{H}_\infty$.

We denote the group of orientation-preserving isometries on $\mathbb{H}^2$ as $\mathrm{Isom}^+(\mathbb{H}^2)$. By identifying $\mathbb{E}^2$ with the complex plane $\mathbb{C}$, each $g \in \mathrm{Isom}^+(\mathbb{H}^2)$ is a mapping in the form $g(z) = \frac{\alpha z+\beta}{\overline{\beta} z+\overline{\alpha}}$, $z \in \mathbb{C}$ with matrix $g = \begin{bmatrix} \alpha & \beta \\ \overline{\beta} & \overline{\alpha} \end{bmatrix}$, where $\alpha, \beta \in \mathbb{C}$ and $|\alpha|^2 - |\beta|^2 = 1$. We are only interested here in one type of orientation-preserving isometries: the *hyperbolic* isometries, also called *translations*. A hyperbolic translation fixes two points at infinity and no point inside $\mathcal{B}$. The geodesic X_g through the two fixed points of a translation g is called the *axis* of g. Points lying on X_g are all translated along X_g by the same fixed distance $\ell(g)$ called the translation *length*. The length can be computed from the matrix as $\ell(g) = 2 \cdot \mathrm{arcosh}\left(\frac{1}{2}\mathrm{Tr}(g)\right)$, where $\mathrm{Tr}(g)$ denotes the trace of the matrix of g. A point that does not lie on X_g is translated by a distance greater than $\ell(g)$ along a curve equidistant from X_g (of course the distance between a point and its image is measured on the geodesic containing them). See Figure 1-Left.

A *hyperbolic surface* is a connected 2-dimensional manifold such that every point has a neighborhood isometric to a disk of $\mathbb{H}^2$. A closed (i.e., compact) and orientable hyperbolic surface is isometric to a quotient of $\mathbb{H}^2$ under the action of a Fuchsian group Γ (i.e., a discrete subgroup of $\mathrm{Isom}^+(\mathbb{H}^2)$) that contains only translations (and the identity). The *Bolza surface* is the simplest possible closed orientable hyperbolic surface. Consider the

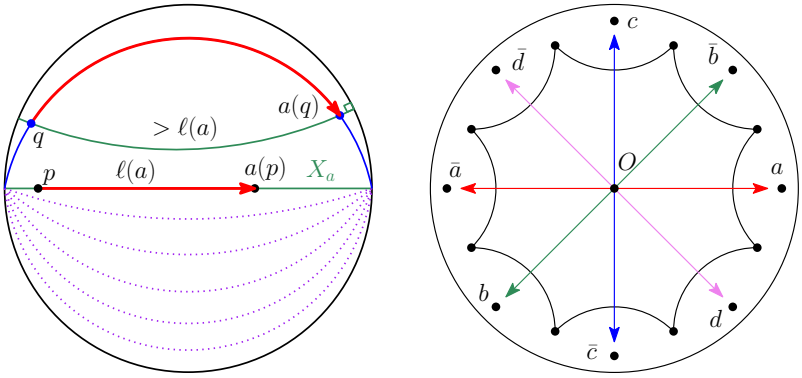


Figure 1 Left: Action of translation a on $\mathbb{H}^2$. **Right:** Regular octagon $\mathcal{D}_O$ and generators of $\mathcal{G}$.

regular hyperbolic octagon $\mathcal{D}_O$ centered at the origin O, with angles equal to $\pi/4$. The four hyperbolic translations a, b, c, and d that identify opposite sides of $\mathcal{D}_O$ generate a Fuchsian group denoted as $\mathcal{G}$. See Figure 1-Right.[1] For simplicity, we also denote as g the image gO of the origin by a translation g of $\mathcal{G}$.

The Bolza surface is defined as the quotient of $\mathbb{H}^2$ under the action of the group $\mathcal{G}$:

$$\mathcal{M} = \mathbb{H}^2/\mathcal{G}.$$

The projection map $\pi : \mathbb{H}^2 \to \mathcal{M} = \mathbb{H}^2/\mathcal{G}$ is a local isometry and a covering projection. The *Dirichlet region* $\mathcal{D}_p(\mathcal{G})$ for $\mathcal{G}$ centered at p is defined as the the closure of the open cell of p in the Voronoi diagram $VD_{\mathbb{H}}(\mathcal{G}p)$ of the infinite set of points $\mathcal{G}p$ in $\mathbb{H}^2$. From the compactness of $\mathcal{M}$, $\mathcal{D}_p(\mathcal{G})$ is a compact convex hyperbolic polygon with finitely many sides. The fact that $\mathcal{G}$ is non-Abelian leads to interesting difficulties. Among other properties, the Dirichlet regions $\mathcal{D}_p(\mathcal{G})$ and $\mathcal{D}_q(\mathcal{G})$ of two different points p and q do not always have the same combinatorics. The set of points $\mathcal{G}O$ is quite degenerate: all vertices of $VD_{\mathbb{H}}(\mathcal{G}O)$ have degree eight. See Figure 2-Left. The octagon $\mathcal{D}_O$ is in fact the Dirichlet region $\mathcal{D}_O(\mathcal{G})$ of the origin. Figure 2-Right illustrates notation that will be used throughout this paper: the vertices of $\mathcal{D}_O$ are denoted as $V_0, \ldots, V_7$ and their associated Delaunay circles are denoted as $C_0, \ldots, C_k$.

Each Dirichlet region $\mathcal{D}_p(\mathcal{G})$ is a *fundamental domain* for the action of $\mathcal{G}$ on $\mathbb{H}^2$, i.e., *(i)* $\mathcal{D}_p(\mathcal{G})$ contains at least one point of the preimage by π of any point of $\mathcal{M}$, and *(ii)* if it contains more than one point of the same preimage, then all these points lie on its boundary.

The group $\mathcal{G}$ has the following *finite presentation*:

$$\mathcal{G} = \left\langle a, b, c, d \mid abcd\bar{a}\bar{b}\bar{c}\bar{d} \right\rangle,$$

which denotes the quotient of the group $\langle a, b, c, d\rangle$ generated by a, b, c, and d, by the normal closure (i.e., the smallest normal subgroup) in $\langle a, b, c, d\rangle$ of the element $\mathcal{R}_{\mathcal{G}} = abcd\bar{a}\bar{b}\bar{c}\bar{d}$, called the *relation* of $\mathcal{G}$. Here and throughout the paper, $\overline{g}$ denotes the inverse of an element $g \in \mathcal{G}$. We use $\mathbb{1}$ to denote the identity of $\mathcal{G}$: $\forall\, g \in \mathcal{G},\ g\overline{g} = \overline{g}g = \mathbb{1}$, and $\mathcal{R}_{\mathcal{G}} = \mathbb{1}$ in $\mathcal{G}$.

The Bolza surface $\mathcal{M}$ is homeomorphic to a double torus. Its area (which is also the area of $\mathcal{D}_O$) is equal to $4\pi(\text{genus}(\mathcal{M}) - 1) = 4\pi$. The generators of $\mathcal{G}$ are naturally ordered around

[1] The octagon is rotated compared to [2]. The notation adopted now seems to be more standard in the literature, see for instance [1].

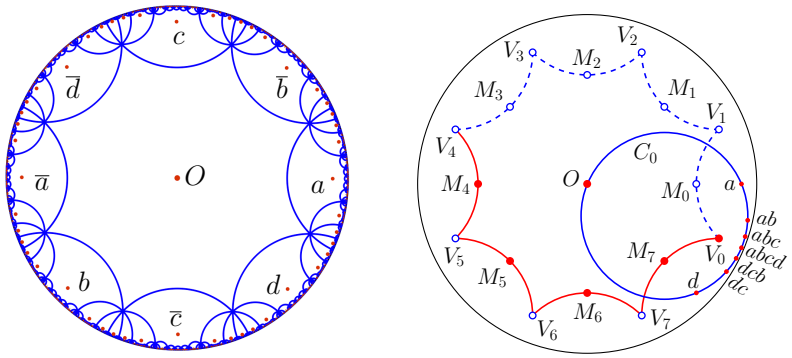


Figure 2 Left: Voronoi diagram of the infinite set of points $\mathcal{G}O$. **Right:** The original domain $\mathcal{D}$ and the images of O around the vertex V_0 of $\mathcal{D}_O$.

the octagon $\mathcal{D}_O$ as an ordered cyclical sequence $\mathcal{A} = \left[a, \overline{b}, c, \overline{d}, \overline{a}, b, \overline{c}, d\right] = [g_0, g_1, \ldots, g_7]$. The matrices of the elements $g_k,\ k = 0, 1, \ldots, 7$, are

$$g_k = \begin{bmatrix} \xi^2 & e^{ik\pi/4}\sqrt{2}\,\xi \\ e^{-ik\pi/4}\sqrt{2}\,\xi & \xi^2 \end{bmatrix}, \text{ where } \xi = \sqrt{1+\sqrt{2}}. \tag{2}$$

The translations g_k all have the same length, which is the systole of $\mathcal{M}$:

$$\mathrm{sys}(\mathcal{M}) = \ell(g_k) = 2 \cdot \mathrm{arcosh}\left(1+\sqrt{2}\right) \approx 3.05714,\ k = 0, 1, \ldots, 7.$$

3 Representation of the triangulation

As mentioned in the introduction, the use of dummy points allows us to always assume that **the set $\mathcal{P}$ of input points satisfies inequality** (1) for the Bolza surface $\mathcal{M}$.

We introduce the *original domain* $\mathcal{D} \subset \mathcal{D}_O$ for $\mathcal{M}$, which contains exactly one point of the fiber under π of each point on the surface $\mathcal{M}$. See Figure 2-Right: $\mathcal{D}$ consists of the interior of $\mathcal{D}_O$, its four "solid" sides, and one vertex of the octagon (chosen to be V_0).[2]

We can consider that all points of $\mathcal{P}$ lie in $\mathcal{D}$. Similarly, we will now define a unique representative of each face of the Delaunay triangulation $DT_{\mathcal{M}}(\mathcal{P})$ of $\mathcal{M}$ defined by $\mathcal{P}$.

3.1 Canonical representative of a face

The definition of the canonical representative of a face will rely on Theorem 2, which is reminiscent of the result proved for the flat torus by Dolbilin and Huson [9] and recalled in [5, Lemma 6.3].

We denote the hyperbolic distance between two points p and q in $\mathbb{H}^2$ as $\mathrm{dist}_{\mathbb{H}}(p, q)$ and the (hyperbolic) segment with endpoints p and q as $[p, q]$. Let us recall our abuse of notation: g denotes both a translation and the point gO. The points $M_k,\ k = 0, \ldots, 7$ visible on Figure 2-Right are defined as the midpoints of V_k and V_{k+1} (indices are meant modulo 8).

Let $\mathcal{U}_{\mathcal{D}}$ be the union of the disks bounded by the circles $C_k, k = 0, 1, \ldots, 7$, and let $\mathcal{C}_{\mathcal{D}}$ be the boundary of $\mathcal{U}_{\mathcal{D}}$. See Figure 3-Left.

[2] $\pi(V_k) = \pi(V_0),\ k = 1, \ldots, 7$: $V_5 = \overline{a}V_0, V_2 = \overline{b}V_5, V_7 = \overline{c}V_2, V_4 = \overline{d}V_2, V_1 = aV_4, V_6 = bV_1, V_3 = cV_6, V_0 = dV_3$.

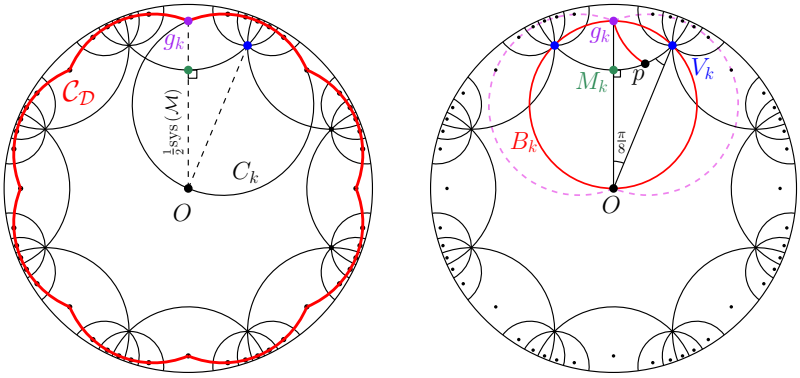


Figure 3 Left: Curve $\mathcal{C}_\mathcal{D}$ (in bold). **Right:** The distance between $\mathcal{C}_\mathcal{D}$ and $\partial\mathcal{D}_O$ is realized as the distance of the points g_k and M_k.

▶ **Lemma 1.** *The distance between $\mathcal{C}_\mathcal{D}$ and $\partial\mathcal{D}_O$ is equal to* $\mathrm{dist}_\mathbb{H}(M_k, g_k) = \frac{1}{2}\mathrm{sys}(\mathcal{M})$, $k = 0, 1, \ldots, 7$.

Proof. Using symmetries, we get (see Figure 3-Right):

$$\mathrm{dist}_\mathbb{H}(\mathcal{C}_\mathcal{D}, \partial\mathcal{D}_O) = \min_{p\in[V_k,M_k],\ q\in C_k\cap\mathcal{C}_\mathcal{D}} \mathrm{dist}_\mathbb{H}(p, q)\,.$$

The hyperbolic circle B_k centered at M_k and passing through O also contains the points V_k, g_k and V_{k+1}: indeed, by definition of the Dirichlet region of O, segment $[V_k, V_{k+1}]$ lies on the bisecting line of O and g_k, moreover $[O, g_k]$ lies on the bisecting line of V_k and V_{k+1}; in addition, $\mathrm{dist}_\mathbb{H}(M_k, V_k) = \mathrm{dist}_\mathbb{H}(O, M_k)$ since the angles of the triangle (O, V_k, M_k) at O and at V_k are both equal to $\pi/8$.

The points O and g_k are the intersections of C_k and C_{k+1}, and the segment $[O, g_k]$ is a diameter of B_k, so B_k is contained in the union of the disks bounded by C_k and C_{k+1}. The segment $[V_k, M_k]$ is a radius of B_k, so for any point $q \in C_k \cap \mathcal{C}_\mathcal{D}$ and for any $p \in [M_k, V_k]$, $\mathrm{dist}_\mathbb{H}(p, B_k) \leq \mathrm{dist}_\mathbb{H}(p, q)$. Equality is attained when $q = g_k$, so: $\mathrm{dist}_\mathbb{H}(\mathcal{C}_\mathcal{D}, \partial\mathcal{D}_O) = \min_{p\in[V_k,M_k]} \mathrm{dist}_\mathbb{H}(p, g_k)$. For every point $p \in [V_k, M_k]$, $\mathrm{dist}_\mathbb{H}(g_k, p) \geq \mathrm{dist}_\mathbb{H}(g_k, M_k)$ because the angle $g_k M_k p$ is right. The result follows. ◀

Let $\mathcal{D}_g$ denote the closure of the region of g in $VD_\mathbb{H}(\mathcal{G}O)$; $\mathcal{D}_g$ is the image of $\mathcal{D}_O$ by the translation g. The infinite set of regions $\mathcal{D}_g$, for $g \in \mathcal{G}$, form a tiling of the plane $\mathbb{H}^2$ (it was shown on Figure 2-Left.) We define $\mathcal{N}$ as the set of translations g in $\mathcal{G}$ for which $\mathcal{D}_g \cap \mathcal{D}_O \neq \emptyset$. The set $\mathcal{N}$ has 48 elements; it is naturally ordered counterclockwise around O, following the boundary of $\mathcal{D}_O$. Each element ν of $\mathcal{N}$ has an index $\mathrm{index}_\mathcal{N}(\nu)$ in this sequence. We choose *abcd* as the first element for the sequence $\mathcal{N}$, i.e., $\mathrm{index}_\mathcal{N}(abcd) = 0$. See Figure 4-Left. We define $\mathcal{D}_\mathcal{N}$ as

$$\mathcal{D}_\mathcal{N} = \bigcup_{g\in\mathcal{N}} \mathcal{D}_g.$$

▶ **Theorem 2.** *Let $\mathcal{P} \subset \mathbb{H}^2$ be a set of points such that inequality* (1) *holds for $\mathcal{M}$. If a 2-face σ of $\mathrm{DT}_\mathbb{H}(\mathcal{GP})$ has at least one of its vertices in $\mathcal{D}_O$, then σ is contained in $\mathcal{D}_\mathcal{N}$.*

From now on, 2-faces will simply be named *faces*, as done in CGAL.

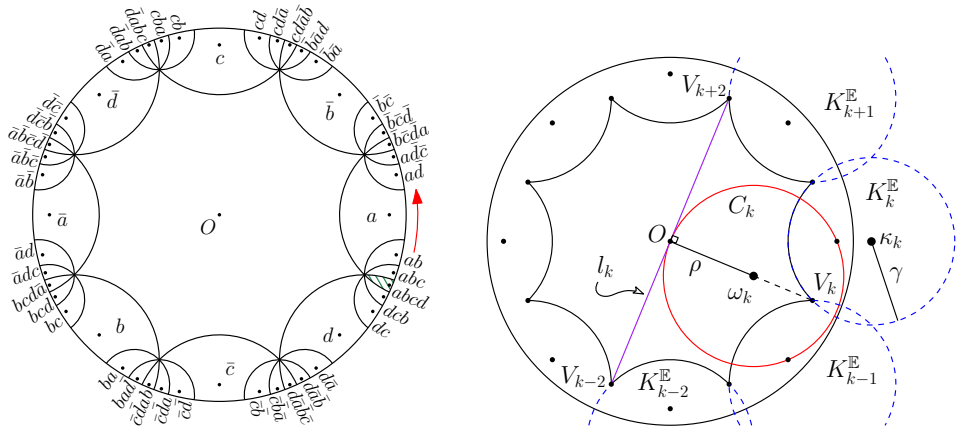


Figure 4 Left: The translations in $\mathcal{N}$. **Right:** Proof of Theorem 2.

Proof. Let σ be a face in $DT_{\mathbb{H}}(\mathcal{GP})$ with at least one vertex in $\mathcal{D}_O$. By definition of $\delta_{\mathcal{P}}$, the circumscribing disk of σ has diameter smaller than $\delta_{\mathcal{P}}$, which is smaller than $\frac{1}{2}\text{sys}(\mathcal{M})$ by inequality (1). Lemma 1 allows us to conclude that this disk is contained in $\mathcal{U}_{\mathcal{D}}$.

We will now prove that $\mathcal{U}_{\mathcal{D}}$ is contained in $\mathcal{D}_{\mathcal{N}}$, by proving that each circle C_k, for $k \in \{0, 1, \ldots, 7\}$ is contained in $\mathcal{D}_{\mathcal{N}}$. A circle C_k is centered at the Voronoi vertex V_k; it passes through the origin O and its images under the action of seven consecutive elements of $\mathcal{N}$. Rotating C_k around V_k by $\pi/4$ maps each of these eight points (and its Voronoi region) to the next one along C_k. This rotational symmetry shows that in order to prove that $C_k \subset \mathcal{D}_{\mathcal{N}}$, it is enough to prove that C_k intersects only the two sides of $\mathcal{D}_O$ that are incident to its hyperbolic center V_k.

Indices below are again taken modulo eight, e.g., we write V_{k+1} instead of $V_{k+1 \bmod 8}$. Let us first show that C_k intersects the sides $[V_{k-1}, V_k]$ and $[V_k, V_{k+1}]$ of $\mathcal{D}_O$. Consider a hyperbolic triangle (O, V_k, V_{k+1}). Its angle at O is $\pi/4$, while the angles at the vertices V_k and V_{k+1} are $\pi/8$. From the Hyperbolic law of sines,[3] we conclude that the length of $[V_k, V_{k+1}]$ is larger than the length of $[O, V_k]$. The result follows, since the segment $[O, V_k]$ is a radius of C_k.

Consider now the line segment $l_k = [V_{k-2}, V_{k+2}], k = 0, 1, \ldots, 7$, which cuts the octagon into two halves. See Figure 4-Right. Both l_k and C_k contain O; moreover l_k is perpendicular to the segment $[O, V_k]$, which is supported by a diameter of C_k. So l_k and C_k are tangent at O and l_k separates C_k from the other half of the octagon, thus C_k cannot intersect any side $[V_{k+j}, V_{k+j+1}]$ of $\mathcal{D}_O$ for $j = 2, 3, 4, 5$.

Using the fact that hyperbolic circles in the Poincaré disk model are Euclidean circles (see Section 2), we continue the proof and the computations in the Euclidean plane $\mathbb{E}^2$. The sides of $\mathcal{D}_O$ are supported by the Euclidean circles $K^{\mathbb{E}}_j = (\kappa_j, \gamma), j = 0, 1, \ldots, 7$ shown on Figure 4-Right. The centers and radii of $K^{\mathbb{E}}_j$, as well as the Euclidean centers ω_k and radii ρ of C_k are given in Table 1 and computed with Maple.

What is left to do is to show that C_k does not intersect either $K^{\mathbb{E}}_{k+1}$ or $K^{\mathbb{E}}_{k-2}$. By symmetry, it suffices to consider $K^{\mathbb{E}}_{k+1}$. The signed Euclidean distance of the circles C_k and $K^{\mathbb{E}}_{k+1}$ is

[3] If A, B, C are the sides of a hyperbolic triangle and $\vartheta_A, \vartheta_B, \vartheta_C$ the angles opposite to each side, then

$$\frac{\sin(\vartheta_A)}{\sinh(A)} = \frac{\sin(\vartheta_B)}{\sinh(B)} = \frac{\sin(\vartheta_C)}{\sinh(C)}.$$

Table 1 Expressions for the Euclidean radii and centers of $K_j^{\mathbb{E}}$ and C_k, $i, k = 0, 1, \ldots, 7$.

Quantity	Notation	Expression	Approximation
radius of $K_j^{\mathbb{E}}$	γ	$\sqrt{\frac{\sqrt{2}-1}{2}}$	0.4551
center of $K_j^{\mathbb{E}}$	κ_j	$e^{ij\pi/4}\sqrt{\frac{\sqrt{2}+1}{2}}$	–
radius of C_k	ρ	$\sqrt{(2-\sqrt{2})(\sqrt{2}-1)}$	0.4926
center of C_k	ω_k	$e^{i(2k+7)\pi/8}\sqrt{3\sqrt{2}-4}$	–

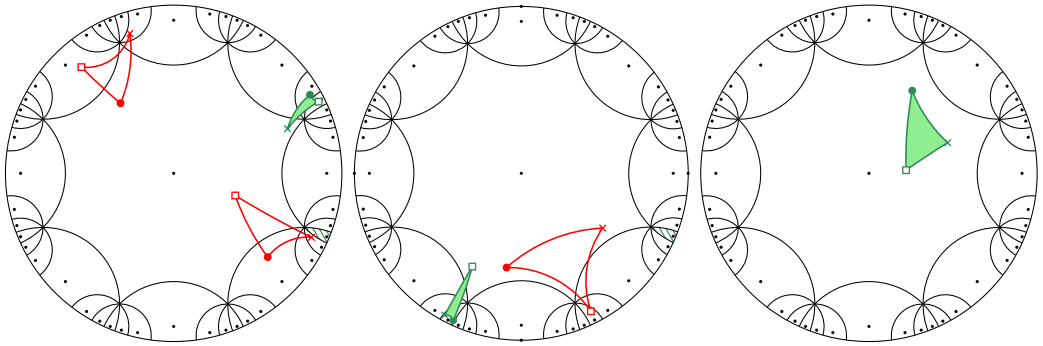


Figure 5 Examples of faces of $DT_{\mathbb{H}}(\mathcal{GP})$ with one, two and three vertices in $\mathcal{D}$, that project to the same face on $\mathcal{M}$. Their respective vertices drawn as a dot project to the same vertex on $\mathcal{M}$ (same for cross and square). The canonical representative is the shaded face.

$\text{dist}_{\mathbb{E}}\left(C_k, K_{k+1}^{\mathbb{E}}\right) = \text{dist}_{\mathbb{E}}\left(\omega_k, \kappa_{k+1}\right) - \rho - \gamma$. Maple calculations yield: $\text{dist}_{\mathbb{E}}\left(C_k, K_{k+1}^{\mathbb{E}}\right) = \frac{\sqrt{\sqrt{2}-1}}{2}\left(3\sqrt{2} - \sqrt{2}\sqrt{4-2\sqrt{2}} - 1\right)$. The last factor is positive: $\left(3\sqrt{2}-1\right)^2 = 19 - 6\sqrt{2}$, $\left(\sqrt{2}\sqrt{4-2\sqrt{2}}\right)^2 = 8 - 4\sqrt{2}$, and clearly $19 - 6\sqrt{2} > 8 - 4\sqrt{2} > 0$. This shows that C_k and $K_{k+1}^{\mathbb{E}}$ do not intersect. ◀

Let $\mathcal{P} \subset \mathcal{D}$ be a set of points satisfying inequality (1) for $\mathcal{M}$. The rest of this section is dedicated to the choice of a unique *canonical representative* σ^c in $DT_{\mathbb{H}}(\mathcal{GP})$ for each face $\boldsymbol{\sigma}$ in $DT_{\mathcal{M}}(\mathcal{P})$.

Let $\boldsymbol{\sigma}$ be a face in $DT_{\mathcal{M}}(\mathcal{P})$. By definition of $\mathcal{D}$, each vertex of $\boldsymbol{\sigma}$ has a unique preimage by π in $\mathcal{D}$, so, the set

$$\Sigma = \left\{\sigma \in \pi^{-1}(\boldsymbol{\sigma}) \;\middle|\; \sigma \text{ has at least one vertex in } \mathcal{D}\right\} \tag{3}$$

contains at most three faces. See Figure 5. When Σ contains only one face, then this face is completely included in $\mathcal{D}$, and we naturally choose it to be σ^c. Let us now assume that Σ contains two or three faces. From Theorem 2, each face $\sigma \in \Sigma$ is contained in $\mathcal{D}_{\mathcal{N}}$. So, for each vertex v of σ, there is a unique translation $\nu(v, \sigma)$ in $\mathcal{N} \cup \{\mathbb{1}\}$ such that v lies in $\nu(v, \sigma)\mathcal{D}$.

We consider all faces in $DT_{\mathbb{H}}(\mathcal{GP})$ oriented counterclockwise. For $\sigma \in \Sigma$, we denote as $v_\sigma^{\text{first\_out}}$ the first vertex of σ (in the counterclockwise order) that is not lying in $\mathcal{D}$. Using the indexing on $\mathcal{N}$ defined above, we can now choose σ^c as the face of Σ whose first vertex lying outside $\mathcal{D}$ is "closest" to the region $abcd\mathcal{D}$ in the counterclockwise order around O:

▶ **Definition 3** (Canonical representative)**.** With the notation defined above, the canonical representative of a face $\boldsymbol{\sigma}$ of $DT_{\mathcal{M}}(\mathcal{P})$ is the face $\sigma^c \in \Sigma$ such that

$$\text{index}_{\mathcal{N}}(\nu(v_{\sigma^c}^{\text{first\_out}}, \sigma^c)) = \min_{\sigma \in \Sigma} \text{index}_{\mathcal{N}}(\nu(v_{\sigma}^{\text{first\_out}}, \sigma)).$$

3.2 Data structure in CGAL

General two-dimensional triangulation data structures in CGAL store the vertices and faces of the triangulation. Each vertex stores a point and a pointer to one of its incident faces. Each face stores three pointers to its vertices v_0, v_1, and v_2, as well as three pointers to its three adjacent faces. Edges are not explicitly stored.

As mentioned above, we can assume that all input points of $\mathcal{P}$ lie in $\mathcal{D}$. We adapt the CGAL structure to store a triangulation of the Bolza surface. Each vertex v of $DT_{\mathcal{M}}(\mathcal{P})$ represents an orbit under the action of $\mathcal{G}$; it stores the point of $\pi^{-1}(v)$ that belongs to $\mathcal{D}$. Faces of $DT_{\mathcal{M}}(\mathcal{P})$ are stored through their canonical representative in $DT_{\mathbb{H}}(\mathcal{GP})$. Concretely, in addition to the pointers to vertices and neighbors, each face σ^c stores the three translations $\nu(v_i, \sigma^c) \in \mathcal{N},\ i = 0, 1, 2$ defined at the end of Section 3.1. In this way, for a given face σ^c in the structure, the corresponding canonical representative is the triangle in $\mathbb{H}^2$ whose vertices are the images by $\nu(v_i, \sigma^c)$ of the point in $\mathcal{D}$ stored in v_i for $i = 0, 1, 2$. The translations $\nu(v_i, \sigma^c)$ play a similar role as the so-called "offsets" of the CGAL Euclidean periodic triangulations.

We choose to represent translations in the faces of the triangulation data structure as words. This is detailed below.

Translations as words. We consider the cyclical sequence $\mathcal{A}$ formed by generators of $\mathcal{G}$ (see Section 2) as an alphabet, and we denote the set of words on $\mathcal{A}$ as $\mathcal{A}^*$. Each translation g in $\mathcal{G}$ can be seen as a word in $\mathcal{A}^*$, also denoted as g. For two translations $g, g' \in \mathcal{G}$, the composition (or multiplication) gg' corresponds to the concatenation of the two words g and g'. Recall that composition is not commutative. We have seen in the two previous sections that we only need to store translations in $\mathcal{N}$. Let us note here that $\mathcal{N}$ is closed under inversion, but not under composition.

The finite presentation of $\mathcal{G}$ captures the fact that a translation $g \in \mathcal{G}$ does not have a unique representation in terms of the generators (see Section 2). To obtain a unique representation of the translations that are involved in our algorithm, we slightly modify Dehn's algorithm. Dehn's algorithm solves the *word problem* (i.e., the problem of deciding whether a given word on the generators of a group is equal to the group identity) in the case of fundamental groups of closed orientable surfaces of genus at least 2 [6, 11].[4]

Let us present our implementation, tailored to our specific case.

We encode each element $g_k,\ k = 0, 1, \ldots, 7$ of $\mathcal{A}$ as its index k. By concatenation, each word of $\mathcal{A}^*$ is encoded as a sequence of integers.

Let w be a non-trivial word in $\mathcal{A}^*$. The first step of the reduction consists in freely reducing w, i.e., removing all sub-words of the form $g\overline{g}$ or $\overline{g}g$ for $g \in \mathcal{A}$. With our encoding, two elements g_i and g_j of $\mathcal{A}$ are inverses in $\mathcal{G}$ if $i = (j + 4) \bmod 8$.

The relation $\mathcal{R}_{\mathcal{G}} = abcd\overline{a}\overline{b}\overline{c}\overline{d}$ is encoded as 05274163. Let us note that any cyclical permutation of $\mathcal{R}_{\mathcal{G}}$ or of its inverse $\overline{\mathcal{R}_{\mathcal{G}}}$ is equal to $\mathbb{1}$ in $\mathcal{G}$. This can be viewed in another

[4] For interesting historical facts on this topic, see [14]. Software solving the word problem can be found for instance in [10, 13].

way by considering $\mathcal{R}_{\mathcal{G}}^{\infty}$, the infinite word formed by infinitely many concatenations of $\mathcal{R}_{\mathcal{G}}$: any subsequence $\mathcal{R}$ of $\mathcal{R}_{\mathcal{G}}^{\infty}$ or $\overline{\mathcal{R}_{\mathcal{G}}^{\infty}}$ with $|\mathcal{R}| = |\mathcal{R}_{\mathcal{G}}|$ is a relation in $\mathcal{G}$, i.e., it reduces to $\mathbb{1}$. (Here $|\cdot|$ denotes the length of a word.) The next step of the reduction consists in detecting a factorization of the (now freely-reduced) word w of the form $w = w_\lambda w_\mu w_\kappa$, where $w_\mu t$ is a relation $\mathcal{R}$ for some $t \in \mathcal{A}^*$ with $|t| < |w_\mu|$. Then $|w_\mu| > |\mathcal{R}_{\mathcal{G}}|/2 = 4$ and w_μ can be substituted in w by $\bar{t}$, which yields the word $w_\lambda \bar{t} w_\kappa$ with length shorter than $|w|$.

In our implementation, to find the sub-word w_μ, we use the fact that a sequence of letters $(g_{k_j})_{j=0,1,\ldots,n},\ g_{k_j} \in \mathcal{A}$, is a sub-word of $\mathcal{R}_{\mathcal{G}}^{\infty}$ of length n if, for every j from 0 to $n-1$, $k_{j+1} = (k_j + 5) \bmod 8$. Similarly, (g_{k_j}) is a sub-word of $\overline{\mathcal{R}_{\mathcal{G}}^{\infty}}$ of length n if for every j from 0 to $n-1$, $k_{j+1} = (k_j - 5) \bmod 8$. It holds that $|\mathcal{R}| < 2|w|$, so all words in $\mathcal{A}^*$ with length less than $2|w|$ can be listed in order to find such a word $\mathcal{R}$.

The two steps are repeated until $w = \mathbb{1}$ or until w cannot be further reduced. In the original algorithm by Dehn, words of length $|\mathcal{R}_{\mathcal{G}}|/2$ are not reduced. In order to have a unique representations of words of length four, we introduce a small modification to the algorithm: whenever we get an irreducible word w with $|w| = 4$, we check whether w is a sub-word of $\overline{\mathcal{R}_{\mathcal{G}}^{\infty}}$. If so, we return $\overline{w}$; in all other cases, we return w.

Dehn's algorithm terminates in a finite number of steps and its time complexity is polynomial in the length of the input word. Note that we reduce words that are formed by the concatenation of two or three words in $\mathcal{N}$; this will become clear in Section 4.2. Since the longest word in $\mathcal{N}$ has four letters, the longest words that we reduce have length 12.

4 Constructing the triangulation

Let us now describe the steps of our implementation of the incremental algorithm that was quickly recalled in the introduction.

4.1 Initialization

The set $\mathcal{Q}$ of 14 dummy points proposed in [2, Section 4.2] is as follows:

- the origin O;
- the eight midpoints P_k of the hyperbolic segments $[O, V_k],\ k = 0, 1, \ldots 7$;
- the midpoints $M_k,\ k = 4, 5, 6, 7$ of the closed sides of $\mathcal{D}$;
- the vertex V_0 of $\mathcal{D}$.

The canonical representatives of the 32 faces forming the Delaunay triangulation of $\mathcal{Q}$ are shown in Figure 6-Left. They can be constructed in four iterations ($i = 0, 1, 2, 3$) by using the numbering shown in Figure 6-Right (but faces are not numbered in the code).

The coordinates of the dummy points are algebraic numbers, as reported in Table 2. They have been computed using Maple. These exact coordinates would increase the algebraic degree of the predicates (studied in Section 5) in an artificial way; therefore, we introduce a set $\mathcal{Q}'$ of rational approximations of the points in $\mathcal{Q}$. See the third column of Table 2. We have verified that $DT_{\mathcal{M}}(\mathcal{Q})$ and $DT_{\mathcal{M}}(\mathcal{Q}')$ have identical combinatorial structures. We initialize the triangulation as $DT_{\mathcal{M}}(\mathcal{Q}')$.[5]

4.2 Finding faces in conflict with a new point

Let $p \in \mathcal{P} \subset \mathcal{D}$ be a new point to be inserted in the Delaunay triangulation. Consider $\boldsymbol{\sigma}$ a face in $DT_{\mathcal{M}}(\mathcal{P})$, and the set Σ defined in (3). We say that σ^c is *in conflict* with the input

[5] Note that any other set of points that satisfies condition (1) could be used to initialize the triangulation.

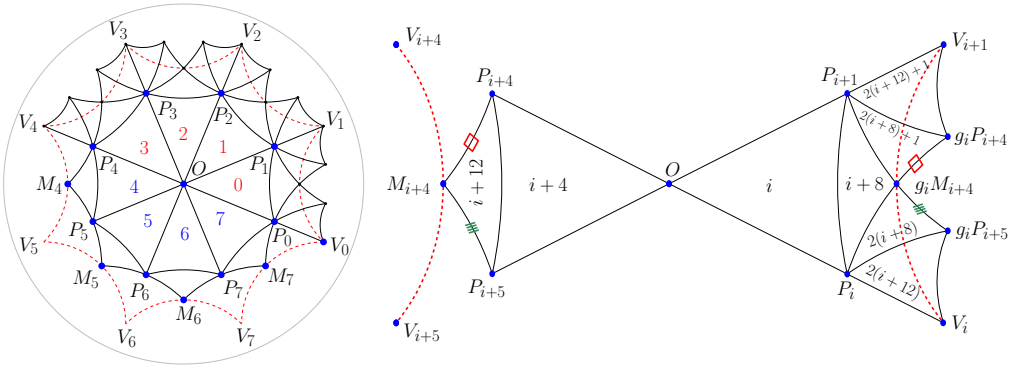


Figure 6 Left: Delaunay triangulation of $\mathcal{M}$ defined by the dummy points. **Right:** Zooming in on the faces created in iteration i. Note the identification of the marked edges.

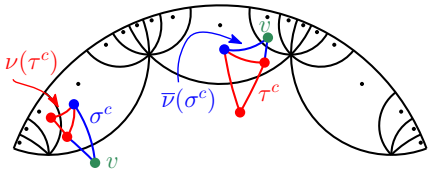


Figure 7 Translating τ^c by $\nu = \nu_{nbr}(\sigma^c, \tau^c)$ gives a face adjacent to σ^c.

point p if there exists a face $\sigma \in \Sigma$ whose circumscribing disk contains p.

Recall that, since hyperbolic circles are Euclidean circles, a Delaunay triangulation in $\mathbb{H}^2$ has exactly the same combinatorics as the Euclidean Delaunay triangulation of the same points. Consequently, the Euclidean Delaunay triangle containing p gives us a hyperbolic Delaunay face in conflict with p; the Euclidean and hyperbolic faces will both be denoted as σ_p, which should not introduce any confusion. To find this triangle, we adapt the so-called *visibility walk* [7]. This walk starts from an arbitrary face, then, for each visited face, it visits one of its neighbors, until a face containing p is found. Before specifying how the neighbor to be visited is specified in the case of the Bolza surface, we introduce the notion of *neighbor translation*.

▶ **Definition 4** (Neighbor translation). Let $\boldsymbol{\sigma}, \boldsymbol{\tau}$ be two adjacent faces in $DT_{\mathcal{M}}(\mathcal{P})$ and σ, τ two of their preimages by π in $DT_{\mathbb{H}}(\mathcal{GP})$. We define the neighbor translation $\nu_{nbr}(\sigma, \tau)$ from σ to τ as the translation of $\mathcal{G}$ such that $\nu_{nbr}(\sigma, \tau)\,\tau$ is adjacent to σ in $DT_{\mathbb{H}}(\mathcal{GP})$.

Let v be a vertex common to $\boldsymbol{\sigma}$ and $\boldsymbol{\tau}$, and let v_σ and v_τ the vertices of σ and τ that project on v by π. We can compute the neighbor translation from σ to τ as

$$\nu_{nbr}(\sigma, \tau) = \nu(v_\tau, \tau)\,\overline{\nu(v_\sigma, \sigma)}.$$

Figure 7 illustrates the neighbor translation of the canonical representatives of $\boldsymbol{\sigma}$ and $\boldsymbol{\tau}$. It can be easily seen that $\nu_{nbr}(\sigma, \tau) = \nu(v, \tau)\,\overline{\nu(v, \sigma)} = \overline{\nu(v, \sigma)}\,\overline{\nu(v, \tau)} = \overline{\nu_{nbr}(\tau, \sigma)}$.

We define the *location translation* ν_{loc} as follows: let σ_p be the Euclidean Delaunay triangle containing p. ν_{loc} is the translation that moves σ_p^c to σ_p.

The location procedure starts from a face incident to O. Then, for each visited face σ of $DT_{\mathbb{H}}(\mathcal{GP})$, we consider the Euclidean edge e defined by two of the vertices of σ. With a simple orientation test, we can check whether the Euclidean line supporting e separates p from the vertex of σ opposite to e. If this is the case, the next visited face is the neighbor τ of σ through e, and we repeat the process, until

Table 2 Exact and rational expressions for the dummy points.

Point	Expression	Rational approximation
V_0	$\left(\frac{2^{3/4}\sqrt{2+\sqrt{2}}}{4}, -\frac{2^{3/4}\sqrt{2-\sqrt{2}}}{4}\right)$	$(97/125, -26/81)$
M_4	$\left(-\sqrt{\sqrt{2}-1}, 0\right)$	$(-9/14, 0)$
M_5	$\left(-\frac{\sqrt{2}\sqrt{\sqrt{2}-1}}{2}, -\frac{\sqrt{2}\sqrt{\sqrt{2}-1}}{2}\right)$	$(-5/11, -5/11)$
M_6	$\left(0, -\sqrt{\sqrt{2}-1}\right)$	$(0, -9/14)$
M_7	$\left(\frac{\sqrt{2}\sqrt{\sqrt{2}-1}}{2}, -\frac{\sqrt{2}\sqrt{\sqrt{2}-1}}{2}\right)$	$(5/11, -5/11)$
P_0	$\left(\frac{2^{1/4}\sqrt{2+\sqrt{2}}}{2\sqrt{2}+2\sqrt{2-\sqrt{2}}}, -\frac{2^{1/4}\sqrt{2-\sqrt{2}}}{2\sqrt{2}+2\sqrt{2-\sqrt{2}}}\right)$	$(1/2, -4/19)$
P_1	$\left(\frac{2^{3/4}\left(\sqrt{2+\sqrt{2}}+\sqrt{2-\sqrt{2}}\right)}{4\sqrt{2}+4\sqrt{2-\sqrt{2}}}, \frac{2^{3/4}\left(\sqrt{2+\sqrt{2}}-\sqrt{2-\sqrt{2}}\right)}{4\sqrt{2}+4\sqrt{2-\sqrt{2}}}\right)$	$(1/2, 4/19)$
P_2	$\left(\frac{2^{1/4}\sqrt{2-\sqrt{2}}}{2\sqrt{2}+2\sqrt{2-\sqrt{2}}}, \frac{2^{1/4}\sqrt{2+\sqrt{2}}}{2\sqrt{2}+2\sqrt{2-\sqrt{2}}}\right)$	$(4/19, 1/2)$
P_3	$\left(\frac{2^{3/4}\left(\sqrt{2-\sqrt{2}}-\sqrt{2+\sqrt{2}}\right)}{4\sqrt{2}+4\sqrt{2-\sqrt{2}}}, \frac{2^{3/4}\left(\sqrt{2+\sqrt{2}}+\sqrt{2-\sqrt{2}}\right)}{4\sqrt{2}+4\sqrt{2-\sqrt{2}}}\right)$	$(-4/19, 1/2)$
P_4	$\left(-\frac{2^{1/4}\sqrt{2+\sqrt{2}}}{2\sqrt{2}+2\sqrt{2-\sqrt{2}}}, \frac{2^{1/4}\sqrt{2-\sqrt{2}}}{2\sqrt{2}+2\sqrt{2-\sqrt{2}}}\right)$	$(-1/2, 4/19)$
P_5	$\left(-\frac{2^{3/4}\left(\sqrt{2+\sqrt{2}}+\sqrt{2-\sqrt{2}}\right)}{4\sqrt{2}+4\sqrt{2-\sqrt{2}}}, \frac{2^{3/4}\left(\sqrt{2-\sqrt{2}}-\sqrt{2+\sqrt{2}}\right)}{4\sqrt{2}+4\sqrt{2-\sqrt{2}}}\right)$	$(-1/2, -4/19)$
P_6	$\left(-\frac{2^{1/4}\sqrt{2-\sqrt{2}}}{2\sqrt{2}+2\sqrt{2-\sqrt{2}}}, -\frac{2^{1/4}\sqrt{2+\sqrt{2}}}{2\sqrt{2}+2\sqrt{2-\sqrt{2}}}\right)$	$(-4/19, -1/2)$
P_7	$\left(\frac{2^{3/4}\left(\sqrt{2+\sqrt{2}}-\sqrt{2-\sqrt{2}}\right)}{4\sqrt{2}+4\sqrt{2-\sqrt{2}}}, -\frac{2^{3/4}\left(\sqrt{2-\sqrt{2}}+\sqrt{2+\sqrt{2}}\right)}{4\sqrt{2}+4\sqrt{2-\sqrt{2}}}\right)$	$(4/19, -1/2)$

- either we find the Euclidean Delaunay face σ_p containing p by visiting only faces that do not cross the border of $\mathcal{D}$; then σ_p is a (canonical) face of $DT_{\mathbb{H}}(\mathcal{GP})$ in conflict with p, and $\nu_{loc} = \mathbb{1}$.
- or, at some point, we visit a (canonical) face $\sigma_{\mathcal{D}}$ included in $\mathcal{D}$ and its (non-canonical) neighbor τ that crosses the border of $\mathcal{D}$. Then the walk continues in non-canonical faces, until we find the Euclidean triangle σ_p containing p. Then ν_{loc} is $\nu_{nbr}(\sigma_{\mathcal{D}}, \tau^c)$ and the canonical face in conflict with p is $\sigma_p^c = \overline{\nu_{loc}}\sigma_p$.

If a (Euclidean) face with edges e_1, e_2, and e_3 is entered through e_1 during the walk, and if none of e_2 and e_3 separates its opposite vertex from p, then the face contains p. So, two orientation tests are enough to conclude that a face contains p (except for the starting face).

The location translation ν_{loc} is also used when looking for all other faces in conflict with p. Starting from σ_p^c and for each face in conflict with p, we recursively examine the translated image under ν_{loc} of each neighbor (obtained with a neighbor translation) that has not yet been visited. We store the set Z^c of canonical faces in conflict with p. Note that Z^c is not necessarily a connected region.

4.3 Insertion

It remains to create the new faces and delete the faces in conflict. The translation ν_{loc} computed in the previous step will again be used. We know that p lies in $\nu_{loc}\sigma_p^c$. We first create a new vertex v_{new} and store p in it.

By construction, the union of all translated faces $\nu_{loc}\nu_{nbr}\left(\sigma_p^c, \tau^c\right)\tau^c$, $\tau^c \in Z^c$ is a topological disk Z in $\mathbb{H}^2$. We identify the sequence of edges E on the border of Z; each edge e is incident to one face in Z and one face that is not in Z. For each face τ^c in Z^c,

we temporarily store the translations $\nu_{loc}\nu_{nbr}(\sigma^c, \tau^c)\,\nu(v_i, \tau^c), i = 0, 1, 2$ directly in its three vertices (not in τ^c, since it will be deleted). Since Z is a topological disk, the result for a given vertex v is independent of the face of Z incident to v that is considered. We store $\mathbb{1}$ in vertex v_{new}.

For each edge $e \in E$, we create a new face τ_e having e as an edge and v_{new} as third vertex. The neighbor of τ_e outside Z^c is the neighbor through e of the face in Z^c incident to e. Two new faces consecutive along E are adjacent. We can now delete all faces in Z.

All that is left to do now is to compute the translations to be stored in the new faces. Let τ_{new} be a newly created face. We retrieve the translations temporarily stored in its vertices v_0, v_1, v_2 and we store them in τ_{new}. Equipped with these translations, τ_{new} is not necessarily canonical. If all translations stored in τ^c_{new} are equal to $\mathbb{1}$, then τ_{new} is contained in $\mathcal{D}$, so it is actually canonical. Otherwise, one of the vertices of τ_{new} is v_{new}; without loss of generality, $v_0 = v_{new}$, and $\nu(v_0, \tau_{new}) = \mathbb{1}$. For $i = 0, 1, 2$ we can easily compute $\Delta_i = \mathrm{index}_{\mathcal{N}}(\nu(v_{\tau_i}^{\mathrm{first\_out}}, \tau_i))$, where τ_i is the image of τ_{new} under $\overline{\nu(v_i, \tau_{new})}$: in each face, $v_{\tau_i}^{\mathrm{first\_out}}$ is the first vertex of τ_i such that $\nu(v_{\tau_i}^{\mathrm{first\_out}}, \tau_i) \neq \mathbb{1}$. Note that we do not actually compute the images of τ_{new}, we only compute translations (as words). We then find the index k for which Δ_k is minimal, and in τ_{new} we store the translations $\overline{\nu(v_k, \tau_{new})}\nu(v_i, \tau_{new})$, $i = 0, 1, 2$. The face τ_{new} has now been canonicalized. Once this is done for all new faces, temporary translations can be removed from the vertices.

5 Algebraic complexity

We follow the so-called *Exact Geometric Computation* paradigm pioneered by Chee Yap [15]. As can be seen in Section 4, the correctness of the combinatorial structure $DT_{\mathcal{M}}(\mathcal{P})$ relies on the exact evaluation of three predicates:

- *SideOfOctagon*, which checks whether an input point lies inside $\mathcal{D}$. This predicate is used as a precondition for the insertion of each point.
- *Orientation*, which checks whether an input point p in $\mathcal{D}$ lies on the right side, the left side, or on an oriented Euclidean segment. This predicate is used when looking for the Euclidean triangle containing an input point.
- *InCircle*, which checks whether an input point p in $\mathcal{D}$ lies inside, outside, or on the boundary of the disk circumscribing an oriented triangle. It is used when looking for all faces in conflict with an input point.

Let the coordinates of a point $p_i \in \mathbb{H}^2$ be denoted as x_i and y_i. The last two predicates can be expressed as signs of determinants:

$$\mathit{Orientation}(p_1, p_2, p_3) = \mathrm{sign}\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix}, \quad \mathit{InCircle}(p_1, p_2, p_3, p_4) = \mathrm{sign}\begin{vmatrix} x_1 & y_1 & x_1^2+y_1^2 & 1 \\ x_2 & y_2 & x_2^2+y_2^2 & 1 \\ x_3 & y_3 & x_3^2+y_3^2 & 1 \\ x_4 & y_4 & x_4^2+y_4^2 & 1 \end{vmatrix}. \quad (4)$$

We assume that all input points (which lie in $\mathcal{D}$) have rational coordinates (recall that this holds for the initial dummy points, see Section 4.1). So, in the above determinants, at least one point (x_i, y_i) is rational. However, the points against which the predicates are testing the new input point are vertices of some face of $DT_{\mathbb{H}}(\mathcal{GP})$ contained in $\mathcal{U}_{\mathcal{D}}$, so they are images of some input points by translations in $\mathcal{N} \cup \{\mathbb{1}\}$. Therefore, the evaluation of the two predicates (4) boils down to determining the sign, considered as an element of $\{-1, 0, 1\}$ of polynomial expressions in rational variables, whose coefficients are lying in some extension field of the rationals, as made precise below.

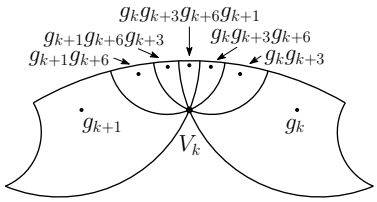


Figure 8 Elements of $\mathcal{N}$ around a vertex V_k.

Table 3 Matrices of translations around a vertex V_k, k. Recall that $\xi = \sqrt{1+\sqrt{2}}$.

$$
\begin{aligned}
g_k &= \begin{bmatrix} 1+\sqrt{2} & e^{ik\pi/4}\sqrt{2}\xi \\ e^{-ik\pi/4}\sqrt{2}\xi & 1+\sqrt{2} \end{bmatrix} \quad \text{(see (2))} \\
g_k g_{k+3} &= \begin{bmatrix} \left(1+\sqrt{2}\right)\left(1-i\sqrt{2}\right) & e^{ik\pi/4}\left(1+i\left(1+\sqrt{2}\right)\right)\xi \\ e^{-ik\pi/4}\left(1-i\left(1+\sqrt{2}\right)\right)\xi & \left(1+\sqrt{2}\right)\left(1+i\sqrt{2}\right) \end{bmatrix} \\
g_k g_{k+3} g_{k+6} &= \begin{bmatrix} -\left(1+\sqrt{2}\right)(1+2i) & e^{ik\pi/4}\left(1+\sqrt{2}\right)(-1+i)\,\xi \\ -e^{-ik\pi/4}\left(1+\sqrt{2}\right)(1+i)\,\xi & -\left(1+\sqrt{2}\right)(1-2i) \end{bmatrix} \\
g_k g_{k+3} g_{k+6} g_{k+1} &= \begin{bmatrix} -2\sqrt{2}-3 & -e^{ik\pi/4}\left(2+\sqrt{2}+i\sqrt{2}\right)\xi \\ -e^{-ik\pi/4}\left(2+\sqrt{2}-i\sqrt{2}\right)\xi & -2\sqrt{2}-3 \end{bmatrix} \\
g_{k+1} g_{k+6} g_{k+3} &= \begin{bmatrix} \left(1+\sqrt{2}\right)(-1+2i) & -e^{ik\pi/4}\left(2+\sqrt{2}\right)i\xi \\ e^{-ik\pi/4}\left(2+\sqrt{2}\right)i\xi & -\left(1+\sqrt{2}\right)(1+2i) \end{bmatrix} \\
g_{k+1} g_{k+6} &= \begin{bmatrix} \left(1+\sqrt{2}\right)+\left(2+\sqrt{2}\right)i & e^{ik\pi/4}\left(1+\sqrt{2}-i\right)\xi \\ e^{-ik\pi/4}\left(1+\sqrt{2}+i\right)\xi & \left(1+\sqrt{2}\right)-\left(2+\sqrt{2}\right)i \end{bmatrix} \\
g_{k+1} &= \begin{bmatrix} 1+\sqrt{2} & e^{ik\pi/4}(1+i)\,\xi \\ e^{-ik\pi/4}(1-i)\,\xi & 1+\sqrt{2} \end{bmatrix}
\end{aligned}
$$

The evaluation of the degree of the predicates requires to perform a case analysis on the different possible positions of the faces in $\mathcal{U}_\mathcal{D}$, i.e., on the possible translations of $\mathcal{N}$ that can be involved in each predicate. The following property shows how we can take symmetries of $\mathcal{D}$ into account to reduce the number of possible cases.

▶ **Lemma 5.** *Let $\boldsymbol{\sigma}$ be a face in $\mathrm{DT}_\mathcal{M}(\mathcal{P})$. Then, for any edge uv of its canonical representative σ^c, such that $\nu(u,\sigma^c) \neq \mathbb{1}$ and $\nu(v,\sigma^c) \neq \mathbb{1}$,*

$$\left|\mathrm{index}_\mathcal{N}(\nu(u,\sigma^c)) - \mathrm{index}_\mathcal{N}(\nu(v,\sigma^c))\right| \leq 7.$$

Proof. We can assume that $\sigma^c \not\subset \mathcal{D}$, otherwise all its three translations are equal to $\mathbb{1}$. Reusing the proof of of Lemma 1 and the notation therein, we see that σ^c is either contained in the disk bounded by B_k, or in the disk bounded by C_k, for some $k \in \{0,\ldots,7\}$. So σ^c can only intersect $\mathcal{D}$ and the seven octagons around some V_k. The result follows. ◀

Figure 8 shows the possible translations of $\mathcal{N}$ involved in a given canonical representative, for some $k \in \{0,\ldots,7\}$. Their matrices are given in Table 3.

For k even, the sine and cosine of $\pm k\pi/4$ have values in $\{-1,0,1\}$, while for k odd they are both equal to $\pm\sqrt{2}/2$. Therefore, up to sign, the above matrices are divided into two "classes". Due to the symmetries of $\mathcal{D}_O$, we actually only need to examine one case in each class, therefore we can focus on the two cases $k=0$ and $k=1$.

▶ **Proposition 1.** *All predicates can be evaluated by determining the sign of rational polynomial expressions of total degree at most 72 in the coordinates of input points.*

Proof. We examine the complexity of the *Orientation* predicate and refer the reader to the appendix for other ones. As mentioned above, at least one point is inside $\mathcal{D}$. Without loss of generality, we assume that $p_3 \in \mathcal{D}$. Let us consider the possible cases for the other two points.

- All three points are inside $\mathcal{D}$. In this case, all the arguments of the predicate are rational, so from (4) we get a polynomial with rational coefficients of total degree 2 in the coordinates of the input points.
- Point p_2 is also in $\mathcal{D}$, and p_1 is outside $\mathcal{D}$. In this case, p_1 can be the image of an input point by 14 possible different translations in $\mathcal{N}$ (seven around V_0 and seven around V_1).
- Only p_3 is inside $\mathcal{D}$. In this case, both p_1 and p_2 can be images of input points under the translations around V_0 and V_1. Of course, we avoid redundancies: if we examine the case $\mathit{Orientation}(g_i p'_1, g_j p'_2, p_3)$, $p'_i, p'_j \in \mathcal{D}$, we do not examine the case $\mathit{Orientation}(g_j p'_1, g_i p'_2, p_3)$ since it would have the same degree. This amounts to 56 cases in total – 28 cases around V_0 and another 28 around V_1.

We have found with Maple that in all cases, the expressions produced by (4) have denominators that are strictly positive and numerators that can be brought into the form

$$\left(A\sqrt{2}+B\right)\xi + C\sqrt{2} + D, \qquad \xi = \sqrt{1+\sqrt{2}}, \tag{5}$$

where A, B, C, D are rational polynomial expressions in the input coordinates. Moreover, the maximum total degree of A, B, C, D is 5. By squaring twice (to eliminate square roots coming from ξ), we get a rational polynomial of degree 20 in rational variables. ◀

The degree itself, as well as the high number of cases (in spite of the reduction) that would need to be considered, show that giving a complete implementation for all polynomial expressions involved in the predicates is hardly feasible. Therefore, we use the type `CORE::Expr` [16] included in the CGAL distribution to compute the coordinates of translated points and directly evaluate the signs of determinants (4). This number type guarantees that predicates are exact.

Our implementation handles degeneracies using symbolic perturbations [8]. Note that there are no degeneracies in the initial triangulation $DT_{\mathcal{M}}(\mathcal{Q}')$.

6 Experimental results

Experiments are run on a MacBook Pro with CPU Intel Core i5 @ 2.9 GHz, 16 GB RAM @ 1867 MHz running the `master` version of CGAL from GitHub, compiled in release mode with clang-700.1.81. We insert random points uniformly distributed with respect to the hyperbolic metric in $\mathcal{D}$. As mentioned in introduction, dummy points are removed after the insertion of new points. Averaged over 10 executions, the running time is 34 seconds for one million points. This is slower than the computation of 2D Euclidean Delaunay triangulations with CGAL, which takes around 12 seconds on average for the same sets of points, using `CORE::Expr` as number type (and about one second with `double` number type). This is due in particular to the much higher arithmetic demand in our case (Proposition 1), as confirmed by preliminary profiling, which shows that almost two thirds of the running time is spent in computations of predicates. For the insertion of one million points, only 0.76% calls to predicates involve images of rational points under translations in $\mathcal{N}$, but these calls account for 36% of the total time spent in predicates.

We have also executed tests in which we insert random points in the triangulation and progressively remove dummy points whenever doing so does not violate condition (1). Over

100 executions, all dummy points are removed with the insertion of at least 17 and at most 72 random points.

Due to lack of space, pictures showing some Delaunay triangulations are shown on the web page.

References

1. N. L. Balazs and A. Voros. Chaos on the pseudosphere. *Physics Reports*, 143(3):109–240, 1986. doi:10.1016/0370-1573(86)90159-6.
2. 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 20:1–20:15, 2016. doi:10.4230/LIPIcs.SoCG.2016.20.
3. A. Bowyer. Computing Dirichlet tessellations. *The Computer Journal*, 24(2):162–166, 1981. doi:10.1093/comjnl/24.2.162.
4. Manuel Caroli and Monique Teillaud. 3D periodic triangulations. In *CGAL User and Reference Manual*. CGAL Editorial Board, 3.5 (and further) edition, 2009-. URL: http://doc.cgal.org/latest/Manual/packages.html#PkgPeriodic3Triangulation3Summary.
5. Manuel Caroli and Monique Teillaud. Delaunay triangulations of closed Euclidean d-orbifolds. *Discrete & Computational Geometry*, 55(4):827–853, 2016. doi:10.1007/s00454-016-9782-6.
6. M. Dehn. Transformation der Kurven auf zweiseitigen Flächen. *Mathematische Annalen*, 72(3):413–421, 1912. doi:10.1007/BF01456725.
7. Olivier Devillers, Sylvain Pion, and Monique Teillaud. Walking in a triangulation. *International Journal of Foundations of Computer Science*, 13:181–199, 2002. URL: https://hal.inria.fr/inria-00102194.
8. Olivier Devillers and Monique Teillaud. Perturbations for Delaunay and weighted Delaunay 3D Triangulations. *Computational Geometry: Theory and Applications*, 44:160–168, 2011. doi:10.1016/j.comgeo.2010.09.010.
9. Nikolai P. Dolbilin and Daniel H. Huson. Periodic Delone tilings. *Periodica Mathematica Hungarica*, 34:1-2:57–64, 1997.
10. The GAP Group. *GAP – Groups, Algorithms, and Programming, Version 4.8.6*, 2016. URL: http://www.gap-system.org.
11. Martin Greendlinger. Dehn's algorithm for the word problem. *Communications on Pure and Applied Mathematics*, 13(1):67–83, 1960. doi:10.1002/cpa.3160130108.
12. Nico Kruithof. 2D periodic triangulations. In *CGAL User and Reference Manual*. CGAL Editorial Board, 4.3 (and further) edition, 2013-. URL: http://doc.cgal.org/latest/Manual/packages.html#PkgPeriodic2Triangulation2Summary.
13. The Magma Development Team. *Magma Computational Algebra System*. URL: http://magma.maths.usyd.edu.au/magma/.
14. John Joseph O'Connor and Edmund Frederick Robertson. The MacTutor History of Mathematics archive, 2003. URL: http://www-history.mcs.st-andrews.ac.uk/HistTopics/Word_problems.html.
15. C. K. Yap and T. Dubé. The exact computation paradigm. In D.-Z. Du and F. K. Hwang, editors, *Computing in Euclidean Geometry*, volume 4 of *Lecture Notes Series on Computing*, pages 452–492. World Scientific, Singapore, 2nd edition, 1995. doi:10.1142/9789812831699_0011.
16. Chee Yap *et al.* The CORE Library Project. URL: http://cs.nyu.edu/exact/core_pages/intro.html.

Lower Bounds for Differential Privacy from Gaussian Width*

Assimakis Kattis[1] and Aleksandar Nikolov[2]

1 **Department of Computer Science, University of Toronto, Toronto, ON, Canada**
kattis@cs.toronto.edu
2 **Department of Computer Science, University of Toronto, Toronto, ON, Canada**
anikolov@cs.toronto.edu

Abstract

We study the optimal sample complexity of a given workload of linear queries under the constraints of differential privacy. The sample complexity of a query answering mechanism under error parameter α is the smallest n such that the mechanism answers the workload with error at most α on any database of size n. Following a line of research started by Hardt and Talwar [STOC 2010], we analyze sample complexity using the tools of asymptotic convex geometry. We study the sensitivity polytope, a natural convex body associated with a query workload that quantifies how query answers can change between neighboring databases. This is the information that, roughly speaking, is protected by a differentially private algorithm, and, for this reason, we expect that a "bigger" sensitivity polytope implies larger sample complexity. Our results identify the *mean Gaussian width* as an appropriate measure of the size of the polytope, and show sample complexity lower bounds in terms of this quantity. Our lower bounds completely characterize the workloads for which the Gaussian noise mechanism is optimal up to constants as those having asymptotically maximal Gaussian width.

Our techniques also yield an alternative proof of Pisier's Volume Number Theorem which also suggests an approach to improving the parameters of the theorem.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases differential privacy, convex geometry, lower bounds, sample complexity

Digital Object Identifier 10.4230/LIPIcs.SoCG.2017.45

1 Introduction

The main goal of private data analysis is to estimate aggregate statistics while preserving individual privacy guarantees. Intuitively, we expect that, for statistics that do not depend too strongly on any particular individual, a sufficiently large database allows computing an estimate that is both accurate and private. A natural question then is to characterize the *sample complexity* under privacy constraints: the smallest database size for which we can privately estimate the answers to a given collection of queries within some allowable error tolerance. Moreover, it is desirable to identify algorithms that are simple, efficient, and have close to the best possible sample complexity. In this work, we study these questions for collections of *linear queries* under the constraints of *approximate differential privacy*.

We model a *database* $\mathcal{D}$ of *size* n as a multiset of n elements (counted with repetition) from an arbitrary finite universe $\mathcal{U}$. Each element of the database corresponds to the data of a single individual. To define a privacy-preserving computation on $\mathcal{D}$, we use the strong

* A full version of the paper is available at https://arxiv.org/abs/1612.02914.

notion of *differential privacy*. Informally, an algorithm is differentially private if it has almost identical behavior on any two databases $\mathcal{D}$ and $\mathcal{D}'$ that differ in the data of a single individual. To capture this concept formally, let us define two databases $\mathcal{D}$ and $\mathcal{D}'$ to be *neighboring* if we can get $\mathcal{D}'$ by replacing a single element of $\mathcal{D}$ with another element from $\mathcal{U}$. Then differential privacy is defined as follows:

▶ **Definition 1** ([5])**.** A randomized algorithm $\mathcal{A}$ that takes as input a database and outputs a random element from the set Y satisfies (ε, δ)-*differential privacy* if for all neighboring databases $\mathcal{D}, \mathcal{D}^{'}$ and all measurable $S \subseteq Y$ we have that:

$$\mathbb{P}[\mathcal{A}(\mathcal{D}) \in S] \leq e^{\varepsilon}\mathbb{P}[\mathcal{A}(\mathcal{D}^{'}) \in S] + \delta,$$

where probabilities are taken with respect to the randomness of $\mathcal{A}$.

One of the most basic primitives in private data analysis, and data analysis in general, are *counting queries* and, slightly more generally, *linear queries*. While interesting and natural in themselves, they are also quite powerful: any statistical query (SQ) learning algorithm can be implemented using noisy counting queries as a black box [10]. In our setting, we specify a linear query by a function $q\colon \mathcal{U} \to [0,1]$ (given by a table of its values for each element of $\mathcal{U}$). Slightly abusing notation, we define the value of the query as $q(\mathcal{D}) = \frac{1}{n}\sum_{e\in\mathcal{D}} q(e)$, where the elements of $\mathcal{D}$ are counted with multiplicity, and n is the size of $\mathcal{D}$. For example, when $q\colon \mathcal{U} \to \{0,1\}$, we can think of q as a property defined on $\mathcal{U}$ and $q(\mathcal{D})$ as the fraction of elements of $\mathcal{D}$ that satisfy the property: this is a *counting query*. We call a set $\mathcal{Q}$ of linear queries a *workload* and an algorithm that answers a query workload a *mechanism*. We denote by $\mathcal{Q}(\mathcal{D}) = (q(\mathcal{D}))_{q\in\mathcal{Q}}$ the vector of answers to the queries in $\mathcal{Q}$. Throughout the paper, we will use the letter m for the size of a workload $\mathcal{Q}$.

Starting from the work of Dinur and Nissim [4], it is known that we cannot hope to answer too many linear queries too accurately while preserving even a very weak notion of privacy. For this reason, we must allow our private mechanisms to make some error. We focus on *average error* (in an L_2 sense). We define the average error of an algorithm $\mathcal{A}$ on a query workload $\mathcal{Q}$ and databases of size at most n as:

$$\mathrm{err}(\mathcal{Q},\mathcal{A},n) = \max_{\mathcal{D}}\left(\mathbb{E}\sum_{q\in\mathcal{Q}}\frac{(\mathcal{A}(\mathcal{D})_q - q(D))^2}{|\mathcal{Q}|}\right)^{1/2} = \max_{\mathcal{D}}\left(\mathbb{E}\frac{1}{m}\|\mathcal{A}(\mathcal{D}) - \mathcal{Q}(\mathcal{D})\|_2^2\right)^{1/2},$$

where the maximum is over all databases $\mathcal{D}$ of size at most n, $\mathcal{A}(\mathcal{D})_q$ is the answer to query q given by the algorithm $\mathcal{A}$ on input $\mathcal{D}$, and expectations are taken with respect to the random choices of $\mathcal{A}$. This is a natural notion of error that also works particularly well with the geometric tools that we use.

In this work we study *sample complexity*: the smallest database size which allows us to answer a given query workload with error at most α. The sample complexity of an algorithm $\mathcal{A}$ with error α is defined as:

$$\mathrm{sc}(\mathcal{Q},\mathcal{A},\alpha) = \min\{n : \mathrm{err}(\mathcal{Q},\mathcal{A},n) \leq \alpha\}.$$

The sample complexity of answering the linear queries $\mathcal{Q}$ with error α under (ε,δ)-differential privacy is defined by:

$$\mathrm{sc}_{\varepsilon,\delta}(\mathcal{Q},\alpha) = \inf\{\mathrm{sc}(\mathcal{Q},\mathcal{A},\alpha) : \mathcal{A} \text{ is } (\varepsilon,\delta)\text{-differentially private}\}.$$

The two main questions we are interested in are:

1. Can we characterize $\mathrm{sc}_{\varepsilon,\delta}(\mathcal{Q},\alpha)$ in terms of a natural property of the workload $\mathcal{Q}$?
2. Can we identify conditions under which simple and efficient (ε,δ)-differentially private mechanisms have nearly optimal sample complexity?

We make progress on both questions. We identify a geometrically defined property of the workload that gives lower bounds on the sample complexity. The lower bounds also *characterize* when one of the simplest differentially private mechanisms, the Gaussian noise mechanism, has nearly optimal sample complexity in the regime of constant α.

Before we can state our results, we need to define a natural geometric object associated with a workload of linear queries. This object has been important in applying geometric techniques to differential privacy [9, 2, 16, 15].

▶ **Definition 2.** The *sensitivity polytope* K of a workload $\mathcal{Q}$ of m linear queries is equal to $K = \mathrm{conv}\{\pm\mathcal{Q}(\mathcal{D}) : \mathcal{D} \text{ is a database of size } 1\}$.

From the above definition, we see that K is a symmetric (i.e. $K = -K$) convex polytope in $\mathbb{R}^m$. The importance of K lies in the fact that it captures how query answers can change between neighboring databases: for any two neighboring databases $\mathcal{D}$ and $\mathcal{D}'$ of size n and n' respectively, $n\mathcal{Q}(\mathcal{D}) - n'\mathcal{Q}(\mathcal{D}') \in K$. This is exactly the information that a differentially private algorithm is supposed to hide. Intuitively, we expect that the larger K is, the larger $\mathrm{sc}_{\varepsilon,\delta}(K,\alpha)$ should be.

We give evidence for the above intuition, and propose the width of K in a random direction as a measure of its "size". Let h_K be the support function of K: $h_K(y) = \max_{x\in K}\langle x, y\rangle$. For a unit vector y, $h_K(y) + h_K(-y)$ is the width of K in the direction of y; for arbitrary y, $h_K(ty)$ scales linearly with t (and is, in fact, a norm). We define the ℓ^*-norm of K, also known as its *Gaussian mean width*, as $\ell^*(K) = \mathbb{E}[h_K(g)]$, where g is a standard Gaussian random vector in $\mathbb{R}^m$. The Gaussian mean width is closely related to the mean width $w(K) = \mathbb{E}\frac{h_K(y)+h_K(-y)}{2}$, where y is distributed according to the rotation invariant probability measure on the unit sphere. It's easy to show that $\ell^*(K) = \mathbb{E}\|g\|_2 w(K)$, where $g \sim N(0, I)$, and $\mathbb{E}\|g\|_2 = \Theta(\sqrt{m})$. The Gaussian mean width of the Euclidean unit ball B_2^m is $\Theta(\sqrt{m})$, and, since $h_K(y) \le h_{B_2^m}(y)$ for any $y \in \mathbb{R}^m$ and any $K \subseteq B_2^m$, we have $\ell^*(K) = O(\sqrt{m})$ for any such K.

The following theorem captures our main result.

▶ **Theorem 3.** *Let $\mathcal{Q}$ be a workload of m linear queries, and let K be its sensitivity polytope. The following holds for all $\varepsilon = O(1)$, $2^{-\Omega(n)} \le \delta \le 1/n^{1+\Omega(1)}$, and any $\alpha \le \frac{\ell^*(K)}{Cm(\log 2m)^2}$, where C is an absolute constant, and $\sigma(\varepsilon,\delta) = (0.5\sqrt{\varepsilon} + \sqrt{2\log(1/\delta)})/\varepsilon$:*

$$\mathrm{sc}_{\varepsilon,\delta}(\mathcal{Q},\alpha) = O\left(\min\left\{\frac{\sigma(\varepsilon,\delta)\ell^*(K)}{\sqrt{m}\alpha^2}, \frac{\sigma(\varepsilon,\delta)\sqrt{m}}{\alpha}\right\}\right);$$

$$\mathrm{sc}_{\varepsilon,\delta}(\mathcal{Q},\alpha) = \Omega\left(\frac{\sigma(\varepsilon,\delta)\ell^*(K)^2}{m^{3/2}(\log 2m)^4\alpha}\right).$$

The upper bound on the sample complexity is achieved by a mechanism running in time polynomial in m, n, and $|\mathcal{U}|$. Moreover, if $\ell^*(K) = \Omega(m)$, then $\mathrm{sc}_{\varepsilon,\delta}(\mathcal{Q},\alpha) = \Theta\left(\frac{\sigma(\varepsilon,\delta)\sqrt{m}}{\alpha}\right)$ for any $\alpha \le 1/C$, where C is an absolute constant.

The sample complexity upper bounds in the theorem above are known from prior work: one is given by the projection mechanism from [16], with the sample complexity upper bound in terms of $\ell^*(K)$ shown in [6]; the other upper bound is given by the Gaussian noise mechanism [4, 7, 5]. The main new contribution in this work are the lower bounds on the

sample complexity. The gap between upper and lower bounds is small when $\ell^*(K)$ is close to its maximal value of m. Indeed, when $\ell^*(K) = \Theta(m)$, our results imply that the Gaussian noise mechanism has optimal sample complexity up to constants. This is, to the best of our knowledge, the first example of a general geometric condition under which a simple and efficient mechanism has optimal sample complexity up to *constant* factors. Moreover, in the constant error regime this condition is also *necessary* for the Gaussian mechanism to be optimal up to constants: when $\ell^*(K) = o(m)$ and $\alpha = \Omega(1)$, the projection mechanism has asymptotically smaller sample complexity than the Gaussian mechanism.

We can prove somewhat stronger results for another natural problem in private data analysis, which we call the *mean point problem*. In this problem, we are given a closed convex set $K \subset \mathbb{R}^m$, and we are asked to approximate the mean $\overline{\mathcal{D}}$ of the database $\mathcal{D}$, where $\mathcal{D} = \{x_1, \ldots, x_n\}$ is a multiset of points in K and $\overline{\mathcal{D}} = \frac{1}{n}\sum_{i=1}^n x_i$. This problem, which will be the focus for most of this paper, has a more geometric flavor, and is closely related to the query release problem for linear queries. In fact, Theorem 3 will essentially follow from a reduction from the results below for the mean point problem.

With respect to the mean point problem, we define the error of an algorithm $\mathcal{A}$ as:

$$\mathrm{err}(K, \mathcal{A}, n) = \sup_{\mathcal{D}} (\mathbb{E}\|\mathcal{A}(\mathcal{D}) - \overline{\mathcal{D}}\|_2^2)^{1/2},$$

where the supremum is over databases $\mathcal{D}$ consisting of at most n points from K, and the expectation is over the randomness of the algorithm. The sample complexity of an algorithm $\mathcal{A}$ with error α is defined as:

$$\mathrm{sc}(K, \mathcal{A}, \alpha) = \min\{n : \mathrm{err}(K, \mathcal{A}, n) \le \alpha\}.$$

The sample complexity of solving the mean point problem with error α over K is defined by:

$$\mathrm{sc}_{\varepsilon,\delta}(K, \alpha) = \min\{\mathrm{sc}(K, \mathcal{A}, \alpha) : \mathcal{A} \text{ is } (\varepsilon, \delta)\text{-differentially private}\}.$$

Our main result for the mean point problem is given in the following theorem:

▶ **Theorem 4.** *Let K be a symmetric convex body contained in the unit Euclidean ball B_2^m in $\mathbb{R}^m$. The following holds for all $\varepsilon = O(1)$, $2^{-\Omega(n)} \le \delta \le 1/n^{1+\Omega(1)}$, and any $\alpha \le \frac{\ell^*(K)}{C\sqrt{m}(\log 2m)^2}$, where C is an absolute constant, and $\sigma(\varepsilon, \delta) = (0.5\sqrt{\varepsilon} + \sqrt{2\log(1/\delta)})/\varepsilon$:*

$$\mathrm{sc}_{\varepsilon,\delta}(K, \alpha) = O\left(\min\left\{\frac{\sigma(\varepsilon,\delta)\ell^*(K)}{\alpha^2}, \frac{\sigma(\varepsilon,\delta)\sqrt{m}}{\alpha}\right\}\right);$$

$$\mathrm{sc}_{\varepsilon,\delta}(K, \alpha) = \Omega\left(\frac{\sigma(\varepsilon,\delta)\ell^*(K)}{(\log 2m)^2\alpha}\right).$$

The upper bound on the sample complexity is achieved by a mechanism running in time polynomial in m, n, and $|\mathcal{U}|$. Moreover, when $\ell^*(K) = \Omega(\sqrt{m})$, then $\mathrm{sc}_{\varepsilon,\delta}(K, \alpha) = \Theta\left(\frac{\sigma(\varepsilon,\delta)\sqrt{m}}{\alpha}\right)$ for any $\alpha \le 1/C$, where C is an absolute constant.

The upper bounds again follow from prior work, and in fact are also given by the projection mechanism and the Gaussian noise mechanism, which can be defined for the mean point problem as well. Notice that the gap between the upper and the lower bound is on the order $\frac{(\log 2m)^2}{\alpha}$. If the lower bound was valid for all values of the error parameter α less than a fixed constant, rather than for $\alpha \le \frac{\ell^*(K)}{C\sqrt{m}(\log 2m)^2}$, Theorem 4 would nearly characterize the optimal sample complexity for the mean point problem for all constant α. Unfortunately, the restriction on α is, in general, necessary (up to the logarithmic terms) for lower bounds

on the sample complexity in terms of $\ell^*(K)$. For example, we can take $K = \gamma B_2^m$, i.e. a Euclidean ball in $\mathbb{R}^m$ with radius γ. Then, $\ell^*(K) = \Theta(\gamma\sqrt{m})$, but the sample complexity is 0 when $\alpha > \gamma$, since the trivial algorithm which ignores the database and outputs 0 achieves error γ. Thus, a more sensitive measure of the size of K is necessary to prove optimal lower bounds. We do, nevertheless, trust that the techniques introduced in this paper bring us closer to this goal.

We conclude this section with a high-level overview of our techniques. Our starting point is a recent tight lower bound on the sample complexity of a special class of linear queries: the 1-way marginal queries. These queries achieve the worst case sample complexity for a family of m linear queries: $\Omega(\sqrt{m}/\alpha)$ [3, 22]. The sensitivity polytope of the 1-way marginals is the cube $[-1, 1]^m$, and it can be shown that the lower bound on the sample complexity of 1-way marginals implies an analogous lower bound on the sample complexity of the mean point problem with $K = Q^m = [-1/\sqrt{m}, 1/\sqrt{m}]^m$. For the mean point problem, it is easy to see that when $K' \subseteq K$, the sample complexity for K' is no larger than the sample complexity for K. Moreover, we can show that the sample complexity of any projection of K is no bigger than the sample complexity of K itself. So, our strategy then is to find a large scaled copy of $Q^{m'}$, $m' \leq m$, inside a projection of K onto a large dimensional subspace whenever $\ell^*(K)$ is large. We solve this geometric problem using deep results from asymptotic convex geometry, namely the Dvoretzky criterion, the low M^* estimate, and the MM^* estimate.

We note that in [3], the authors mention a similar idea of showing a lower bound on the sample complexity of an arbitrary family of *counting* queries $\mathcal{Q}$ by embedding the 1-way marginals into $\mathcal{Q}$. For average error, their approach gives a lower bound of $\Omega(\frac{\sqrt{d}}{\alpha})$ (ignoring the dependence on ε and δ) for any $\alpha \leq \frac{1}{10}\sqrt{\frac{d}{m}}$, where d is the the VC-dimension of the set system $\{S_e : e \in \mathcal{U}\}$ and $S_e = \{q \in \mathcal{Q} : q(e) = 1\}$. This lower bound is at least as strong as our lower bounds, since $\frac{\ell^*(K)}{\sqrt{m}} \leq C\sqrt{d}$ for a sufficiently large constant C. (This inequality is a well-known consequence of Dudley's chaining inequality and estimates on entropy numbers in terms of VC-dimension, e.g. Theorem 14.12. in [11].) Our results, however, hold for arbitrary linear queries, and not just counting queries. Moreover, our lower bound is in terms of the efficiently computable quantity $\ell^*(K)$, while there is evidence that computing VC-dimension is hard [18]. Thus, our lower bound can be seen as an efficiently computable relaxation of the VC-dimension lower bound, and also as a generalization of it to linear queries.

Our techniques also yield an alternative proof of the volume number theorem of Milman and Pisier [14]. Besides avoiding the quotient of subspace theorem, our proof yields an improvement in the volume number theorem, conditional on the well-known conjecture (see e.g. Chapter 6 in [1]) that any symmetric convex body K has a position (affine image) TK for which $\ell^*(TK)\ell(TK) = O(m\sqrt{\log 2m})$, where $\ell(K)$ is the expected K-norm of a standard Gaussian. More details about this connection are given in Section 6.

1.1 Prior Work

Most closely related to our work are the results of Nikolov, Talwar, and Zhang [16], who gave a private mechanism (also based on the projection mechanism, but more involved) which has nearly optimal sample complexity (with respect to average error), up to factors polynomial in $\log m$ and $\log |\mathcal{U}|$. This result was subsequently improved by Nikolov [15], who showed that the $\log m$ factors can be replaced by $\log n$. While these results are nearly optimal for subconstant values of the error parameter α, i.e. the optimality guarantees do not depend on $1/\alpha$, factors polynomial in $\log |\mathcal{U}|$ can be prohibitively large. Indeed, in many natural settings, such as that of marginal queries, $|\mathcal{U}|$ is exponential in the number of queries m, so the competitiveness ratio can be polynomial in m.

The line of work that applies techniques from convex geometry to differential privacy started with the beautiful paper of Hardt and Talwar [9], whose results were subsequently strengthened in [2]. These papers focused on the "large database" regime (or, in our language, the setting of subconstant error), and pure differential privacy ($\delta = 0$).

2 Preliminaries

We begin with the introduction of some notation. Throughout the paper we use C, C_1, etc., for absolute constants, whose value may change from line to line. We use $\langle \cdot, \cdot \rangle$ for the standard inner product on $\mathbb{R}^m$, $\|\cdot\|_2$ for the standard Euclidean norm, and $\|\cdot\|_1$ for the ℓ_1 norm in $\mathbb{R}^m$. We define B_1^m and B_2^m to be the ℓ_1 and ℓ_2 unit balls in $\mathbb{R}^m$ respectively, while $Q^m = [-\frac{1}{\sqrt{m}}, \frac{1}{\sqrt{m}}]^m \subseteq \mathbb{R}^m$ will refer to the m-dimensional hypercube, normalized to be contained in the unit Euclidean ball. We use I_m for the identity operator on $\mathbb{R}^m$, as well as for the $m \times m$ identity matrix. For a given subspace E, we define $\Pi_E \colon \mathbb{R}^m \to \mathbb{R}^m$ as the orthogonal projection operator onto E. Moreover, when $T \colon E \to F$ is a linear operator between the subspaces $E, F \subseteq \mathbb{R}^m$, we define $\|T\| = \max\{\|Tx\|_2 : \|x\|_2 = 1\}$ as its operator norm, which is also equal to its largest singular value $\sigma_1(T)$. For the diameter of a set K we use the nonstandard, but convenient, definition $\operatorname{diam} K = \max\{\|x\|_2 : x \in K\}$. For sets symmetric around 0, this is equivalent to the standard definition, but scaled up by a factor of 2. We use $N(\mu, \Sigma)$ to refer to the Gaussian distribution with mean μ and covariance Σ, and we use the notation $x \sim N(\mu, \Sigma)$ to denote that x is distributed as a Gaussian random variable with mean μ and covariance Σ. For an $m \times m$ symmetric matrix (or equivalently a self-adjoint operator from ℓ_2^m to ℓ_2^m) A we use $A \succeq 0$ to denote that A is positive semidefinite, i.e. $\langle x, Ax \rangle \geq 0$ for any $x \in \mathbb{R}^m$. For positive semidefinite matrices/operators A, B, we use the notation $A \preceq B$ to denote $B - A \succeq 0$.

2.1 Convex Geometry

In this section, we outline the main geometric tools we use in later sections. For a more detailed treatment, we refer to the lecture notes by Vershynin [23] and the books by Pisier [21] and Artstein-Avidan, Giannopoulos, and Milman [1].

Throughout, we define a *convex body* K as a compact subset of $\mathbb{R}^m$ with non-empty interior. A convex body K is *(centrally) symmetric* if and only if $K = -K$. We define the *polar body* K° of K as: $K^\circ = \{y : \langle x, y \rangle \leq 1 \ \forall x \in K\}$. The following basic facts are easy to verify and very useful.

▶ **Fact 5.** *For convex bodies $K, L \subseteq \mathbb{R}^m$, $K \subseteq L \Leftrightarrow L^\circ \subseteq K^\circ$.*

▶ **Fact 6** (Section/Projection Duality)**.** *For a convex body $K \subseteq \mathbb{R}^m$ and a subspace $E \subseteq \mathbb{R}^m$:*

1. $(K \cap E)^\circ = \Pi_E(K^\circ)$;
2. $(\Pi_E(K))^\circ = K^\circ \cap E$.

In both cases, the polar is taken in the subspace E.

▶ **Fact 7.** *For any invertible linear map T and any convex body K, $T(K)^\circ = T^{-*}(K^\circ)$, where T^{-*} is the inverse of the adjoint operator T^*.*

A simple special case of Fact 7 is that, for any convex body K, $(rK)^\circ = \frac{1}{r}K^\circ$. Using this property alongside Fact 5 and Fact 6, we have the following useful corollary.

▶ **Corollary 8.** *For a convex body $K \subseteq \mathbb{R}^m$ and $E \subseteq \mathbb{R}^m$ a subspace with $k = \dim E$, the following two statements are equivalent:*

1. $\Pi_E(rB_2^m) \subseteq \Pi_E(K)$;
2. $K^\circ \cap E \subseteq \frac{1}{r}(B_2^m \cap E)$,

where, as before, taking the polar set is considered in the subspace E. Notice that the second statement is also equivalent to $\operatorname{diam}(K^\circ \cap E) \leq \frac{1}{r}$.

Our work relies on appropriately quantifying the "size" of (projections and sections of) a convex body. It turns out that, for our purposes, the right measure of size is related to the notion of *width*, captured by the *support function*. Recall from the introduction that the support function of a convex body $K \subset \mathbb{R}^m$ is given by $h_K(y) = \max_{x \in K} \langle x, y \rangle$ for every $y \in \mathbb{R}^m$.

The support function is intimately related to the *Minkowski norm* $\|\cdot\|_K$, defined for a symmetric convex body $K \subseteq \mathbb{R}^m$ by $\|x\|_K = \min\{r \geq 0 : x \in rK\}$, for every $x \in \mathbb{R}^m$. It is easy to verify that $\|\cdot\|_K$ is indeed a norm. The support function h_K is identical to the Minkowski norm of the polar body K° (which is also the dual norm to $\|\cdot\|_K$): $h_K(y) = \|y\|_{K^\circ}$ for every $y \in \mathbb{R}^m$.

Now we come to the measure of the "size" of a convex body which will be central to our results: the Gaussian mean width of the body, defined next.

▶ **Definition 9.** The Gaussian mean width and Gaussian mean norm of a symmetric convex body $K \subseteq \mathbb{R}^m$ are defined respectively as:

$$\ell^*(K) = \mathbb{E}\|g\|_{K^\circ} = \mathbb{E}[h_K(g)], \qquad \ell(K) = \mathbb{E}\|g\|_K,$$

where $g \sim N(0, I_m)$ is a standard Gaussian random variable.

The next lemma gives an estimate of how the mean width changes when applying a linear transformation to K. The lemma is standard and the proof is deferred to the full version of the paper.

▶ **Lemma 10.** *For any symmetric convex body $K \subset \mathbb{R}^m$, and any linear operator $T\colon \ell_2^m \to \ell_2^m$:*

$$\ell^*(T(K)) \leq \|T\|\ell^*(K).$$

Similar to approaches in previous works ([9], [16]), we exploit properties inherent to a specific position of K to prove lower bounds on its sample complexity.

▶ **Definition 11** (ℓ-position)**.** A convex body $K \subseteq \mathbb{R}^m$ is in ℓ-position if for all linear operators $T\colon \ell_2^m \to \ell_2^m$:

$$\ell^*(K) \cdot \ell(K) \leq \ell^*(T(K)) \cdot \ell(T(K)).$$

Clearly, K is in ℓ-position if and only if K° is in ℓ-position, since $\ell^*(K) = \ell(K^\circ)$ for any convex body K. Note further that the product $\ell^*(K) \cdot \ell(K)$ is scale-invariant, in the sense that $\ell^*(rK) \cdot \ell(rK) = \ell^*(K) \cdot \ell(K)$ for any nonnegative real r. This is because, for any $x, y \in \mathbb{R}^m$, $\|x\|_{rK} = \frac{1}{r}\|x\|_K$, and $h_{rK}(y) = rh_K(y)$, so $\ell^*(rK) = r\ell^*(K)$ and $\ell(rK) = \frac{1}{r}\ell(K)$.

We will relate the Gaussian mean width of K to another measure of its size, and the size of its projections and sections, known as Gelfand width. A definition follows.

▶ **Definition 12** (Gelfand width)**.** For a symmetric convex body $K \subset \mathbb{R}^m$, the *Gelfand width of order k* of K (with respect to the ℓ_2 norm) is defined as:

$$c_k(K) = \inf_E \inf\{r : K \cap E \subseteq r(B_2^m \cap E)\} = \inf_E \sup\{\|x\|_2 : x \in K \cap E\},$$

where the first infimum is over subspaces $E \subseteq \mathbb{R}^m$ of co-dimension at most $k-1$ (i.e. of dimension at least $m-k+1$). When $k > m$, we define $c_k(K) = 0$.

Note that $c_k(K) = \inf_E \operatorname{diam}(K \cap E)$, where the infimum is over subspaces $E \subseteq \mathbb{R}^m$ of codimension at most $k-1$. Observe also that for any K, $c_k(K)$ is non-increasing in k. It is well-known that the infimum in the definition is actually achieved [19].

2.2 Known Bounds

In this section, we recall some known differentially private mechanisms, with bounds on their sample complexity, as well as a lower bound on the optimal sample complexity. We start with the lower bound:

▶ **Theorem 13** ([3, 22]). *For all $\varepsilon = O(1)$, $2^{-\Omega(n)} \leq \delta \leq 1/n^{1+\Omega(1)}$ and $\alpha \leq 1/10$:*

$$\mathrm{sc}_{\varepsilon,\delta}(Q^m, \alpha) = \Omega\left(\frac{\sqrt{m \log 1/\delta}}{\alpha\varepsilon}\right). \tag{1}$$

Next we recall one of the most basic mechanisms in differential privacy, the Gaussian mechanism. A proof of the privacy guarantee, with the constants given below, can be found in [16].

▶ **Theorem 14** (Gaussian Mechanism [4, 7, 5]). *Let $\mathcal{D} = \{x_1, \ldots, x_n\}$ be such that $\forall i$: $\|x_i\|_2 \leq \sigma$. If $w \sim N(0, \sigma(\varepsilon,\delta)^2\sigma^2 I_m)$, $\sigma(\varepsilon,\delta) = (\sqrt{\varepsilon} + 2\sqrt{2\log(1/\delta)})/\varepsilon$ and $I_m \in \mathbb{R}^{m\times m}$ is the identity matrix, then the algorithm $\mathcal{A}_{GM}$ defined by $\mathcal{A}_{GM}(\mathcal{D}) = \overline{\mathcal{D}} + \frac{1}{n}w$ is (ε,δ)-differentially private.*

▶ **Corollary 15.** *For any symmetric convex $K \subseteq B_2^m$:*

$$\mathrm{sc}_{\varepsilon,\delta}(K, \alpha) = O\left(\frac{\sqrt{m \log 1/\delta}}{\alpha\varepsilon}\right).$$

In the rest of the paper we will use the notation $\sigma(\varepsilon,\delta) = \frac{\sqrt{\varepsilon}+2\sqrt{2\log(1/\delta)}}{\varepsilon}$ from the theorem statement above.

Finally, we also present the projection mechanism from [16], which post-processes the output of the Gaussian mechanism by projecting onto K.

▶ **Theorem 16** (Projection Mechanism [16, 6]). *Let $K \subseteq B_2^m$ be a symmetric convex body, and define $\mathcal{A}_{PM}$ to be the algorithm that, on input $\mathcal{D} = \{x_1, \ldots, x_n\} \subset K$, outputs:*

$$\hat{y} = \arg\min\{\|\hat{y} - \tilde{y}\|_2^2 : \hat{y} \in K\},$$

where $\tilde{y} = \overline{\mathcal{D}} + \frac{1}{n}w$, $w \sim N(0, \sigma(\varepsilon,\delta)^2 I_m)$. Then $\mathcal{A}_{PM}$ satisfies (ε,δ)-differential privacy and has sample complexity:

$$\mathrm{sc}(K, \mathcal{A}_{PM}, \alpha) = O\left(\frac{\sigma(\varepsilon,\delta)\ell^*(K)}{\alpha^2}\right).$$

▶ **Corollary 17.** *For any symmetric convex $K \subseteq B_2^m$:*

$$\mathrm{sc}_{\varepsilon,\delta}(K, \alpha) = O\left(\frac{\sigma(\varepsilon,\delta)\ell^*(K)}{\alpha^2}\right).$$

3 Basic Properties of Sample Complexity

In this section, we prove some fundamental properties of sample complexity that will be extensively used in later sections. The proofs Lemmas 18,20 and Theorem 23 are deferred to the full version of the paper.

▶ **Lemma 18.** *$L \subseteq K \Rightarrow \forall \alpha \in (0,1) : \mathrm{sc}_{\varepsilon,\delta}(L,\alpha) \leq \mathrm{sc}_{\varepsilon,\delta}(K,\alpha)$.*

▶ **Corollary 19.** *For all $\varepsilon = O(1)$, $2^{-\Omega(n)} \leq \delta \leq 1/n^{1+\Omega(1)}$ and $\alpha \leq 1/10$:*

$$\mathrm{sc}_{\varepsilon,\delta}(B_2^m,\alpha) = \Omega\left(\frac{\sqrt{m\log 1/\delta}}{\alpha\varepsilon}\right).$$

Proof. Since $Q^m \subseteq B_2^m$, this follows directly from Lemma 18 and Theorem 13. ◀

▶ **Lemma 20.** *For any $\alpha \in (0,1)$, any linear operator $T\colon \mathbb{R}^m \to \mathbb{R}^m$ and any symmetric convex body $K \subset \mathbb{R}^m$:*

$$\mathrm{sc}_{\varepsilon,\delta}(K,\alpha) \geq \mathrm{sc}_{\varepsilon,\delta}(T(K),\alpha\cdot\|T\|).$$

▶ **Corollary 21.** *For any $t > 0$:*

$$\mathrm{sc}_{\varepsilon,\delta}(tK,t\alpha) = \mathrm{sc}_{\varepsilon,\delta}(K,\alpha).$$

Proof. Taking $T = tI_m$ in Lemma 20, where I_m is the identity on $\mathbb{R}^m$, the lemma implies $\mathrm{sc}_{\varepsilon,\delta}(tK,t\alpha) \leq \mathrm{sc}_{\varepsilon,\delta}(K,\alpha)$. Since this inequality holds for any t and K, we may apply it to $K' = tK$ and $t' = 1/t$, and we get $\mathrm{sc}_{\varepsilon,\delta}(K,\alpha) = \mathrm{sc}_{\varepsilon,\delta}((1/t)tK,(1/t)t\alpha) \leq \mathrm{sc}_{\varepsilon,\delta}(tK,t\alpha)$. ◀

Since for any subspace E of $\mathbb{R}^m$, the corresponding orthogonal projection Π_E has operator norm 1, we also immediately get the following corollary of Lemma 20:

▶ **Corollary 22.** *For any subspace E:*

$$\mathrm{sc}_{\varepsilon,\delta}(K,\alpha) \geq \mathrm{sc}_{\varepsilon,\delta}(\Pi_E(K),\alpha).$$

In the next theorem, we combine the lower bound in Theorem 19 and the properties we proved above in order to give a lower bound on the sample complexity of an arbitrary symmetric convex body K in terms of its geometric properties. In the following sections we will relate this geometric lower bound to the mean Gaussian width of K.

▶ **Theorem 23** (Geometric Lower Bound)**.** *For all $\varepsilon = O(1)$, $2^{-\Omega(n)} \leq \delta \leq 1/n^{1+\Omega(1)}$, any convex symmetric body $K \subseteq \mathbb{R}^m$, any $1 \leq k \leq m$ and any $\alpha \leq 1/(10c_k(K^\circ))$:*

$$\mathrm{sc}_{\varepsilon,\delta}(K,\alpha) = \Omega\left(\frac{\sqrt{\log 1/\delta}}{\alpha\varepsilon}\cdot\frac{\sqrt{m-k+1}}{c_k(K^\circ)}\right).$$

4 Optimality of the Gaussian Mechanism

In this section, we present the result that the Gaussian mechanism is optimal, up to constant factors, when $K \subseteq B_2^m$ is sufficiently large. More specifically, if the Gaussian mean width of K is asymptotically maximal, then we can get a tight lower bound on the sample complexity of the Gaussian mechanism. This is summarized in the theorem below.

▶ **Theorem 24.** *For all $\varepsilon < O(1)$, $2^{-\Omega(n)} \leq \delta \leq 1/n^{1+\Omega(1)}$, sufficiently small constant α, and any symmetric convex body $K \subseteq B_2^m$, if*

$$\ell^*(K) = \Omega(\sqrt{m}),$$

then:

$$\mathrm{sc}_{\varepsilon,\delta}(K,\alpha) = \Theta\left(\frac{\sqrt{m \log 1/\delta}}{\alpha\varepsilon}\right),$$

and $\mathrm{sc}_{\varepsilon,\delta}(K,\alpha)$ is achieved, up to constants, by the Gaussian mechanism.

By Theorem 15 we have an upper bound for the Gaussian mechanism defined previously. To prove its optimality, we use a classical result from convex geometry, known as Dvoretzky's criterion, to show a matching lower bound for the sample complexity. This result relates the existence of a nearly-spherical section of a given convex body to the Gaussian mean norm. It was a key ingredient in Milman's probabilistic proof of Dvoretzky's theorem: see Matoušek's book [12] for an exposition.

▶ **Theorem 25** ([13]; Dvoretzky's Criterion)**.** *For every symmetric convex body $K \subseteq \mathbb{R}^m$ such that $B_2^m \subseteq K$, and every $\beta < 1$, there exists a constant $c(\beta)$ and a subspace E with dimension $\dim E \geq c(\beta)\ell(K)^2$ for which:*

$$(1-\beta)\frac{\ell(K)}{\sqrt{m}}B_2^m \cap E \subseteq K \cap E \subseteq (1+\beta)\frac{\ell(K)}{\sqrt{m}}B_2^m \cap E.$$

Proof of Theorem 24. Given the matching upper bound on sample complexity in Theorem 15, it suffices to show the equivalent lower bound, namely that:

$$\mathrm{sc}_{\varepsilon,\delta}(K,\alpha) = \Omega\left(\frac{\sqrt{m \log 1/\delta}}{\alpha\varepsilon}\right).$$

To this end, we will show that there exists a $k \leq (1-c)m + 1$, for an absolute constant c, so that $c_k(K^\circ) = O(1)$. Then the lower bound will follow directly from Theorem 23.

We will prove the claim above by applying Dvoretzky's criterion to K°. By Theorem 5, $K \subseteq B_2^m \Rightarrow B_2^m \subseteq K^\circ$. We can then apply Dvoretzky's criterion with $\beta = 1/2$, ensuring that there exists a subspace E of dimension $\dim E \geq c(1/2)\ell(K^\circ)^2$ for which:

$$K^\circ \cap E \subseteq \frac{\ell(K^\circ)}{2\sqrt{m}}B_2^m \cap E.$$

Let us define $k = m - \dim E + 1$; then $k \leq m - c(1/2)\ell(K^\circ)^2 + 1 = m - c(1/2)\ell^*(K)^2 + 1$. Since, by assumption $\ell^*(K) = \Omega(m)$, there exists a constant c so that $k \leq (1-c)m + 1$. Finally, by the definition of Gelfand width, $c_k(K^\circ) \leq \frac{\ell(K^\circ)}{2\sqrt{m}} = O(1)$, as desired. This completes the proof. ◀

5 Gaussian Width Lower Bounds in ℓ-position

In Section 4 we showed that the Gaussian Mechanism is optimal when the Gaussian mean width of K is asymptotically as large possible. Our goal in this and the following section is to show general lower bounds on sample complexity in terms of $\ell^*(K)$. This is motivated by the sample complexity upper bound in terms of $\ell^*(K)$ provided by the projection mechanism.

It is natural to follow the strategy from Section 4: use Dvoretzky's criterion to find a nearly-spherical projection of K of appropriate radius and dimension. An inspection of the proof of Theorem 24 shows that the sample complexity lower bound we get this way is $\Omega\left(\frac{\ell^*(K)^2}{\sqrt{m}}\right)$ (ignoring the dependence on ε, δ, and α here, and in the rest of this informal discussion). Recall that we are aiming for a lower bound of $\Omega(\ell^*(K))$, so we are off by a factor of $\frac{\ell^*(K)}{\sqrt{m}}$. Roughly speaking, the problem is that Dvoretzky's criterion does too much: it guarantees a spherical section of K°, while we only need a bound on the diameter of the section. In order to circumvent this difficulty, we use a different result from asymptotic convex geometry, the low M^*-estimate, which bounds the diameter of a random section of K°, without also producing a large ball contained inside the section. A technical difficulty is that the resulting upper bound on the diameter is in terms of the Gaussian mean K-norm, rather than the (reciprocal of the) mean width. When K is in ℓ-position, this is not an issue, because results of Pisier, Figiel, and Tomczak-Jaegermann show that in that case $\ell(K)\ell^*(K) = O(\log m)$. In this section we assume that K is in ℓ-position, and we remove this requirement in the subsequent section.

The main result of this section is summarized below.

▶ **Theorem 26.** *For all $\varepsilon = O(1)$, $2^{-\Omega(n)} \leq \delta \leq 1/n^{1+\Omega(1)}$, all symmetric convex bodies $K \subseteq \mathbb{R}^m$ in ℓ-position, and for $\alpha \leq \frac{\ell^*(K)}{C\sqrt{m}\log 2m}$, where C is an absolute constant:*

$$\mathrm{sc}_{\varepsilon,\delta}(K,\alpha) = \Omega\left(\frac{\sqrt{\log 1/\delta}}{\alpha\varepsilon}\cdot\frac{\ell^*(K)}{\log 2m}\right).$$

The following two theorems are the main technical ingredients we need in the proof of Theorem 26.

▶ **Theorem 27** ([8], [20]; MM^* Bound)**.** *There exists a constant C such that for every symmetric convex body $K \subset \mathbb{R}^m$ in ℓ-position:*

$$\ell(K)\cdot\ell^*(K) \leq C\cdot m\log 2m.$$

It is an open problem whether this bound can be improved to $m\sqrt{\log 2m}$. This would be tight for the cube Q^m. This improvement would lead to a corresponding improvement in our bounds.

▶ **Theorem 28** ([17]; Low M^* estimate)**.** *' There exists a constant C such that for every symmetric convex body $K \subset \mathbb{R}^m$ there exists a subspace $E \subseteq \mathbb{R}^m$ with $\dim E = m - k$ for which:*

$$\mathrm{diam}(K\cap E) \leq C\cdot\frac{\ell^*(K)}{\sqrt{k}}.$$

Combining Theorems 27 and 28, we get the following key lemma.

▶ **Lemma 29.** *There exists a constant C such that for every symmetric convex body $K \subset \mathbb{R}^m$ in ℓ-position, and every $\beta \in (0, 1-1/m)$, there exists a subspace E of dimension at least βm satisfying:*

$$\mathrm{diam}(K\cap E) \leq C\frac{\sqrt{m}\log 2m}{\sqrt{1-\beta}\cdot\ell(K)}.$$

Proof. Let $k = \lfloor (1-\beta)m \rfloor \geq 1$. Using the low M^* estimate on K, there exists a subspace E with $\dim E = m - k = \lceil \beta m \rceil$ for which:

$$\operatorname{diam}(K \cap E) \leq C_1 \cdot \frac{\ell^*(K)}{\sqrt{k}}.$$

By the MM^* upper bound, since K is in ℓ-position, we have that:

$$\ell^*(K) \leq C_2 \cdot \frac{m \log 2m}{\ell(K)},$$

and, combining the two inequalities, we get that:

$$\operatorname{diam}(K \cap E) \leq C_1 C_2 \frac{m \log 2m}{\sqrt{k} \cdot \ell(K)} \leq C \frac{\sqrt{m} \log m}{\sqrt{1-\beta} \cdot \ell(K)},$$

for an appropriate constant C. This completes the proof. ◀

The proof of the desired lower bound now follows easily from this lemma.

Proof of Theorem 26. By Theorem 23, it suffices to show that

$$\max_{k=1}^{m} \frac{\sqrt{m-k+1}}{c_k(K^\circ)} = \Omega\left(\frac{\ell^*(K)}{\log 2m}\right). \tag{2}$$

Indeed, if k^* is the value of k for which the maximum on the left hand side is achieved, then $\frac{\sqrt{m-k^*+1}}{c_{k^*}(K^\circ)}$ is a lower bound on the sample complexity for all $\alpha \leq 1/(10 c_{k^*}(K^\circ))$, and by (2):

$$\frac{1}{10 c_{k^*}(K^\circ)} = \Omega\left(\frac{\ell^*(K)}{\sqrt{m-k^*+1} \cdot \log 2m}\right) = \Omega\left(\frac{\ell^*(K)}{\sqrt{m} \cdot \log 2m}\right).$$

In the rest of the proof, we establish (2).

Since K (and thus also K°) are in ℓ-position by assumption, from Lemma 29 applied to K° we have that there exists a subspace E such that $\dim E \geq m/2$ and:

$$\operatorname{diam}(K^\circ \cap E) = O\left(\frac{\sqrt{m} \log 2m}{\ell(K^\circ)}\right) = O\left(\frac{\sqrt{m} \log 2m}{\ell^*(K)}\right).$$

Setting $k_E = m - \dim E + 1 \leq m/2 + 1$, and because $c_{k_E}(K^\circ) \leq \operatorname{diam}(K^\circ \cap E)$ by definition, we get that:

$$\max_{k=1}^{m} \frac{\sqrt{m-k+1}}{c_k(K^\circ)} \geq \frac{\sqrt{\dim E}}{\operatorname{diam}(K^\circ \cap E)} = \Omega\left(\frac{\ell^*(K)}{\log 2m}\right),$$

as desired. ◀

6 Gaussian Width Lower Bounds for Arbitrary Bodies

In this section, we remove the assumption that K is in ℓ-position from the previous section. Instead, we use a recursive charging argument in order to reduce to the ℓ-position case. The resulting guarantee is worse than the one we proved for bodies in ℓ-position by a logarithmic factor.

The main lower bound result of this section is the following theorem.

▶ **Theorem 30.** *For all $\varepsilon = O(1)$, $2^{-\Omega(n)} \leq \delta \leq 1/n^{1+\Omega(1)}$, any symmetric convex body $K \subset \mathbb{R}^m$, and any $\alpha \leq \frac{\ell^*(K)}{C\sqrt{m}(\log 2m)^2}$, where C is an absolute constant:*

$$\mathrm{sc}_{\varepsilon,\delta}(K,\alpha) = \Omega\left(\frac{\sigma(\varepsilon,\delta)\ell^*(K)}{(\log 2m)^2\alpha}\right).$$

The lower bound follows from the geometric lemma below, which may be of independent interest.

▶ **Lemma 31.** *There exists a constant C such that, for any symmetric convex body $K \subset \mathbb{R}^m$,*

$$\ell^*(K) \leq C(\log 2m)\left(\sum_{i=1}^{m} \frac{1}{\sqrt{i}\cdot c_{m-i+1}(K^\circ)}\right). \tag{3}$$

Lemma 31 is closely related to the volume number theorem of Milman and Pisier [14], which states that the inequality (3) holds with $\frac{1}{c_{m-i+1}(K^\circ)}$ replaced by the volume number $v_i(K)$, defined as:

$$v_i(K) = \sup_{E:\dim E=i} \frac{\mathrm{vol}(\Pi_E(K))^{1/i}}{(\mathrm{vol}(\Pi_E(B_2^m))^{1/i}},$$

where the supremum is over subspaces E of $\mathbb{R}^m$. Inequality (3) is stronger than the volume number theorem, because $\frac{1}{c_{m-i+1}(K^\circ)} \leq v_i(K)$. Indeed, setting $r = \frac{1}{c_{m-i+1}(K^\circ)}$, by Theorem 8 and the definition of Gelfand width we have that there exists a subspace E of dimension i such that $r\Pi_E(B_2^m) \subseteq \Pi_E K$. Therefore, $\mathrm{vol}(\Pi_E(K)) \geq r^i \,\mathrm{vol}(\Pi_E(B_2^m))$, which implies the desired inequality.

Even though the volume number theorem is weaker than (3), the proof given by Pisier in his book [21], with minor modifications, appears to yield the stronger inequality we need. In the full version of the paper we give an alternative proof, which only uses the low M^* estimate, the MM^* estimate, and elementary linear algebra.

Proof of Theorem 30. As in the proof of Theorem 26, it is sufficient to prove that:

$$\max_{k=1}^{m} \frac{\sqrt{m-k+1}}{c_k(K^\circ)} = \Omega\left(\frac{\ell^*(K)}{(\log 2m)^2}\right). \tag{4}$$

But this inequality follows easily from Lemma 31 and the trivial case of Hölder's inequality:

$$\begin{aligned}\ell^*(K) \leq C(\log 2m)\left(\sum_{i=1}^{m} \frac{1}{\sqrt{i}\cdot c_{m-i+1}(K^\circ)}\right) &\leq C(\log 2m)\left(\sum_{i=1}^{m}\frac{1}{i}\right)\cdot\left(\max_{i=1}^{m}\frac{\sqrt{i}}{c_{m-i+1}(K^\circ)}\right)\\ &= O((\log 2m)^2)\cdot\left(\max_{i=1}^{m}\frac{\sqrt{i}}{c_{m-i+1}(K^\circ)}\right).\end{aligned}$$

Then, the proof of the theorem follows from (4) analogously to the proof of Theorem 26. ◀

We now have everything in place to prove our main result for the mean point problem.

Proof of Theorem 4. The upper bounds on sample complexity follow from Theorem 14, Theorem 15, Theorem 16, and Theorem 17. The lower bounds follow from Theorem 30. The statement after "moreover" follows from Theorem 15 and Theorem 24. ◀

7 From Mean Point to Query Release

All the bounds we proved so far were for the mean point problem. In this section we show reductions between this problem, and the query release problem, which allow us to translate our lower bounds to the query release setting and prove Theorem 3. We will show that the problem of approximating $\mathcal{Q}(\mathcal{D})$ for a query workload $\mathcal{Q}$ under differential privacy is nearly equivalent to approximating the mean point problem with universe $K' = \frac{1}{\sqrt{m}}K$, where K is the sensitivity polytope of $\mathcal{Q}$. We state this reduction next, and defer its proof to the full version of the paper.

▶ **Lemma 32.** *Let $\mathcal{Q}$ be a workload of m linear queries over the universe $\mathcal{U}$ with sensitivity polytope K. Define $K' = \frac{1}{\sqrt{m}}K$. Then, we have the inequalities:*

$$\mathrm{sc}_{\varepsilon,\delta}(\mathcal{Q},\alpha) \le \mathrm{sc}_{\varepsilon,\delta}(K',\alpha); \tag{5}$$

$$\mathrm{sc}_{\varepsilon,\delta}(K',\alpha) \le \max\left\{\mathrm{sc}_{2\varepsilon,2\delta}(\mathcal{Q},\alpha/4), \frac{16\,\mathrm{diam}(K')}{\alpha^2}\right\}. \tag{6}$$

Moreover, we can use an (ε,δ)-differentially private algorithm $\mathcal{A}'$ as a black box to get an (ε,δ)-differentially private algorithm $\mathcal{A}$ such that $\mathrm{sc}(\mathcal{Q},\mathcal{A},\alpha) = \mathrm{sc}(K',\mathcal{A}',\alpha)$. $\mathcal{A}$ makes a single call to $\mathcal{A}'$, and performs additional computation of worst-case complexity $O(mn)$, where n is the size of the database.

We also use a simple lemma that relates the sample complexity at an error level α to the sample complexity at a lower error level $\alpha' < \alpha$. The proof is a padding argument and can be found in [22].

▶ **Lemma 33.** *For any workload $\mathcal{Q}$, any $0 < \alpha' < \alpha < 1$, and any privacy parameters ε, δ, we have*

$$\mathrm{sc}_{\varepsilon,\delta}(\mathcal{Q},\alpha') = \Omega\left(\frac{\alpha}{C\alpha'}\right)\cdot \mathrm{sc}_{\varepsilon,\delta}(\mathcal{Q},\alpha),$$

for an absolute constant C.

We are now ready to finish the proof of our main result for the query release problem.

Proof of Theorem 3. The upper bounds on sample complexity follow from the upper bounds in Theorem 4 together with Lemma 32.

Denote $K' = \frac{1}{\sqrt{m}}K$, and let $\alpha_0 = \frac{\ell^*(K')}{C\sqrt{m}(\log 2m)^2} = \frac{\ell^*(K)}{Cm(\log 2m)^2}$ be the smallest error parameter for which Theorem 30 holds. Then, by Theorem 30 and Lemma 32:

$$\max\left\{\mathrm{sc}_{\varepsilon,\delta}(\mathcal{Q},\alpha_0), \frac{\mathrm{diam}(K)}{\sqrt{m}\alpha_0^2}\right\} = \Omega\left(\frac{\sigma(\varepsilon,\delta)\ell^*(K)}{\sqrt{m}(\log 2m)^2\alpha_0}\right).$$

It is easy to show that $\mathrm{sc}_{\varepsilon,\delta}(\mathcal{Q},\alpha_0) = \Omega(\mathrm{diam}(K)/(\alpha_0\sqrt{m}))$ for all sufficiently small ε and δ. Therefore, we have:

$$\mathrm{sc}_{\varepsilon,\delta}(\mathcal{Q},\alpha_0) = \Omega\left(\frac{\sigma(\varepsilon,\delta)\ell^*(K)}{\sqrt{m}(\log 2m)^2}\right).$$

By Lemma 33, we get that for any $\alpha \le \alpha_0$ the sample complexity is at least:

$$\mathrm{sc}_{\varepsilon,\delta}(\mathcal{Q},\alpha) = \Omega\left(\frac{\sigma(\varepsilon,\delta)\ell^*(K)\alpha_0}{\sqrt{m}(\log 2m)^2\alpha}\right) = \Omega\left(\frac{\sigma(\varepsilon,\delta)\ell^*(K)^2}{m^{3/2}(\log 2m)^4\alpha}\right).$$

An analogous proof, with $\alpha_0 = 1/C$ set to the smallest error parameter for which Theorem 24 holds, establishes the statement after "moreover". ◀

References

1. Shiri Artstein-Avidan, Apostolos Giannopoulos, and Vitali D. Milman. *Asymptotic geometric analysis. Part I*, volume 202 of *Mathematical Surveys and Monographs*. American Mathematical Society, Providence, RI, 2015. doi:10.1090/surv/202.
2. Aditya Bhaskara, Daniel Dadush, Ravishankar Krishnaswamy, and Kunal Talwar. Unconditional differentially private mechanisms for linear queries. In *Proceedings of the 44th Symposium on Theory of Computing*, STOC'12, pages 1269–1284, New York, NY, USA, 2012. ACM. doi:10.1145/2213977.2214089.
3. Mark Bun, Jonathan Ullman, and Salil Vadhan. Fingerprinting codes and the price of approximate differential privacy. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, pages 1–10. ACM, 2014.
4. Irit Dinur and Kobbi Nissim. Revealing information while preserving privacy. In *Proceedings of the 22nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 202–210. ACM, 2003.
5. Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography Conference*, pages 265–284. Springer, 2006.
6. Cynthia Dwork, Aleksandar Nikolov, and Kunal Talwar. Using convex relaxations for efficiently and privately releasing marginals. In *30th Annual Symposium on Computational Geometry, SOCG'14, Kyoto, Japan, June 08-11, 2014*, page 261. ACM, 2014. doi:10.1145/2582112.2582123.
7. Cynthia Dwork and Kobbi Nissim. Privacy-preserving datamining on vertically partitioned databases. In *Annual International Cryptology Conference*, pages 528–544. Springer, 2004.
8. T. Figiel and Nicole Tomczak-Jaegermann. Projections onto Hilbertian subspaces of Banach spaces. *Israel J. Math.*, 33(2):155–171, 1979. doi:10.1007/BF02760556.
9. Moritz Hardt and Kunal Talwar. On the geometry of differential privacy. In *Proceedings of the 42nd ACM Symposium on Theory of Computing*, STOC'10, pages 705–714, New York, NY, USA, 2010. ACM. doi:10.1145/1806689.1806786.
10. Michael Kearns. Efficient noise-tolerant learning from statistical queries. *J. ACM*, 45(6):983–1006, 1998. doi:10.1145/293347.293351.
11. Michel Ledoux and Michel Talagrand. *Probability in Banach spaces*. Classics in Mathematics. Springer-Verlag, Berlin, 2011. Isoperimetry and processes, Reprint of the 1991 edition.
12. Jiří 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.
13. V. D. Milman. A new proof of A. Dvoretzky's theorem on cross-sections of convex bodies. *Funkcional. Anal. i Priložen.*, 5(4):28–37, 1971.
14. V. D. Milman and G. Pisier. Gaussian processes and mixed volumes. *Ann. Probab.*, 15(1):292–304, 1987. URL: http://links.jstor.org/sici?sici=0091-1798(198701)15:1<292:GPAMV>2.0.CO;2-A&origin=MSN.
15. Aleksandar Nikolov. An improved private mechanism for small databases. In *Automata, Languages, and Programming – 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part I*, volume 9134 of *Lecture Notes in Computer Science*, pages 1010–1021. Springer, 2015. doi:10.1007/978-3-662-47672-7_82.
16. Aleksandar Nikolov, Kunal Talwar, and Li Zhang. The geometry of differential privacy: the sparse and approximate cases. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 351–360. ACM, 2013. doi:10.1145/2488608.2488652.

17 Alain Pajor and Nicole Tomczak-Jaegermann. Subspaces of small codimension of finite-dimensional Banach spaces. *Proc. Amer. Math. Soc.*, 97(4):637–642, 1986. doi:10.2307/2045920.

18 Christos H. Papadimitriou and Mihalis Yannakakis. On limited nondeterminism and the complexity of the V-C dimension. *J. Comput. Syst. Sci.*, 53(2):161–170, 1996. doi:10.1006/jcss.1996.0058.

19 Allan Pinkus. *n-widths in approximation theory*, volume 7 of *Ergebnisse der Mathematik und ihrer Grenzgebiete (3) [Results in Mathematics and Related Areas (3)]*. Springer-Verlag, Berlin, 1985. doi:10.1007/978-3-642-69894-1.

20 G. Pisier. Sur les espaces de Banach K-convexes. In *Seminar on Functional Analysis, 1979–1980 (French)*, pages Exp. No. 11, 15. École Polytech., Palaiseau, 1980.

21 Gilles Pisier. *The volume of convex bodies and Banach space geometry*, volume 94 of *Cambridge Tracts in Mathematics*. Cambridge University Press, Cambridge, 1989. doi:10.1017/CBO9780511662454.

22 Thomas Steinke and Jonathan Ullman. Between pure and approximate differential privacy. *CoRR*, abs/1501.06095, 2015. URL: http://arxiv.org/abs/1501.06095.

23 R. Vershynin. Lectures in geometric functional analysis. 2009. URL: http://www-personal.umich.edu/~romanv/papers/GFA-book.pdf.

Constrained Triangulations, Volumes of Polytopes, and Unit Equations

Michael Kerber[1], Robert Tichy[2], and Mario Weitzer[3]

1 **Institute of Geometry, Graz University of Technology, Graz, Austria**
kerber@tugraz.at
2 **Institute of Analysis and Number Theory, Graz University of Technology, Graz, Austria**
tichy@tugraz.at
3 **Institute of Analysis and Number Theory, Graz University of Technology, Graz, Austria**
weitzer@math.tugraz.at

Abstract

Given a polytope $\mathcal{P}$ in $\mathbb{R}^d$ and a subset U of its vertices, is there a triangulation of $\mathcal{P}$ using d-simplices that all contain U? We answer this question by proving an equivalent and easy-to-check combinatorial criterion for the facets of $\mathcal{P}$. Our proof relates triangulations of $\mathcal{P}$ to triangulations of its "shadow", a projection to a lower-dimensional space determined by U. In particular, we obtain a formula relating the volume of $\mathcal{P}$ with the volume of its shadow. This leads to an exact formula for the volume of a polytope arising in the theory of unit equations.

1998 ACM Subject Classification F.2.2 [Nonnumerical Algorithms and Problems] Geometrical Problems and Computations, F.2.1 [Numerical Algorithms and Problems] Number-Theoretic Computations

Keywords and phrases constrained triangulations, simplotopes, volumes of polytopes, projections of polytopes, unit equations, S-integers

Digital Object Identifier 10.4230/LIPIcs.SoCG.2017.46

1 Introduction

1.1 Problem statement and results

Let $\mathcal{P}$ be a convex polytope in $\mathbb{R}^d$, that is, the convex hull of a finite point set V, and let U be a subset of V. We ask for a triangulation of (the interior of) $\mathcal{P}$ with the property that every d-simplex in the triangulation contains all points of U as vertices, calling it a *U-spinal triangulation*. A simple example is the *star triangulation* of $\mathcal{P}$ (Figure 1), where all d-simplices contain a common vertex $\mathbf{p}$, and U is the singleton set consisting of that point. Another example is the d-hypercube with U being a pair of opposite points (Figure 2). Indeed, the hypercube can be triangulated in a way that all d-simplices contain the space diagonal spanned by U [16, 10].

We are interested in what combinations of $\mathcal{P}$ and U admit spinal triangulations. Our results provide a simple combinatorial answer for this question: Denoting by n the cardinality of U, a U-spinal triangulation of $\mathcal{P}$ exists if and only if each facet of $\mathcal{P}$ contains at least $n-1$ vertices of U. In that case, we call U a *spine* of $\mathcal{P}$. More generally, we provide a complete characterization of spinal triangulations: let Φ denote the orthogonal projection of $\mathbb{R}^d$ to the orthogonal complement of the lower-dimensional flat spanned by U. Φ maps U to $\mathbf{0}$ by construction, and $\mathcal{P}$ is mapped to a *shadow* $\hat{\mathcal{P}} := \Phi(\mathcal{P})$. We obtain a U-spinal triangulation

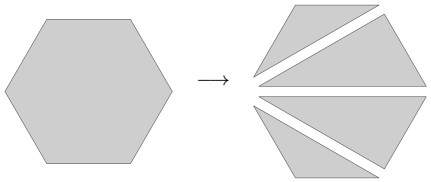


Figure 1 A star triangulation of a hexagon.

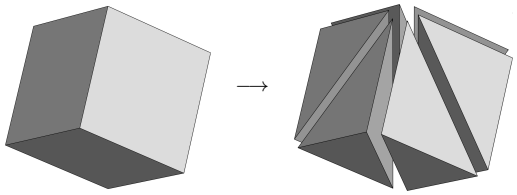


Figure 2 A U-spinal triangulation of a cube, where U consists of the two vertices on a space diagonal.

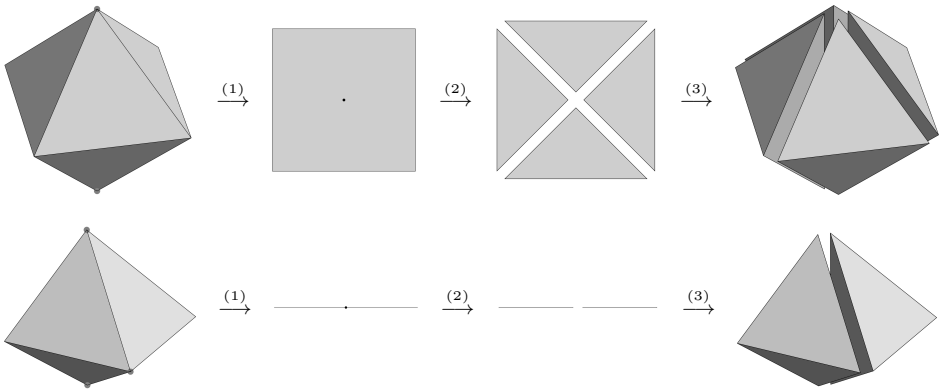


Figure 3 Two examples of the lifting process: (1) Project polytope $\mathcal{P}$ to the orthogonal complement of the flat spanned by U (vertices marked by prominent dots) to obtain shadow $\hat{\mathcal{P}}$. (2) Star-triangulate $\hat{\mathcal{P}}$ with respect to the origin. (3) Lift star triangulation of $\hat{\mathcal{P}}$ to obtain U-spinal triangulation of $\mathcal{P}$. Note: every facet of $\mathcal{P}$ contains exactly $|U| - 1$ points of U in both examples.

of $\mathcal{P}$ by first star-triangulating $\hat{\mathcal{P}}$ with respect to $\mathbf{0}$ and then lifting each maximal simplex to $\mathbb{R}^d$ by taking the preimage of its vertices under Φ (Figure 3). Vice versa, every spinal triangulation can be obtained in this way.

An important consequence of our characterization is that a spine allows us to relate the volumes of a convex polytope $\mathcal{P}$ and its shadow $\hat{\mathcal{P}}$ (with respect to that spine) by a precise equation. An application of this observation leads to our second result: an exact volume formula of an important polytope arising in number theory which we call the *Everest polytope*. We show that this polytope is the shadow of a higher-dimensional *simplotope*, the product of simplices, whose volume is easy to determine.

1.2 Number theoretic background

In the following we briefly discuss the number-theoretic background of the Everest polytope. G. R. Everest [12, 13] studied various counting problems related to Diophantine equations. In particular, he proved asymptotic results for the number of values taken by a linear form whose variables are restricted to lie inside a given finitely generated subgroup of a number field. This includes norm form- and discriminant form equations, normal integral bases and related objects. Everest's work contains important contributions to the quantitative theory of S-unit equations and makes use of Baker's theory of linear forms in logarithms and Schmidt's subspace theorem from Diophantine approximation; see for instance [30, 14, 2]. Later, other authors [15] applied the methods of Everest to solve combinatorial problems in algebraic number fields. The corresponding counting results involve various important arithmetic constants, one of them being the volume of a certain convex polytope.

In order to introduce Everest's constant, we use basic facts from algebraic number theory. Let K be a number field, $N = N_{K/\mathbb{Q}}$ the field norm and S a finite set of places of K including the archimedian ones. We denote by $O_{K,S} = \{\alpha \in K : |\alpha|_v \leq 1 \text{ for all } v \notin S\}$ the ring of S-integers and its unit group by $U_{K,S}$; the group of S-units. Let $c_0, \ldots, c_n$ denote given non-zero algebraic numbers. During the last decades, a lot of work is devoted to the study of the values taken by the expression $c_0x_0 + \ldots + c_nx_n$, where the x_n are allowed to run through $U_{K,S}$, see for instance [26, 19]. A specific instance of this kind of general S-unit equations is the following combinatorial problem. As usual, two S-integers α and β are said to be associated (for short $\alpha \sim \beta$) if there exists an S-unit ϵ such that $\alpha = \beta\epsilon$. It is well-known that the group of S-units $U_{K,S}$ is a free abelian group with $s = |S| - 1$ generators, ω_K and $\mathrm{Reg}_{K,S}$ denote as usual the number of roots of unity and the S-regulator of K, respectively (for the basic concepts of algebraic number theory see [25]). Then for given $n \in \mathbb{N}, q > 0$ the counting function $u(n, q)$ is defined as the number of equivalence classes $[\alpha]_\sim$ such that

$$N(\alpha) := \prod_{v \in S} |\alpha|_v \leq q, \ \alpha = \sum_{i=1}^{n} \varepsilon_i,$$

where $\varepsilon_i \in U_{K,S}$ and no subsum of $\varepsilon_1 + \ldots + \varepsilon_n$ vanishes. From the work of Everest [12, 13], the following asymptotic formula can be derived:

$$u(n, q) = \frac{c(n-1, s)}{n!} \left(\frac{\omega_K (\log q)^s}{\mathrm{Reg}_{K,S}} \right)^{n-1} + o((\log q)^{(n-1)s-1+\varepsilon})$$

for arbitrary $\varepsilon > 0$. Here $c(n-1, s)$ is a positive constant, and its exact value has been known only in special cases; see [3] for more details. In general, $c(n, s)$ is given as the volume of a convex polytope in $\mathbb{R}^{ns}$ that we define and study in Section 4. Our results show that

$$c(n, s) = \frac{1}{(s!)^{n+1}} \frac{((n+1)s)!}{(ns)!},$$

which can also be written in terms of a multinomial coefficient as $\binom{(n+1)s}{s,\ldots,s} \frac{1}{(ns)!}$.

1.3 Geometric background

Our results fall into the category of *constrained triangulations of convex polytopes*. Triangulations of polytopes are a classic topic in discrete geometry; an infamous question is the quest for triangulating a d-hypercube with a minimal number of simplices [6]. Precise answers are only known up to dimension 7 using computer-assisted proofs [22]. A contemporary overview on results relating to triangulations of polytopes and more general point configurations is provided by de Loera, Rambau, and Santos [8]. It includes a discussion on the triangulation of simplotopes, a geometric object whose study goes back to Hadwiger [20], and has been studied, for instance, in the context of combinatorics [17], game theory [32] and algebraic geometry [18, Ch. 7]. Simplotopes admit a standard triangulation, the so-called *staircase triangulation*, which can easily be described in combinatorial terms. A by-product of our results is that simplotopes can also be triangulated by a family of spinal triangulations.

An n-element subset U of the vertex set of a polytope with the property that each facet contains exactly $n-1$ points of U is called a *special simplex* in the literature. Special simplices have been studied by Athanasiadis [1] to relate the Ehrhart polynomial of integer polytopes with special simplex with the h-vector of the shadow. This results found applications in the study of toric rings and Gorenstein polytopes [21, 5]. Polytopes with special simplices are further studied by de Wolff [9]. His classification yields, among other results, upper bounds

for the number of faces of a polytope with special simplices. Remarkably, special simplices with two vertices, called *spindles*, are also used by Santos for his celebrated counterexample for the Hirsch conjecture [29]. While these works employ similar techniques as our work, for instance, lifting triangulations of the shadow to triangulations of the polytope, the (more elementary) questions of this paper are not addressed in the related work. We also point out that despite the close relation to special simplices, the notion of spines introduced in this paper is slightly more general because a facet is allowed to contain all vertices of U.

Otherwise, constraining triangulations has mostly been considered for low-dimensional problems under an algorithmic angle. For instance, a *constrained Delaunay triangulation* is a triangulation which contains a fixed set of pre-determined simplices; apart from these constraints, it tries to be "as Delaunay as possible"; see Shewchuk's work [31] for details. While our work is related in spirit, there appears to be no direct connection to this framework, because our constraint does not only ensure the presence of certain simplices in the triangulation, but rather constrains all d-simplices at once.

Computing volumes of high-dimensional convex polytopes is another notoriously hard problem, from a computational perspective [24, Sec. 13][11] as well as for special cases. A famous example is the *Birkhoff polytope* of all doubly-stochastic $n \times n$-matrices, whose exact volume is known exactly only up to $n = 10$ [27, 7]. Our contribution provides a novel technique to compute volumes of polytopes through lifting into higher dimensions. We point out that lifting increases the dimension, so that the lift of a polytope is not the image of a linear transformation. Therefore, the well-known formula $\mathrm{vol}(A\mathcal{P}) = \sqrt{\det(A^T A)}\,\mathrm{vol}(\mathcal{P})$ with $A \in \mathbb{R}^{e\times d}$ and $e \geq d$ does not apply to our case.

1.4 Organization

We start by introducing the basic concepts from convex geometry in Section 2. We proceed with our structural result on spinal triangulations, in Section 3. We calculate the vertices of the Everest polytope in Section 4 and define a map from a simplotope to the Everest polytope in Section 5, leading to the volume formula for the Everest polytope. We conclude with some additional remarks in Section 6.

2 Geometric concepts

Let M be an arbitrary subset of $\mathbb{R}^d$ with some integer $d \geq 1$. The *dimension* of M is the dimension of the smallest affine subspace of $\mathbb{R}^d$ containing M. We say that M is *full-dimensional* if its dimension is equal to d. Throughout the entire paper, V will always stand for a finite point set in $\mathbb{R}^d$ that is full-dimensional and in *convex position*, that is $x \notin \mathrm{conv}(V \setminus \{x\})$ for every $x \in V$, where $\mathrm{conv}(\cdot)$ denotes the *convex hull* in $\mathbb{R}^d$.

2.1 Polytopes and simplicial complexes

We use the following standard definitions (compare, for instance, Ziegler [33]): A *polytope* $\mathcal{P}$ is the convex hull of a finite point set in $\mathbb{R}^d$ in which case we say that the point set *spans* $\mathcal{P}$. A hyperplane $\mathbb{H} \subseteq \mathbb{R}^d$ is called *supporting* (for $\mathcal{P}$) if $\mathcal{P}$ is contained in one of the closed half-spaces induced by $\mathbb{H}$. A *face* of $\mathcal{P}$ is either $\mathcal{P}$ itself, or the intersection of $\mathcal{P}$ with a supporting hyperplane. If a face is neither the full polytope nor empty, we call it *proper*. A face of dimension ℓ is also called ℓ-face of $\mathcal{P}$, with the convention that the empty set is a (-1)-face. We call the union of all proper faces of $\mathcal{P}$ the *boundary* of $\mathcal{P}$, and the points of $\mathcal{P}$ not on the boundary the *interior* of $\mathcal{P}$. 0-faces are called the *vertices* of $\mathcal{P}$, and we let $V(\mathcal{P})$

denote the set of vertices. With ℓ being the dimension of $\mathcal{P}$, we call $(\ell-1)$-faces *facets*, and $(\ell-2)$-faces *ridges* of $\mathcal{P}$. Any face $\mathcal{F}$ of $\mathcal{P}$ is itself a polytope whose vertex set is $V(\mathcal{P}) \cap \mathcal{F}$.

It is well-known that every point $\mathbf{p} \in \mathcal{P}$ (and only those) can be written as a *convex combination* of vertices of $\mathcal{P}$, that is $\mathbf{p} = \sum_{\mathbf{v} \in V(\mathcal{P})} \lambda_{\mathbf{v}} \mathbf{v}$ with real values $\lambda_{\mathbf{v}} \geq 0$ for all $\mathbf{v}$ and $\sum_{\mathbf{v} \in V(\mathcal{P})} \lambda_{\mathbf{v}} = 1$. By Carathéodory's theorem, there exists a convex combination with at most $d+1$ non-zero entries, that is, $\mathbf{p} = \sum_{i=1}^{d+1} \lambda_i \mathbf{v}_i$ with $\mathbf{v}_i \in V(\mathcal{P})$, $\lambda_i \geq 0$ and $\sum \lambda_i = 1$.

An *ℓ-simplex* σ with $\ell \in \{-1, \ldots, d\}$ is a polytope of dimension ℓ that has exactly $\ell+1$ vertices. Every point in a simplex is determined by a unique convex combination of the vertices. A *simplicial complex* $\mathcal{C}$ in $\mathbb{R}^d$ is a set of simplices in $\mathbb{R}^d$ such that for a simplex σ in $\mathcal{C}$, all faces of σ are in $\mathcal{C}$ as well, and if σ and τ are in $\mathcal{C}$, the intersection $\sigma \cap \tau$ is a common face of both (note that the empty set is a face of any polytope). We let $V(\mathcal{C})$ denote the set of all vertices in $\mathcal{C}$. The *underlying space* $\bigcup \mathcal{C}$ of $\mathcal{C}$ is the union of its simplices. We call a simplex in $\mathcal{C}$ *maximal* if it is not a proper face of another simplex in $\mathcal{C}$. A simplicial complex equals the set of its maximal simplices together with all their faces and is therefore uniquely determined by its maximal simplices. Also, the underlying space of $\mathcal{C}$ equals the union of its maximal simplices.

In what follows, we let $\mathcal{P} := \mathrm{conv}(V)$ be the polytope spanned by V as fixed above. In particular, $\dim(\mathcal{P}) = d$ because V is full-dimensional and $V(\mathcal{P}) = V$ because V is in convex position.

2.2 Spines

We call $U \subseteq V$ with $|U| = n$ a *spine* of V if each facet of $\mathcal{P}$ contains at least $n-1$ points of U. Trivially, a one-point subset of V is a spine. If $\mathcal{P}$ is a simplex, any non-empty subset of vertices is a spine. For a hypercube, every pair of opposite vertices forms a spine, but no other spines with two or more elements exist.

We derive a geometric characterization of spines next. The *U-span (in V)* is the set of all d-simplices σ satisfying $U \subseteq V(\sigma) \subseteq V$. Equivalently, it is the set of all d-simplices with vertices in V that have $\mathrm{conv}(U)$ as a common face. Clearly, each simplex of the U-span is contained in $\mathcal{P}$, and the same is true for the union of all simplices in the U-span.

▶ **Lemma 1.** *Let $U \subseteq V$ with $|U| = n$. Then, U is a spine of V if and only if the union of all U-span simplices is equal to $\mathcal{P}$, that is, if every point in $\mathcal{P}$ belongs to at least one simplex in the U-span.*

Proof. We prove both directions of the equivalence separately. For "⇒", we proceed by induction on n. The statement is true for $n = 0$ by Carathéodory's theorem. Let U be a set with at least one element, $\mathbf{u} \in U$ arbitrary, and $\mathbf{p} \in \mathcal{P} \setminus \{\mathbf{u}\}$. The ray starting in $\mathbf{u}$ through $\mathbf{p}$ leaves the polytope in a point $\bar{\mathbf{p}}$, and this point lies on (at least) one facet $\mathcal{F}$ of $\mathcal{P}$ that does not contain $\mathbf{u}$. By assumption, $\mathcal{F}$ contains all points in $\bar{U} := U \setminus \{\mathbf{u}\}$. We claim that $\bar{U}$ is a spine of $\bar{V} := V \cap \mathcal{F}$. To see that, note that $\mathcal{F}$ is spanned by $\bar{V}$ and the facets of $\mathcal{F}$ (considered as a polytope in $\mathbb{R}^{d-1}$) are the ridges of $\mathcal{P}$ contained in $\mathcal{F}$. Every such ridge $\mathcal{R}$ is the intersection of $\mathcal{F}$ and another facet $\mathcal{F}'$ of $\mathcal{P}$. By assumption, $\mathcal{F}'$ also contains at least $n-1$ points of U and it follows at once that $\mathcal{R}$ contains $n-2$ points of $\bar{U}$. So, $\bar{U}$ is a spine of $\bar{V}$, and by induction hypothesis, there exists a $(d-1)$-simplex $\bar{\sigma}$ in the $\bar{U}$-span in $\bar{V}$ that contains $\bar{\mathbf{p}}$. The vertices of $\bar{\sigma}$ together with $\mathbf{u}$ span a simplex σ that contains $\mathbf{p}$ and σ is in the U-span by construction.

The direction "⇐" is clear if $n \in \{0, 1\}$, so we may assume that $n \geq 2$ and proceed by contraposition. If U is not a spine, we have a facet $\mathcal{F}$ of $\mathcal{P}$ such that less than $n-1$ points

of U lie on $\mathcal{F}$. Then, every simplex σ in the U-span has at least 2 vertices not on $\mathcal{F}$, and therefore at most $d-1$ vertices on $\mathcal{F}$. This implies that $\sigma \cap \mathcal{F}$ is at most $(d-2)$-dimensional. Therefore, the (finite) union of all U-span simplices cannot cover the $(d-1)$-dimensional facet $\mathcal{F}$, which means that the U-span is not equal to $\mathcal{P}$. ◀

From now on, we use the (equivalent) geometric characterization from the preceding lemma and the combinatorial definition of a spine interchangeably. A useful property is that spines extend to faces in the following sense.

▶ **Lemma 2.** *Let U be a spine of V, and let $\mathcal{F}$ be an ℓ-face of $\mathcal{P}$. Then $\bar{U} := U \cap \mathcal{F}$ is a spine of $\bar{V} := V \cap \mathcal{F}$, both considered as point sets in $\mathbb{R}^\ell$. In particular, $\mathcal{F}$ contains at least $n - (d - \ell)$ points of U.*

Proof. For every simplex σ in the U-span, let $\bar{\sigma} := \sigma \cap \mathcal{F}$. Clearly, $\bar{\sigma}$ is itself a simplex, spanned by the vertices $V(\bar{\sigma}) = V(\sigma) \cap \mathcal{F}$, and is of dimension at most ℓ. Because U is a spine of V, the union of all $\bar{\sigma}$ covers $\mathcal{F} = \operatorname{conv}(\bar{V})$. Moreover, $V(\bar{\sigma})$ contains $\bar{U}$. If $\bar{\sigma}$ is not of dimension ℓ, we can find an ℓ-simplex in the $\bar{U}$-span of $\bar{V}$ that has $\bar{\sigma}$ as a face just by adding suitable vertices from $\bar{V}$. This implies that the union of the $\bar{U}$-span covers $\mathcal{F}$. The "in particular" part follows by downward induction on ℓ. ◀

2.3 Star triangulations

Let $V' \subseteq \mathbb{R}^d$ be a finite point set that is full-dimensional, but not necessarily in convex position. We call a simplicial complex $\mathcal{C}$ a *triangulation* of V' if $V(\mathcal{C}) = V'$ and $\bigcup \mathcal{C} = \operatorname{conv}(V')$. In this case, we also call $\mathcal{C}$ a triangulation of the polytope $\operatorname{conv}(V')$. In a triangulation of V', every maximal simplex must be of dimension d.

We will consider several types of triangulations in this paper. For the first type, we assume that $V' = V \cup \{\mathbf{0}\}$, where $\mathbf{0} = (0, \ldots, 0) \notin V$, V is in convex position (as fixed before), and either $\mathbf{0}$ lies in $\operatorname{conv}(V)$ or V' is in convex position as well. We define a *star triangulation* of V' as a triangulation where all d-simplices contain $\mathbf{0}$ as a vertex. The elementary proof of the following result can be looked up at [23].

▶ **Lemma 3.** *A star triangulation of V' exists.*

2.4 Pulling triangulations

As usual, let V be a point set in convex position spanning a polytope $\mathcal{P}$ in $\mathbb{R}^d$, and let $\mathbf{p}_1 \in V$. We can describe a star triangulation with respect to $\mathbf{p}_1$ also as follows: Triangulate each facet of $\mathcal{P}$ that does not contain $\mathbf{p}_1$ such that the triangulations agree on their common boundaries. Writing $\Sigma := \{\sigma_1, \ldots, \sigma_m\}$ for the maximal simplices triangulating these facets, it is not difficult to see that a (star) triangulation of $\mathcal{P}$ is given by the maximal simplices

$$\mathbf{p}_1 * \Sigma := \{\mathbf{p}_1 * \sigma_1, \ldots, \mathbf{p}_1 * \sigma_m\},$$

where $\mathbf{v} * \sigma$ is the simplex spanned by $\mathbf{v}$ and the vertices of σ. Recursively star-triangulating the facets not containing $\mathbf{p}_1$ in the same way, this construction yields the *pulling triangulation*.

To define the triangulation formally, we fix a total order $\mathbf{p}_1 \prec \mathbf{p}_2 \prec \ldots \prec \mathbf{p}_n$ on V. For a single point, we set $\operatorname{Pull}(\{\mathbf{p}\}) := \{\mathbf{p}\}$. For any face $\mathcal{F}$ of $\mathcal{P}$ with positive dimension, let $\mathbf{p}_k$ denote the smallest vertex of $\mathcal{F}$ with respect to $\prec$. Then

$$\operatorname{Pull}(\mathcal{F}) := \mathbf{p}_k * \bigcup_{\substack{\mathcal{R} \text{ facet of } \mathcal{F} \\ \mathbf{p}_k \notin \mathcal{R}}} \operatorname{Pull}(\mathcal{R}).$$

The following result follows directly by induction on the dimension of the faces. See [4, Sec.5.6], [8, Lemma 4.3.6]:

▶ **Theorem 4.** Pull($\mathcal{P}$) *are the maximal simplices of a triangulation of* $\mathcal{P}$.

3 Spinal triangulations

Fix a simplicial complex $\mathcal{C}$ with vertex set $V(\mathcal{C})$ in $\mathbb{R}^d$ and a set $U \subseteq V(\mathcal{C})$ of size at most $d+1$. Let σ denote the simplex spanned by U. We call $\mathcal{C}$ *U-spinal* if every maximal simplex of $\mathcal{C}$ contains σ as a face. If $\mathcal{C}$ is a triangulation of V, we talk about a *U-spinal triangulation* accordingly. U-spinal triangulations are closely related to spines: if a U-spinal triangulation of V exists, then U is a spine of V, because all maximal simplices of the triangulation lie in the U-span. For the previously discussed spine of a hypercube consisting of two opposite points, also a spinal triangulation exists, consisting of $d!$ d-simplices that all share the diagonal connecting these points. This construction is called *staircase triangulation* [8] or *Freudenthal triangulation* [10].

We show next that a spine always induces a spinal triangulation. Let $\mathcal{P}$ be a polytope spanned by a finite full-dimensional point set $V \subseteq \mathbb{R}^d$ in convex position and let $U = \{\mathbf{u}_1, \ldots, \mathbf{u}_n\}$ be a spine of V. We fix a total order $\prec$ on V where $\mathbf{u}_1 \prec \mathbf{u}_2 \prec \ldots \prec \mathbf{u}_n$ are the n smallest elements, preceeding all points in $V \setminus U$.

▶ **Lemma 5.** *The pulling triangulation with respect to* $\prec$ *is a U-spinal triangulation.*

Proof. We prove the statement by induction on n. Let $\mathcal{T}$ denote the pulling triangulation with respect to $\prec$. For $n = 1$, every maximal simplex of $\mathcal{T}$ contains $\mathbf{u}_1$ by construction. For $n > 1$, every maximal simplex is a join of $\mathbf{u}_1$ with a $(d-1)$-simplex σ that is contained in a facet $\mathcal{F}$ of $\mathcal{P}$ that does not contain $\mathbf{u}_1$. σ, however, is itself a maximal simplex of the pulling triangulation of $\mathcal{F}$. By the spine property, $U' := \{\mathbf{u}_2, \ldots, \mathbf{u}_n\}$ are vertices of $\mathcal{F}$ and form a spine by Lemma 2. By induction, the pulling triangulation of $\mathcal{F}$ is U'-spinal. Therefore, σ contains all vertices of U', and the join with $\mathbf{u}_1$ contains all vertices of U. ◀

3.1 Folds and lifts

As before, let $\mathcal{P}$ be a polytope spanned by a finite full-dimensional point set $V \subseteq \mathbb{R}^d$ in convex position, $U \subseteq V$ a spine of V with n elements, and set $e := d - n + 1$. Assume without loss of generality that the origin is among the points in U. Furthermore let $\mathbb{A}_U$ be the subspace of $\mathbb{R}^d$ spanned by U. It is easy to see that the spine points are affinely independent, so that the dimension of $\mathbb{A}_U$ is $n-1$. Let $\mathbb{A}_U^\perp$ be the orthogonal complement, which is of dimension e. Let $\Phi_U : \mathbb{R}^d \to \mathbb{A}_U^\perp$ the (orthogonal) projection of $\mathbb{R}^d$ to $\mathbb{A}_U^\perp$. For notational convenience, we use the short forms $\hat{x} := \Phi_U(x)$ and $\hat{X} := \Phi_U(X)$ for the images of points and sets in $\mathbb{R}^d$.

Fix a U-spinal triangulation $\mathcal{T}$ and let σ be a maximal simplex of $\mathcal{T}$. Recall that the vertices of σ are the points of U, which all map to $\mathbf{0}$ under Φ_U, and e additional vertices $\mathbf{v}_1, \ldots, \mathbf{v}_e$. Hence, $\hat{\sigma}$ is the convex hull of $\{\mathbf{0}, \hat{\mathbf{v}}_1, \ldots, \hat{\mathbf{v}}_e\}$. Moreover, since σ has positive (d-dimensional) volume, its projection $\hat{\sigma}$ has positive (e-dimensional) volume as well. It follows that $\hat{\sigma}$ is a e-simplex spanned by $\{\mathbf{0}, \hat{\mathbf{v}}_1, \ldots, \hat{\mathbf{v}}_e\}$. Consequently, with $\sigma_1, \ldots, \sigma_t$ being the maximal simplices of $\mathcal{T}$, we call its *fold* the set of simplices consisting of $\hat{\sigma}_1, \ldots, \hat{\sigma}_m$ and all their faces. The following statement is a reformulation of [1, Prop.2.3] and [28, Prop.3.12]

▶ **Lemma 6.** *The fold of a U-spinal triangulation $\mathcal{T}$ is a star triangulation (with respect to the origin).*

Proof. All maximal simplices of the fold contain the origin by construction. Moreover, their union covers $\hat{\mathcal{P}}$ because $\mathcal{T}$ covers $\mathcal{P}$. Finally, we argue that the fold of two distinct maximal simplices σ_1, σ_2 cannot overlap: Let $\mathbb{H}$ be a hyperplane that separates σ_1 and σ_2. Since $\mathbb{A}_U$ is contained in the affine span of $\sigma_1 \cap \sigma_2$, $\mathbb{H}$ contains $\mathbb{A}_U$. Then, $\Phi_U(\mathbb{H})$ is a hyperplane in $\mathbb{R}^e$ which separates the two e-simplices $\hat{\sigma_1}$ and $\hat{\sigma_2}$. ◀

We will now define the converse operation to get from a star triangulation in $\mathbb{R}^e$ to a U-spinal triangulation in $\mathbb{R}^d$. We first show that pre-images of vertices are well-defined.

▶ **Lemma 7.** *If $\mathbf{v} \in V \setminus U$ then $\hat{\mathbf{v}} \neq \mathbf{0}$. Furthermore, if $\mathbf{v} \neq \mathbf{w} \in V \setminus U$ then $\hat{\mathbf{v}} \neq \hat{\mathbf{w}}$.*

Proof. Since U is a spine of V, there is a d-simplex σ in the U-span in V which has $\mathbf{v}$ among its vertices. If $\mathbf{v}$ is in the kernel of Φ_U, then $\mathbf{v}$ and the points in U (which span the kernel of Φ_U) are linearly dependent and σ cannot be full-dimensional, which is a contradiction.

For the second part, we assume to the contrary that $\mathbf{v} \neq \mathbf{w}$ but $\hat{\mathbf{v}} = \hat{\mathbf{w}}$. If there is also a d-simplex σ in the U-span which has both $\mathbf{v}$ and $\mathbf{w}$ among its vertices, $\hat{\sigma}$ cannot be full-dimensional, which is a contradiction. Otherwise, if no such d-simplex σ exists, the d-simplices incident to $\mathbf{v}$ triangulate a neighborhood of $\mathbf{v}$ within $\mathcal{P}$, and the same is true for $\mathbf{w}$, with the two sets of simplices being disjoint. It follows that their projections under Φ_U have to overlap, contradicting Lemma 6. ◀

For an e-simplex $\hat{\sigma} \subseteq \mathbb{R}^e$ with vertices in $\hat{V}$ and containing $\mathbf{0}$ as vertex, the *lifted* d-simplex $\sigma \subseteq \mathbb{R}^d$ is spanned by the pre-image of $V(\hat{\sigma})$ under $\Phi_U\big|_V$ (the restriction of Φ_U to V). Note the slight abuse of notation as we chose "$\hat{\sigma}$" as the name of a simplex before even defining the simplex σ, but the naming is justified because $\hat{\sigma}$ indeed is equal to $\Phi_U(\sigma)$ in this case. Given a star triangulation of $\hat{V}$, its *lift* is given by the set of lifts of its maximal simplices, together with all their faces.

Our goal is to show that the lift of a star triangulation is a U-spinal triangulation of $\mathcal{P}$. As a first step, we observe that such a lift is a U-spinal simplicial complex.

▶ **Lemma 8.** *The lift of a star triangulation of $\hat{V}$ is a U-spinal simplicial complex in $\mathbb{R}^d$ whose underlying space is a subset of $\mathcal{P}$.*

Proof. Fix a star triangulation $\hat{\mathcal{T}}$ and let $\mathcal{T}$ denote its lift. For any simplex in $\mathcal{T}$, all faces are included by construction. We need to show that for two simplices σ and τ in $\mathcal{T}$, $\sigma \cap \tau$ is a face of both. We can assume without loss of generality that σ and τ are maximal, hence d-simplices. By construction, σ and τ are the lifts of e-simplices $\hat{\sigma}$ and $\hat{\tau}$ in $\hat{\mathcal{T}}$. Clearly, $\hat{\sigma}$ and $\hat{\tau}$ intersect because they share the vertex $\mathbf{0}$. Moreover, since $\hat{\sigma}$ and $\hat{\tau}$ belong to a triangulation, their intersection is a common face, spanned by a set of vertices $\{\mathbf{0}, \hat{\mathbf{v}}_1 \ldots, \hat{\mathbf{v}}_k\}$. Hence, there exists a hyperplane $\hat{\mathbb{H}}$ in $\mathbb{R}^e$ separating $\hat{\sigma}$ and $\hat{\tau}$ such that $\hat{\sigma} \cap \hat{\mathbb{H}} = \operatorname{conv}\{\mathbf{0}, \hat{\mathbf{v}}_1 \ldots, \hat{\mathbf{v}}_k\} = \hat{\tau} \cap \hat{\mathbb{H}}$. Let $\mathbb{H}$ denote the preimage of $\hat{\mathbb{H}}$ under Φ_U. Then, $\mathbb{H}$ is a separating hyperplane for σ and τ, and $\sigma \cap \mathbb{H} = \operatorname{conv}\{\mathbf{u}_1, \ldots, \mathbf{u}_n, \mathbf{v}_1, \ldots, \mathbf{v}_k\} = \tau \cap \mathbb{H}$ as one can readily verify. This shows that the lift is a simplicial complex. Its underlying space lies in $\mathcal{P}$ because every lifted simplex does. It is U-spinal because the lift of every simplex contains U by definition. ◀

3.2 Volumes

The converse of Lemma 6 follows from the fact that all lifts of star triangulations have the same volume, as we will show next.

▶ **Lemma 9.** *Let σ be a simplex with vertices in V that contains $U = \{\mathbf{u}_1, \dots, \mathbf{u}_n\}$. Then*

$$\binom{d}{n-1}\mathrm{vol}(\sigma) = \mathrm{vol}(U)\mathrm{vol}(\hat{\sigma}),$$

where $\mathrm{vol}(U)$ denotes the volume of the simplex spanned by U.

Proof. For a k-simplex $\tau = \{v_0, \dots, v_k\}$, let $p(\tau)$ denote the parallelotope spanned by $v_1 - v_0, \dots, v_k - v_0$. It is well-known that $\mathrm{vol}(p(\tau)) = k!\mathrm{vol}(\tau)$. Rewriting the claimed volume by expanding the binomial coefficient and noting that $d - (n-1) = e$ yields

$$\underbrace{d!\mathrm{vol}(\sigma)}_{\mathrm{vol}(p(\sigma))} = \underbrace{(n-1)!\mathrm{vol}(U)}_{\mathrm{vol}(p(U))}\underbrace{e!\mathrm{vol}(\hat{\sigma})}_{\mathrm{vol}(p(\hat{\sigma}))}.$$

To prove the relation between the volumes of paralleotopes, we assume without loss of generality that $\mathbf{u}_1 = \mathbf{0}$. Let $\mathbb{A}_U$ denote the linear subspace spanned by $u_2, \dots, u_n$, and let $\mathbb{A}_U^{\mathbf{q}}$ denote the parallel affine subspace that contains $\mathbf{q} \in \mathbb{A}_U^{\perp}$. Then, $\mathrm{vol}(p(\sigma) \cap \mathbb{A}_U) = \mathrm{vol}(p(U))$ by definition. By Cavalieri's principle, every parallel cross-section of $p(\sigma)$ has the same volume. More precisely,

$$\mathrm{vol}(p(\sigma) \cap \mathbb{A}_U^{\mathbf{q}}) = \begin{cases} \mathrm{vol}(p(U)) & \text{if } \mathbf{q} \in p(\hat{\sigma}) \\ 0 & \text{otherwise} \end{cases}.$$

Using Fubini's theorem, the volume of $p(\sigma)$ can be expressed as an integral over all cross-sections, which yields

$$\mathrm{vol}(p(\sigma)) = \int_{\mathbf{q} \in \mathbb{A}_U^{\perp}} \mathrm{vol}(p(\sigma) \cap \mathbb{A}_U^{\mathbf{q}})dq = \int_{\mathbf{q} \in p(\hat{\sigma})} \mathrm{vol}(p(U))dq = \mathrm{vol}(p(U))\mathrm{vol}(p(\hat{\sigma})). \quad \blacktriangleleft$$

▶ **Lemma 10.** *Let U be a spine of V, and $\mathcal{T}$ denote the lift of a star triangulation of $\hat{V}$ Then, with the notation of Lemma 9,*

$$\binom{d}{n-1}\mathrm{vol}\left(\bigcup \mathcal{T}\right) = \mathrm{vol}(U)\mathrm{vol}(\hat{\mathcal{P}}),$$

where $\bigcup \mathcal{T}$ is the underlying space of $\mathcal{T}$. In particular, the underlying spaces of the lifts of all star triangulations of $\hat{V}$ have the same volume.

Proof. The statement follows directly from Lemma 9 because the relation holds for any simplex in the star triangulation and its lift. ◀

With that, we can prove our first main theorem.

▶ **Main Theorem 1** (Lifting theorem)**.** *Let $\mathcal{P}$ be a polytope spanned by a full-dimensional finite point set $V \subseteq \mathbb{R}^d$ in convex position and $U := \{\mathbf{u}_1, \dots, \mathbf{u}_n\} \subseteq V$. Then,*

U is a spine of V if and only if there exists a U-spinal triangulation of V.

In this case, for $n \geq 1$, the U-spinal triangulations of V are exactly the lifts of the star triangulations of $\hat{V}$, the orthogonal projection of V to the orthogonal complement of the affine space spanned by U. Furthermore, if $n \geq 2$,

$$\binom{d}{n-1}\mathrm{vol}(\mathcal{P}) = \mathrm{vol}(U)\mathrm{vol}(\hat{\mathcal{P}}).$$

Proof. For the equivalence, the "if"-part follows directly from the geometric characterization of spines (Lemma 1), and the "only if" part follows from Lemma 5.

For the second part, given any U-spinal triangulation $\mathcal{T}^*$, its fold $\hat{\mathcal{T}}^*$ is a star triangulation by Lemma 6, and by lifting that star triangulation, we obtain $\mathcal{T}^*$ back. Vice versa, starting with any star triangulation $\hat{\mathcal{T}}$, we know that its lift is a simplicial complex contained in $\mathcal{P}$ by Lemma 8. By Lemma 10, the lifts of $\hat{\mathcal{T}}$ and $\hat{\mathcal{T}}^*$ have the same volume, but the lift of the latter is $\mathcal{P}$. It follows that also the lift of the former is a triangulation of $\mathcal{P}$, proving the second claim.

The claim about the volumes follows at once by applying Lemma 10 on $\mathcal{T}^*$ and $\hat{\mathcal{T}}^*$. ◀

We remark that not every U-spinal triangulation is a pulling triangulation. This follows from the fact that each pulling triangulation is *regular* (see [8, p.181]), but examples of non-regular spinal triangulations are known (one such example is given in [8, p.306]).

4 The Everest polytope

For $n, s \in \mathbb{N}$, define $\mathcal{E}_{n,s} := \{\mathbf{x} \in \mathbb{R}^{ns} \mid g_{n,s}(\mathbf{x}) \leq 1\}$ where $g_{n,s} : \mathbb{R}^{ns} \to \mathbb{R}$ with

$$(x_{1,1}, \ldots, x_{n,s}) \mapsto \sum_{j=1}^{s} \max\{0, x_{1,j}, \ldots, x_{n,j}\} + \max\left\{0, -\sum_{j=1}^{s} x_{1,j}, \ldots, -\sum_{j=1}^{s} x_{n,j}\right\}.$$

It is not difficult to verify that $\mathcal{E}_{n,s}$ is bounded and the intersection of finitely many halfspaces of $\mathbb{R}^{ns}$. We call it the (n, s)-*Everest polytope*. It is well-known [3] that the number-theoretic constant $c(n, s)$ discussed in the introduction is equal to the volume of $\mathcal{E}_{n,s}$.

4.1 Vertex sets

In order to describe the vertices of $\mathcal{E}_{n,s}$ we introduce the following point sets which we also utilize in later parts of the paper. Note that we identify $\mathbb{R}^{ns}$ and $\mathbb{R}^{n\times s}$ which explains the meaning of "row" and "column" in the definition. Let $\mathbf{e}_s(i)$ denote the i-th s-dimensional unit (row) vector with the convention that $\mathbf{e}_s(0) = \mathbf{0}$. We define the following sets in $\mathbb{R}^{ns}$:

$$V_{n,s} := \left\{ \begin{pmatrix} -\mathbf{e}_s(j_1) \\ \vdots \\ -\mathbf{e}_s(j_n) \end{pmatrix} \;\middle|\; j_1, \ldots, j_n \in \{0, \ldots, s\} \right\},$$

$$U_{n,s} := \left\{ \begin{pmatrix} -\mathbf{e}_s(j) \\ \vdots \\ -\mathbf{e}_s(j) \end{pmatrix} \;\middle|\; j \in \{0, \ldots, s\} \right\},$$

$$P_{n,s} := V_{n,s} - (U_{n,s} \setminus \{\mathbf{0}\}) = \{\mathbf{v} - \mathbf{u} \mid \mathbf{v} \in V_{n,s} \wedge \mathbf{u} \in U_{n,s} \setminus \{\mathbf{0}\}\}.$$

It can be readily verified that $V_{n,s}$ is the set of points in $\{-1, 0\}^{ns}$ such that there is at most one -1 per row, $U_{n,s}$ is the set of points in $V_{n,s}$ such that all -1's (if there are any) are contained in a single column, and $P_{n,s}$ is the set of points in $\{-1, 0, 1\}^{ns}$ such that

- all '1's (if there are any) are in a unique "1-column",
- all entries of the 1-column are either 0 or 1,
- all rows with a 1 contain at most one -1,
- all rows without a 1 contain only '0's.

▶ **Lemma 11.** *$P_{n,s} \cap V_{n,s} = \{\mathbf{0}\}$, $U_{n,s} \subseteq V_{n,s}$, $|V_{n,s}| = (s+1)^n$, $|U_{n,s}| = s+1$, and $|P_{n,s}| = s(s+1)^n - s + 1$.*

Proof. Follows directly from the definitions and from basic combinatorics. ◀

▶ **Theorem 12.** *The set of vertices of $\mathcal{E}_{n,s}$ is given by $(P_{n,s} \cup V_{n,s}) \setminus \{\mathbf{0}\} = (V_{n,s} - U_{n,s}) \setminus \{\mathbf{0}\}$.*

We split the proof into several parts which will be treated in the following lemmas. For the rest of this section, let i and j (with a possible subscript) denote elements of $\{1, \dots, n\}$ and $\{1, \dots, s\}$, respectively, let $\mathbf{v} = (v_{1,1}, \dots, v_{n,s})$ be a vertex of $\mathcal{E}_{n,s}$, and set

$$m_j := \max\{0, v_{1,j}, \dots, v_{n,j}\} \text{ for all } j,$$
$$s_i := -\sum_{j=1}^{s} v_{i,j} \text{ for all } i,$$
$$m := \max\{0, s_1, \dots, s_n\}.$$

Then

$$g_{n,s}\begin{pmatrix} v_{1,1} & \cdots & v_{1,s} \\ \vdots & & \vdots \\ v_{n,1} & \cdots & v_{n,s} \end{pmatrix} = \max\begin{Bmatrix} 0 \\ v_{1,1} \\ \vdots \\ v_{n,1} \end{Bmatrix} + \cdots + \max\begin{Bmatrix} 0 \\ v_{1,s} \\ \vdots \\ v_{n,s} \end{Bmatrix} + \max\begin{Bmatrix} 0 \\ -v_{1,1} - \cdots - v_{1,s} \\ \vdots \\ -v_{n,1} - \cdots - v_{n,s} \end{Bmatrix}$$
$$= m_1 + \cdots + m_s + m = 1.$$

Furthermore it can easily be verified that $v_{i,j} \in [-1, 1]$ for all i and j, $m_j \in [0, 1]$ for all j, $s_i \in [-1, 1]$ for all i, and $m \in [0, 1]$.

In the proofs below, we will repeatedly apply the following argument: if there is an $\varepsilon > 0$ and an $\mathbf{x} \in \mathbb{R}^{ns}$ such that $\mathbf{v} \pm \delta\mathbf{x} \in \mathcal{E}_{n,s}$ for all $\delta \in [0, \varepsilon]$, $\mathbf{v}$ cannot be a vertex of $\mathcal{E}_{n,s}$ (since it is in the interior of an at least 1-dimensional face).

▶ **Lemma 13.** *If $m_j = 0$ for all j then $\mathbf{v} \in V_{n,s}$.*

Proof. Since all m_j are equal to zero, all $v_{i,j}$ have to be non-positive, so all s_i are non-negative. Also we get that m is equal to 1 which implies that at least one of the s_i is equal to 1. Suppose that $v_{i_0,j_0} \in (-1, 0)$ for some i_0, j_0. If $s_{i_0} < 1$, then for $\varepsilon := \min\{-v_{i_0,j_0}, 1 - s_{i_0}\} > 0$ and $\delta \in [0, \varepsilon]$ we get that $g_{n,s}(v_{1,1}, \dots, v_{i_0,j_0} \pm \delta, \dots, v_{n,s}) = 1$, so $(v_{1,1}, \dots, v_{i_0,j_0} \pm \delta, \dots, v_{n,s})$ is on the boundary of $\mathcal{E}_{n,s}$ and $\mathbf{v}$ cannot be a vertex of $\mathcal{E}_{n,s}$.

If on the other hand s_{i_0} is equal to 1, then there is a $j_1 \neq j_0$ such that $v_{i_0,j_1} \in (-1, 0)$. But then we get $g_{n,s}(v_{1,1}, \dots, v_{i_0,j_0} \pm \delta, \dots, v_{i_0,j_1} \mp \delta, \dots, v_{n,s}) = 1$ where we set $\varepsilon := \min\{-v_{i_0,j_0}, -v_{i_0,j_1}, v_{i_0,j_0} + 1, v_{i_0,j_1} + 1\} > 0$ and $\delta \in [0, \varepsilon]$, so again $\mathbf{v}$ cannot be a vertex of $\mathcal{E}_{n,s}$.

Thus we get that all $v_{i,j}$ are either -1 or 0 and it is clear that in any given row i_0 only one of the $v_{i_0,j}$ can be -1 (as they sum up to $-s_{i_0} \geq -1$). Therefore $\mathbf{v} \in V_{n,s}$ (see [23] for an extended proof). ◀

▶ **Lemma 14.** *If $m_{j_0} = 1$ for some j_0 then $\mathbf{v} \in P_{n,s}$.*

Proof. Since m_{j_0} is equal to 1, all other m_j and m have to be equal to zero. Thus all v_{i,j_0} are non-negative and at least one of them is equal to 1. Also, all other $v_{i,j}$ (i.e. if $j \neq j_0$) are non-positive and so are all s_i. Just as in the proof of Lemma 13 we can show that all $v_{i,j}$ are either -1, 0, or 1; we omit the details. Furthermore it is clear that if v_{i_0,j_0} is equal to 1 for

some i_0, then there cannot be more than one -1 in the i_0-th row (as $s_{i_0} \leq 0$). By the same reasoning, if v_{i_0,j_0} is equal to 0, there cannot be any -1 in the i_0-th row at all. Considering the definition of $P_{n,s}$ we thus see that $\mathbf{v} \in P_{n,s}$. ◀

▶ **Lemma 15.** *If $m_j \neq 1$ for all j then $m_j = 0$ for all j.*

Proof. The proof works similar to that of Lemma 13; see [23]. ◀

Proof of Theorem 12. Lemma 15 implies that if $\mathbf{v}$ is a vertex of $\mathcal{E}_{n,s}$, then we are in the situation of either Lemma 13 or Lemma 14, hence $\mathbf{v} \in P_{n,s} \cup V_{n,s}$. Furthermore it is clear that $\mathbf{0}$ is not a vertex of $\mathcal{E}_{n,s}$. Also, it follows from the definition of $P_{n,s}$ that

$$E_{n,s} := (P_{n,s} \cup V_{n,s}) \setminus \{\mathbf{0}\} = ((V_{n,s} - (U_{n,s} \setminus \{\mathbf{0}\})) \cup V_{n,s}) \setminus \{\mathbf{0}\} = (V_{n,s} - U_{n,s}) \setminus \{\mathbf{0}\}\,.$$

We are left to show that $E_{n,s} \subseteq V(\mathcal{E}_{n,s})$. First we observe that $g_{n,s}(\mathbf{v}) = 1$ for all $\mathbf{v} \in E_{n,s}$. We consider the case that $n, s \geq 2$ and assume that there is a $\mathbf{v} \in E_{n,s}$ that is not a vertex of $\mathcal{E}_{n,s}$. Since $\mathbf{v}$ is on the boundary of $\mathcal{E}_{n,s}$ but not a vertex of $\mathcal{E}_{n,s}$, it is contained in the interior of an at least 1-dimensional face $\mathcal{F}$ of $\mathcal{E}_{n,s}$. Let $\mathbf{w}$ be any vertex of $\mathcal{F}$. Then $\mathbf{w} \in E_{n,s}$ and $\mathbf{v} \neq \mathbf{w}$.

Now let $\mathbf{a} \in E_{2,2}$, $\mathbf{a} \neq \mathbf{b} \in E_{2,2} \cup \{\mathbf{0}\}$, and consider the convex combination $\alpha\mathbf{a} + (1-\alpha)\mathbf{b}$, $\alpha \in \mathbb{R}$. By plugging in all possible values of $\mathbf{a}$ and $\mathbf{b}$ one can verify that if $\alpha > 1$ then $g_{2,2}(\alpha\mathbf{a} + (1-\alpha)\mathbf{b}) > 1$.

Let $\mathbf{v}', \mathbf{w}' \in \mathbb{R}^{2\times s}$ be submatrices consisting of 2 rows of $\mathbf{v}$ and $\mathbf{w}$ respectively, such that $\mathbf{v}' \neq \mathbf{0}$ and $\mathbf{v}' \neq \mathbf{w}'$. By definition of $E_{n,s}$, $\mathbf{v}'$ and $\mathbf{w}'$ thus respectively contain submatrices of the form $\mathbf{a}$ and $\mathbf{b}$ from above while the remaining entries are padded with zeros. It follows from the definition of $g_{n,s}$ that $g_{n,s}(\alpha\mathbf{v} + (1-\alpha)\mathbf{w}) > 1$ if $\alpha > 1$, which contradicts the fact that $\mathbf{v}$ is in the interior of $\mathcal{F}$. Hence, $E_{n,s} \subseteq V(\mathcal{E}_{n,s})$ if $n, s \geq 2$. If $n = 1$ and $s \geq 2$ one can proceed analogously by considering $\mathbf{a} \in E_{1,2}$, $\mathbf{a} \neq \mathbf{b} \in E_{1,2} \cup \{\mathbf{0}\}$; same goes for $s = 1$ and $\mathbf{a} \in E_{1,1}$, $\mathbf{a} \neq \mathbf{b} \in E_{1,1} \cup \{\mathbf{0}\}$. ◀

▶ **Corollary 16.** *The number of vertices of $\mathcal{E}_{n,s}$ is given by $(s+1)^{n+1} - s - 1$.*

Proof. Follows directly from Theorem 12 and Lemma 11. ◀

5 Projections of simplotopes

We will establish a relation between the Everest polytope $\mathcal{E}_{n,s}$ and a special polytope known as simplotope. This relation will allow the comparison of the volumes of the two polytopes even though they are of different dimension.

5.1 Simplotopes

For $s \in \mathbb{N}$, the s-simplex Δ_s is spanned by the points $(\mathbf{0}, -\mathbf{e}_s(1), \ldots, -\mathbf{e}_s(s))$ in $\mathbb{R}^s$, with $\mathbf{e}_s(i)$ the i-th standard vector in $\mathbb{R}^s$, as before. A *simplotope* is a Cartesian product of the form $\Delta_{s_1} \times \ldots \times \Delta_{s_n}$ with positive integers $s_1, \ldots, s_n$. Note that in the literature, simplotopes are usually defined in a combinatorially equivalent way using the standard s-simplex spanned by $(s+1)$-unit vectors in $\mathbb{R}^{s+1}$. We restrict to the case that all s_i are equal, and we call

$$\mathcal{S}_{n,s} = \underbrace{\Delta_s \times \ldots \times \Delta_s}_{n \text{ times}}$$

the (n,s)-*simplotope* for $n, s \in \mathbb{N}$.

For instance, d-hypercubes are d-fold products of line segments, and therefore (n,s)-simplotopes with $n = d$ and $s = 1$. It is instructive to visualize a point in $\mathcal{S}_{n,s}$ as an $n \times s$-matrix with real entries in $[0,1]$, where the sums of the entries in each row do not exceed 1. One can readily verify that the set of vertices of the (n,s)-simplotope is equal to $V_{n,s}$, as given in the beginning of Section 4. Moreover, it is straight-forward to verify that each facet of $\mathcal{S}_{n,s}$ is given by

$$\underbrace{\Delta_s \times \ldots \times \Delta_s}_{i \text{ times}} \times \mathcal{F} \times \underbrace{\Delta_s \times \ldots \times \Delta_s}_{n-i-1 \text{ times}}$$

where $\mathcal{F}$ is a facet of Δ_s and $0 \leq i \leq n-1$. It follows at once:

▶ **Theorem 17.** *$U_{n,s}$ is a spine of $V_{n,s} = V(\mathcal{S}_{n,s})$.*

5.2 A linear transformation

We call the matrix

$$\Pi_{n,s} := \begin{pmatrix} & -I_s \\ I_{ns} & \vdots \\ & -I_s \end{pmatrix} \in \mathbb{R}^{(ns)\times((n+1)s)},$$

where I_d is the identity matrix of dimension d, the (n,s)-*SE-transformation* ("SE" stands for "**S**implotope $\leftrightarrow$ **E**verest polytope"). We show that the name is justified, as it maps the $(n+1,s)$-simplotope onto the (n,s)-Everest polytope. See [23] for a proof.

▶ **Theorem 18.** *$\Pi_{n,s}(V(\mathcal{S}_{n+1,s}) \setminus U_{n+1,s}) = V(\mathcal{E}_{n,s})$, $U_{n+1,s} \setminus \{\mathbf{0}\}$ is a basis of $\ker(\Pi_{n,s})$ (in particular, $\Pi_{n,s}U_{n+1,s} = \{\mathbf{0}\}$), and $\mathbf{0}$ is contained in the interior of $\mathcal{E}_{n,s}$. In particular, $\Pi_{n,s}\mathcal{S}_{n+1,s} = \mathcal{E}_{n,s}$.*

We are now ready to prove a formula for the Everest polytope.

▶ **Main Theorem 2.** *The volume of the (n,s)-Everest polytope is given by*

$$\mathrm{vol}(\mathcal{E}_{n,s}) = \frac{((n+1)s)!}{(ns)!(s!)^{n+1}}.$$

Proof. We want to apply the Lifting Theorem (Main Theorem 1) with $\mathcal{P} \leftarrow \mathcal{S}_{n+1,s}$, $\hat{\mathcal{P}} \leftarrow \mathcal{E}_{n,s}$, $d \leftarrow (n+1)s$ and $e \leftarrow ns$. However, a minor modification is needed, because the SE-transformation $\Pi_{n,s}$ is not a projection matrix. So, let $\tilde{\Pi}_{n,s} := \begin{pmatrix} \Pi_{n,s} \\ 0\ I_s \end{pmatrix} \in \mathbb{R}^{((n+1)s)\times((n+1)s)}$, $\Pi := (I_{ns}\ 0) \in \mathbb{R}^{(ns)\times((n+1)s)}$, and let $\tilde{\mathcal{S}}_{n+1,s} := \tilde{\Pi}_{n,s}\mathcal{S}_{n+1,s}$ denote the transformed simplotope. Clearly, $\mathrm{vol}(\tilde{\mathcal{S}}_{n+1,s}) = \mathrm{vol}(\mathcal{S}_{n+1,s})$, $\Pi\tilde{\mathcal{S}}_{n+1,s} = \mathcal{E}_{n,s}$, and the transformed spine points $\tilde{U}_{n+1,s} := \tilde{\Pi}_{n,s}U_{n+1,s}$ span the kernel of Π. Using Main Theorem 1 on $\tilde{\mathcal{S}}_{n+1,s}$ and $\mathcal{E}_{n,s}$, we obtain

$$\binom{(n+1)s}{ns}\mathrm{vol}(\tilde{\mathcal{S}}_{n+1,s}) = \mathrm{vol}(\tilde{U}_{n+1,s})\mathrm{vol}(\mathcal{E}_{n,s}).$$

Furthermore, $\mathrm{vol}(\tilde{U}_{n+1,s}) = \frac{1}{s!}$ since $\tilde{U}_{n+1,s} = \{(\mathbf{0},\ldots,\mathbf{0},-\mathbf{e}_s(j)) \mid j \in \{0,\ldots,s\}\}$. Moreover, $\mathcal{S}_{n+1,s}$ is the $(n+1)$-fold product of simplices Δ_s spanned by $\mathbf{0}$ and s unit vectors. Hence, the volume of Δ_s is $1/s!$, and by Fubini's theorem,

$$\mathrm{vol}(\tilde{\mathcal{S}}_{n+1,s}) = \mathrm{vol}(\mathcal{S}_{n+1,s}) = (\mathrm{vol}(\Delta_s))^{n+1} = \frac{1}{(s!)^{n+1}}.$$

Plugging in the formulas for $\mathrm{vol}(\tilde{U}_{n+1,s})$ and $\mathrm{vol}(\tilde{\mathcal{S}}_{n+1,s})$ into the formula given by the lifting theorem the claim follows by rearranging terms. ◀

6 Conclusions and further remarks

Main Theorem 1 combines several new results. It answers the question of the existence of a triangulation of a polytope under the constraint that a given subset of the vertices of the polytope must be contained in every maximal simplex of the triangulation. Furthermore, it characterizes all such triangulations and provides a method to compute one (or all) efficiently from the lift of a star triangulation. Finally, it generalizes the well-known relation $\mathrm{vol}(AM) = |\det(A)|\,\mathrm{vol}(M)$ where M is a measurable subset of $\mathbb{R}^d$ and $A \in \mathbb{R}^{d\times d}$ to certain cases where A is not a square matrix. In particular, it allows us to express the volume of an object in $\mathbb{R}^d$ in terms of the volume its "shadow" in $\mathbb{R}^e$, and vice versa.

The shadow that a cube in $\mathbb{R}^3$ casts if the light shines parallel to any of its space diagonals is a regular hexagon. Assuming a cube with side length ℓ, the theorem implies that the volume of the cube and the volume (area) of its shadow (the hexagon) differ by a factor of $\sqrt{3}/\ell$ which provides an alternative method to compute the volume of a hexagon from the volume of a cube. By lifting the "complicated" hexagon to a higher dimensional space it gains more symmetries and becomes the comparatively simple cube. In the same fashion, the complicated Everest polytope is the shadow of the simpler simplotope which allowed the computation of its volume in Main Theorem 2.

Starting with a polytope and a spine, it is easy to determine the volume of the "shadow" with respect to the spine using our theorem. On the other hand, there is no easy way to tell if a given shape is the shadow of some higher dimensional object and in the case of the Everest polytope, this is the interesting direction. We pose the question of whether other polytopes (e.g., the Birkhoff polytope) can be expressed as shadows of other polytopes. For that purpose, it might be worthwhile to find general methods or at least good heuristics to determine if a complicated shape can be recognized as the shadow of some simpler object.

Acknowledgements. We thank Raman Sanyal, Volkmar Welker, and the anonymous referees for helpful discussions and recommendations that have led to significant simplifications of our exposition.

References

1 C. Athanasiadis. Ehrhart polynomials, simplicial polytopes, magic squares and a conjecture of Stanley. *J. Reine Angew. Math.*, 583:163–174, 2005.

2 A. Baker and G. Wüstholz. Logarithmic forms and group varieties. *J. Reine Angew. Math.*, 442:19–62, 1993.

3 F. Barroero, C. Frei, and R. Tichy. Additive unit representations in rings over global fields – a survey. *Publ. Math. Debrecen*, 79(3–4):291–307, 2011.

4 M. Beck and R. Sanyal. *Combinatorial Reciprocity Theorems.* American Mathematical Society, 2016. In preparation, available at `http://math.sfsu.edu/beck/crt.html`.

5 W. Bruns and T. Römer. h-vectors of Gorenstein polytopes. *J. Combin. Theory Ser. A*, 114:65–76, 2007.

6 H. Croft, K. Falconer, and R. Guy. *Unsolved Problems in Geometry.* Springer, 1991.

7 J. de Loera, F. Liu, and R. Yoshida. A generating function for all semi-magic squares and the volume of the Birkhoff polytope. *J. Algebraic Combin.*, pages 113–139, 2009.

8 J. de Loera, J. Rambau, and F. Santos. *Triangulations.* Springer, 2010.

9 T. de Wolff. Polytopes with special simplices. arXiv:1009.6158.

10 H. Edelsbrunner and M. Kerber. Dual complexes of cubical subdivisions of $\mathbb{R}^n$. *Discrete Comput. Geom.*, 47(2):393–414, 2012.

11 I. Emiris and V. Fisikopoulos. Efficient random-walk methods for approximating polytope volume. In *Proc. of the 13th Annual Symp. on Comp. Geom.*, SOCG'14, pages 318:318–318:327, 2014.

12 G. R. Everest. A "Hardy-Littlewood" approach to the S-unit equation. *Compos. Math.*, 70(2):101–118, 1989.

13 G. R. Everest. Counting the values taken by sums of S-units. *J. Number Theory*, 35(3):269–286, 1990.

14 J. Evertse and H. P. Schlickewei. A quantitative version of the absolute subspace theorem. *J. Reine Angew. Math.*, 548:21–127, 2002.

15 C. Frei, R. Tichy, and V. Ziegler. On sums of S-integers of bounded norm. *Monatsh. Math.*, 175(2):241–247, 2014.

16 H. Freudenthal. Simplizialzerlegung beschränkter Flachheit. *Ann. of Math.*, pages 580–582, 1942.

17 R. Freund. Combinatorial theorems on the simplotope that generalize results on the simplex and cube. *Math. Oper. Res.*, 11(1):169–179, 1986.

18 I. Gelfand, M. Kapranov, and A. Zelevinsky. *Discriminants, Resultants and Multidimensional Determinants*. Birkhäuser, 2008.

19 K. Győry and K. Yu. Bounds for the solutions of S-unit equations and decomposable form equations. *Acta Arith.*, 123(1):9–41, 2006.

20 H. Hadwiger. *Vorlesungen über Inhalt, Oberfläche und Isoperimetrie*. Springer, 1957.

21 T. Hibi and H. Ohsugi. Special simplices and Gorenstein toric rings. *J. Combin. Theory Ser. A*, 113:718–725, 2006.

22 R. Hughes and M. Anderson. Simplexity of the cube. *Discrete Math.*, 158:99–150, 1996.

23 M. Kerber, R. Tichy, and M. Weitzer. Constrained triangulations, volumes of polytopes, and unit equations. *arXiv, 1609.05017*, 2016.

24 J. Matoušek. *Lectures in Discrete Geometry*. Springer, 2002.

25 J. Neukirch. *Algebraic number theory*, volume 322 of *Grundlehren der mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences]*. Springer, 1999.

26 K. Nishioka. Algebraic independence by Mahler's method and S-unit equations. *Compositio Math.*, 92(1):87–110, 1994.

27 I. Pak. Four questions on Birkhoff polytope. *Ann. Comb.*, 4:83–90, 2000.

28 V. Reiner and V. Welker. On the Charney-Davis and Neggers-Stanley conjectures. *J. Combin. Theory Ser. A*, 109(2):247–280, 2005.

29 F. Santos. A counterexample to the Hirsch conjecture. *Ann. of Math.*, 176:383–412, 2012.

30 H. P. Schlickewei. S-unit equations over number fields. *Invent. Math.*, 102(1):95–108, 1990.

31 J. Shewchuk. General-dimensional constrained Delaunay and constrained regular triangulations, I: Combinatorial properties. *Discrete Comput. Geom.*, 39:580–637, 2008.

32 G. van der Laan and A. Talman. On the computation of fixed points in the product space of unit simplices and an application to noncooperative n person games. *Math. Oper. Res.*, 7(1):1–13, 1982.

33 G. Ziegler. *Lectures on Polytopes*. Springer, 2007.

Proper Coloring of Geometric Hypergraphs*

Balázs Keszegh[1] and Dömötör Pálvölgyi[2]

1 **Alfréd Rényi Institute of Mathematics, Hungarian Academy of Sciences, Budapest, Hungary**
keszegh@renyi.hu

2 **Department of Pure Mathematics and Mathematical Statistics, University of Cambridge, UK**
dom@cs.elte.hu

Abstract

We study whether for a given planar family $\mathcal{F}$ there is an m such that any finite set of points can be 3-colored such that any member of $\mathcal{F}$ that contains at least m points contains two points with different colors. We conjecture that if $\mathcal{F}$ is a family of pseudo-disks, then $m = 3$ is sufficient. We prove that when $\mathcal{F}$ is the family of all homothetic copies of a given convex polygon, then such an m exists. We also study the problem in higher dimensions.

1998 ACM Subject Classification G.2.2 [Graph Theory] Hypergraphs

Keywords and phrases discrete geometry, decomposition of multiple coverings, geometric hypergraph coloring

Digital Object Identifier 10.4230/LIPIcs.SoCG.2017.47

1 Introduction

In the present paper, we primarily focus on the following proper coloring problem. Given a finite set of points in the plane, S, we want to color the points of S with a small number of colors such that every member of some given geometric family $\mathcal{F}$ that intersects S in many points will contain at least two different colors.

Pach conjectured in 1980 [28] that for every convex set D there is an m such that any finite set of points admits a 2-coloring such that any translate of D that contains at least m points contains both colors. This conjecture inspired a series of papers studying the problem and its variants – for a recent survey, see [31]. Eventually, the conjecture was shown to hold in the case when D is a convex polygon in a series of papers [29, 41, 37], but disproved in general [30]. In fact, the conjecture fails for any D with a smooth boundary, e.g., for a disk.

It follows from basic properties of generalized Delaunay triangulations (to be defined later) and the Four Color Theorem that for any convex D it is possible to 4-color any finite set of points such that any *homothetic copy*[1] of D that contains at least two points will contain at least two colors. Therefore, the only case left open is when we have 3 colors. We conjecture that for 3 colors the following holds.

▶ **Conjecture 1.** *For every plane convex set D there is an m such that any finite set of points admits a 3-coloring such that any homothetic copy of D that contains at least m points contains two points with different colors.*

* First author is supported by the Hungarian National Research, Development and Innovation Office – NKFIH under the grant K 116769. Second author is supported by the Marie Skłodowska-Curie action of the EU, under grant IF 660400.

[1] A homothetic copy or homothet of a set is a scaled and translated copy of it (rotations are *not* allowed).

The special case of Conjecture 1 when D is a disk has been posed earlier by the first author [17], and is also still open. Our main result is the proof of Conjecture 1 for convex polygons.

▶ **Theorem 2.** *For every convex n-gon D there is an m such that any finite set of points admits a 3-coloring such that any homothetic copy of D that contains at least m points contains two points with different colors.*

We would like to remark that the constructions from [30] do not exclude the possibility that for convex polygons the strengthening of Theorem 2 using only 2 colors instead of 3 might also hold; this statement is known to hold for triangles [19] and squares[2] [1].

The constant m which we get from our proof depends not only on the number of sides, but also on the shape of the polygon. However, we conjecture that this dependence can be removed, and in fact the following stronger conjecture holds for any *pseudo-disk arrangement*. We define a pseudo-disk arrangement as a family of planar bodies whose boundaries are Jordan curves such that any member of the family intersects the boundary of any other member in a connected curve.[3]

▶ **Conjecture 3.** *For any pseudo-disk arrangement any finite set of points admits a 3-coloring such that any pseudo-disk that contains at least 3 points contains two points with different colors.*

This conjecture also has a natural dual counterpart.

▶ **Conjecture 4.** *The members of any pseudo-disk arrangement admit a 3-coloring such that any point that is contained in at least 3 pseudo-disks is contained in two pseudo-disks with different colors.*

We believe that these are fundamental problems about geometric hypergraphs, and find it quite surprising that they have not been studied much.

The rest of this paper is organized as follows. In the rest of this section, we give an overview of related results. In Section 2 we give the definition and basic properties of generalized Delaunay triangulations. In Section 4 we prove Theorem 2, using the proof method of [1]. In Section 5 we study the higher dimensional variants of the problem and present some constructions. In Section 6 we briefly discuss further related topics.

Previous results

Most earlier papers on colorings and geometric ranges focused not on proper colorings, but on *polychromatic colorings* and its dual, *cover-decomposition*. In the polychromatic k-coloring problem our goal is to color the points of some finite S with k colors such that every member of some family $\mathcal{F}$ that contains many points of S contains *all* k colors. Gibson and Varadarajan [13] have shown that for every convex polygon D there is a c_D such that every finite set of points can be k-colored such that any translate of D that contains at least $m_k = c_D k$ points contains all k colors. Whether such a polychromatic k-coloring exists for homothetic copies of convex polygons for any m_k is an open problem, which would be a significant strengthening of our Theorem 2. This conjecture has only been proved in a

[2] And since affine transformation have no effect on the question, also for parallelograms.

[3] This is slightly non-standard, as usually it is assumed that any two boundaries intersect at most twice (the structure of the boundary curves is also called a *pseudo-circle arrangement*). We use our definition as in our case any family of homothets of a convex set forms a pseudo-disk arrangement, see, e.g., [27].

series of papers for triangles [7, 8, 19, 20, 21, 23] and very recently [1] for squares using the Four Color Theorem. The derived upper bound on m_k is polynomial in k in both cases. It is, however, conjectured in a much more general setting [35] that whenever m_k exists, it is linear in k, just like for the translates of convex polygons. This is also known for *axis-parallel bottomless rectangles*:[4] any finite set of points can be k-colored such that any axis-parallel bottomless rectangle that contains at least $m_k = 3k - 2$ points contains all k colors [3]. (The value of m_k is known to be optimal only for $k = 2$ [17].) Their proof reduces the problem to coloring a one-dimensional *dynamic point set* with respect to intervals, which turns the problem into a variant of online colorings. We will not introduce these notions here; for some related results, see [8, 18, 23].

The dual notion of polychromatic colorings is *cover-decomposition*. In the cover-decomposition problem we are given some finite family $\mathcal{F}$ that covers some region m_k-fold (i.e., each point of the region is contained in at least m_k members of $\mathcal{F}$) and our goal is to partition $\mathcal{F}$ into k families that each cover the region. By considering the respective underlying incidence hypergraphs in the polychromatic coloring and in the cover-decomposition problems, one can see that they are about colorings of dual hypergraphs.[5] In fact, the two problems are equivalent for translates of a given set, as the following observation shows.

▶ **Observation 5** (Pach [28]). *For any set D, if $\mathcal{H}$ is the inclusion hypergraph of some points and some translates of D, then the dual hypergraph of $\mathcal{H}$ is also such an inclusion hypergraph.*

Combining this with the result of Gibson and Varadarajan [13], we get that for every convex polygon D there is a c_D such that if $\mathcal{F}$ is an $(m_k = c_D k)$-fold covering of a region by the translates of D, then $\mathcal{F}$ can be decomposed into k coverings of the same region. It follows from the proofs about polychromatic k-colorings for triangles that the same holds for coverings by the homothets of a triangle, with a polynomial bound on m_k (this function is slightly weaker than what is known for the polychromatic k-coloring problem). By homothets of other convex polygons, however, surprisingly for any m it is possible to construct an indecomposable m-fold covering [26]. The homothets of the square are the only family which is known to behave differently for polychromatic coloring and cover-decomposition.

Proper colorings of (primal and dual) geometric hypergraphs have been first studied systematically in [17], for halfplanes and axis-parallel bottomless rectangles, proving several lower and upper bounds. Other papers mainly studied the dual variant of our question. Smorodinsky [39] has shown that any pseudo-disk family can be colored with a bounded number of colors such that every point covered at least twice is covered by at least two differently colored disks. He also proved that 4 colors are sufficient for disks, and later this was generalized by Cardinal and Korman [9] to the homothetic copies of any convex body. Smorodinsky has also shown that any family of n axis-parallel rectangles can be colored with $O(\log n)$ colors such that every region covered at least twice is covered by at least two differently colored rectangles. This was shown to be optimal by Pach and Tardos [32]; they proved that there is a C such that for every m there is a family of n axis-parallel rectangles such that for any $(C\frac{\log n}{m \log m})$-coloring of the family there is a point covered by exactly m rectangles, all of the same color. It was shown by Chen et al. [10], answering a question of Brass, Moser and Pach [5], that for every c and m there is a finite point set S such that for

[4] A bottomless rectangle is a planar set of the form $\{(x, y) \mid a \le x \le b, y \le c\}$.

[5] The dual of a hypergraph $\mathcal{H} = (V, \mathcal{E})$ is the hypergraph $\overline{\mathcal{H}}$ with vertex set $\mathcal{E}$, edge set V, and with the incidences reversed, i.e., in $\overline{\mathcal{H}}$ a vertex corresponding to $e \in \mathcal{E}$ is contained in the edge corresponding to $v \in V$ if and only if v is contained in e in $\mathcal{H}$.

every c-coloring of S there is an axis-parallel rectangle containing m points that are all of the same color. This latter construction is the closest to the problem that we study. It also shows why the pseudo-disk property is crucial in Conjecture 3.

Rotation invariant families have also been studied. It was shown in [33] using the Hales-Jewett theorem [16] that for every c and m there is a finite planar point set S such that for every c-coloring of S there is a line containing m points that are all of the same color. Using duality, they have also shown that this implies that for every c and m there is a finite collection of lines such that for every c-coloring of the lines there is a point covered by exactly m lines, all of the same color. Halfplanes, on the other hand, behave much more like one-dimensional sets and admit polychromatic colorings. It was shown in [40], improving on earlier results [2, 17, 34], that any finite point set can be k-colored such that any halfplane that contains $m_k = 2k - 1$ points contains all k colors, and this is best possible. In the dual, they have shown that any finite set of halfplanes can be k-colored such that any point that is covered by at least $m_k = 3k - 2$ halfplanes is covered by all k colors. This bound is not known to be best possible, except for $k = 2$ [12]. Except for this last sharpness bound, all other results were extended to pseudo-halfplanes in [22].

2 Generalized Delaunay triangulations

With a slight perturbation of the points, it is enough to prove Theorem 2 (or any similar statement) for the case when the points are in a *general position with respect to* the convex polygon D in the sense that no two points are on a line parallel to a side of D and no four points are on the boundary of a homothet of D. In the following, we always suppose that our point set S is in general position with respect to D. We will also suppose that D is open – this does not alter the validity of the statements and makes some of the arguments simpler to present.

We say that a halfplane H is *supporting D at a side ab* of D if H contains D and ab is on the boundary of H. A point $s \in S$ is *extremal* (for a side ab) if a translate of a halfplane supporting D at a side ab contains s but no other point of S.

We define a plane graph whose vertices are the points of S, called the *generalized Delaunay triangulation* of S with respect to D, and we denote it by $\mathcal{DT}_D(S)$, or when clear from the context, simply by $\mathcal{DT}$. As it leads to no confusion, we will not differentiate between the points and their associated vertices. Two points of S are connected by a straight-line edge in $\mathcal{DT}$ if there is a homothet of D that contains only them from S. It follows [4, 24] that $\mathcal{DT}$ is a well-defined connected plane graph whose inner faces are triangles. We recall a few simple statements about $\mathcal{DT}$, most of which also appeared in [1].

▶ **Proposition 6.** *If D' is a homothet of D, the points $D' \cap S$ induce a connected subgraph of $\mathcal{DT}_D(S)$.*

▶ **Corollary 7** ([1])**.** *If D' is a homothet of D and e is an edge of $\mathcal{DT}$ that splits D' into two parts, then one of these parts does not contain any point from S.*

We continue with a proposition that is quite similar to a statement of [1].

▶ **Proposition 8.** *Suppose that D' and D_y are two homothets of D, $x, y, y', z \in D_y \cap S$ and $y, y' \in D'$ but $x, z \notin D'$, x and z are neighbors of y in $\mathcal{DT}$, and for one of the two cones whose sides are the halflines starting from y as yx and yz, denoted by C, we have $C \cap D' \subset D_y$ and $y' \in C \cap D'$. (See Figure 1 for an illustration.) Then y' has a neighbor in $\mathcal{DT}$ that is contained in $D' \cap D_y$.*

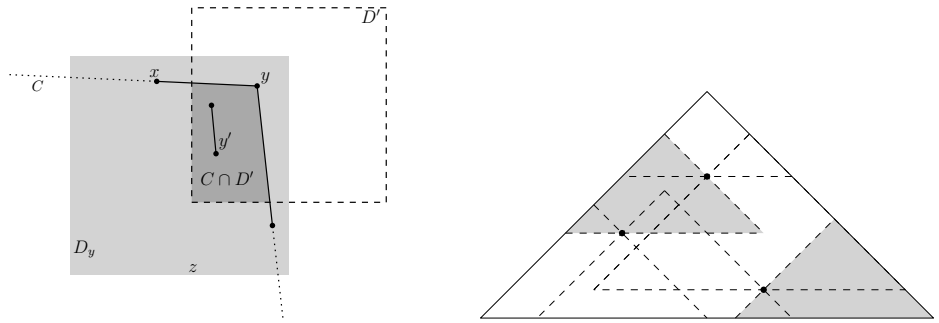


Figure 1 Illustration for Proposition 8 on the left and for Theorem 10 for triangles to the right (shading is only to improve visibility.)

Proof. Using Proposition 6, y' has a neighbor in D'. Since $\mathcal{DT}$ is planar, this neighbor must be in $C \cap D' \subset D' \cap D_y$. ◀

3 Framework

In this section we outline the main idea behind the proof of Theorem 2. As discussed in Section 2, we can suppose that S is in general position with respect to D, and we can consider the generalized Delaunay triangulation $\mathcal{DT} = \mathcal{DT}_D(S)$. We will take an initial coloring of S that has some nice properties. More specifically, we need a 3-coloring for which the assumptions of the following lemma hold for $c = 3$ and for some constant t that only depends on D.

▶ **Lemma 9.** *For every convex polygon D for every c and t there is an m such that if for a c-coloring of a point set S and a set of points $R \subset S$ it holds that*

(i) *for every homothet D' if $D' \cap S$ is monochromatic with at least t vertices, D' contains a point of R,*

(ii) *for every homothet D' if D' contains t points from R colored with the same color, D' also contains a point from $S \setminus R$ that has the same color,*

then there is a c-coloring of S such that no homothet that contains at least m points of S is monochromatic.

To prove Lemma 9, we use the following theorem about the so-called *self-coverability* of convex polygons.

▶ **Theorem 10** ([20])**.** *Given a closed convex polygon D and a collection of k points in its interior, we can take $c_D k$ homothets of D whose union is D such that none of the homothets contains any of the given points in its interior, where c_D is a constant that depends only on D.*

Proof of Lemma 9. The proper c-coloring will be simply taking the c-coloring given in the hypothesis, and recoloring each vertex in R arbitrarily to a different color. Now we prove the correctness of this new coloring. Let D' be a homothet of D containing at least m points (where m is to be determined later).

Suppose first that D' contains $m \geq ct$ points from R. Using the pigeonhole principle, D' contains at least t points from R that originally had the same color. Using (ii), D' will have a point both in R and in $S \setminus R$ that had the same color. These points will have different colors after the recoloring, thus D' will not be monochromatic.

Otherwise, suppose that D' contains m points of which less than ct are from R. Apply Theorem 10 with D' and $R' = D' \cap R$. This gives $c_D ct$ homothets (where c_D comes from Theorem 10), each of which might contain at most three points on their boundaries (which include the points from R'), thus by the pigeonhole principle at least one homothet, D'', contains no points from R and at least $\frac{m-3c_D ct}{c_D ct}$ points from $S \setminus R$. If we set $m = c_D ct(t+3)$, this is at least t. Thus, by (i), D'' was not monochromatic before the recoloring. As the recoloring does not affect points in $S \setminus R$, after the recoloring D'' (and so also D') still contains two points that have different colors. Thus $m = c_D ct(t+3)$ is a good choice for m in both cases. ◀

Therefore, to prove Theorem 2, we only need to show that we can find a coloring with three colors that satisfy the conditions of Lemma 9 for some t.

4 Proof of Theorem 2

In this section we prove Theorem 2, that is, we show that for every convex polygon D there is an m such that any finite set of points S admits a 3-coloring such that there is no monochromatic homothet of D that contains at least m points. If one could find a 3-coloring where every monochromatic component of $\mathcal{DT}$ is bounded, then that would immediately prove Theorem 2. This, however, is not true in general [25], only for bounded degree graphs [11], but the $\mathcal{DT}$ can have arbitrarily high degree vertices for any convex polygon, thus we cannot apply this result. Instead, we use the following result (whose proof is just a couple of pages).

▶ **Theorem 11** (Poh [38]; Goddard [14]). *The vertices of any planar graph can be 3-colored such that every monochromatic component is a path.*

To prove Theorem 2, apply Theorem 11 to $\mathcal{DT}$ to obtain a 3-coloring where every monochromatic component is a path. As discussed in Section 3, it is sufficient to show that for $t = 4n + 12$ (where n denotes the number of sides of D) there is a set of points $R \subset S$ for which

- **(i)** for every homothet D' if $D' \cap S$ is monochromatic with at least t vertices, D' contains a point of R,
- **(ii)** for every homothet D' if D' contains t points from R colored with the same color, D' also contains a point from $S \setminus R$ that has the same color.

Now we describe how to select R. First, partition every monochromatic path that has at least t vertices into subpaths, called *sections*, such that the number of vertices of each section is at least $\frac{t}{4}$ but at most $\frac{t}{2}$. We call such a section *cuttable* if there is a monochromatic homothet of D that contains all of its points. R will consist of exactly one point from each cuttable section. These points are selected arbitrarily from the non-extremal points of each section, except that they are required to be non-adjacent on their monochromatic path. By ruling out the extremal points and the two end vertices of a section, by the pigeonhole principle we can select such a point from each section if $\frac{t}{4} \geq n + 3$. For an $r \in R$ we denote its section by σ_r and a (fixed) monochromatic homothet containing it by D_r.

Now we prove that R satisfies the requirements (i) and (ii).

To prove (i), suppose that a homothet D' is monochromatic with at least t vertices. Using Proposition 6, the subgraph induced on these vertices is connected. As any monochromatic connected component is a path, D' contains at least t consecutive vertices of a monochromatic path, and thus also a section. Because of D' this section is cuttable, and thus contains a point of R.

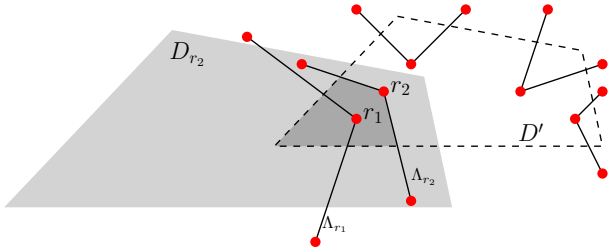


Figure 2 Proof of Proposition 12.

To prove (ii), suppose that a homothet D' contains t points from R colored with the same color, red. Denote these points by R'. It is sufficient to prove that for some $r \in R'$ another point from σ_r is also contained in D'. Suppose the contrary. As the neighbors of r in σ_r are red but not in R, they must be outside D', or (ii) holds. For each $r \in R'$, denote the geometric embedding of the two edges adjacent to r in σ_r by Λ_r. Therefore, Λ_r will intersect the boundary of D' in two points for each $r \in R'$. We claim that these points are usually on the same side of D'.

▶ **Proposition 12.** *Both intersection points of Λ_r and the boundary of D' are on the same side of D' for all but at most n points of $r \in R'$.*

Proof. Suppose that there are more than n points $r \in R'$ for which Λ_r intersects D' in two sides. For each such point $r \in R'$, for (at least) one of the two cones whose sides are the halflines starting in Λ_r, denoted by C_r, we have $C_r \cap D' \subset D_r$. Since the intersection $C_r \cap D'$ is a connected curve, it contains a vertex of D'. Using the pigeonhole principle, there are two points, $r_1, r_2 \in R'$, such that C_{r_1} and C_{r_2} contain the same vertex of D'. (See Figure 2.) In this case r_1 and r_2 would be in the configuration described in Proposition 8, one playing the role of y, the other of y'. The neighbor of y' in $D' \cap D_y$ given by Proposition 8 must also be red, as it is contained in D_y. As the red neighbors of any red point of R are not in R, we have found a red point from $S \setminus R$ in D', proving (ii). ◀

Divide the points $r \in R'$ for which Λ_r intersects only one side of D' into n groups, $R'_1, \ldots, R'_n$, depending on which side is intersected. By the pigeonhole principle there is a group, R'_i, that contains at least $\frac{t-n}{n} \geq 3$ points. Suppose without loss of generality that the side ab intersected by Λ_r for $r \in R'_i$ is horizontal, bounding D' from below. For each $r \in R'_i$, fix and denote by x_r a point from σ_r whose y-coordinate is larger than the y-coordinate of r. (Such a point exists because no $r \in R$ is extremal in σ_r.) Denote the path from r to x_r in σ_r by P_r, and the neighbor of r in P_r by q_r.

The geometric embedding of P_r starts above ab with r, then goes below ab as $q_r \notin D'$, and finally x_r is again above the line ab. Denote the first intersection (starting from r) of the embedding of the path P_r with the line ab by $\alpha_r = r\bar{q}_r \cap \bar{ab}$, and the next intersection by β_r. Since $|R'_i| \geq 3$, without loss of generality, there are $r_1, r_2 \in R'_i$ such that β_{r_1} is to the left of α_{r_1} and β_{r_2} is to the left of α_{r_2}. For readability and simplicity, let $x_i = x_{r_i}$, $P_i = P_{r_i}$, $q_i = q_{r_i}$, $\alpha_i = \alpha_{r_i}$, $\beta_i = \beta_{r_i}$.

Without loss of generality suppose that α_1 is to the left of α_2. Recall that P_2 contains only red points, of which only r_2 is in R. Therefore, no other vertex of P_2 can be in D'. If β_2 is to the right of α_1, then one of the edges of P_2 would separate r_1 and r_2 in the sense described in Corollary 7. (See Figure 3.) As this cannot happen, β_2 is to the left of α_1.

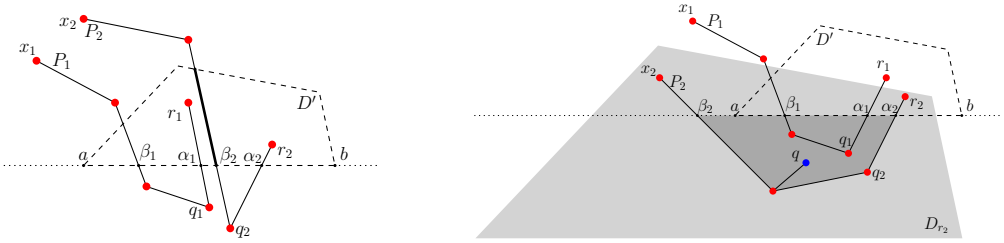


Figure 3 The two cases at the end of the proof of Theorem 2. To the left, β_2 is to the right of α_1, the part of the edge splitting D' is bold. To the right, β_2 is to the left of α_1, the shaded regions must contain q_1.

This implies that $q_1 \notin P_2$ is in the convex hull of P_2 below the ab line. Take the point $q \in S \setminus P_2$ with the smallest y-coordinate such that q is in the convex hull of P_2 below the ab line. As q is not an extremal point of S, it is connected in $\mathcal{DT}$ to some point in S whose y-coordinate is smaller. By the definition of q, this neighbor must be in P_2. As the end vertices of P_2, r_2 and x_2, are above the ab line, q is connected to an inner vertex of a monochromatic red path. Since every monochromatic component is a path, q cannot be red. The homothet D_{r_2} contains the red vertices of P_2 and thus all the points in the convex hull of P_2. But D_{r_2} is monochromatic, so it cannot contain the non-red point q, a contradiction.

This finishes the proof of Theorem 2.

5 Higher dimensions

In this section we study the following natural extension of the problem to higher dimensions. Given a finite set of points $S \in \mathbb{R}^d$ and a family $\mathcal{F}$, can we c-color S such that every $F \in \mathcal{F}$ contains at least two colors? First we show that for $d = 3$ there might be hope to find such colorings for natural families, but not in higher dimensions. Define a (positive) *hextant* in $\mathbb{R}^4$ as the set of points $\{(x, y, z, w) \mid x \geq x_0, y \geq y_0, z \geq z_0, w \geq w_0\}$ for some real numbers x_0, y_0, z_0, w_0. Cardinal noticed that hextants can simulate the axis-parallel rectangles of an appropriate subplane of $\mathbb{R}^4$ and thus the following holds.

▶ **Theorem 13** (Cardinal[6]). *For any c and m there is a finite point set S such that for every c-coloring of S there is a hextant that contains exactly m points of S, all of the same color.*

Proof. As mentioned in the introduction, Chen et al. [10] have shown that for any c and m there is a finite planar point set S such that for every c-coloring of S there is an axis-parallel rectangle that contains exactly m points of S, all of the same color. Place this construction on the $\Pi = \{(x, y, z, w) \mid x + y = 0,\ z + w = 0\}$ subplane of $\mathbb{R}^4$. A hextant $\{(x, y, z, w) \mid x \geq x_0, y \geq y_0, z \geq z_0, w \geq w_0\}$ intersects Π in $\{(x, y, z, w) \mid x_0 \leq x = -y \leq -y_0, z_0 \leq z = -w \leq -w_0\}$, which is a rectangle whose sides are parallel to the lines $\{x + y = 0, z = w = 0\}$ and $\{x = y = 0, z + w = 0\}$, respectively. Taking these perpendicular lines as axes, thus any "axis-parallel" rectangle of Π is realizable by an appropriate hextant, and the theorem follows. ◀

For $d = 3$, however, the following might hold.

[6] Cardinal (personal communication) stated this for $c = 2$ using the same reduction based on [33] about axis-parallel rectangles; Theorem 13 is only more general because we use a stronger result [10] about axis-parallel rectangles.

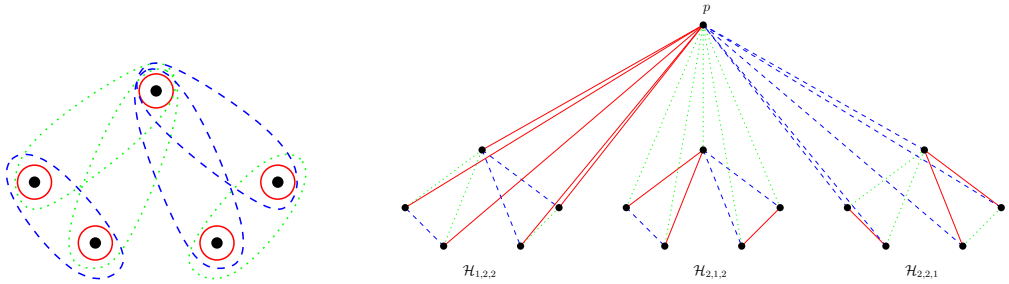


Figure 4 $\mathcal{H}(1,2,2)$ drawn with sets (left) and $\mathcal{H}(2,2,2)$ drawn as graph (right). Different colors represent the edges from the different families $\mathcal{E}_i$.

▶ **Conjecture 14.** *For every convex set $D \subset \mathbb{R}^3$ there is an m such that any finite set of points admits a 4-coloring such that any homothetic copy of D that contains at least m points contains at least two colors.*

As an anonymous referee called our attention to it,[7] a 3-dimensional *Delaunay triangulation* of any number of points might induce a complete graph (for a recent proof, see [15]), so it is not even clear that Conjecture 14 is true with any number of colors instead of 4, unlike it was in 2 dimensions.

The reason why Conjecture 14 is stated with 4 colors is the following construction.

▶ **Theorem 15.** *For every m there is a finite set of points $S \in \mathbb{R}^3$ such that for any 3-coloring of S there is a unit ball that contains exactly m points of S, all of the same color.*

Earlier such a construction with unit balls was only known for 2-colorings [33]. For 2-colorings the analogue of Theorem 15 was also shown to hold when the family is the translates of any polyhedron instead of unit balls [36]. The only known positive result is that for *octants* any finite set of points can be 2-colored such that any octant that contains at least 9 points contains both colors [19, 23]. We do not, however, know the answer for 3-colorings and the translates or homothets of polyhedra.

The rest of this section contains a sketch of the proof of Theorem 15. The reason why we only sketch the proof is that it is a simple modification of the planar construction with similar properties for unit disks from [30].

Abstract hypergraph

First we define the abstract hypergraph that will be realized with unit balls. It is a straightforward generalization of the hypergraph defined first in [36]. Instead of a single parameter, m, the induction will be on three parameters, k, ℓ and m. For any k, ℓ, m we define the (multi)hypergraph $\mathcal{H}(k,\ell,m) = (V(k,\ell,m), \mathcal{E}(k,\ell,m))$ recursively. The edge set $\mathcal{E}(k,\ell,m)$ will be the disjoint union of three sets, $\mathcal{E}(k,\ell,m) = \mathcal{E}_1(k,\ell,m) \cup \mathcal{E}_2(k,\ell,m) \cup \mathcal{E}_3(k,\ell,m)$. All edges belonging to $\mathcal{E}_1(k,\ell,m)$ will be of size k, all edges belonging to $\mathcal{E}_2(k,\ell,m)$ will be of size ℓ, and all edges belonging to $\mathcal{E}_3(k,\ell,m)$ will be of size m. We will prove that in every 3-coloring of $\mathcal{H}(k,\ell,m)$ with colors c_1, c_2 and c_3 there will be an edge in $\mathcal{E}_i(k,\ell,m)$ such that all of its vertices are colored c_i for some $i \in \{1,2,3\}$. If $k = \ell = m$, we get an m-uniform hypergraph that cannot be properly 3-colored.

[7] We have claimed the opposite in the first version of our manuscript.

Now we give the recursive definition. Define $\mathcal{H}(1,1,1)$ as a hypergraph on one vertex with three edges containing it, with one edge in each of $\mathcal{E}_1(1,1,1)$, $\mathcal{E}_2(1,1,1)$ and $\mathcal{E}_3(1,1,1)$. If at least one of k,ℓ,m is bigger than 1, define $\mathcal{H}(k,\ell,m)$ recursively from $\mathcal{H}(k-1,\ell,m)$, $\mathcal{H}(k,\ell-1,m)$, $\mathcal{H}(k,\ell,m-1)$ by adding a "new" vertex p as follows.

$$V(k,\ell,m) = V(k-1,\ell,m) \uplus V(k,\ell-1,m) \uplus V(k,\ell,m-1) \uplus \{p\}.$$

If $k=1$, then $\mathcal{E}_1(1,\ell,m) = \{\{v\} : v \in V(1,\ell,m)\}$, otherwise

$$\mathcal{E}_1(k,\ell,m) = \{e \cup \{p\} : e \in \mathcal{E}_1(k-1,\ell,m)\} \uplus \mathcal{E}_1(k,\ell-1,m) \uplus \mathcal{E}_1(k,\ell,m-1).$$

Similary, if $\ell=1$, then $\mathcal{E}_2(k,1,m) = \{\{v\} : v \in V(k,1,m)\}$, otherwise

$$\mathcal{E}_2(k,\ell,m) = \{e \cup \{p\} : e \in \mathcal{E}_2(k,\ell-1,m)\} \uplus \mathcal{E}_2(k-1,\ell,m) \uplus \mathcal{E}_2(k,\ell,m-1),$$

and if $m=1$, then $\mathcal{E}_3(k,\ell,1) = \{\{v\} : v \in V(k,\ell,1)\}$, otherwise

$$\mathcal{E}_3(k,\ell,m) = \{e \cup \{p\} : e \in \mathcal{E}_3(k,\ell,m-1)\} \uplus \mathcal{E}_3(k-1,\ell,m) \uplus \mathcal{E}_3(k,\ell-1,m).$$

▶ **Lemma 16.** *In every 3-coloring of $\mathcal{H}(k,\ell,m)$ with colors c_1,c_2 and c_3 there is an edge in $\mathcal{E}_i(k,\ell,m)$ such that all of its vertices are colored c_i for some $i \in \{1,2,3\}$. Therefore, $\mathcal{H}(k,\ell,m)$ has no proper 3-coloring.*

The proof is a simple modification of the respective statement from [36].

Proof. If $k=\ell=m=1$, the statement holds. Otherwise, suppose without loss of generality that the color of p is c_1. If $k=1$, we are done as $\{p\} \in \mathcal{E}_1(1,\ell,m)$. Otherwise, consider the copy of $\mathcal{H}(k-1,\ell,m)$ contained in $\mathcal{H}(k,\ell,m)$. If it contains an edge in $\mathcal{E}_2(k-1,\ell,m)$ or $\mathcal{E}_3(k-1,\ell,m)$ such that its vertices are all colored c_2 or all colored c_3, respectively, we are done. Otherwise, it contains an $e \in \mathcal{E}_1(k-1,\ell,m)$ such that its vertices are all colored c_1. But then all the vertices of $(e \cup \{p\}) \in \mathcal{E}_1(k,\ell,m)$ are also all colored c_1, we are done. ◀

Geometric realization

Now we sketch how to realize $\mathcal{H}(k,\ell,m)$ by unit balls in $\mathbb{R}^3$. The construction will build on the construction of [30], where the edges belonging to $\mathcal{E}_1(k,\ell,1) \uplus \mathcal{E}_2(k,\ell,1)$ of $\mathcal{H}(k,\ell,1)$ were realized by unit disks.

The vertices $V(k,\ell,m)$ will be embedded as a point set, $S(k,\ell,m)$, and the edge set $\mathcal{E}_i(k,\ell,m)$ as a collection of unit balls, $\mathcal{B}_i(k,\ell,m)$, where a point is contained in a ball if and only if the respective vertex is in the respective edge. All the points of $S(k,\ell,m)$ will be placed in a small neighborhood of the origin. The centers of the balls from $\mathcal{B}_1(k,\ell,m)$, $\mathcal{B}_2(k,\ell,m)$ and $\mathcal{B}_3(k,\ell,m)$ will be close to $(0,-1,0)$, $(0,1,0)$ and $(0,0,-1)$, respectively. The realization of $\mathcal{H}(1,1,1)$ contains only one point, the origin, and one ball in each family, centered appropriately close to the required center.

Suppose that not all of k,ℓ,m are 1, and we have already realized the hypergraphs $\mathcal{H}(k-1,\ell,m)$, $\mathcal{H}(k,\ell-1,m)$ and $\mathcal{H}(k,\ell,m-1)$. Place the new point p in the origin, and shift the corresponding realizations (i.e., the point sets, $S(k-1,\ell,m)$, $S(k,\ell-1,m)$ and $S(k,\ell,m-1)$, and the collection of balls, $\mathcal{B}(k-1,\ell,m)$, $\mathcal{B}(k,\ell-1,m)$ and $\mathcal{B}(k,\ell,m-1)$) by the following vectors, where $\epsilon = \epsilon(k,\ell,m)$ is a small enough number, but such that $\epsilon(k-1,\ell,m)$, $\epsilon(k,\ell-1,m)$ and $\epsilon(k,\ell,m-1)$ are all $O(\epsilon^5(k,\ell,m))$.

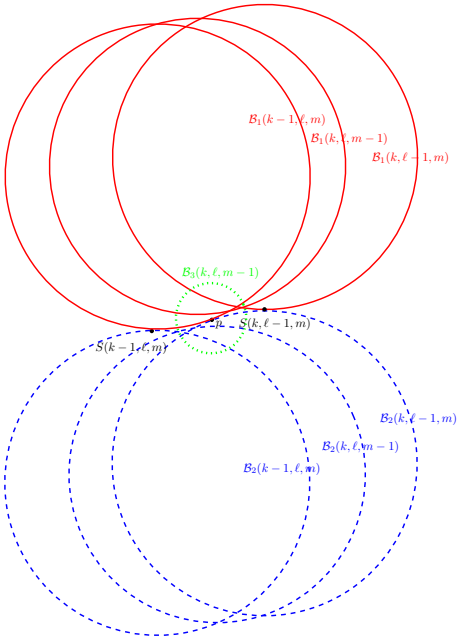


Figure 5 The intersection of $\mathcal{H}(k,\ell,m)$ with the $z=0$ plane. Point sets/collections of balls that are at distance $O(\epsilon^5)$ are represented by a single point/ball. As the balls $\mathcal{B}_3(k-1,\ell,m)$ intersect in a $O(\epsilon^5)$ vicinity of $S(k-1,\ell,m)$ and the balls $\mathcal{B}_3(k,\ell-1,m)$ intersect in a $O(\epsilon^5)$ vicinity of $S(k,\ell-1,m)$, they are not drawn to avoid overcrowding the picture.

1. Shift $\mathcal{H}(k-1,\ell,m)$ by $(2\epsilon-1.5\epsilon^3, 2\epsilon^2, 0)$.
2. Shift $\mathcal{H}(k,\ell-1,m)$ by $(-2\epsilon+1.5\epsilon^3, -2\epsilon^2, 0)$.
3. Shift $\mathcal{H}(k,\ell,m-1)$ by $(0,0,2\epsilon^2)$.

For an illustration, see Figure 5.

▶ **Proposition 17.** *The above construction realizes* $\mathcal{H}(k,\ell,m)$.

The proof of this proposition is a routine calculation, we only show some parts.

Proof. Denote by o_B the center of the ball B and denote by $dist(p,q)$ the Euclidean distance of two points p,q.

1. $p \in B \in \mathcal{B}_1(k-1,\ell,m)$:

$$dist^2(p,o_B) = (2\epsilon-1.5\epsilon^3)^2 + (1-2\epsilon^2)^2 + O(\epsilon^5) = 1-2\epsilon^4+O(\epsilon^5) < 1.$$

2. $p \notin B \in \mathcal{B}_1(k,\ell-1,m)$:

$$dist^2(p,o_B) = (2\epsilon-1.5\epsilon^3)^2 + (1+2\epsilon^2)^2 + O(\epsilon^5) = 1+4\epsilon^2+O(\epsilon^3) > 1.$$

3. $p \notin B \in \mathcal{B}_1(k,\ell,m-1)$:

$$dist^2(p,o_B) = 1^2 + (2\epsilon^2)^2 + O(\epsilon^5) = 1+4\epsilon^4+O(\epsilon^5) > 1.$$

4. If $s \in S(k,\ell-1,m)$, then $s \notin B \in \mathcal{B}_1(k-1,\ell,m)$:

$$dist^2(s,o_B) = (4\epsilon-3\epsilon^3)^2 + (1-4\epsilon^2)^2 + O(\epsilon^5) = 1+8\epsilon^2+O(\epsilon^3) > 1.$$

5. If $s \in S(k, \ell-1, m)$, then $s \notin B \in \mathcal{B}_1(k, \ell, m-1)$:

$$dist^2(s, o_B) = (2\epsilon - 1.5\epsilon^3)^2 + (1 - 2\epsilon^2)^2 + (2\epsilon^2)^2 + O(\epsilon^5) = 1 + 2\epsilon^4 + O(\epsilon^5) > 1.$$

6. If $s \in S(k, \ell-1, m)$, then $s \notin B \in \mathcal{B}_3(k, \ell, m-1)$:

$$dist^2(s, o_B) = (2\epsilon - 1.5\epsilon^3)^2 + (2\epsilon^2)^2 + (1 - 2\epsilon^2)^2 + O(\epsilon^5) = 1 + 2\epsilon^4 + O(\epsilon^5) > 1.$$

7. If $s \in S(k, \ell, m-1)$, then $s \notin B \in \mathcal{B}_1(k-1, \ell, m)$:

$$dist^2(s, o_B) = (2\epsilon - 1.5\epsilon^3)^2 + (1 - 2\epsilon^2)^2 + (2\epsilon^2)^2 + O(\epsilon^5) = 1 + 2\epsilon^4 + O(\epsilon^3) > 1.$$

The other incidences can be checked similarly and thus Proposition 17 follows. ◀

Lemma 16 and Proposition 17 imply Theorem 15 by selecting $k = \ell = m$, therefore this also finishes the proof of Theorem 15.

6 Further remarks

Combining Theorems 2 and 10, for any convex polygon, D, and for any finite point set, S, we can first find a 3-coloring of S using Theorem 2 such that every large (in the sense that it contains many points of S) homothet of D contains two differently colored points, then using Theorem 10 we can conclude that every very large homothet of D contains many points from at least two color classes, and finally we can recolor every color class separately using Theorem 2. This proves that for every k there is a 3^k-coloring such that every large homothet of D contains at least 2^k colors. Of course, the colors that we use when recoloring need not be different for each color class, so we can also prove for example that there is a 6-coloring such that every large homothet of D contains at least 3 colors. What are the best bounds of this type that can be obtained?

Given a planar graph, G, and a pair of paths on three vertices, uvw and $u'vw'$, we say that the paths *cross* at v if u, u', w, w' appear in this order around v. A possible equivalent reformulation of Conjecture 3 is the following. Is it true that for any planar graph and any pairwise non-crossing collection of its paths on three vertices, $\mathcal{P}$, there is a 3-coloring of the vertices such that every path from $\mathcal{P}$ is non-monochromatic?

Finally, we would like to draw attention to the study of *realizable hypergraphs*. Unfortunately, *planar hypergraphs* are traditionally defined dully as a hypergraph whose (bipartite) incidence graph is planar. Instead, it would be more natural to define them as the hypergraphs realizable by a pseudo-disk arrangement in the sense that the vertices are embedded as points and the edges as pseudo-disks such that a point is contained in a pseudo-disk if and only if the respective vertex is in the respective edge. This was done in [6], where it was proved that such a hypergraph on n vertices can have at most $O(k^2n)$ edges that each contain at most k points, while there can be at most $3n-6$ edges containing exactly two points, matching Euler's bound for planar graphs. Despite [6], these hypergraphs received little attention and even simple statements are highly trivial; see the recent proof by Kisfaludi-Bak[8] that the complete 3-uniform hypergraph on 5 vertices is not realizable by pseudo-disks. We believe that these hypergraphs deserve more attention.

[8] http://mathoverflow.net/a/257212/955.

Acknowledgment. We would like to thank our anonymous referees for several suggestions that improved the presentation of our results, and to Arnau Padrol for explaining to us the example in [15].

References

1 Eyal Ackerman, Balázs Keszegh, and Máté Vizer. Coloring points with respect to squares. In Sándor P. Fekete and Anna Lubiw, editors, *32nd International Symposium on Computational Geometry, SoCG 2016, June 14-18, 2016, Boston, MA, USA*, volume 51 of *LIPIcs*, pages 5:1–5:16. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2016.

2 Greg Aloupis, Jean Cardinal, Sébastien Collette, Stefan Langerman, and Shakhar Smorodinsky. Coloring geometric range spaces. *Discrete & Computational Geometry*, 41(2):348–362, 2009.

3 Andrei Asinowski, Jean Cardinal, Nathann Cohen, Sébastien Collette, Thomas Hackl, Michael Hoffmann, Kolja B. Knauer, Stefan Langerman, Michal Lason, Piotr Micek, Günter Rote, and Torsten Ueckerdt. Coloring hypergraphs induced by dynamic point sets and bottomless rectangles. In Frank Dehne, Roberto Solis-Oba, and Jörg-Rüdiger Sack, editors, *Algorithms and Data Structures – 13th International Symposium, WADS 2013, London, ON, Canada, August 12-14, 2013. Proceedings*, volume 8037 of *Lecture Notes in Computer Science*, pages 73–84. Springer, 2013.

4 Prosenjit Bose, Paz Carmi, Sébastien Collette, and Michiel H. M. Smid. On the stretch factor of convex delaunay graphs. *J. of Computational Geometry*, 1(1):41–56, 2010.

5 Peter Brass, William O. J. Moser, and János Pach. *Research problems in discrete geometry.* Springer, 2005.

6 Sarit Buzaglo, Rom Pinchasi, and Günter Rote. Topological hypergraphs. In János Pach, editor, *Thirty Essays on Geometric Graph Theory*, pages 71–81. Springer New York, 2013. doi:10.1007/978-1-4614-0110-0_6.

7 Jean Cardinal, Kolja Knauer, Piotr Micek, and Torsten Ueckerdt. Making triangles colorful. *J. of Computational Geometry*, 4:240–246, 2013.

8 Jean Cardinal, Kolja Knauer, Piotr Micek, and Torsten Ueckerdt. Making octants colorful and related covering decomposition problems. *SIAM J. on Discrete Math.*, 28(4):1948–1959, 2014.

9 Jean Cardinal and Matias Korman. Coloring planar homothets and three-dimensional hypergraphs. *Computational Geometry*, 46(9):1027–1035, 2013.

10 Xiaomin Chen, János Pach, Mario Szegedy, and Gábor Tardos. Delaunay graphs of point sets in the plane with respect to axis-parallel rectangles. *Random Struct. Algorithms*, 34(1):11–23, 2009. doi:10.1002/rsa.20246.

11 Louis Esperet and Gwenaël Joret. Colouring planar graphs with three colours and no large monochromatic components. *Combinatorics, Probability & Computing*, 23(4):551–570, 2014.

12 Radoslav Fulek. Coloring geometric hypergraph defined by an arrangement of half-planes. In *Proceedings of the 22nd Annual Canadian Conference on Computational Geometry, Winnipeg, Manitoba, Canada, August 9-11, 2010*, pages 71–74, 2010.

13 Matt Gibson and Kasturi R. Varadarajan. Decomposing coverings and the planar sensor cover problem. In *50th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2009, October 25-27, 2009, Atlanta, Georgia, USA*, pages 159–168. IEEE Computer Society, 2009. doi:10.1109/FOCS.2009.54.

14 Wayne Goddard. Acyclic colorings of planar graphs. *Discrete Math.*, 91(1):91–94, 1991.

15 B. Gonska and A. Padrol. Neighborly inscribed polytopes and delaunay triangulations. *Advances in Geometry*, 16(3):349–360, 2016.

16 A. W. Hales and R. I. Jewett. Regularity and positional games. *Trans. Amer. Math. Soc.*, 106:222–229, 1963.

17 Balázs Keszegh. Coloring half-planes and bottomless rectangles. *Computational Geometry*, 45(9):495–507, 2012.

18 Balázs Keszegh, Nathan Lemons, and Dömötör Pálvölgyi. Online and quasi-online colorings of wedges and intervals. *Order*, 33(3):389–409, 2016.

19 Balázs Keszegh and Dömötör Pálvölgyi. Octants are cover-decomposable. *Discrete & Computational Geometry*, 47(3):598–609, 2012.

20 Balázs Keszegh and Dömötör Pálvölgyi. Convex polygons are self-coverable. *Discrete & Computational Geometry*, 51(4):885–895, 2014.

21 Balázs Keszegh and Dömötör Pálvölgyi. Octants are cover-decomposable into many coverings. *Computational Geometry*, 47(5):585–588, 2014.

22 Balázs Keszegh and Dömötör Pálvölgyi. An abstract approach to polychromatic coloring: Shallow hitting sets in aba-free hypergraphs and pseudohalfplanes. In Ernst W. Mayr, editor, *Graph-Theoretic Concepts in Computer Science – 41st International Workshop, WG 2015, Garching, Germany, June 17-19, 2015, Revised Papers*, volume 9224 of *Lecture Notes in Computer Science*, pages 266–280. Springer, 2015.

23 Balázs Keszegh and Dömötör Pálvölgyi. More on decomposing coverings by octants. *J. of Computational Geometry*, 6(1):300–315, 2015.

24 Rolf Klein. *Concrete and abstract Voronoi diagrams*, volume 400. Springer Science & Business Media, 1989.

25 Jon M. Kleinberg, Rajeev Motwani, Prabhakar Raghavan, and Suresh Venkatasubramanian. Storage management for evolving databases. In *38th Annual Symposium on Foundations of Computer Science, FOCS'97, Miami Beach, Florida, USA, October 19-22, 1997*, pages 353–362. IEEE Computer Society, 1997.

26 István Kovács. Indecomposable coverings with homothetic polygons. *Discrete & Computational Geometry*, 53(4):817–824, 2015.

27 L. Ma. *Bisectors and Voronoi Diagrams for Convex Distance Functions.* PhD thesis, FernUniversität Hagen, Germany, 2000.

28 János Pach. Decomposition of multiple packing and covering. In *2. Kolloquium über Diskrete Geometrie*, pages 169–178. Institut für Mathematik der Universität Salzburg, 1980.

29 János Pach. Covering the plane with convex polygons. *Discrete & Computational Geometry*, 1:73–81, 1986. doi:10.1007/BF02187684.

30 János Pach and Dömötör Pálvölgyi. Unsplittable coverings in the plane. *Advances in Mathematics*, 302:433–457, 2016.

31 János Pach, Dömötör Pálvölgyi, and Géza Tóth. Survey on decomposition of multiple coverings. In Imre Bárány, Károly J. Böröczky, Gábor Fejes Tóth, and János Pach, editors, *Geometry – Intuitive, Discrete, and Convex*, volume 24 of *Bolyai Society Mathematical Studies*, pages 219–257. Springer Berlin Heidelberg, 2013.

32 János Pach and Gábor Tardos. Coloring axis-parallel rectangles. *J. of Combinatorial Theory, Series A*, 117(6):776–782, 2010. doi:10.1016/j.jcta.2009.04.007.

33 János Pach, Gábor Tardos, and Géza Tóth. Indecomposable coverings. *Canadian mathematical bulletin*, 52(3):451–463, 2009.

34 János Pach and Géza Tóth. Decomposition of multiple coverings into many parts. In *Proceedings of the twenty-third annual symposium on Computational geometry*, pages 133–137. ACM, 2007.

35 Dömötör Pálvölgyi. Decomposition of geometric set systems and graphs, phd thesis. *arXiv preprint arXiv:1009.4641*, 2010.

36 Dömötör Pálvölgyi. Indecomposable coverings with concave polygons. *Discrete & Computational Geometry*, 44(3):577–588, 2010. doi:10.1007/s00454-009-9194-y.

37 Dömötör Pálvölgyi and Géza Tóth. Convex polygons are cover-decomposable. *Discrete & Computational Geometry*, 43(3):483–496, 2010.

38 K. S. Poh. On the linear vertex-arboricity of a planar graph. *J. of Graph Theory*, 14(1):73–75, 1990.

39 Shakhar Smorodinsky. On the chromatic number of geometric hypergraphs. *SIAM J. on Discrete Math.*, 21(3):676–687, 2007.

40 Shakhar Smorodinsky and Yelena Yuditsky. Polychromatic coloring for half-planes. *J. of Combinatorial Theory, Series A*, 119(1):146–154, 2012.

41 Gábor Tardos and Géza Tóth. Multiple coverings of the plane with triangles. *Discrete & Computational Geometry*, 38(2):443–450, 2007. doi:10.1007/s00454-007-1345-4.

Computing Representative Networks for Braided Rivers*

Maarten Kleinhans[1], Marc van Kreveld[2], Tim Ophelders[3], Willem Sonke[4], Bettina Speckmann[5], and Kevin Verbeek[6]

1 **Faculty of Geosciences, Utrecht University, Utrecht, The Netherlands**
`m.g.kleinhans@uu.nl`
2 **Dept. of Information and Computing Sciences, Utrecht University, Utrecht, The Netherlands**
`m.j.vankreveld@uu.nl`
3 **Dept. of Mathematics and Computer Science, TU Eindhoven, Eindhoven, The Netherlands**
`t.a.e.ophelders@tue.nl`
4 **Dept. of Mathematics and Computer Science, TU Eindhoven, Eindhoven, The Netherlands**
`w.m.sonke@tue.nl`
5 **Dept. of Mathematics and Computer Science, TU Eindhoven, Eindhoven, The Netherlands**
`b.speckmann@tue.nl`
6 **Dept. of Mathematics and Computer Science, TU Eindhoven, Eindhoven, The Netherlands**
`k.a.b.verbeek@tue.nl`

Abstract

Drainage networks on terrains have been studied extensively from an algorithmic perspective. However, in drainage networks water flow cannot bifurcate and hence they do not model *braided rivers* (multiple channels which split and join, separated by sediment bars). We initiate the algorithmic study of braided rivers by employing the descending quasi Morse-Smale complex on the river bed (a polyhedral terrain), and extending it with a certain ordering of bars from the one river bank to the other. This allows us to compute a graph that models a representative channel network, consisting of lowest paths. To ensure that channels in this network are sufficiently different we define a *sand function* that represents the volume of sediment separating them. We show that in general the problem of computing a maximum network of non-crossing channels which are δ-different from each other (as measured by the sand function) is NP-hard. However, using our ordering between the river banks, we can compute a maximum δ-different network that respects this order in polynomial time. We implemented our approach and applied it to simulated and real-world braided rivers.

1998 ACM Subject Classification F.2.2 Analysis of Algorithms and Problem Complexity

Keywords and phrases Braided rivers, Morse-Smale complex, persistence, network extraction, polyhedral terrain

Digital Object Identifier 10.4230/LIPIcs.SoCG.2017.48

* T. Ophelders, W. Sonke and B. Speckmann are supported by the Netherlands Organisation for Scientific Research (NWO) under project no. 639.023.208, and K. Verbeek under project no. 639.021.541. M. Kleinhans is supported by the Dutch Technology Foundation STW (grant Vici 016.140.316/13710; part of the Netherlands Organisation for Scientific Research (NWO), and partly funded by the Ministry of Economic Affairs).

1 Introduction

Geomorphology is the study of the shape of natural terrains and the processes that create them. One of these processes is erosion due to water flow. In mountainous areas, valleys are nearly always shaped by rivers,[1] which slowly transport solid material downstream. Due to gravity, water usually follows the direction of steepest descent, although inertia may result in deviations. The combination of terrain shape and water flow gives rise to various computational problems that have been studied in geomorphology, geographic information science (GIS), and computational geometry.

One prominent problem is the computation of drainage networks, also referred to as flows [1, 2, 4]. Here computations are based on elevation data only and the shape of the terrain is used to determine where rivers will form (see [26] for an extensive overview). A second problem of interest concerns local minima. Due to erosion local minima are more rare in natural terrains than local maxima. Minima in terrains are the bottoms of pits that will be filled with water. When a pit overflows at the lowest saddle surrounding it, the saddle may be eroded and eventually both pit and saddle will be removed simultaneously. For this reason certain terrain types do not have local minima and minor local minima are often measurement errors. Such errors are clearly undesirable when studying flow on terrains and hence, minor local minima are removed by computational means [15, 16, 17, 25]. A third commonly studied problem deals with watersheds. Watershed boundaries are steepest ascent paths following the ridges of mountains [5, 19, 26].

Braided rivers. The usual models (as discussed above) for water flow in terrains allow rivers to merge, which is natural because side valleys join main valleys. And clearly, if water always follows the direction of steepest descent, a river cannot split (except due to degeneracies). Yet splitting happens in deltas and various river types, in particular braided river systems [12, 11] (see Figure 1 for an example). Such systems have islands called bars ranging in length from about one water depth to ten times the overall river width, separating different channels of the same river over their length after which the channels confluence [20]. Bifurcations are currently still poorly understood in geomorphology [14]. They are quite dynamic in the sense that erosion or sedimentation leads to perpetual changes in the division of water and sediment at bifurcations, which affects downstream morphological development of channels and bars locally [3, 6, 23] and over surprisingly long distances downstream [22]. This makes braided river systems highly dynamic on the seasonal time scale. Within a year the appearance and hydrological functioning is also highly variable due to changing water levels during floods and low flow periods that emerge and submerge bars.

Many fundamental questions remain how rivers form shallow, ecologically valuable shoals, carve deep channels for shipping, collapse banks with property on it, and flood built-up areas. Part of the reason for slow progress is that sediment transport is a nonlinear function of flow velocity, leading to self-amplifying changes in depth and width of channels. This leads to changes in water and sediment conveyance, which in turn affects downstream channels that split around other bars and so on ad nearly infinitum [14]. We know this quasi-quantitatively from increasing gridded data of bed elevation and flow properties collected with advanced remote sensing techniques [11, 18], in numerical two-phase modelling [23] and in controlled laboratory experiments with small-scale braided rivers [9]. However, how small changes at

[1] We use 'rivers' as unifying terminology for flowing water, including erosive rivulets, streams, brooks, and large sediment-laden rivers that build their own landscapes through sedimentation.

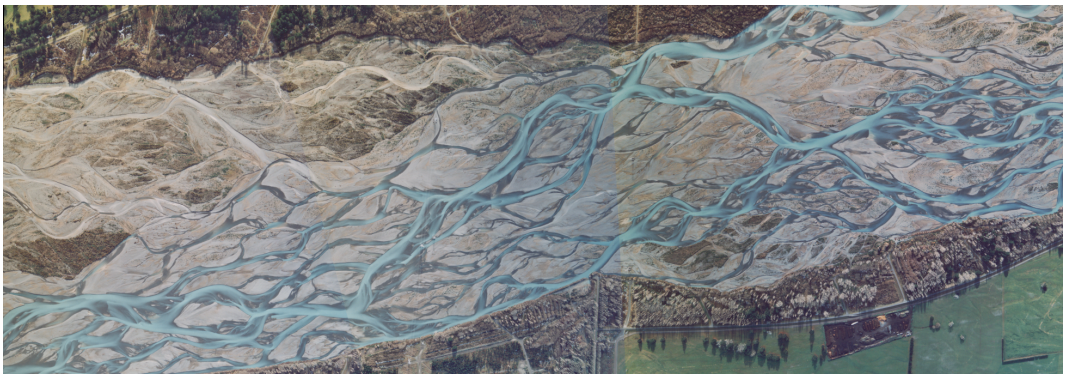


Figure 1 Orthoimagery of the Crossbank reach of the Waimakariri river, a braided gravel river in New Zealand. Thanks to Murray Hicks from NIWA Christchurch, NZ.

bifurcations and confluences propagate and grow or decay through the channel network and how that leads to changes in the larger-scale network structure remains unknown. A main reason is that there is currently no technique to extract morphologically meaningful networks, nor to rigorously connect the networks at different water levels [12, 18].

Modeling braided rivers, where channels can both split and merge, is considerably more complex than modeling standard drainage networks, where all rivers flow only downhill and do not bifurcate. In this paper we initiate the study of braided rivers from the perspective of computational geometry and topology.

Modeling braided rivers in a nutshell. To model a braided river we first need a representation of the basic geometry, independent of water level. We hence use the elevation of the river bed as a starting point. In meandering rivers the so-called *thalweg* is often used as a basic representation of the river. The thalweg is defined as the deepest part of a continuous channel. In a meandering river that is simply one linear feature, which is usually not the channel center line but more sinuous as it follows the deeper pools in the outer bends. We are striving for a similar representation for braided rivers, consisting of linear features along *lowest paths* in each channel. These linear features can merge and bifurcate, that is, they form a planar graph or network. The use of graphs to model and analyze braided rivers was recently pioneered in [18]. In a river, channels are deepest where water flows (or used to flow) the fastest. This is due to the non-linearity of sediment transport: when velocity doubles, sediment transport increases eight to perhaps hundredfold, which leads to the erosion of channels and deposition in bars. We define lowest paths intuitively as the paths that do not go higher than they need to go to connect the endpoints. To find lowest paths we use a *descending quasi Morse-Smale complex* [10, 24]. We show that lowest paths must always lie on this complex, except for, possibly, the ends of the path.

A representative network for a braided river should not necessarily contain all possible channels. Topologically speaking a tiny local maximum in the river bed creates two channels. We can use persistence to simplify our input and avoid such situations. But still, too many channels might remain. We would hence like to select a set of channels which are sufficiently different from each other. We model how different two channels are with a function (the *sand function*) that relates to the volume of sediment the river has to move before the two channels become one. More volume needs more time to be removed [13]. A bar of very small volume separating two channels requires insignificant time to be removed, so the channels are not significantly different. But a large bar with a large volume may require multiple

floods to be shaved off or cut through by a new channel, meaning that the two channels separated by this bar are significantly different.

Our objective now is to compute a representative network of channels that is optimal in some sense. We require (*i*) each channel to be (mostly) on the descending quasi Morse-Smale complex (that is, *locally lowest*), (*ii*) any two channels to be sufficiently *different* (specified by a parameter δ and the sand function), and (*iii*) the representative network to be *maximum*. Unfortunately, the exact formulation of this problem is NP-hard. We deal with this by computing a *striation*: a left-bank-to-right-bank sequence of non-crossing paths for the whole braided river. We provide three different heuristics to compute a striation, each with various strengths and weaknesses. We then limit the representative network to select channels only from the striation. To make this selection, we compute the sand function between every two channels. For this we compute a monotone isotopy between any two channels, to ensure that only sand between the channels is measured, and without multiplicity. We require this isotopy to be consistent with the striation. We then present three different models to compute the sand function along a single matching path of the isotopy, which is then integrated over the entire isotopy to compute the sand function between two channels. The isotopy that minimizes the sand function is chosen. We show that the resulting sand function can be computed efficiently between any two channels of the striation. Finally, we can use a simple greedy algorithm to select the representative network from the striation.

Organization. Section 2 gives the definitions and the problem statement. Section 3 describes the quasi Morse-Smale complex and shows that lowest paths lie mostly on this complex. Section 4 shows NP-hardness of computing a representative network and introduces three ways of computing a striation. Section 5 gives three ways of defining the sand function and algorithms to compute it. Section 6 describes how to compute a representative network if a striation and a sand function are given. Section 7 reports on experiments of an implementation applied to data from a numerical model and to data collected from the real-world Waimakariri river (see Fig. 1). Finally, a discussion is given in Section 8.

2 Definitions and problem statement

Let $G = (V, E)$ be a triangulation of a topological disk M in the plane, and let $h\colon M \to \mathbb{R}$ be the *height* of the points in M, where the edges (respectively triangles) of G interpolate h linearly between its vertices (respectively edges). So G can be viewed as a simplicial 2-complex in $\mathbb{R}^3$ by adding h as a third dimension. Furthermore, let $\sigma \in V$ be a *source* and $\tau \in V$ be a *sink*, both on the boundary ∂M. The source and sink are assumed to be vertices for simplicity, but our approach can be extended to the case where σ and τ are connected components on the boundary of M. We refer to (G, h, σ, τ) as a *river*, and we refer to the volume $\{(x, y, z) \mid (x, y) \in M, z \in \mathbb{R}, z \leq h(x, y)\}$ as *sand*. We define the amount of sand above a point (x, y, z) as $\max(0, h(x, y) - z)$.

Let $\pi^+_{\partial M}$ and $\pi^-_{\partial M}$ (respectively clockwise and counterclockwise) be the two paths from σ to τ along the boundary of M. We call a path π from σ to τ *semi-simple* if it has no self-intersections, but it may coincide with itself, see the red path in Fig. 2. In such self-coinciding parts, these parts are symbolically separated. Let $\mathcal{P}$ be a set of semi-simple, pairwise non-crossing paths from σ to τ along edges of G. For two semi-simple paths π_0 and π_1 in $\mathcal{P}$, let $D(\pi_0, \pi_1)$ be the region bounded by and including π_0 and π_1. So two semi-simple paths π_0 and π_1 from σ to τ have no proper crossings if and only if $D(\pi^+_{\partial M}, \pi_0) \subseteq D(\pi^+_{\partial M}, \pi_1)$ or $D(\pi^+_{\partial M}, \pi_1) \subseteq D(\pi^+_{\partial M}, \pi_0)$.

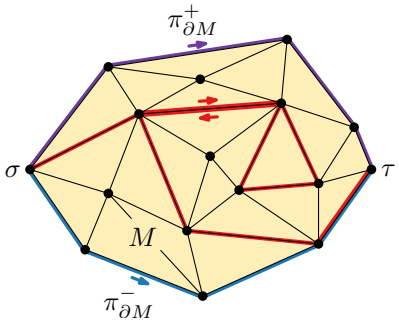


Figure 2 The disk M and three paths of $\mathcal{P}$ without proper crossings, including the two paths $\pi^+_{\partial M}$ and $\pi^-_{\partial M}$ and a backtracking path.

A *homotopy* $\eta\colon [0,1]^2 \to M$ from π_0 to π_1 is a continuous map, such that $\eta(p,0) = \pi_0(\alpha_\eta(p))$, $\eta(p,1) = \pi_1(\beta_\eta(p))$, $\eta(0,t) = \sigma$ and $\eta(1,t) = \tau$, where reparameterizations α_η and $\beta_\eta\colon [0,1] \to [0,1]$ are continuous non-decreasing surjections aligning π_0 and π_1. We refer to $(\alpha_\eta, \beta_\eta)$ as the *matching* between π_0 and π_1 given by η. We can equip a homotopy η with a height function $\zeta\colon [0,1]^2 \to \mathbb{R}$ and define the surface $\Sigma^\zeta_\eta\colon [0,1]^2 \to \mathbb{R}^3$ as $\Sigma^\zeta_\eta(p,t) = (\eta(p,t), \zeta(p,t))$. Define the *volume* above Σ^ζ_η as the total volume of sand above the points of Σ^ζ_η (counted with multiplicity) given by

$$vol(\Sigma^\zeta_\eta) = \iint_{[0,1]\times[0,1]} \max(0, h(\eta(p,t)) - \zeta(p,t)) \left\| \frac{\partial \eta}{\partial p} \times \frac{\partial \eta}{\partial t} \right\| \mathrm{d}p\,\mathrm{d}t.$$

Generally, we will choose $\eta(p,t)$ in such a way that it does not surpass the height of $\eta(p,0)$ or $\eta(p,1)$, so as to measure at least the volume of sand 'above' an extremal path (π_0 or π_1).

We measure the similarity between two paths using a *sand function* $d\colon \mathcal{P} \times \mathcal{P} \to \mathbb{R}$, and we say a path π_0 is δ-*dissimilar* to π_1 if and only if $d(\pi_0, \pi_1) \geq \delta$. The function d will generally not be a metric, since it is generally not symmetric, does not satisfy the triangle inequality, and $d(\pi_0, \pi_1)$ can be 0 for distinct paths π_0 and π_1. Intuitively, we define $d(\pi_0, \pi_1)$ in such a way that measures the volume of sand that lies between π_0 and π_1. Since the height of both paths may vary along the length of the river, it is unclear how to define this volume in a natural way. We define $d(\pi_0, \pi_1) = vol(\Sigma^\zeta_\eta)$, so the sand measured by d depends largely on the homotopy η and the corresponding height function ζ. We discuss how to choose η between π_0 and π_1 in Section 5.

Our goal is to compute a network whose paths represent channels in a river. Essentially, such paths minimize the distance spent at high elevations. We define the cost of a path as its *lexicographic height* [21]. For a path $\pi\colon [0,1] \to M$, define $\pi_h\colon [0,1] \to M \times \mathbb{R}$ to be the path $(\pi(p), h(\pi(p)))$ over the terrain, and let $\rho(\pi, z)$ be the length of the path π_h that has height at least z. We say a path π_0 is *lower* than π_1 if and only if there exists a $z^* \in \mathbb{R}$, such that for all $z \geq z^*$, $\rho(\pi_0, z) = \rho(\pi_1, z)$ and for all $\varepsilon > 0$, there is some $z' \in (z^* - \varepsilon, z^*)$ with $\rho(\pi_0, z') < \rho(\pi_1, z')$. A path is *lowest* if no lower paths exist. We motivate this choice by the property of Lemma 2 that lowest paths follow steepest descent – a property often assumed for water flow as well [26]. Instead of stopping in local minima, lowest paths can proceed by taking a steepest descent from a saddle point in reverse (note that this is *not* a steepest ascent path from the minimum).

Call a subset $\Pi \subseteq \mathcal{P}$ of paths a δ-*network* if no pair of paths in Π has proper crossings, and $d(\pi_0, \pi_1) \geq \delta$ if the lexicographic height of $\pi_0 \in \Pi$ is at least that of $\pi_1 \in \Pi$. Intuitively, a *representative* δ-network is one that contains as many lowest paths as possible. More precisely,

if Π and Π' are δ-networks, then Π' is *better* than Π if there exists some $k \leq \min\{|\Pi|+1, |\Pi'|\}$, such that for each $i < k$, the i-th lowest path of Π and that of Π' are equally low, and either $|\Pi| < k$, or the k-th lowest path of Π' is lower than that of Π. A δ-network is *representative* if no better δ-networks exist.

Assumptions and problem statement. Assume that a polyhedral terrain is given: a triangulation with n vertices that have a height, and linear interpolation over edges and triangles. We assume that all vertices have a different height and any two edges incident to the same vertex have different slope. The latter assumption ensures that steepest descent and steepest ascent over edges is unique from every vertex. We artificially connect all vertices of the boundary of the terrain to an extra vertex v_∞ that is higher than all vertices on the boundary. All local minima on the boundary will stay local minima, whereas all local maxima on the boundary become regular. We do not need a geometric embedding for this modification.

Given such a modified terrain, a source σ, a sink τ, and a difference parameter δ, we study the problem of computing a representative δ-network over the edges of the triangulation for various choices of the sand function d.

3 Morse-Smale complex and lowest paths

An important topological tool for computing a representative δ-network, and lowest paths in particular, is the Morse-Smale complex. Here we briefly introduce some of the concepts related to (quasi) Morse-Smale complexes that are relevant to our work. We do so in a simplified manner to make the paper more accessible (full technical details are given in [10]). We furthermore specify the relation between Morse-Smale complexes and lowest paths.

3.1 Morse-Smale complex

Let $\mathbb{M}$ be a smooth, compact 2-dimensional manifold without boundary, and let $h\colon \mathbb{M} \to \mathbb{R}$ be a height function on $\mathbb{M}$. A point p on $\mathbb{M}$ is *critical* with respect to h if all partial derivatives vanish at p. Otherwise, p is called *regular*. There are three types of critical points: (local) minima, (local) maxima, and saddle points. For each regular point p we define the path of *steepest ascent* (or *steepest descent*) as the path that follows the gradient of h at p. These paths are also known as *integral lines* and are open at both ends, with at each end a critical point. Using these integral lines, we can subdivide $\mathbb{M}$ as follows: two regular points p and q belong to the same cell if the integral lines through p and q end at the same critical points on both sides. The resulting complex is known as the *Morse-Smale complex*, or MS-complex in short. Note that if one of the endpoints of an integral line is a saddle point, then the corresponding cell is 1-dimensional. We refer to 2-dimensional cells of the MS-complex as *MS-cells*, 1-dimensional cells as *MS-edges*, and the vertices simply correspond to the critical points. It can be shown ([10]) that every MS-cell is a quadrilateral with a minimum, a saddle, a maximum, and again a saddle along the boundary of the cell in that order.

Our input consists of a triangulation, and all paths must follow the edges of the triangulation. Therefore, the MS-complex as defined above is not directly useful for our purpose. Instead, we use a quasi MS-complex as defined in [10]. A quasi MS-complex has the same combinatorial structure as an ordinary MS-complex, but it can be required to follow the edges of a triangulation. Let v be a vertex of the triangulation, and let $S(v)$ be the *edge star* of v consisting of the set of edges incident to v. Note that, if we order the edges of $S(v)$ around v, then the endpoints of two neighboring edges must be connected by an edge in the triangulation. The *lower edge star* $S^{\downarrow}(v)$ consists of the subset of edges whose

endpoints are lower than v with respect to h. Symmetrically, we can define the upper edge star $S^{\uparrow}(v) = S(v) \setminus S^{\downarrow}(v)$ (recall that we have no horizontal edges). The lower edge star can naturally be subdivided into *wedges* of consecutive edges in $S^{\downarrow}(v)$ separated by edges in $S^{\uparrow}(v)$. We can now classify points based on the wedges in its lower edge star: minima have 0 wedges, regular points have 1 wedge but not the whole edge star, saddles have 2 or more wedges, and maxima have one complete wedge ($S(v) = S^{\downarrow}(v)$). Given these definitions, we can construct a quasi MS-complex as follows. From every saddle point v we construct a steepest descent path in every wedge of $S^{\downarrow}(v)$ until it reaches a minimum. Similarly, we construct a steepest ascent path in every wedge of $S^{\uparrow}(v)$ until it reaches a maximum. This construction can lead to crossings, as a steepest descent path may cross a steepest ascent path (two steepest descent paths may merge, but will never cross). Therefore, to ensure the correct combinatorial structure, the construction of a path must also end if it encounters an already constructed path. The resulting subdivision is a quasi MS-complex and thus depends on the order in which the paths are constructed. Since, for our purposes, the steepest descent paths are more important, we require that all steepest descent paths are constructed first. In fact, we do not need the steepest ascent paths, and hence we will completely omit these paths from the construction. The resulting complex is commonly referred to as a *descending* (quasi) MS-complex [24]. The cells of a descending MS-complex are bounded by alternating minima and saddle points, and every cell contains exactly one maximum. In the remainder of this paper we refer to the descending quasi MS-complex as constructed above simply as the MS-complex, unless stated otherwise. The same rule applies to the components of the complex, namely the MS-cells and MS-edges.

The quasi MS-complex and therefore the descending quasi MS-complex can be computed in $O(n \log n)$ time [10].

3.2 Lowest paths

The following lemmas state that the lowest paths lie mostly on the MS-complex. The proofs can be found in the full version of the paper. We also sketch how lowest paths can be computed.

▶ **Lemma 1.** *Let π be the lowest path between two vertices u and v in G. Then the highest point of π is at u, v, or a saddle point.*

▶ **Lemma 2.** *Let u and v be two adjacent vertices on a lowest path π with $h(u) > h(v)$. Then the edge (u, v) must be the edge of steepest descent among the edges in the wedge of $S^{\downarrow}(u)$ that contains (u, v).*

▶ **Lemma 3.** *Let π be the lowest path between two vertices u and v in G, where both u and v lie on MS-edges. Then all vertices of π lie on MS-edges.*

▶ **Lemma 4.** *Let π be the lowest path between two vertices u and v in G, and let u' and v' be the first vertices on an MS-edge encountered by following the steepest descent path from u and v, respectively. Then π is the concatenation of the steepest descent path from u to u', the lowest path from u' to v', and the steepest descent path from v to v' in reverse.*

We can compute lowest paths efficiently using the MS-complex. Specifically, we construct a lowest path tree from the source vertex as follows. We start with a disconnected set of vertices consisting of the source and all minima. Then we add all saddle points in increasing order of height. For every saddle, we add the saddle point and the MS-edges to adjacent minima in the MS-complex. However, if two (or more) minima are already in the same connected

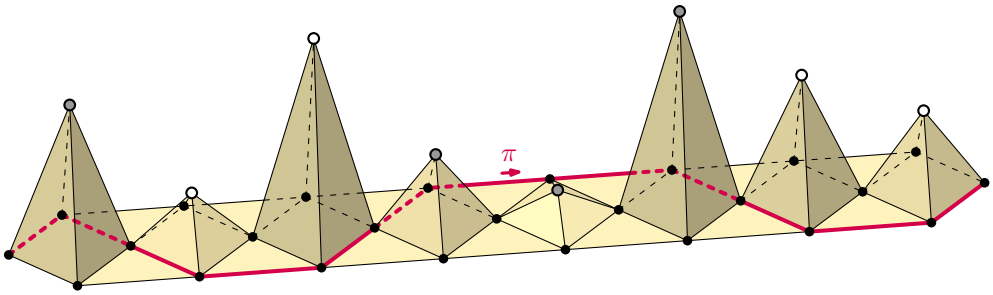


Figure 3 Example of a path π such that $\{\pi^+_{\partial M}, \pi, \pi^-_{\partial M}\}$ is a δ-network if π partitions the total volume of the pyramids equally. Points in the different subsets are shown in white and gray.

component in the lowest path tree, then we remove the first edges of the corresponding MS-edges (the edges incident to the saddles), except for the first edge that descends the steepest. After all saddle points are added, we compute the steepest descent edge for all vertices internal to MS-cells. This results in the correct lowest path tree by Lemma 3 and Lemma 4. The method runs in $O(n + m \log m)$ time using sorting, where m is the number of critical vertices.

4 Striation

Before we specify the sand function needed to construct a δ-network, we can consider the complexity of the problem of computing a representative δ-network for a generic sand function. Let π_0 and π_1 be two semi-simple non-crossing paths from σ to τ. For any reasonable sand function d, the value of $d(\pi_0, \pi_1)$ should measure only sand in the region $D(\pi_0, \pi_1)$ in between the paths. To enforce this, we can restrict the domain of the homotopy η from π_0 to π_1 to $D(\pi_0, \pi_1)$. Furthermore, we want to ensure that each volume of sand is measured at most once. This can be achieved by restricting η to be an isotopy,[2] as well as monotone.[3]

Now consider an input terrain consisting of a sequence of pyramids with different heights as shown in Figure 3, where all non-peak vertices are at height 0. Let π_0 and π_1 be two paths from source to sink at height 0 and let P be the set of pyramids in $D(\pi_0, \pi_1)$. We say that a sand function d is *well-behaved* if $d(\pi_0, \pi_1) = \sum_{p \in P} vol(p)$ where $vol(p)$ is the sand volume of the pyramid p. It is easy to see that computing a representative δ-network for a terrain of this type is NP-hard if the sand function is well-behaved, by reduction from Partition (see Fig. 3). We note that all sand functions that we use are well-behaved.

To make the problem tractable, we put a restriction on the paths that can be used in a representative δ-network. It is not sufficient to restrict the paths to be monotone, since the NP-hardness reduction still works for monotone paths. As a stronger restriction we can specify a monotone isotopy η between the boundaries $\pi^-_{\partial M}$ and $\pi^+_{\partial M}$. We require every path in a δ-network to be a level curve of η. This ensures that the candidate paths cannot cross each other or themselves. The resulting δ-network then strongly depends on the choice of η. Naturally we want to choose η in such a way that many low paths are included as candidates. To that end we use the MS-complex. Instead of completely specifying η, we specify only a discrete subset thereof consisting of suitable candidate paths. We define a *striation* $\mathcal{S}$ as an

[2] That is, a homotopy whose intermediate curves have no self-intersections.
[3] That is, for $t \leq t'$, $\eta(\cdot, t)$ and $\eta(\cdot, t')$ have no proper crossings and $\eta(\cdot, t) \subseteq D(\eta(\cdot, 0), \eta(\cdot, t'))$.

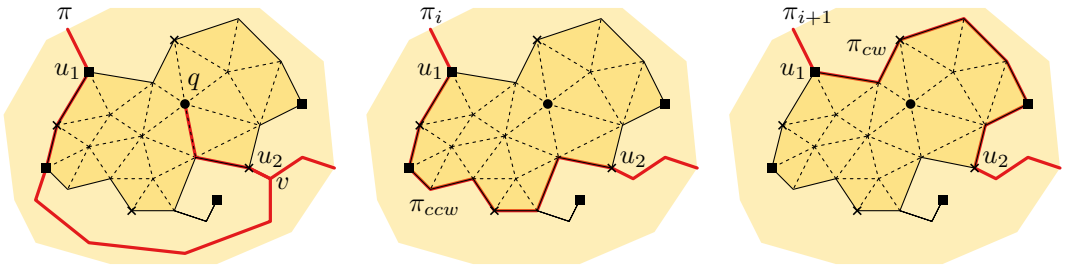


Figure 4 Computing the striation paths around the MS-complex cell c of a maximum q; the cell c is shown darker and triangulated.

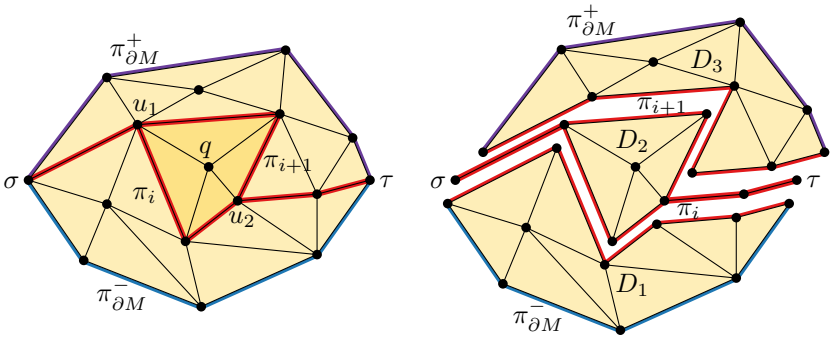


Figure 5 Splitting the triangulation by the striation paths around an MS-complex cell.

ordered set of non-crossing paths $\mathcal{S} = \{\pi_0, \ldots, \pi_r\}$ from σ to τ with $\pi_0 = \pi^-_{\partial M}$ and $\pi_r = \pi^+_{\partial M}$. Every path in a striation must be composed of MS-edges and between every two consecutive paths π_i and π_{i+1} in a striation there can be at most one MS-cell and possibly several one-dimensional features. The one-dimensional features arise from overlapping MS-edges or from the way the striation is computed. Note that every striation can be completed to a monotone isotopy between $\pi^-_{\partial M}$ and $\pi^+_{\partial M}$.

Computing a striation. The hardness result of the previous section also directly implies that computing a striation that includes a representative δ-network is NP-hard. Therefore we consider several heuristics. For every heuristic we require that the lowest path between source and sink is in the striation. This is possible, since by Lemma 3, the lowest path is composed of MS-edges.

Two of our three heuristics for computing the striation use the *persistence* of local maxima. The persistence of a local maximum is the difference in height to a saddle with which it is paired, and a local maximum is paired with the lowest saddle on a highest path to a higher maximum. It can be computed in linear time from the contour tree, which can be computed in $O(n + m \log m)$ time [7, 8] when there are m critical points. All local maxima except for v_∞ will be paired and have their persistence defined.

Iterated lowest path. As a first step we compute the lowest path π from source to sink. We then subdivide G along π into two parts: $D_1 = D(\pi, \pi^-_{\partial M})$ and $D_2 = D(\pi^+_{\partial M}, \pi)$. Next, we recursively apply the algorithm in D_1 and D_2. The obtained striations $\mathcal{S}_1$ and $\mathcal{S}_2$ can then be concatenated to obtain the final striation $\mathcal{S} = \{\mathcal{S}_1, \pi, \mathcal{S}_2\}$. Finally we can add $\pi^-_{\partial M}$ and $\pi^+_{\partial M}$ to the striation. The recursion stops when G contains at most one MS-cell. We have to be careful when computing π, since π must be distinct from $\pi^-_{\partial M}$ and $\pi^+_{\partial M}$.

More precisely, π must have an MS-cell on both sides. Since π is also a path in D_1 and D_2, we must often compute the second or third lowest path. We can do so by computing the lowest paths through all edges that are not in the lowest path tree. The lowest path that is not one of the boundary paths can then easily be obtained.

Highest persistence first. The first path π is obtained by computing the lowest path from source to sink that passes through the maximum q with the highest persistence (excluding v_∞). Since π actually consists of two lowest paths (from source to q, and from q to sink), π has the form of a path π' with a special vertex v from which there is a path to q and back to v (Lemma 4). We now subdivide G as follows. Let c be the MS-cell containing q, and let u_1 and u_2 be the first and last vertices of π that are on the boundary of c, respectively (see Fig. 4). Furthermore, let π_{cw} and π_{ccw} be the paths between u_1 and u_2 along the boundary of c in clockwise and counterclockwise direction, respectively. We can now obtain the path π_i as the concatenation of the subpath of π from σ to u_1, the path π_{ccw}, and the subpath of π from u_2 to τ. Similarly, we can obtain π_{i+1} by replacing π_{ccw} by π_{cw} in π_i. If $u_1 = u_2$, then either π_i or π_{i+1} may backtrack from u_1. In that case we can replace the respective path with π'. The paths π_i and π_{i+1} subdivide G into three parts (see Fig. 5): $D_1 = D(\pi_i, \pi^-_{\partial M})$, $D_2 = D(\pi_{i+1}, \pi_i)$, and $D_3 = D(\pi^+_{\partial M}, \pi_{i+1})$. Since D_2 contains only one MS-cell, we recurse only in D_1 and D_3 to obtain $\mathcal{S}_1$ and $\mathcal{S}_3$. The final striation then consists of $\mathcal{S} = \{\mathcal{S}_1, \pi_{i+1}, \pi_i, \mathcal{S}_3\}$. During recursive calls we do not recompute the persistence of maxima, but use the persistence computed initially.

Hybrid. We first compute the lowest path π from source to sink. Next we pick the maximum q with the highest persistence among all maxima for which π contains an MS-edge on the boundary of the corresponding MS-cell. This heuristic then proceeds in the same way as the highest persistence first heuristic, using the lowest path through q.

It is easy to verify that all heuristics produce a striation and that this striation includes the lowest path from source to sink (none of the chosen paths can cross the lowest path). The complexity of the striation – the summed complexities of its paths – is $O(nm)$ in all cases, where m is the number of local maxima. The first heuristic may seem the most natural. However, lowest paths computed in recursive steps will often stay close to the original lowest path and differ only in bars with low persistence. It may make more sense to deviate first in the bar with highest persistence, as this may lead directly to a path that is δ-dissimilar to the lowest path. This is what the second heuristic is designed for. However, the lowest path through the maximum with highest persistence might require a long backtracking path to reach this maximum. This will force many paths to go around possibly the wrong side of a bar. Note that, in a sense, this problem is dual to the problem of the first heuristic. The final heuristic, a hybrid of the first two, tries to alleviate this problem by requiring the chosen maximum to be close to the lowest path.

5 Sand function

Before we can compute a representative δ-network from a striation, we need to define the sand function and show how to compute it for two paths. Let π_i and π_j $(i < j)$ be two paths in the striation. To compute $d(\pi_i, \pi_j)$ we first need to specify a monotone isotopy η between π_i and π_j. For consistency, we require η to be monotone with respect to the striation. An isotopy η is *striation monotone* with respect to a striation $\mathcal{S} = \{\pi_0, \ldots, \pi_r\}$ if for every k $(i \le k \le j)$ there exists a t_k such that $\eta(\cdot, t) \subseteq D(\pi_i, \pi_k)$ for all $t \le t_k$ and $\eta(\cdot, t) \subseteq D(\pi_k, \pi_j)$ for all $t \ge t_k$. Intuitively, for every path π_k in the striation between π_i and π_j there must exist a level curve of η that matches π_k, and other level curves cannot

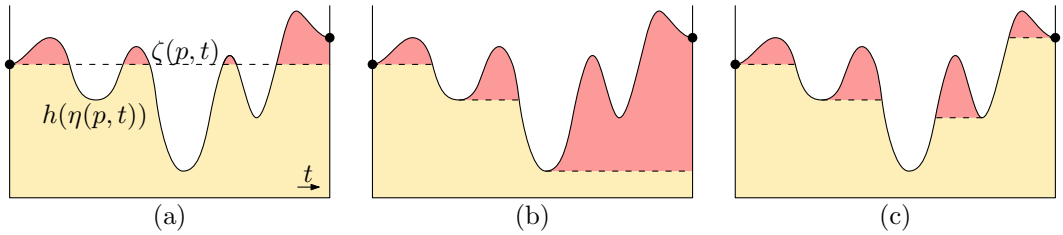


Figure 6 The area of sand (shaded) above curve $\zeta(p,\cdot)$ (dashed), where ζ is minimizing in the (a) water level model, (b) water flow model, (c) symmetric flow model.

cross π_k. Similarly, a path $f\colon [0,1] \to M$ with $f(0) \in \pi_i$ and $f(1) \in \pi_j$ is *striation monotone* with respect to a striation $\mathcal{S} = \{\pi_0, \ldots, \pi_r\}$ if for every k $(i \leq k \leq j)$ there exist a t_k such that $f(t) \in D(\pi_i, \pi_k)$ for $t \leq t_k$ and $f(t) \in D(\pi_k, \pi_j)$ for $t \geq t_k$. Note that every *matching curve* $\eta(p,\cdot)$ of a striation monotone isotopy η is striation monotone.

We now define the sand function along a matching curve $\eta(p,t)$ of the isotopy (for some fixed p) between π_i and π_j. This function can then be extended to a sand function $d(\pi_i, \pi_j)$ from π_i to π_j as described in Section 2. We consider three different models (see Fig. 6):

Water level model. This sand function simply measures the amount of sand above $h(\eta(p,0))$. Intuitively, the sand function measures the minimum amount of sand that needs to be removed such that π_i and π_j merge if the water level is at the height of π_i. Formally, we set $\zeta(p,t) = h(\eta(p,0))$.

Water flow model. This sand function measures the amount of sand that needs to be removed such that the function $h(\eta(p,\cdot))$ becomes non-increasing. Formally, we set $\zeta(p,t) = \min_{t' \leq t} h(\eta(p,t'))$.

Symmetric flow model. This sand function measures the amount of sand that needs to be removed such that $h(\eta(p,t))$ is unimodal. Formally, we set $\zeta(p,t) = \max(\min_{t' \leq t} h(\eta(p,t')), \min_{t' \geq t} h(\eta(p,t')))$.

Note that the sand functions in the water level model and the water flow model are not symmetric, whereas the sand function in the symmetric flow model is symmetric. In general we do not assume that the sand function is symmetric.

Finally, we need to compute the isotopy η between two paths π_i and π_j. In general we define the sand function as $d(\pi_i, \pi_j) = \inf_\eta vol(\Sigma_\eta^\zeta)$, where ζ is defined by the model as described above. Below we describe how to compute $d(\pi_i, \pi_j)$ only for the water flow model, but it is straightforward to extend the same approach to the other models.

Following the definition of the water flow model, we say that π_i is *similar* to π_j if there exists a striation monotone isotopy η such that all matching curves are non-increasing in h. Note that $d(\pi_i, \pi_j) = 0$ if π_i is similar to π_j. To be able to argue the correctness of our computation, we introduce a different but related definition. We say that π_i is *pseudo-similar* to π_j if, for every point q in $D(\pi_i, \pi_j)$, there exists a point $p \in \pi_i$ and a striation monotone path π from p to q such that π is non-increasing in h. Omitted proofs can be found in the full version of the paper.

▶ **Lemma 5.** *If π_i is pseudo-similar to π_j, then $d(\pi_i, \pi_j) = 0$.*

Computing the sand function. We now show how to compute $d(\pi_i, \pi_j)$ efficiently in the water flow model. The algorithm can operate directly on the MS-complex. First consider two consecutive paths π_i and π_{i+1} in the striation. Let $B(\pi_i, \pi_{i+1})$ and $T(\pi_i, \pi_{i+1})$ be the sets of critical points that are bounding the MS-cell c_i between π_i and π_{i+1} and are on π_i

and π_{i+1}, respectively. Now let h^* be the maximum height among all points in $B(\pi_i, \pi_{i+1})$ and assign h^* to c_i. We claim that the volume of sand in c_i above h^* equals $d(\pi_i, \pi_{i+1})$. For non-consecutive paths π_i and π_j, we need to propagate the heights. After assigning h^* to c_i, we also lower the height of all points in $T(\pi_i, \pi_{i+1})$ to h^* (points with lower height than h^* keep their height). We continue the propagation until we reach π_j. In case there is a one-dimensional feature between two consecutive paths, then we also need to propagate the height over this one-dimensional feature separately from propagating over MS-cells. However, note that one-dimensional features do not add any volume to the sand function. This process assigns certain height values h_k^* to each of the MS-cells c_k between π_i and π_j, and $d(\pi_i, \pi_j) = \sum_{k=i}^{j-1} vol(c_k, h_k^*)$, where $vol(c_k, h_k^*)$ is the volume of sand in c_k above h_k^*.

▶ **Lemma 6.** *If $D(\pi_i, \pi_j)$ is altered by lowering every MS-cell c_k in $D(\pi_i, \pi_j)$ to h_k^*, then π_i is pseudo-similar to π_j.*

▶ **Lemma 7.** *For any two paths π_i and π_j in the striation $d(\pi_i, \pi_j) \geq \sum_{k=i}^{j-1} vol(c_k, h_k^*)$.*

Proof. For the sake of contradiction, assume that we can make π_i similar to π_j by removing less than $\sum_{k=i}^{j-1} vol(c_k, h_k^*)$ volume of sand from $D(\pi_i, \pi_j)$. By Lemma 5 we can then also make π_i pseudo-similar to π_j by removing the same volume of sand. Now there must be some point q in some MS-cell c_k (if q is on an MS-edge, then let k be as small as possible) that has height higher than h_k^* in the altered $D(\pi_i, \pi_j)$. We now show by induction on k that there is no striation monotone non-decreasing path from q to π_i, refuting that π_i is pseudo-similar to π_j. If $k = i$, then any non-decreasing path from q to π_i must end at some point q' on the boundary of c_i. But since the height of q' can be at most the maximum height of all points in $B(\pi_i, \pi_{i+1})$, which is h_i^*, this is not possible. If $k > i$, then any non-decreasing path from q to π_i must hit π_k at some point q' at a height above h_k^*. If $q' \in \pi_i$, then there must exist a point among $B(\pi_k, \pi_{k+1})$ that is on π_i and has height above h_k^*. However, this is not possible by construction. Otherwise, q' lies on some MS-cell c_l with $l < k$. By construction $h_l^* \leq h_k^*$ and thus q' has height higher than h_l^*. The result now follows by induction. ◀

Lemma 6 and Lemma 7 prove that we can compute $d(\pi_i, \pi_j)$ as $\sum_{k=i}^{j-1} vol(c_k, h_k^*)$. To compute the values h_k^* we simply need to propagate the height values on the MS-complex by following the striation. To compute $vol(c_k, x)$ efficiently for some height x, we can preprocess each MS-cell. Note that $vol(c_k, x)$ is a monotone function of $O(n_k)$ complexity, where n_k is the number of vertices in c_k. After computing this function in $O(n_k \log n_k)$ time using a plane sweep, we can query $vol(c_k, x)$ for any height x in $O(\log n_k)$ time. The total preprocessing time is $O(n \log n)$, after which the sand function from any π_i to any π_j can be determined in $O(|i - j| \log n) = O(m \log n)$ time.

To compute the sand functions for other models, we only need slight modifications. For the water level model we need the following modification: when we propagate a height h_k^* to all points in $T(\pi_i, \pi_{i+1})$, we also update heights lower than h_k^* to h_k^*. The symmetric flow model requires the propagation in both directions, that is from π_i to π_j and from π_j to π_i. The height assigned to an MS-cell is then the maximum height assigned by any direction of the propagation. The correctness proofs are very similar to those of the water flow model.

6 Representative network

To obtain our representative network, we sort the $O(m)$ paths of our striation $\mathcal{S} = \{\pi_0, \ldots, \pi_r\}$ based on lexicographic height. Each path π_i contains $O(n)$ edges. We preprocess each path in $O(n \log n)$ time to compute its height profile function [21]. After this, we can compare

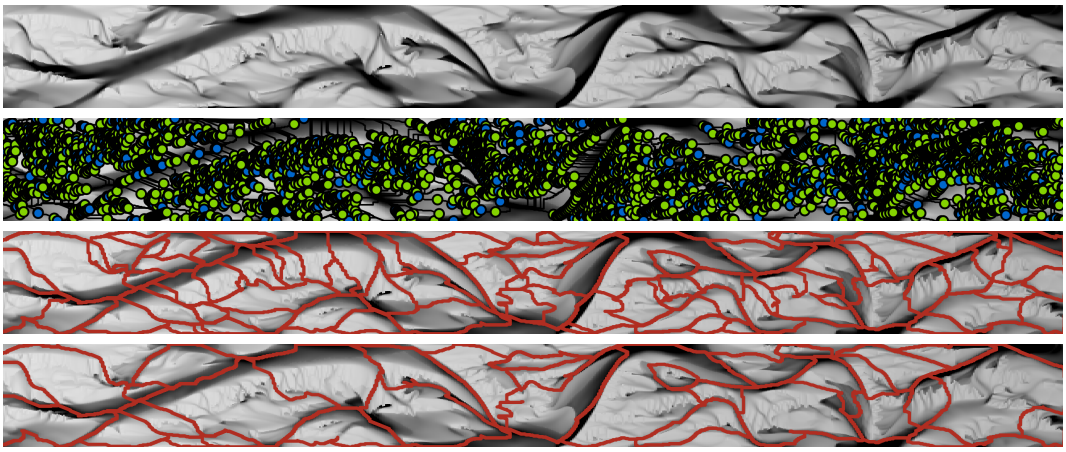


Figure 7 A numerically modeled river, the Morse-Smale complex overlaid, and two representative δ-networks for two different values of δ.

the lexicographic heights of two striation paths in $O(n)$ time. Hence, sorting the paths by lexicographic height takes $O(mn \log n + mn \log m) = O(mn \log n)$ time. By our non-degeneracy assumptions, all paths have different lexicographic height. Assume $\mathcal{S}$ is the sorted set of paths.

To compute the representative network, we initialize the representative network as an empty set $\mathcal{R}$ of paths. We iterate over the paths in $\mathcal{S}$ in order of increasing lexicographic height, and we add a path π to $\mathcal{R}$ if each path π' already in $\mathcal{R}$ has $d(\pi, \pi') \geq \delta$. This can be tested in $O(m \log n)$ time by testing only the two paths of $\mathcal{R}$ in between which π would be inserted. Overall, this yields an $O(mn \log n)$ running time given the striation. By construction we obtain a representative δ-network, that is, no better δ-network exists.

7 Experimental results

We performed experiments on two data sets: a numerically modeled river and a real-world braided river. To do so we implemented the highest persistence first heuristic for the striation and the sand function in the water flow model. The numerically modeled river was created by a state-of-the-art model suite that is used in the civil engineering and fluvial and coastal morphology disciplines worldwide, indicating its usefulness and quality [23]. The real-world case is an iconic braided gravel river in New Zealand called Waimakariri, one of the largest in the world of this type, that was the first with spatial cover of the bed through remote sensing techniques [11].

Figure 7 shows our results for the numerically modeled river. The representative network is capturing most channels of the river. Channels that are added in the denser network cross big bars and are in most cases located in small tie-channels that formed at water levels when water was spilling over the bar. These are essential in the natural braiding dynamics in that these can be the locations where bars split during floods. Furthermore, the sparser network also avoids deep short channels which are connected on one or no end. Some of these formed as filling lows between two merging bars rather than active channels, and the fact that they only partake in the dense network when connected at small tie-channels is evidence that the network method represents the river channel network in these important aspects.

Figure 8 shows a representative network for the Waimakariri. This complex topography has many large remnant channels that became inactive while smaller channels may be

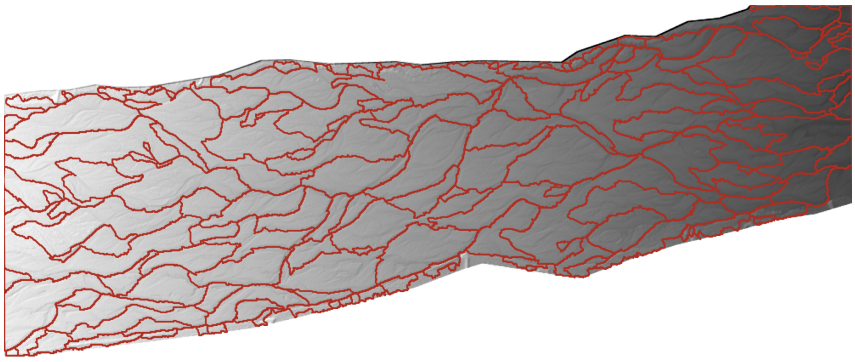


Figure 8 Representative network for the Waimakariri river (same area as shown in Fig. 1).

Figure 9 Mountains splitting the river.

growing. The network captures some smaller channels and leaves out the unconnected remnant channels. The network sometimes loops to flow upstream, meaning that such channels should either not be connected or that minor tie-channels were missed, which can now for the first time be investigated through comparison of sparser and denser networks.

8 Discussion and future work

The representative δ-networks computed using our approach already look very promising, even though we have implemented only one striation heuristic and only one sand function. The next step is a thorough testing on different river bed terrain data using each combination of a striation and a sand function, and with different values of δ. The output can be further assessed by geomorphologists, for example by comparing the computed networks with the corresponding actual river networks at different water levels. Other striations and other sand functions can also be considered, and may give better results in certain scenarios.

In particular, our striation heuristic may give counter-intuitive results when M includes the banks of the river. To illustrate this, consider the two high mountains on both sides of the river banks in Figure 9. It can occur that such mountains are the most significant maxima, and cause the majority of the river to go over the banks, instead of between the mountains. For this reason, we may want to base the significance of maxima also on their distance to the main channel of the river. Our hybrid heuristic already tries to address this issue, but a more refined approach may be needed.

In the future we also want to consider time-varying data on rivers, both as a tool to improve the results by eliminating features that do not persist over time, and as a way to analyze how the river structure evolves over time. However, to perform such an analysis efficiently, we may need to simplify our model.

References

1 Pankaj Agarwal, Mark de Berg, Prosenjit Bose, Katrin Dobrint, Marc van Kreveld, Mark Overmars, Marko de Groot, Thomas Roos, Jack Snoeyink, and Sidi Yu. The complexity

of rivers in triangulated terrains. In *Proc. 8th Canadian Conference on Computational Geometry CCCG'96*, pages 325–330, 1996.

2 Lars Arge, Jeffrey S. Chase, Patrick Halpin, Laura Toma, Jeffrey S. Vitter, Dean Urban, and Rajiv Wickremesinghe. Efficient flow computation on massive grid terrain datasets. *GeoInformatica*, 7(4):283–313, 2003.

3 Peter Ashmore. Channel morphology and bed load pulses in braided, gravel-bed streams. *Geografiska Annaler: Series A, Physical Geography*, 73(1):37–52, 1991.

4 Mark de Berg, Otfried Cheong, Herman Haverkort, Jung-Gun Lim, and Laura Toma. The complexity of flow on fat terrains and its I/O-efficient computation. *Computational Geometry*, 43(4):331–356, 2010.

5 Mark de Berg and Constantinos Tsirogiannis. Exact and approximate computations of watersheds on triangulated terrains. In *Proc. 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 74–83. ACM, 2011.

6 Walter Bertoldi, Luca Zanoni, and Marco Tubino. Planform dynamics of braided streams. *Earth Surface Processes and Landforms*, 34:547–557, 2009.

7 Hamish Carr, Jack Snoeyink, and Ulrike Axen. Computing contour trees in all dimensions. *Computational Geometry*, 24(2):75–94, 2003.

8 Yi-Jen Chiang, Tobias Lenz, Xiang Lu, and Günter Rote. Simple and optimal output-sensitive construction of contour trees using monotone paths. *Computational Geometry*, 30(2):165–195, 2005.

9 Wout M. van Dijk, Wietse I. van de Lageweg, and Maarten G. Kleinhans. Formation of a cohesive floodplain in a dynamic experimental meandering river. *Earth Surface Processes and Landforms*, 38, 2013.

10 Herbert Edelsbrunner, John Harer, and Afra Zomorodian. Hierarchical Morse complexes for piecewise linear 2-manifolds. In *Proc. 17th Annual ACM Symposium on Computational Geometry*, pages 70–79, 2001.

11 D. Murray Hicks, Maurice J. Duncan, and Jeremy M. Walsh. New views of the morphodynamics of large braided rivers from high-resolution topographic surveys and time-lapse video. In *The Structure, Function and Management Implications of Fluvial Sedimentary Systems (Proceedings)*, pages 373–380. IAHS Publ. no. 276, 2002.

12 Alan D. Howard, Mary E. Keetch, and C. Linwood Vincent. Topological and geometrical properties of braided streams. *Water Resources Research*, 6(6), 1970.

13 Maarten G. Kleinhans. Flow discharge and sediment transport models for estimating a minimum timescale of hydrological activity and channel and delta formation on Mars. *Journal of Geophysical Research*, 110, 2005.

14 Maarten G. Kleinhans, Robert I. Ferguson, Stuart N. Lane, and Richard J. Hardy. Splitting rivers at their seams: bifurcations and avulsion. *Earth Surface Processes and Landforms*, 38(1):47–61, 2013.

15 Thierry de Kok, Marc van Kreveld, and Maarten Löffler. Generating realistic terrains with higher-order Delaunay triangulations. *Computational Geometry*, 36(1):52–65, 2007.

16 Marc van Kreveld and Rodrigo I. Silveira. Embedding rivers in triangulated irregular networks with linear programming. *International Journal of Geographical Information Science*, 25(4):615–631, 2011.

17 Yuanxin Liu and Jack Snoeyink. Flooding triangulated terrain. In *Developments in Spatial Data Handling*, pages 137–148. Springer, 2005.

18 Wouter A. Marra, Maarten G. Kleinhans, and Elisabeth A. Addink. Network concepts to describe channel importance and change in multichannel systems: test results for the Jamuna river, Bangladesh. *Earth Surface Processes and Landforms*, 39(6):766–778, 2014.

19 Michael McAllister and Jack Snoeyink. Extracting consistent watersheds from digital river and elevation data. In *Proc. ASPRS/ACSM Annu. Conf*, volume 138, 1999.

20 Gary Parker. On the cause and characteristic scales of meandering and braiding in rivers. *Journal of Fluid Mechanics*, 76(3):457–480, 1976.

21 Günter Rote. Lexicographic Fréchet matchings. In *Abstracts of the 30th European Workshop on Computational Geometry*, 2014.

22 Filip Schuurman, Maarten G. Kleinhans, and Hans Middelkoop. Network response to disturbances in large sand-bed braided rivers. *Earth Surface Dynamics*, 4(1):25–45, 2016.

23 Filip Schuurman, Wouter A. Marra, and Maarten G. Kleinhans. Physics-based modeling of large braided sand-bed rivers: Bar pattern formation, dynamics, and sensitivity. *Journal of Geophysical Research: Earth Surface*, 118(4):2509–2527, 2013.

24 Nithin Shivashankar, Senthilnathan M, and Vijay Natarajan. Parallel computation of 2D Morse-Smale complexes. *IEEE Transactions on Visualization and Computer Graphics*, 18(10):1757–1770, 2012.

25 Rodrigo I. Silveira and René van Oostrum. Flooding countries and destroying dams. *International Journal of Computational Geometry & Applications*, 20(3):361–380, 2010.

26 Sidi Yu, Marc van Kreveld, and Jack Snoeyink. Drainage queries in TINs: from local to global and back again. In *Advances in GIS Research II: Proc. 7th International Symposium on Spatial Data Handling*, pages 829–842, 1997.

A Proof of the Orbit Conjecture for Flipping Edge-Labelled Triangulations

Anna Lubiw[1], Zuzana Masárová[2], and Uli Wagner[3]

1 **School of Computer Science, University of Waterloo, Waterloo, ON, Canada**
alubiw@uwaterloo.ca
2 **IST Austria, Klosterneuburg, Austria**
zuzana.masarova@ist.ac.at
3 **IST Austria, Klosterneuburg, Austria**
uli@ist.ac.at

Abstract

Given a triangulation of a point set in the plane, a *flip* deletes an edge e whose removal leaves a convex quadrilateral, and replaces e by the opposite diagonal of the quadrilateral. It is well known that any triangulation of a point set can be reconfigured to any other triangulation by some sequence of flips. We explore this question in the setting where each edge of a triangulation has a label, and a flip transfers the label of the removed edge to the new edge. It is not true that every labelled triangulation of a point set can be reconfigured to every other labelled triangulation via a sequence of flips, but we characterize when this is possible. There is an obvious necessary condition: for each label l, if edge e has label l in the first triangulation and edge f has label l in the second triangulation, then there must be some sequence of flips that moves label l from e to f, ignoring all other labels. Bose, Lubiw, Pathak and Verdonschot formulated the *Orbit Conjecture*, which states that this necessary condition is also sufficient, i.e. that *all* labels can be simultaneously mapped to their destination if and only if *each* label individually can be mapped to its destination. We prove this conjecture. Furthermore, we give a polynomial-time algorithm to find a sequence of flips to reconfigure one labelled triangulation to another, if such a sequence exists, and we prove an upper bound of $O(n^7)$ on the length of the flip sequence.

Our proof uses the topological result that the sets of pairwise non-crossing edges on a planar point set form a simplicial complex that is homeomorphic to a high-dimensional ball (this follows from a result of Orden and Santos; we give a different proof based on a shelling argument). The dual cell complex of this simplicial ball, called the *flip complex*, has the usual flip graph as its 1-skeleton. We use properties of the 2-skeleton of the flip complex to prove the Orbit Conjecture.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases triangulations, reconfiguration, flip, constrained triangulations, Delaunay triangulation, shellability, piecewise linear balls

Digital Object Identifier 10.4230/LIPIcs.SoCG.2017.49

1 Introduction

The flip operation is fundamental to the study of triangulations of point sets in the plane. A flip removes one edge and replaces it by the opposite diagonal of the resulting quadrilateral, so long as that quadrilateral is convex. Lawson [18] proved the foundational result that any triangulation can be transformed into any other triangulation of the same point set via a sequence of flips. His second proof of this result [19] used the approach that is more widely known – showing that any triangulation can be flipped to the Delaunay triangulation, which then acts as a "hub" through which we can flip any triangulation to any other.

The result that any triangulation can be flipped to any other is captured succinctly by saying that the *flip graph* is connected, where the *flip graph* has a vertex for each triangulation of the given point set, and an edge when two triangulations differ by one flip. The special case of a point set in convex position has been very thoroughly studied. In this case triangulations correspond to binary trees, and a flip corresponds to a rotation. The flip graph in this case is the 1-skeleton of a polyhedron called the *associahedron*.

The use of flips to reconfigure triangulations is relevant to the study of associahedra [28] and mixing [22]. Flips are also important in practice for mesh generation and for finding triangulations that optimize certain quality measures [3, 13]. The survey by Bose and Hurtado [6] discusses these and many other aspects of flips.

Despite the extensive work on flips, it is only recently that the question of where edges go under flip operations has been investigated. This can be formalized by attaching a label to each edge in a triangulation. Throughout, we fix a set P of n points in general position, and we identify triangulations with their edge sets (i.e., a triangulation of P is a maximal set T of pairwise non-crossing edges spanned by P). A *labelled triangulation* $\mathcal{T}$ of P is a pair (T, ℓ) where T is a triangulation of P and ℓ is a *labelling function* that maps the edges of T one-to-one onto the labels $1, 2, \ldots, t_P$. Here t_P is the number of edges in any triangulation of P. When we perform a flip operation on $\mathcal{T}$, the label of the removed edge is transferred to the new edge.

We can now capture "where an edge goes" under flip operations. We say that edges e and f lie in the same *orbit* if we can attach label l to e in some triangulation and apply some sequence of flips to arrive at a triangulation in which edge f has label l. The orbits are exactly the connected components of a graph that Eppstein [14] called the *quadrilateral graph* – this graph has a vertex for every one of the possible $\binom{n}{2}$ edges formed by point set P, with e and f being adjacent if they cross and their four endpoints form a convex quadrilateral that is empty of other points. In particular, this implies that there is a polynomial-time algorithm to find the orbits. The orbits can be very different depending on P. For a point set in convex position, all the non-convex hull edges are in a single orbit [7], but at the other extreme, a point set with no empty convex pentagon has the property that in any triangulation, the edges are all in distinct orbits [14].

Orbits tell us where each individual edge label can go, but not how they combine. The main question we address in this paper is: when is there a sequence of flips to reconfigure one labelled triangulation of point set P to another labelled triangulation of P? A necessary condition is that, for each label l, the edges with label l in the two triangulations must lie in the same orbit. Bose et al. [7] conjectured that this condition is also sufficient. As our main result we prove this "Orbit Conjecture," and strengthen it by providing a polynomial-time algorithm and a bound on the length of the flip sequence.

▶ **Theorem 1** (Orbit Theorem). *Given two edge-labelled triangulations $\mathcal{T}_1$ and $\mathcal{T}_2$ of a point set, there is a flip sequence that transforms one into the other if and only if for every label l, the edges of $\mathcal{T}_1$ and $\mathcal{T}_2$ having label l belong to the same orbit. Furthermore, there is a polynomial-time algorithm that tests whether the condition is satisfied, and if it is, computes a flip sequence of length $O(n^7)$ to transform $\mathcal{T}_1$ to $\mathcal{T}_2$.*

The orbit theorem is stated for triangulations $\mathcal{T}_1$ and $\mathcal{T}_2$ that may have different edge sets, but – since we know how to use flips to change the edge set – the crux of the matter is the special case where the two triangulations have the same edge set T but different label functions ℓ_1 and ℓ_2. In other words, we are given a permutation of the edge labels of a triangulation, and we seek a flip sequence to realize the permutation. Furthermore, since every permutation is a composition of transpositions, we concentrate first on finding a flip

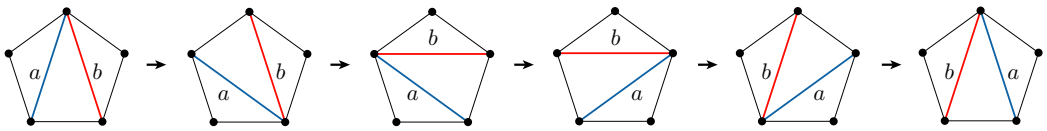


Figure 1 Five flips swap the edge labels (a and b) of two diagonals of a convex pentagon. In the flip graph these five flips form a 5-cycle.

sequence to transpose (or "swap") two labels. This idea of reducing the problem to the case of swaps appears in [7].

One insight to be gained from previous work is that empty convex pentagons in the point set seem to be crucial for swapping edge labels. Certainly, an empty convex pentagon provides a label swap – Figure 1 shows how the edge labels of two diagonals of an empty convex pentagon can be swapped by a sequence of five flips. In the other direction, the special cases of the orbit theorem that were proved by Bose et al. [7] for convex and spiral polygons involved moving pairs of labels into empty convex pentagons and swapping them there. Furthermore, Eppstein [14] showed that in a triangulation of a point set with no empty convex pentagons, no permutations of edge labels are possible via flips.

The foundation of our proof is to make this intuition about empty convex pentagons rigorous. In particular, we show that the only elementary operation that is needed for label permutation is to transpose two labels by moving them into an empty convex pentagon and swapping them there. More formally, given a labelled triangulation $\mathcal{T} = (\mathcal{T}, \ell)$, an *elementary swap* of edges e and f in T is a transposition of the labels of e and f that is accomplished as follows: perform a sequence, σ, of flips on $\mathcal{T}$ to get to a triangulation $\mathcal{T}'$ in which the labels $\ell(e)$ and $\ell(f)$ are attached to the two diagonals of an empty convex pentagon; then perform the 5-flip sequence, π, that transposes these two labels; then perform the sequence σ^{-1}. We say that the sequence $\sigma\pi\sigma^{-1}$ *realizes* the elementary swap. Observe that the effect of $\sigma\pi\sigma^{-1}$ on $\mathcal{T}$ is to transpose the labels of e and f while leaving all other labels unchanged. We will prove that an elementary swap can always be realized by a flip sequence of length $O(n^6)$, and furthermore, that such a sequence can be found in polynomial time.

One of our main results is the following, from which the Orbit Theorem can readily be derived:

▶ **Theorem 2.** *In a labelled triangulation $\mathcal{T}$, two edges are in the same orbit if and only if there is an elementary swap between them.*

In order to prove Theorem 2, we use the following key result:

▶ **Theorem 3** (Elementary Swap Theorem)**.** *Given a labelled triangulation $\mathcal{T}$, any permutation of the labels that can be realized by a sequence of flips can be realized by a sequence of elementary swaps.*

This theorem is proved using topological properties of the *flip complex*, whose 1-skeleton is the flip graph. A result of Orden and Santos [24] can be used to show that the flip complex has the topology of a high-dimensional ball[1]. We give an alternate proof of this. We use the 2-skeleton of the flip complex, and show that its 2-cells correspond to cycles in the flip graph of two types: quadrilaterals, which do not permute labels; and pentagons, which correspond precisely to the 5-cycles of flips shown in Figure 1. Then we prove the Elementary Swap

[1] Technically speaking, the flip complex is homotopy equivalent to a ball.

Theorem by translating it into a result about decomposing closed walks in the flip graph into simpler *elementary walks*.

Although there is a rich literature on associahedra and on cell complexes associated with triangulations of point sets, we are not aware of any previous combinatorial results on triangulations that require topological proofs, as our proof of the Orbit Theorem seems to.

We now briefly describe the rest of our method after the Elementary Swap Theorem is established. In order to prove Theorem 2, we need one more ingredient about the structure of elementary swaps: we will show that any sequence of elementary swaps that moves the label of edge e to edge f can be "completed" to get the label of f back to e, and that, in fact, the resulting sequence provides an elementary swap of e and f.

The high-level idea of our proof of Theorem 2 is then as follows: From our hypothesis that two edges e and f lie in the same orbit, we show that there is a sequence of flips that permutes the labels of triangulation $\mathcal{T}$, taking the label of e to f. The Elementary Swap Theorem then gives us a sequence of elementary swaps to do the same (this is the significant step of the proof). Finally, from the structure of elementary swaps we can then find an elementary swap of e and f.

Our paper is organized as follows. In Section 3 we prove the Elementary Swap Theorem using topological methods. In Section 4 we prove the properties of elementary swaps that were mentioned above. In top-down fashion, we begin in Section 2 by expanding on the high-level ideas, and proving the Orbit Theorem assuming the results in the later sections.

1.1 Background

The diameter of the flip graph of a point set gives the worst-case number of flips required to reconfigure one triangulation to another. For unlabelled triangulations, the diameter of the flip graph is known to be $\Theta(n^2)$, with the upper bound proved by Lawson [18] and the lower bound proved by Hurtado et al. [16]. For the special case of points in convex position, there is an exact bound of $2n - 10$ [28, 26]. The problem of finding the distance in the flip graph between two given triangulations of a point set is NP-hard [20], and even APX-hard [25]. The problem remains NP-hard for triangulations of a polygon [1], but the complexity status is open for the case of points in convex position. For further results on flips, see the survey by Bose and Hurtado [6].

The *labelled flip graph* of a point set has a vertex for every labelled triangulation of the point set and an edge when two labelled triangulations differ by a flip. Bose et al. [7] formulated the Orbit Conjecture and proved it for the special case of triangulations of any convex polygon, showing that the labelled flip graph has a single connected component (ignoring convex hull edges, which cannot flip), and giving a tight bound of $\Theta(n \log n)$ on its diameter. Araujo-Pardo et al. [2] independently proved the Orbit Conjecture for convex polygons, and introduced "colorful associahedra" which generalize associahedra to the setting of labelled (or coloured) triangulations. Bose et al. also proved the Orbit Conjecture for spiral polygons. In this case the labelled flip graph may be disconnected but each connected component has diameter $O(n^2)$, which is a tight bound.

The best known lower bound on the diameter of a connected component of the labelled flip graph for a point set is $\Omega(n^3)$ [7]. There is a large gap between this lower bound and our upper bound of $O(n^7)$.

The Orbit Theorem holds for combinatorial triangulations [7], and for pseudotriangulations [8]. In both these cases there is a single orbit, so the labelled flip graph is connected. There are also some related results using variants of the flip operation, for example, Cano et al. [9] reconfigured edge-labelled non-maximal plane graphs by "rotating" edges around one

of their endpoints; again there is a single orbit. A related result where there are multiple orbits is an analogue of the Orbit Theorem for labelled (or "ordered") bases of a matroid – one labelled basis can be turned into another labelled basis via basis exchange steps if and only if elements with the same label lie in the same connected component of the matroid [21].

For more general problems of reconfiguring one structure to another via elementary steps, see [17, 30].

1.2 Preliminaries and Definitions

Most definitions were given above, but we fill in a few missing details. Throughout, we assume a set of n point in general position in the plane. A point set determines $\binom{n}{2}$ *edges* which are the line segments between pairs of points. Two edges *cross* if they intersect in a point that is interior to at least one of the two edges. An *empty convex k-gon* is a subset of k points that forms a convex polygon with no point of P in its interior. A *diagonal* of a convex polygon is an edge joining two points that are not consecutive on the polygon boundary.

Several times in our proofs we will use the result that if two unlabelled triangulations of the same point set have a subset, S, of *constrained* edges in common, then there is a sequence of flips that transforms one triangulation into the other, without ever flipping any edge of S, i.e. the edges in S remain fixed throughout the flip sequence. This was first proved by Dyn et al. [12], and can alternatively be proved using constrained Delaunay triangulations [3].

2 Proof of the Orbit Theorem

In this section we prove the Orbit Theorem assuming the Elementary Swap Theorem (Theorem 3, proved in Section 3), and assuming the following two results on elementary swaps. The first result shows that every elementary swap can be realized by a relatively short flip sequence that can be found efficiently, and the second result gives us a way to combine elementary swaps so that, after moving e's label to f, we can get f's label back to e. These lemmas will be proved in Section 4.

▶ **Lemma 4.** *If there is an elementary swap between two edges in a triangulation $\mathcal{T}$ then there is a flip sequence of length $O(n^6)$ to realize the elementary swap, and, furthermore, this sequence can be found in polynomial time.*

▶ **Lemma 5.** *Let $\mathcal{T}$ be a labelled triangulation containing two edges e and f. If there is a sequence of elementary swaps on $\mathcal{T}$ that takes the label of edge e to edge f, then there is an elementary swap of e and f in $\mathcal{T}$.*

We prove the Orbit Theorem in stages, first Theorem 2 (the case of swapping two labels in a triangulation), then the more general case of permuting edge labels in a triangulation, and finally the full result.

Proof of Theorem 2. The "if" direction is clear, so we address the "only if" direction. Suppose that $\mathcal{T} = (T, \ell)$ is the given edge-labelled triangulation and that e and f are edges of T that are in the same orbit. Then there is a sequence of flips that changes $\mathcal{T}$ to an edge-labelled triangulation $\mathcal{T}' = (T', \ell')$ where T' contains f and $\ell'(f) = \ell(e)$. We now apply the result that any constrained triangulation of a point set can be flipped to any other. Fix edge f and flip T' to T. Applying the same flip sequence to the labelled triangulation $\mathcal{T}'$ yields an edge-labelling of triangulation T in which edge f has the label $\ell(e)$. Thus we have a sequence of flips that permutes the labels of $\mathcal{T}$ and moves the label of e to f.

By the Elementary Swap Theorem (Theorem 3) there is a sequence of elementary swaps whose effect is to move the label of edge e to edge f. By Lemma 5 there is an elementary swap of e and f in $\mathcal{T}$. ◀

▶ **Theorem 6** (Edge Label Permutation Theorem)**.** *Let T be a triangulation of a point set with two edge-labellings ℓ_1 and ℓ_2 such that for each label l, the edge with label l in ℓ_1 and the edge with label l in ℓ_2 are in the same orbit. Then there is a sequence of $O(n)$ elementary swaps to transform the first labelling to the second. Such a sequence can be realized via a sequence of $O(n^7)$ flips, which can be found in polynomial time.*

Proof. The idea is to effect the permutation as a sequence of swaps. If every edge has the same label in ℓ_1 and ℓ_2 we are done. So consider a label l that is attached to a different edge in ℓ_1 and in ℓ_2. Suppose $\ell_1(e) = l$ and $\ell_2(f) = l$, with $e \neq f$. By hypothesis, e and f are in the same orbit. By Theorem 2 there is an elementary swap of e and f in (T, ℓ_1) which results in a new labelling ℓ_1' that matches ℓ_2 in one more edge (namely the edge f) and still has the property that for every label l, the edge with label l in ℓ_1' and the edge with label l in ℓ_2 are in the same orbit. Thus we can continue this process until all edge labels match those of ℓ_2. In total we use $O(n)$ elementary swaps. These can be realized via a sequence of $O(n^7)$ flips by Lemma 4. Furthermore, the sequence can be found in polynomial time. ◀

We can now prove the Orbit Theorem.

Proof of Theorem 1. The necessity of the condition is clear, and we can test it in polynomial time by finding all the orbits, so we address sufficiency. The idea is to reconfigure $\mathcal{T}_1$ to have the same underlying unlabelled triangulation as $\mathcal{T}_2$ and then apply the previous theorem. The details are as follows. Let $\mathcal{T}_1 = (T_1, \ell_1)$ and $\mathcal{T}_2 = (T_2, \ell_2)$. There is a sequence σ of $O(n^2)$ flips to reconfigure the unlabelled triangulation T_1 to T_2, and σ can be found in polynomial time. Applying σ to the labelled triangulation $\mathcal{T}_1$ yields a labelled triangulation $\mathcal{T}_3 = (T_2, \ell_3)$. Note that for every label l, the edges of $\mathcal{T}_1$ and $\mathcal{T}_3$ having label l belong to the same orbit. This is because flips preserve orbits (by definition of orbits). Thus by Theorem 6 there is a flip sequence τ that reconfigures $\mathcal{T}_3$ to $\mathcal{T}_2$, and this flip sequence can be found in polynomial time and has length $O(n^7)$. The concatenation of the two flip sequences, $\sigma\tau$, reconfigures $\mathcal{T}_1$ to $\mathcal{T}_2$, has length $O(n^7)$, and can be found in polynomial time. ◀

3 Proof of the Elementary Swap Theorem

As mentioned in the introduction, we prove the Elementary Swap Theorem using topological properties of the *flip complex*, whose 1-skeleton (i.e. vertices and edges) is the flip graph. In fact, we will only need the 2-cells of the flip complex, not any higher-dimensional structure. We will show that 2-cells of the flip complex correspond to 4- and 5-cycles in the flip graph.

The basic idea is as follows. We will translate the Elementary Swap Theorem to a statement about walks in the flip graph. The hypothesis of the Elementary Swap Theorem is that we have a sequence of flips that permutes the edge labels of a triangulation T. In the flip graph, this sequence corresponds to a closed walk w that starts and ends at triangulation T. Our main topological result is that the flip complex has a trivial fundamental group, which will imply that such a closed walk w can be decomposed into simpler *elementary walks*. Each elementary walk starts at T, traces a path in the flip graph, then traverses the edges of a 2-cell, then retraces the path back to T. The edge-label permutation induced by an elementary walk depends on the 2-cell. If the 2-cell is a 4-cycle, the permutation is the identity; and if the 2-cell is a 5-cycle, then the permutation is a transposition, and

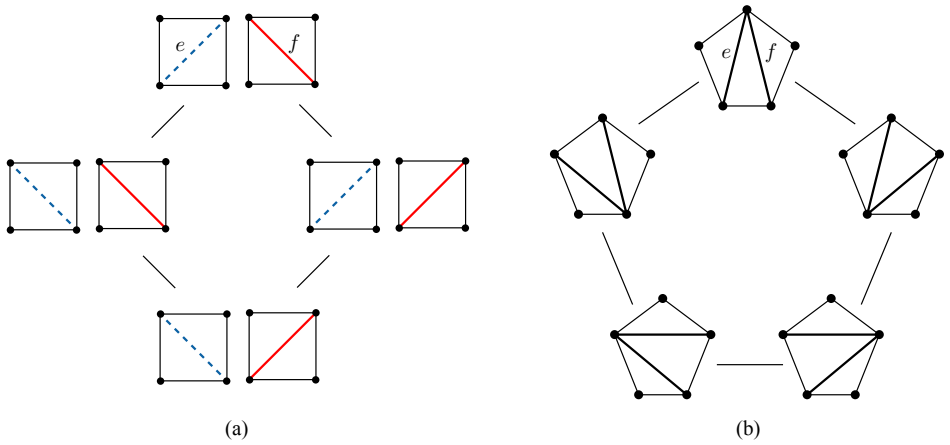


Figure 2 (a) Triangulations that differ in the diagonals of two internally disjoint quadrilaterals form an *elementary* 4*-cycle* in the flip graph. The cycle does not permute the labels (shown as red and blue). (b) Triangulations that differ in the diagonals of a convex pentagon form an *elementary 5-cycle* in the flip graph. This cycle permutes labels as shown in Figure 1.

the elementary walk corresponds to an elementary swap. Altogether, this implies that the permutation induced by the closed walk w can be expressed as a composition of elementary swaps, which proves the Elementary Swap Theorem.

Before stating our main topological theorem, we first define the special cycles that will be shown to correspond to 2-cells of the flip graph. In the same way that an edge of the flip complex corresponds to two triangulations that differ on one edge, every 2-cell of the flip complex corresponds to a set of triangulations that differ on two edges. Define an *elementary* 4*-cycle* to be a cycle of the flip graph obtained in the following way. Take a triangulation T and two edges $e, f \in T$ whose removal leaves two internally disjoint convex quadrilaterals in T. Each quadrilateral can be triangulated in two ways, which results in four triangulations that contain $F := T \setminus \{e, f\}$. These four triangulations form a 4-cycle in the flip graph, as shown in Figure 2(a). Observe that a traversal of the cycle corresponds to a sequence of flips that returns edge-labels to their original positions.

Define an *elementary* 5*-cycle* to be a cycle of the flip graph obtained in the following way. Take a triangulation T and two edges $e, f \in T$ whose removal leaves a convex pentagon in T. There are five triangulations that contain $F := T \setminus \{e, f\}$, and they form a 5-cycle in the flip graph, as shown in Figure 2(b). Observe that the sequence of flips around such a cycle permutes labels of e and f as shown in Figure 1.

Our main topological theorem is the following.

▶ **Theorem 7.** *Let P be a set of n points in general position in the plane. There is a high-dimensional cell complex $\mathbb{X} = \mathbb{X}(P)$, which we call the* flip complex*, such that:*

1. *The 1-skeleton of $\mathbb{X}$ is the flip graph of P;*
2. *There is a one-to-one correspondence between the 2-cells of $\mathbb{X}$ and the elementary 4-cycles and elementary 5-cycles of the flip graph of P;*
3. *$\mathbb{X}$ has the topology of (i.e., is homotopy equivalent to) a high-dimensional ball; therefore its* fundamental group*, $\pi_1(\mathbb{X})$, is trivial.*

In what follows, we will use a number of notions from combinatorial topology; some of these we will recall along the way, but others we will only describe informally or leave undefined and instead refer the reader to standard textbooks for further background (in

particular, we refer the reader to [5, Appendix 4.7] and [15] for background on *regular cell complexes*, *shellability*, and *piecewise linear balls and spheres*, to [29] for background on the fundamental group of cell complexes, and to [15, 23] for background on *dual complexes*; we will provide more detailed references for specific results below).

Theorem 7 follows from a result of Orden and Santos [24]; we are grateful to F. Santos for bringing this reference to our attention. In fact, Orden and Santos show something stronger: There exists a simple polytope $\mathbb{Y} = \mathbb{Y}(P)$ and a face F of $\mathbb{Y}$ such that $\mathbb{X}$ can be taken to be the complement of the star of F in $\mathbb{Y}$.

Before becoming aware of the work of Orden and Santos, we found a different proof of Theorem 7 that starts out by considering the simplicial complex $\mathbb{T} = \mathbb{T}(P)$ whose faces are the sets of pairwise non-crossing edges (line segments) spanned by P. This complex $\mathbb{T}$ is shown to be a *shellable simplicial ball* (by an argument based on constrained Delaunay triangulations), and $\mathbb{X}$ is then constructed as the *dual complex* of $\mathbb{T}$. We hope that this alternative proof of Theorem 7 is of some independent interest and present it in Sections 3.2 and 3.3 below. Before that, in Section 3.1, we show how to derive the Elementary Swap Theorem from Theorem 7.

3.1 From Topology to the Elementary Swap Theorem

In this section we use Theorem 7 to prove the Elementary Swap Theorem. We begin by defining elementary walks. A *walk* in the flip graph is a sequence $T_0, T_1, \ldots, T_k$ of triangulations (possibly with repetitions) such that T_{i-1} and T_i differ by a flip. We will refer to T_0 and T_k as the start and the end of the walk, respectively. A walk is *closed* if it starts and ends at the same triangulation. If w_1 and w_2 are walks such that the end of w_1 equals the start of w_2 then we can define their *composition* $w_1 w_2$ in the obvious way. Furthermore, if $w = (T = T_0, T_1, \ldots, T_k)$ is a walk, we will use the notation $w^{-1} = (T_k, T_{k-1}, \ldots, T_0)$ for the *inverse walk*.

Fix a triangulation T_0. An *elementary quadrilateral walk* is a closed walk of the form wzw^{-1}, where z is an elementary 4-cycle in the flip graph, and w is a walk from T_0 to some triangulation on z. An *elementary pentagonal walk* is defined analogously, with z an elementary 5-cycle.

It is straightforward to check the effect of these elementary walks on labellings:

▶ **Lemma 8.** *Let (T_0, ℓ) be a labelled triangulation. An elementary quadrilateral walk does not permute the labels. An elementary pentagonal walk swaps the labels of two edges (e and f in Figure 2(b)) and leaves all other labels fixed; this corresponds exactly to the notion of an elementary swap introduced earlier.*

Another operation that does not affect the permutation of labels induced by a closed walk is the following. A *spur* ww^{-1} starting and ending at T is an arbitrary walk w starting at T, immediately followed by the *inverse walk*. If w_1 and w_2 are walks in the flip graph such that w_1 ends at a triangulation T and w_2 starts there, and if s is a spur at T, then we say that the walk $w_1 s w_2$ differs from $w_1 w_2$ by a *spur insertion* The inverse operation is called a *spur deletion*.

▶ **Lemma 9.** *If two closed walks w and w' in the flip graph differ only by a finite number of spur insertions and deletions then they yield the same permutation of edge labels.*

Proof. A flip immediately followed by its inverse flip has no effect on labels. The lemma follows by induction on the length of a spur and the number of spur insertions and deletions. ◀

By Lemmas 8 and 9, the Elementary Swap Theorem directly reduces to the following, which we prove using Theorem 7:

▶ **Proposition 10.** *Let w be a closed walk in the flip graph starting and ending at T_0. Then, up to a finite number of spur insertions and deletions, w can be written as the composition of finitely many elementary walks.*

Proof. We use the well-known fact that the fundamental group of a cell complex can be defined *combinatorially* in terms of closed walks in the 1-skeleton and this definition is equivalent to the usual topological definition in terms of continuous loops, see [27, Chap. 7] or [29, Chap. 4]. In particular, in a cell complex with trivial fundamental group any two closed walks in the 1-skeleton starting at the same vertex are related by a finite number of spur insertions, deletions and so-called 2-cell relations.

We describe the combinatorial definition of the fundamental group of the flip complex $\mathbb{X}$ in detail. By Theorem 7, the 1-skeleton of $\mathbb{X}$ is the flip graph of P. Fix a *base triangulation* T_0, and, for every triangulation T, fix a walk p_T from T_0 to T. Given two triangulations T_1, T_2 that differ by a flip, we form the closed walk w_{T_1,T_2} in the flip graph, called a *generating walk*, that goes from T_0 to T_1 along p_{T_1}, then flips to T_2, and then returns to T_0 along $p_{T_2}^{-1}$. It is easy to see that, up to a finite number of spur insertions and deletions, every closed walk starting and ending at T_0 can be written as a composition of generating walks.

We say that walks w and w' are 2-*cell related* if we can express them as $w = w_1w_2$ and $w' = w_1zw_2$, where z is a closed walk traversing the boundary of a 2-cell (an elementary cycle) exactly once in either orientation. Notice that w_1w_2 and $w_1zz^{-1}w_2$ differ only by the spur zz^{-1}, hence, up to spur insertion and deletion, being 2-cell related is symmetric.

Also, notice the *precomposition property*: if w and w' are 2-cell related as above and if w is precomposed with the closed walk $w_1zw_1^{-1}$ then the result $w'' = (w_1zw_1^{-1})w = w_1z(w_1^{-1}w_1)w_2$ differs from w' only by the spur $w_1^{-1}w_1$. By Theorem 7, a boundary of a 2-cell is an elementary 4- or 5-cycle and so the walk $w_1zw_1^{-1}$ above is an elementary walk.

Two walks in the flip graph are called equivalent if they differ by a finite number of spur insertion and/or deletions and by applying a finite number of 2-cell relations. It is not hard to check that this defines an equivalence relation, and the fundamental group $\pi_1(\mathbb{X})$ is given as the set of equivalence classes of closed walks starting and ending at T_0.

By Theorem 7, the fundamental group of the flip complex $\mathbb{X}$ is trivial. This translates into the fact that every closed walk starting and ending at T_0 is equivalent to the trivial walk. By the precomposition property, this means that, up to a finite number of spur insertions and deletions, every closed walk is a composition of finitely many elementary walks. ◀

3.2 The Simplicial Complex of Plane Graphs

Let P be a set of n points in general position in the plane. Let E be the set of edges (closed line segments) spanned by P. Two edges $e, f \in E$ are said to be *non-crossing* if they are disjoint or if they intersect in a single point of P that is an endpoint of both edges. We say that a subset $F \subseteq E$ is *non-crossing* if every pair of distinct edges $e, f \in F$ is non-crossing. If G is non-crossing and $F \subseteq G$ then F is non-crossing as well. Thus, the non-crossing sets of edges form an abstract simplicial complex

$$\mathbb{T} = \mathbb{T}(P) := \{F \colon F \subseteq E, F \text{ non-crossing}\},$$

which we call the *complex of plane graphs on P*. We collect some basic properties of $\mathbb{T}$:

1. The *facets* (inclusion-maximal faces) of $\mathbb{T}$ are exactly the triangulations of P (every non-crossing set of edges $F \subseteq E$ can be extended to a triangulation). Thus, the simplicial complex $\mathbb{T}$ is of dimension $m-1$, where m is the number of edges in any triangulation of P, and it is *pure*, i.e., every face of $\mathbb{T}$ is contained in a face of dimension $m-1$.
2. Every face F of $\mathbb{T}$ of dimension $m-2$ is contained in either one or two triangulations. In the latter case, F corresponds to a flip between these two triangulations.

We will show that the topology of $\mathbb{T}$ is particularly simple, namely that $\mathbb{T}$ is a homeomorphic to an $(m-1)$-dimensional ball. Furthermore, there is a combinatorial certificate (*shellability*) for this homeomorphism. This implies that the homeomorphism is particularly nice and that $\mathbb{T}$ is a *piecewise-linear ball*. We refer to [15] and [5, Appendix 4.7] for more details and further references on shellability and piecewise-linear balls, spheres, and manifolds. In this extended abstract, we will leave the notion of piecewise-linearity undefined – the only property that we will need is that it ensures that the construction of the *dual cell complex* $\mathbb{T}^*$ (see Proposition 13 below) is well-behaved.

We recall that a pure d-dimensional simplicial complex is *shellable* if there exists a total ordering of its facets $F_1, F_2, \cdots, F_N$ (called a *shelling order*) such that, for every $2 \le j \le N$, the intersection of F_j with the simplicial complex generated by the preceding facets[2] is pure of dimension $d-1$.

We will need the following result (which appears implicitly in [4], and explicitly in [10]; see [5, Prop. 4.7.22] for a short proof):

▶ **Proposition 11.** *Suppose $\mathbb{K}$ is a finite d-dimensional simplicial complex that is a pseudomanifold, i.e., $\mathbb{K}$ is pure and every $(d-1)$-dimensional face of $\mathbb{K}$ is contained in at most two d-faces. If $\mathbb{K}$ is shellable then $\mathbb{K}$ is either a piecewise-linear ball or a piecewise-linear sphere. The former case occurs iff there is at least one $(d-1)$-dimensional face that is contained in only one d-face of $\mathbb{K}$.*[3]

▶ **Theorem 12.** *$\mathbb{T}$ is* shellable*, and hence a piecewise-linear $(m-1)$-dimensional ball.*

Proof. We observed earlier that $\mathbb{T}$ is a pure $(m-1)$-dimensional simplicial complex, and that every $(m-2)$-dimensional face of $\mathbb{T}$ is contained in at most two $(m-1)$-dimensional faces, hence $\mathbb{T}$ is a pseudomanifold. Moreover, if T is a triangulation of P and if $e \in T$ is a non-flippable edge (e.g., if e is a convex hull edge) then $F := T \setminus \{e\}$ is an $(m-2)$-dimensional face of $\mathbb{T}$ that is contained in a unique $(m-1)$-face, namely T.

Thus, by Proposition 11, it suffices to show that $\mathbb{T}$ is shellable, i.e., to exhibit a shelling order for the facets of $\mathbb{T}$.

With every triangulation T of P, we associate the sorted vector of angles $\alpha(T) = (\alpha_1(T), \alpha_2(T), \cdots, \alpha_{3t}(T))$, where $\alpha_1(T) \le \alpha_2(T) \le \cdots \le \alpha_{3t}(T)$ are the angles occurring in the triangulation T. We order the triangulations of P by sorting the corresponding angle vectors $\alpha(T)$ lexicographically from largest to smallest; if the point set is in general position, this defines a total ordering

$$T_1, T_2, \ldots, T_N, \qquad \alpha(T_1) >_{\text{LEX}} \alpha(T_2) >_{\text{LEX}} \cdots >_{\text{LEX}} \alpha(T_N), \tag{1}$$

where N is the number of triangulations of P.

---

[2] More formally, for any set F, let 2^F denote the simplicial complex of all subsets of F. Then the requirement for a shelling is that, for $2 \le j \le N$, the intersection of the complexes 2^{F_j} and $\bigcup_{i<j} 2^{F_i}$ be pure of dimension $d-1$.

[3] We remark that the property of being a shellable pseudomanifold (which is a combinatorial and algorithmically verifiable condition) is strictly stronger than being a piecewise-linear ball or sphere, which in turn is strictly stronger than being a simplicial complex homeomorphic to a ball or sphere.

It is well known (see, for example, [11, Chap. 3.4]) that in this ordering, T_1 is the Delaunay triangulation of P. Moreover, if we consider only triangulations containing a particular plane subgraph corresponding to a face F of $\mathbb{T}$ and the corresponding subsequence of the angle vectors, the first of these vectors corresponds to the Delaunay triangulation constrained to F.

We claim that the triangulation ordering (1) defines a shelling. For this, we need to prove that the following holds for $2 \leq j \leq N$: If F is a face of $\mathbb{T}$ that is contained in $T_j \cap T_i$ for some $i < j$, then there exists an $(m-2)$-dimensional face G of T_j and some $i' < j$ such that $F \subseteq G = T_{i'} \cap T_j$.

To see this, consider the subsequence $T_{k_1}, T_{k_2}, \ldots$ of the sequence (1) consisting only of those triangulations that contain the edge set F. Then T_{k_1} is the constrained Delaunay triangulation with respect to the edge set F, and T_i and T_j both appear in that subsequence; in particular, $T_j \neq T_{k_1}$ since T_i precedes it. Since every triangulation containing F can be transformed to the constrained Delaunay triangulation T_{k_1}, (see, e.g., the description of the Lawson flip algorithm in [11]) there must exist an edge $e \in T_j \setminus T_{k_1}$ such that flipping e (a Lawson flip) increases the angle vector; thus, the triangulation resulting from flipping e is some T_k with $k < j$ and satisfies $F \subseteq T_k \cap T_j$ as desired. ◀

3.3 Boundary and Interior Faces of $\mathbb{T}$, and the Dual Flip Complex $\mathbb{X}$

Let $\mathbb{B}$ be a piecewise-linear ball of dimension d. By definition, the *boundary* $\partial\mathbb{B}$ of $\mathbb{B}$ is the subcomplex of $\mathbb{B}$ consisting of all faces F for which there exists a $(d-1)$-dimensional face G of $\mathbb{B}$, with $F \subseteq G$, such that G is contained in a unique d-dimensional face of $\mathbb{B}$. (In the case $\mathbb{B} = \mathbb{T}$, the latter condition means that $G = T \setminus \{e\}$ for some triangulation T and some edge $e \in T$ that is not flippable.) A face F of $\mathbb{B}$ that does not lie in $\partial\mathbb{B}$ is called an *interior face*.

To define the flip complex $\mathbb{X}$, we need the notion of *dual cells* and the dual cell decomposition of a piecewise-linear ball; for the precise definition, we refer to [15, Sec. I.6] or [23, §64 and §70].[4] Here, we simply collect the properties that we will need:

▶ **Proposition 13.** *Let $\mathbb{B}$ be a d-dimensional piecewise-linear ball.*

1. *For each interior k-dimensional face F of $\mathbb{B}$, one can define a* dual cell *F^* (a certain subcomplex of the* barycentric subdivision *of $\mathbb{B}$ that is a piecewise-linear ball of dimension $d-k$ [15, Lemma I.19]).*
2. *The construction reverses inclusion, i.e., for interior faces F, G of $\mathbb{B}$, $F \subseteq G$ iff $F^* \supseteq G^*$.*
3. *The dual cells of the interior faces of $\mathbb{B}$ form a regular cell complex, denoted $\mathbb{B}^*$ and called the* dual cell complex. *$\mathbb{B}^*$ need not be a manifold or pure d-dimensional, but it is homotopy equivalent to $\mathbb{B}$ [23, Lem. 70.1].*[5]

We define the *flip complex* $\mathbb{X} := \mathbb{T}^*$ as the dual complex of the simplicial complex $\mathbb{T}$.

Proof of Theorem 7. By Prop. 13, $\mathbb{X} = \mathbb{T}^*$ is a regular cell complex that is homotopy equivalent to the ball $\mathbb{T}$; consequently, the fundamental group $\pi_1(\mathbb{X})$ vanishes.

It remains to show the characterization of the vertices, edges, and 2-cells of $\mathbb{X}$.

[4] In [23], the terminology *dual blocks* is used instead of dual cells, since the construction is described in a more general setting (for arbitrary triangulated manifolds or homology manifolds) in which the dual blocks need not be cells (homeomorphic to balls). In the setting of piecewise-linear manifolds, in particular piecewise-linear balls, however, this technical issue does not arise.

[5] More specifically, the dual complex of a piecewise-linear manifold with boundary is a deformation retraction of the manifold. For manifolds without boundary, the dual complex is piecewise-linearly homeomorphic to the original manifold.

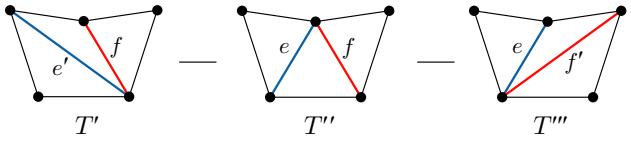


Figure 3 The case of the proof of Theorem 7 where e and f lie in a non-convex pentagon.

The vertices of $\mathbb{X}$ correspond (are dual) to the faces of $\mathbb{T}$ of the highest dimension $(m-1) = \dim \mathbb{T}$, i.e., to the triangulations of P (these are automatically interior faces of $\mathbb{T}$).

The edges of $\mathbb{X}$ correspond to $(m-2)$-dimensional faces F of $\mathbb{T}$ that are *interior*, i.e., such that F is contained in two triangulations T and T' that differ by a flip. Thus, the 1-skeleton of $\mathbb{X}$ is exactly the flip graph of P.

Every 2-cell of $\mathbb{X}$ is the dual cell F^* of an *interior* face F of $\mathbb{T}$ of dimension $\dim F = m-3$.

Consider an arbitrary triangulation T containing F, i.e., F is obtained from T by deleting two edges e, f. If one of these edges, say e, were not flippable in T, then $T \setminus \{e\}$ and hence F would lie in the boundary $\partial\mathbb{T}$ and not be interior. Thus, both edges e and f must be flippable.

If e and f are not incident to a common triangle in T, (or, equivalently, removing both e and f from T creates two internally disjoint convex quadrilaterals) then there exist four triangulations containing F and these form an elementary 4-cycle in the flip graph. The 4-cycle is by definition the boundary of the dual cell F^*.

If e and f are incident to a common triangle in T, and the union of the three triangles of T containing either e or f forms a convex pentagon, then there are five triangulations containing F and these form an elementary 5-cycle in the flip graph. The 5-cycle is by definition the boundary of the dual cell F^*.

It remains to consider the case that the union of the three triangles of T containing e or f is a non-convex pentagon with a single reflex vertex, see Figure 3. In that case, there are three triangulations of P containing F, corresponding to the three triangulations of such a non-convex pentagon. These triangulations form a path of length 2 in the flip graph, say T', T'', T''' in that order. Then T' contains an edge (f in Figure 3) that is not flippable in T', hence $T' \setminus \{f\}$ is a boundary face containing F, i.e., F is not interior and does not give rise to a dual 2-cell of $\mathbb{X}$.

Hence, every 2-cell of $\mathbb{X}$ corresponds to an elementary 4- or 5-cycle of the flip graph.

Conversely, every elementary 4- or 5-cycle of the flip graph gives rise to a 2-cell F^* of $\mathbb{X}$: more precisely, F^* corresponds to the intersection of the triangulations in the elementary cycle. ◀

4 Proofs of Properties of Elementary Swaps

In this section we prove Lemmas 4 and 5.

Proof of Lemma 4. Construct a graph G_D called the *double quadrilateral graph*. Vertices of the graph G_D are pairs of non-crossing edges on the point set P, and we define two vertices (e_1, f_1) and (e_2, f_2) of G_D to be adjacent if either $e_1 = e_2$ and f_1 and f_2 are adjacent in the quadrilateral graph, or if $f_1 = f_2$ and e_1 and e_2 are adjacent in the quadrilateral graph. (Recall that two edges a and b are adjacent in the quadrilateral graph if a and b cross and their four endpoints form an empty quadrilateral.)

In the graph G_D we identify some vertices as "swap vertices". These are the vertices (g, h) such that g and h are diagonals of some empty convex pentagon in the point set. Note that the swap vertices can be identified in polynomial time.

We claim that there is an elementary swap of e and f in labelled triangulation $\mathcal{T} = (\mathcal{T}, \ell)$ if and only if there is a path in G_D from vertex (e, f) to a swap vertex. For the forward direction, suppose there is such an elementary swap. It begins with a sequence σ of flips from $\mathcal{T}$ to a labelled triangulation $\mathcal{T}'$ in which labels $\ell(e)$ and $\ell(f)$ are attached to two diagonals g and h of some empty convex pentagon. The subsequence of σ consisting of those flips that apply to an edge whose current label is $\ell(e)$ or $\ell(f)$ corresponds to a path in G_D from (e, f) to the swap vertex (g, h).

For the other direction, let π be a path in G_D from (e, f) to a swap vertex. It suffices to show that the path π provides a sequence of flips, σ, that takes $\mathcal{T}$ to some labelled triangulation $\mathcal{T}'$ in which labels $\ell(e)$ and $\ell(f)$ are attached to two diagonals of an empty convex pentagon, because the rest of the elementary swap is then determined. Consider the first edge of π and suppose without loss of generality that it goes from (e, f) to (e, f') (the case when e changes is similar). Then e and f' are non-crossing. Because f and f' are adjacent in the quadrilateral graph, they cross and form an empty convex quadrilateral Q. Note that e does not intersect the interior of Q, since Q is empty and e does not cross f or f'. We apply the result that any constrained triangulation can be flipped to any other with $O(n^2)$ flips. Fix edges e and f in T and flip $\mathcal{T}$ to a labelled triangulation that contains the edges of Q. In this triangulation, we can flip f to f', transferring $\ell(f)$ to f'. We continue in this way to realize each edge of π via $O(n^2)$ flips, arriving finally at a labelled triangulation in which labels $\ell(e)$ and $\ell(f)$ are attached to edges that are the diagonals of some empty convex pentagon in the point set. Fixing the two diagonals, we can flip to a triangulation that contains the edges of the convex pentagon, and at this point we are done.

Because the graph G_D has $O(n^4)$ vertices, the diameter of any of its connected components is $O(n^4)$. Thus, if there is an elementary swap that exchanges the labels of edges e and f, then there is one corresponding to a path in G_D of length $O(n^4)$. We can explicitly construct G_D and find such a path in polynomial time. As argued above, every edge of G_D can be realized by $O(n^2)$ flips. This proves that, for any elementary swap, we can construct a sequence of $O(n^6)$ flips to realize it, and the construction takes polynomial time. ◀

Proof of Lemma 5. An elementary swap in triangulation $\mathcal{T}$ acts on two edges of $\mathcal{T}$. We define a graph G_S called the *elementary swap graph* of $\mathcal{T}$. G_S has a vertex for every edge of $\mathcal{T}$, and we define vertices e and f to be adjacent in G_S if there is an elementary swap of e and f in $\mathcal{T}$.

By hypothesis, there is a sequence of elementary swaps that takes the label of edge e to edge f. Observe that no sequence of elementary swaps will take the label of edge e outside the connected component of G_S that contains e. Therefore e and f must lie in the same connected component of G_S. We will now show that each connected component of G_S is a clique. This implies that there is an elementary swap of e and f, and completes our proof.

Consider a simple path $(e_0, e_1), (e_1, e_2), \ldots, (e_{k-1}, e_k)$ in G_S. Let σ_i, $i = 1, \ldots, k$ be a flip sequence that realizes the elementary swap (e_{i-1}, e_i), and let $\sigma = \sigma_1\sigma_2 \ldots \sigma_{k-1}$. Observe that σ takes the label of e_0 to e_{k-1}, and does not change the label of e_k (by the assumption that the path is simple). By definition of an elementary swap, the flip sequence σ_k has the form $\rho\pi\rho^{-1}$ where ρ is a sequence of flips that moves the labels of e_{k-1} and e_k into an empty convex pentagon, and π is the sequence of five flips that exchanges the labels of e_{k-1} and e_k.

Consider the flip sequence $\sigma\sigma_k\sigma^{-1} = \sigma\rho\pi\rho^{-1}\sigma^{-1} = \sigma\rho\pi(\sigma\rho)^{-1}$. The first part of this flip sequence, $\sigma\rho$, moves the labels of e_0 and e_k into an empty convex pentagon; the middle

part, π, exchanges them; and the final part, $(\sigma\rho)^{-1}$ reverses the first part. Therefore this flip sequence realizes an elementary swap of e_0 and e_k. ◀

5 Conclusions

We have characterized when two labelled triangulations of a set of n points belong to the same connected component of the labelled flip graph, and proved that the diameter of each connected component is bounded by $O(n^7)$. We conclude with some open problems:

1. Reduce the gap between the upper bound, $O(n^7)$, and the best known lower bound of $O(n^3)$ [7] on the diameter of a component of the labelled flip graph.
2. We did not analyze the run-time of our algorithms in the main text. A crude bound is $O(n^8)$, with the bottleneck being the explicit construction in the proof of Lemma 4 of the double quadrilateral graph which has $O(n^4)$ vertices and thus $O(n^8)$ edges. This bound can surely be improved.
3. What is the complexity of the following flip distance problem for labelled triangulations: Given two labelled triangulations and a number k, is there a flip sequence of length at most k to transform the first triangulation to the second one? This problem is NP-complete in the unlabelled setting, but knowing the mapping of edges might make the problem easier.

Acknowledgements. This research was initiated at the 2016 Bellairs Workshop on Geometry and Graphs.

References

1 Oswin Aichholzer, Wolfgang Mulzer, and Alexander Pilz. Flip distance between triangulations of a simple polygon is NP-complete. *Discrete & Computational Geometry*, 54(2):368–389, 2015. doi:10.1007/s00454-015-9709-7.

2 Gabriela Araujo-Pardo, Isabel Hubard, Deborah Oliveros, and Egon Schulte. Colorful associahedra and cyclohedra. *Journal of Combinatorial Theory, Series A*, 129:122–141, 2015. doi:10.1016/j.jcta.2014.09.001.

3 Marshall Bern and David Eppstein. Mesh generation and optimal triangulation. In Ding-Zhu Du and Frank Hwang, editors, *Computing in Euclidean geometry*, volume 1 of *Lecture Notes Series on Computing*, pages 23–90. World Scientific, 1992. doi:10.1142/9789814355858_0002.

4 R. H. Bing. Some aspects of the topology of 3-manifolds related to the Poincaré conjecture. In *Lectures on modern mathematics, Vol. II*, pages 93–128. Wiley, New York, 1964.

5 Anders Björner, Michel Las Vergnas, Bernd Sturmfels, Neil White, and Günter M. Ziegler. *Oriented Matroids*, volume 46 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, Cambridge, 2nd edition, 1999. doi:10.1017/CBO9780511586507.

6 Prosenjit Bose and Ferran Hurtado. Flips in planar graphs. *Computational Geometry Theory and Applications*, 42(1):60–80, 2009. doi:10.1016/j.comgeo.2008.04.001.

7 Prosenjit Bose, Anna Lubiw, Vinayak Pathak, and Sander Verdonschot. Flipping edge-labelled triangulations. *arXiv:1310.1166*, 2013. To appear in *Computational Geometry*. URL: http://arxiv.org/abs/1310.1166.

8 Prosenjit Bose and Sander Verdonschot. Flips in edge-labelled pseudo-triangulations. *Computational Geometry*, 60:45–54, 2017.

9 Javier Cano, José-Miguel Díaz-Báñez, Clemens Huemer, and Jorge Urrutia. The edge rotation graph. *Graphs and Combinatorics*, 29(5):1207–1219, 2013. doi:10.1007/s00373-012-1201-z.

10 Gopal Danaraj and Victor Klee. Shellings of spheres and polytopes. *Duke Mathematical Journal*, 41(2):443–451, 1974.

11 Satyan L. Devadoss and Joseph O'Rourke. *Discrete and Computational Geometry*. Princeton University Press, 2011.

12 N. Dyn, I. Goren, and S. Rippa. Transforming triangulations in polygonal domains. *Computer Aided Geometric Design*, 10:531–536, 1993.

13 Herbert Edelsbrunner. *Geometry and Topology for Mesh Generation*. Cambridge University Press, Cambridge, 2001. doi:10.1017/CBO9780511530067.

14 David Eppstein. Happy endings for flip graphs. *Journal of Computational Geometry*, 1(1):3–28, 2010. doi:10.20382/jocg.v1i1a2.

15 J. F. P. Hudson. *Piecewise Linear Topology*. W. A. Benjamin, Inc., New York-Amsterdam, 1969.

16 Ferran Hurtado, Marc Noy, and Jorge Urrutia. Flipping edges in triangulations. *Discrete & Computational Geometry*, 22(3):333–346, 1999. doi:10.1007/PL00009464.

17 Takehiro Ito, Erik D. Demaine, Nicholas J. A. Harvey, Christos H. Papadimitriou, Martha Sideri, Ryuhei Uehara, and Yushi Uno. On the complexity of reconfiguration problems. *Theoretical Computer Science*, 412(12-14):1054–1065, 2011. doi:10.1016/j.tcs.2010.12.005.

18 Charles L. Lawson. Transforming triangulations. *Discrete Mathematics*, 3(4):365–372, 1972.

19 Charles L. Lawson. Software for C^1 surface interpolation. In *Mathematical Software III*, pages 161–194. Academic Press, New York, 1977.

20 Anna Lubiw and Vinayak Pathak. Flip distance between two triangulations of a point set is NP-complete. *Computational Geometry*, 49:17–23, 2015. doi:10.1016/j.comgeo.2014.11.001.

21 Anna Lubiw and Vinayak Pathak. Reconfiguring ordered bases of a matroid. *arXiv:1612.00958*, 2016.

22 Michael Molloy, Bruce Reed, and William Steiger. On the mixing rate of the triangulation walk. In *DIMACS-AMS volume on Randomization Methods in Algorithm Design*, volume 43 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 179–190. AMS, 1999.

23 James R. Munkres. *Elements of Algebraic Topology*. Addison-Wesley Publishing Company, Menlo Park, CA, 1984.

24 David Orden and Francisco Santos. The polytope of non-crossing graphs on a planar point set. *Discrete & Computational Geometry*, 33(2):275–305, 2005. doi:10.1007/s00454-004-1143-1.

25 Alexander Pilz. Flip distance between triangulations of a planar point set is APX-hard. *Computational Geometry*, 47(5):589–604, 2014. doi:10.1016/j.comgeo.2014.01.001.

26 Lionel Pournin. The diameter of associahedra. *Advances in Mathematics*, 259:13–42, 2014. doi:10.1016/j.aim.2014.02.035.

27 Herbert Seifert and William Threlfall. *A Textbook of Topology*, volume 89 of *Pure and Applied Mathematics*. Academic Press, 1980.

28 Daniel D. Sleator, Robert E. Tarjan, and William P. Thurston. Rotation distance, triangulations, and hyperbolic geometry. *Journal of the American Mathematical Society*, 1(3):647–681, 1988. doi:10.2307/1990951.

29 John Stillwell. *Classical Topology and Combinatorial Group Theory*, volume 72 of *Graduate Texts in Mathematics*. Springer-Verlag, 2nd edition, 1993. doi:10.1007/978-1-4612-4372-4.

30 Jan van den Heuvel. The complexity of change. *Surveys in Combinatorics*, 409:127–160, 2013.

A Spectral Gap Precludes Low-Dimensional Embeddings

Assaf Naor

Mathematics Department, Princeton University, Princeton, NJ, USA
naor@math.princeton.edu

Abstract

We prove that there is a universal constant $C > 0$ with the following property. Suppose that $n \in \mathbb{N}$ and that $\mathsf{A} = (a_{ij}) \in M_n(\mathbb{R})$ is a symmetric stochastic matrix. Denote the second-largest eigenvalue of A by $\lambda_2(\mathsf{A})$. Then for *any* finite-dimensional normed space $(X, \|\cdot\|)$ we have

$$\forall\, x_1, \ldots, x_n \in X, \qquad \dim(X) \geqslant \frac{1}{2} \exp\left(C \frac{1-\lambda_2(\mathsf{A})}{\sqrt{n}} \left(\frac{\sum_{i=1}^n \sum_{j=1}^n \|x_i - x_j\|^2}{\sum_{i=1}^n \sum_{j=1}^n a_{ij} \|x_i - x_j\|^2} \right)^{\frac{1}{2}} \right).$$

It follows that if an n-vertex $O(1)$-expander embeds with average distortion $D \geqslant 1$ into X, then necessarily $\dim(X) \gtrsim n^{c/D}$ for some universal constant $c > 0$. This is sharp up to the value of the constant c, and it improves over the previously best-known estimate $\dim(X) \gtrsim (\log n)^2/D^2$ of Linial, London and Rabinovich, strengthens a theorem of Matoušek, and answers a question of Andoni, Nikolov, Razenshteyn and Waingarten.

1998 ACM Subject Classification F.2.2 Geometrical Problems and Computations

Keywords and phrases Metric embeddings, dimensionality reduction, expander graphs, nonlinear spectral gaps, nearest neighbor search, complex interpolation, Markov type

Digital Object Identifier 10.4230/LIPIcs.SoCG.2017.50

1 Introduction

Given $n \in \mathbb{N}$ and a symmetric stochastic matrix $\mathsf{A} \in M_n(\mathbb{R})$, the eigenvalues of A will be denoted below by $1 = \lambda_1(\mathsf{A}) \geqslant \ldots \geqslant \lambda_n(\mathsf{A}) \geqslant -1$. Here we prove the following statement.

▶ **Theorem 1.** *There is a universal constant $C > 0$ with the following property. Fix $n \in \mathbb{N}$ and a symmetric stochastic matrix $\mathsf{A} = (a_{ij}) \in M_n(\mathbb{R})$. For any finite-dimensional normed space $(X, \|\cdot\|)$,*

$$\forall\, x_1, \ldots, x_n \in X, \qquad \dim(X) \geqslant \frac{1}{2} \exp\left(C \frac{1-\lambda_2(\mathsf{A})}{\sqrt{n}} \left(\frac{\sum_{i=1}^n \sum_{j=1}^n \|x_i - x_j\|^2}{\sum_{i=1}^n \sum_{j=1}^n a_{ij} \|x_i - x_j\|^2} \right)^{\frac{1}{2}} \right). \quad (1)$$

We shall next explain a noteworthy geometric consequence of Theorem 1 that arises from an examination of its special case when the matrix A is the normalized adjacency matrix of a connected graph. Before doing so, we briefly recall some standard terminology related to metric embeddings.

Suppose that $(\mathcal{M}, d)$ is a finite metric space and $(X, \|\cdot\|)$ is a normed space. For $L \geqslant 0$, a mapping $\phi : \mathcal{M} \to X$ is said to be L-Lipschitz if $\|\phi(x) - \phi(y)\| \leqslant L d(x, y)$ for every $x, y \in \mathcal{M}$. For $D \geqslant 1$, one says that $\mathcal{M}$ embeds into X with (bi-Lipschitz) distortion D if there is a D-Lipschitz mapping $\phi : \mathcal{M} \to X$ such that $\|\phi(x) - \phi(y)\| \geqslant d(x, y)$ for every $x, y \in \mathcal{M}$. Following Rabinovich [46], given $D \geqslant 1$ one says that $\mathcal{M}$ embeds into

X with average distortion D if there exists a D-Lipschitz mapping $\phi : \mathcal{M} \to X$ such that $\sum_{x,y\in\mathcal{M}} \|\phi(x) - \phi(y)\| \geqslant \sum_{x,y\in\mathcal{M}} d(x,y)$.

For $n \in \mathbb{N}$ write $[n] = \{1, \ldots, n\}$. Fix $k \in \{3, \ldots, n\}$ and let $\mathsf{G} = ([n], E_{\mathsf{G}})$ be a k-regular connected graph whose vertex set is $[n]$. The shortest-path metric that is induced by G on $[n]$ is denoted $d_{\mathsf{G}} : [n] \times [n] \to \mathbb{N} \cup \{0\}$. A simple (and standard) counting argument (e.g. [31]) gives

$$\frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n} d_{\mathsf{G}}(i,j) \gtrsim \frac{\log n}{\log k}, \tag{2}$$

where in (2), as well as in the rest of this article, we use the following (standard) asymptotic notation. Given two quantities $Q, Q' > 0$, the notations $Q \lesssim Q'$ and $Q' \gtrsim Q$ mean that $Q \leqslant \mathsf{K}Q'$ for some universal constant $\mathsf{K} > 0$. The notation $Q \asymp Q'$ stands for $(Q \lesssim Q') \wedge (Q' \lesssim Q)$. If we need to allow for dependence on certain parameters, we indicate this by subscripts. For example, in the presence of an auxiliary parameter ψ, the notation $Q \lesssim_\psi Q'$ means that $Q \leqslant c(\psi)Q'$, where $c(\psi) > 0$ is allowed to depend only on ψ, and similarly for the notations $Q \gtrsim_\psi Q'$ and $Q \asymp_\psi Q'$.

The normalized adjacency matrix of the graph G, denoted A_{G}, is the matrix whose entry at $(i,j) \in [n] \times [n]$ is equal to $\frac{1}{k}\mathbf{1}_{\{i,j\}\in E_{\mathsf{G}}}$. Denote from now on $\lambda_2(\mathsf{G}) = \lambda_2(\mathsf{A}_{\mathsf{G}})$. Let $(X, \|\cdot\|)$ be a finite-dimensional normed space. Fix $D \geqslant 1$ and a mapping $\phi : [n] \to X$ that satisfies

$$\left(\frac{1}{|E_{\mathsf{G}}|}\sum_{\{i,j\}\in E_{\mathsf{G}}} \|\phi(i) - \phi(j)\|^2\right)^{\frac{1}{2}} = \left(\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{n} (\mathsf{A}_{\mathsf{G}})_{ij}\|\phi(i) - \phi(j)\|^2\right)^{\frac{1}{2}} \leqslant D. \tag{3}$$

Condition (3) holds true, for example, if ϕ is D-Lipschitz as a mapping from $([n], d_{\mathsf{G}})$ to $(X, \|\cdot\|)$. Let $\eta > 0$ be the implicit constant in the right hand side of (2), and suppose that ϕ also satisfies

$$\left(\frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n} \|\phi(i) - \phi(j)\|^2\right)^{\frac{1}{2}} \geqslant \eta\frac{\log n}{\log k}. \tag{4}$$

Due to (2) and the Cauchy–Schwarz inequality, conditions (3) and (4) hold true simultaneously (for an appropriately chosen ϕ) if e.g. $([n], d_{\mathsf{G}})$ embeds with average distortion D into $(X, \|\cdot\|)$. At the same time, by an application of Theorem 1 with $x_i = \phi(i)$ and $\mathsf{A} = \mathsf{A}_{\mathsf{G}}$,

$$\dim(X) \gtrsim e^{\frac{C\eta(1-\lambda_2(\mathsf{A}))\log n}{D\log k}} = n^{\frac{C\eta(1-\lambda_2(\mathsf{A}))}{D\log k}}.$$

For ease of later reference, we record this conclusion as the following corollary.

▶ **Corollary 2.** *There exists a universal constant $\rho \in (0,\infty)$ such that for every $n \in \mathbb{N}$ and $k \in [n]$, if $\mathsf{G} = ([n], E_{\mathsf{G}})$ is a connected n-vertex k-regular graph and $D \geqslant 1$, then the dimension of any normed space $(X, \|\cdot\|)$ into which the metric space $([n], d_{\mathsf{G}})$ embeds with average distortion D must satisfy $\dim(X) \gtrsim n^{c(\mathsf{G})/D}$, where $c(\mathsf{G}) = \rho(1 - \lambda_2(\mathsf{A}))/\log k$.*

For every $n \in \mathbb{N}$ there exists a 4-regular graph $\mathsf{G}_n = ([n], E_{\mathsf{G}_n})$ with $\lambda_2(\mathsf{G}_n) \leqslant 1 - \delta$, where $\delta \in (0,1)$ is a universal constant; see the survey [18] for this statement as well as much more on such *expander graphs*. It therefore follows from Corollary 2 that for every $n \in \mathbb{N}$ there exists an n-point metric space $\mathcal{M}_n$ with the property that its embeddability into any normed space with average distortion D forces the dimension of that normed space to be at least $n^{c/D}$, where $c > 0$ is a universal constant. The significance of this statement will be discussed in Section 1.1 below.

The desire to obtain Corollary 2 was the goal that initiated our present investigation, because Corollary 2 resolves (negatively) a question that was posed by Andoni, Nikolov, Razenshteyn and Waingarten [3, Section 1.6] in the context of their work on efficient approximate nearest neighbor search (NNS). Specifically, they devised in [3] an approach for proving a hardness result for NNS that requires the existence of an n-vertex expander that embeds with bi-Lipschitz distortion $O(1)$ into some normed space of dimension $n^{o(1)}$. Corollary 2 shows that no such expander exists. One may view this statement as a weak indication that perhaps an algorithm for NNS in general norms could be designed with better performance than what is currently known, but we leave this interesting algorithmic question for future research and refer to [3] for a full description of this connection. The previously best-known bound in the context of Corollary 2 was due to Linial, London and Rabinovich in [28, Proposition 4.2], where it was shown that if G is $O(1)$-regular and $\lambda_2(\mathsf{G}) = 1 - \Omega(1)$, then any normed space X into which G embeds with average distortion D must satisfy $\dim(X) \gtrsim (\log n)^2/D^2$. The above exponential improvement over [28] is sharp, up to the value of c, as shown by Johnson, Lindenstrauss and Schechtman [21].

1.1 Dimensionality reduction

The present work relates to fundamental questions in mathematics and computer science that have been extensively investigated over the past three decades, and are of major current importance. The overarching theme is that of *dimensionality reduction*, which corresponds to the desire to "compress" n-point metric spaces using representations with few coordinates, namely embeddings into $\mathbb{R}^k$ with (hopefully) k small, in such a way that pairwise distances could be (approximately) recovered by computing lengths in the image with respect to an appropriate norm on $\mathbb{R}^k$. Corollary 2 asserts that this cannot be done in general if one aims for compression to $k = n^{o(1)}$ coordinates. In essence, it states that a spectral gap induces an inherent (power-type) high-dimensionality even if one allows for recovery of pairwise distances with large multiplicative errors, or even while only approximately preserving two averages of the squared distances: along edges and all pairs, corresponding to (3) and (4), respectively. In other words, we isolate two specific averages of pairwise squared distances of a finite collection of vectors in an arbitrary normed space, and show that if the ratio of these averages is roughly (i.e., up to a fixed but potentially large factor) the same as in an expander then the dimension of the ambient space must be large.

In addition to obtaining specific results along these lines, there is need to develop techniques to address dimensionality questions that relate nonlinear (metric) considerations to the linear dimension of the vector space. Our main conceptual contribution is to exhibit a new approach to a line of investigations that previously yielded comparable results using algebraic techniques. In contrast, here we use an analytic method arising from a recently developed theory of nonlinear spectral gaps.

Adopting the terminology of [28, Definition 2.1], given $D \in [1, \infty)$, $n \in \mathbb{N}$ and an n-point metric space $\mathcal{M}$, define a quantity $\dim_D(\mathcal{M}) \in \mathbb{N}$, called the (distortion-D) *metric dimension* of $\mathcal{M}$, to be the minimum $k \in \mathbb{N}$ for which there exists a k-dimensional normed space $X_{\mathcal{M}}$ such that $\mathcal{M}$ embeds into $X_{\mathcal{M}}$ with distortion D. We always have $\dim_D(\mathcal{M}) \leqslant \dim_1(\mathcal{M}) \leqslant n - 1$ by the classical Fréchet isometric embedding [17] into ℓ_∞^{n-1}. In their seminal work [20], Johnson and Lindenstrauss asked [20, Problem 3] whether $\dim_D(\mathcal{M}) = O(\log n)$ for some $D = O(1)$ and every n-point metric space $\mathcal{M}$. Observe that the $O(\log n)$ bound arises naturally here, as it cannot be improved due to a standard volumetric argument when one considers embeddings of the n-point equilateral space; see also Remark 4 below for background on the Johnson–Lindenstrauss question in the context of the Ribe program.

Nevertheless, Bourgain proved [11, Corollary 4] that this question has a negative answer. He showed that for arbitrarily large $n \in \mathbb{N}$ there is an n-point metric space $\mathcal{M}_n$ such that $\dim_D(\mathcal{M}) \gtrsim (\log n)^2/(D \log\log n)^2$ for every $D \in [1,\infty)$. He also posed in [11] the natural question of determining the asymptotic behavior of the maximum of $\dim_D(\mathcal{M})$ over all n-point metric spaces $\mathcal{M}$. It took over a decade for this question to be resolved.

In terms of upper bounds, Johnson, Lindenstrauss and Schechtman [21] proved that there exists a universal constant $\alpha > 0$ such that for every $D \geqslant 1$ and $n \in \mathbb{N}$ we have $\dim_D(\mathcal{M}) \lesssim_D n^{\alpha/D}$ for any n-point metric space $\mathcal{M}$. In [29, 30], Matoušek improved this result by showing that one can actually embed $\mathcal{M}$ with distortion D into ℓ_∞^k for some $k \in \mathbb{N}$ satisfying $k \lesssim_D n^{\alpha/D}$, i.e., the target normed space need not depend on $\mathcal{M}$ (Matoušek's proof is also simpler than that of [21], and it yields a smaller value of α; see the exposition in Chapter 15 of the monograph [32]).

In terms of lower bounds, an asymptotic improvement over [11] was made by Linial, London and Rabinovich [28, Proposition 4.2], who showed that for arbitrarily large $n \in \mathbb{N}$ there exists an n-point metric space $\mathcal{M}_n$ such that $\dim_D(\mathcal{M}_n) \gtrsim (\log n)^2/D^2$ for every $D \in [1,\infty)$. For small distortions, Arias-de-Reyna and Rodríguez-Piazza proved [4] the satisfactory assertion that for arbitrarily large $n \in \mathbb{N}$ there exists an n-point metric space $\mathcal{M}_n$ such that $\dim_D(\mathcal{M}_n) \gtrsim_D n$ for every $1 \leqslant D < 2$. For larger distortions, it was asked in [4, page 109] whether for every $D \in (2,\infty)$ and $n \in \mathbb{N}$ we have $\dim_D(\mathcal{M}) \lesssim_D (\log n)^{O(1)}$ for any n-point metric space $\mathcal{M}$. In [30], Matoušek famously answered this question negatively by proving Theorem 3 below via a clever argument that relies on (a modification of) graphs of large girth with many edges and an existential counting argument (inspired by ideas of Alon, Frankl and Rödl [1]) that uses the classical theorem of Milnor [37] and Thom [50] from real algebraic geometry.

▶ **Theorem 3** (Matoušek). *For every $D \geqslant 1$ and arbitrarily large $n \in \mathbb{N}$, there is an n-point metric space $\mathcal{M}_n(D)$ such that $\dim_D\big(\mathcal{M}_n(D)\big) \gtrsim_D n^{c/D}$, where $c > 0$ is a universal constant.*

Due to the upper bound that was quoted above, Matoušek's theorem satisfactorily answers the questions of Johnson–Lindenstrauss and Bourgain, up to the universal constant c. Corollary 2 also resolves these questions, via an approach for deducing dimensionality lower bounds from rough (bi-Lipschitz) metric information that differs markedly from Matoušek's argument.

Our solution has some new features. The spaces $\mathcal{M}_n(D)$ of Theorem 3 can actually be taken to be independent of the distortion D, while the construction of [30] depends on D (it is based on graphs of girth of order D). One could alternatively achieve this by considering the disjoint union of the spaces $\{\mathcal{M}_n(2^k)\}_{k=0}^m$ for $m \asymp \log n$, which is a metric space of size $O(n \log n)$. More importantly, rather than using an ad-hoc construction (relying also on a non-constructive existential statement) as in [30], here we specify a natural class of metric spaces, namely the shortest-path metrics on expanders (see also Remark 5 below), for which Theorem 3 holds. Obtaining this result for this concrete class of metric spaces is needed to answer the question of [3] that was quoted above. Finally, Matoušek's approach based on the Millnor–Thom theorem uses the fact that the embedding has controlled bi-Lipschitz distortion, while our approach is robust in the sense that it deduces the stated lower bound on the dimension from an embedding with small average distortion.

▶ Remark 4. The *Ribe program* aims to uncover an explicit "dictionary" between the local theory of Banach spaces and general metric spaces, inspired by an important rigidity theorem of Ribe [47] that indicates that a dictionary of this sort should exist. See the introduction of [12] as well as the surveys [22, 38, 6] and the monograph [44] for more on this topic. While more recent research on dimensionality reduction is most often motivated by the need to compress data, the initial motivation of the question of Johnson and Lindenstrauss [20] that

we quoted above arose from the Ribe program. It seems simplest to include here a direct quotation of Matoušek's explanation in [30, page 334] for the origin of the investigations that led to Theorem 3.

> *...This investigation started in the context of the local Banach space theory, where the general idea was to obtain some analogs for general metric spaces of notions and results dealing with the structure of finite dimensional subspaces of Banach spaces. The distortion of a mapping should play the role of the norm of a linear operator, and the quantity* $\log n$*, where* n *is the number of points in a metric space, would serve as an analog of the dimension of a normed space. Parts of this programme have been carried out by Bourgain, Johnson, Lindenstrauss, Milman and others...*

Despite many previous successes of the Ribe program, not all of the questions that it raised turned out to have a positive answer (see e.g. [33]). Theorem 3 is among the most extreme examples of failures of natural steps in the Ribe program, with the final answer being exponentially worse than the initial predictions. Corollary 2 provides a further explanation of this phenomenon.

▶ Remark 5. The reasoning prior to Corollary 2 gives the following statement that applies to regular graphs that need not have bounded degree. Fix $\beta > 0$ and $n \in \mathbb{N}$. Suppose that $\mathsf{G} = ([n], E_{\mathsf{G}})$ is a connected regular graph that satisfies $(1 - \lambda_2(\mathsf{G})) \sum_{i=1}^n \sum_{j=1}^n d_{\mathsf{G}}(i,j) \geqslant \beta n^2 \log n$. Then, $\dim_D(\mathsf{G}) \gtrsim n^{C\beta/D}$ for every $D \geqslant 1$, where $C > 0$ is the universal constant of Theorem 1 and we use the notation $\dim_D([n], d_{\mathsf{G}}) = \dim_D(\mathsf{G})$. Let $\mathrm{diam}(\mathsf{G})$ be the diameter of $([n], d_{\mathsf{G}})$ and suppose (for simplicity) that G is vertex-transitive (e.g., G can be the Cayley graph of a finite group). Then, it is simple to check that $n^2 \,\mathrm{diam}(\mathsf{G}) \geqslant \sum_{i=1}^n \sum_{j=1}^n d_{\mathsf{G}}(i,j) \geqslant n^2 \,\mathrm{diam}(\mathsf{G})/4$ (see. e.g. equation (4.24) in [40]), and therefore the above reasoning shows that every vertex-transitive graph satisfies

$$\forall\, D \geqslant 1, \qquad \dim_D(\mathsf{G}) \gtrsim e^{\frac{C}{4D}(1-\lambda_2(\mathsf{G}))\,\mathrm{diam}(\mathsf{G})}. \tag{5}$$

In particular, it follows from (5) that if $([n], d_{\mathsf{G}})$ embeds with distortion $O(1)$ into some normed space of dimension $(\log n)^{O(1)}$, then necessarily $(1 - \lambda_2(\mathsf{G}))\,\mathrm{diam}(\mathsf{G}) \lesssim \log\log n$.

There are many examples of Cayley graphs $\mathsf{G} = ([n], E_{\mathsf{G}})$ for which $\lambda_2(\mathsf{G}) = 1 - \Omega(1)$ and $\mathrm{diam}(\mathsf{G}) \gtrsim \log n$ (see e.g. [2, 43]). In all such examples, (5) asserts that $\dim_D(\mathsf{G}) \gtrsim n^{c/D}$ for some universal constant $c > 0$. The Cayley graph that was studied in [23] (a quotient of the Hamming cube by a good code) now shows that there exist arbitrarily large n-point metric spaces $\mathcal{M}_n$ with $\dim_1(\mathcal{M}_n) \lesssim \log n$ (indeed, $\mathcal{M}_n$ embeds isometrically into ℓ_1^k for some $k \lesssim \log n$), yet $\mathcal{M}_n$ has a $O(1)$-Lipschitz quotient (see [9] for the relevant definition) that does not embed with distortion $O(1)$ into any normed space of dimension $n^{o(1)}$. To the best of our knowledge, it wasn't previously known that the metric dimension $\dim_D(\cdot)$ can become asymptotically larger (and even increase exponentially) under Lipschitz quotients, which is yet another major departure from the linear theory, in contrast to what one would normally predict in the context of the Ribe program.

2 Proof of Theorem 1

Modulo the use of a theorem about nonlinear spectral gaps which is a main result of [40], our proof of Theorem 1 is not long. We rely here on an argument that perturbs any finite-dimensional normed space (by complex interpolation with its distance ellipsoid) so as to make the result of [40] become applicable, and we proceed to show that by optimizing over the size of the perturbation one can deduce the desired dimensionality-reduction lower bound. This idea is the main conceptual contribution of the present work. We begin with an informal overview of this argument.

2.1 Overview

The precursors of our approach are the works [26] and [25] about the impossibility of dimensionality reduction in ℓ_1 and ℓ_∞, respectively. It was shown in [26] (respectively [25]) that a certain n-point metric space $\mathcal{M}_1$ (respectively $\mathcal{M}_\infty$) does not admit a low-distortion embedding into $X = \ell_1^k$ (respectively $X = \ell_\infty^k$) with k small, by arguing that if k were indeed small then there would be a normed space Y that is "close" to X, yet any embedding of $\mathcal{M}_1$ (respectively $\mathcal{M}_\infty$) into Y incurs large distortion. This leads to a contradiction, provided that the assumed embedding of $\mathcal{M}_1$ (respectively $\mathcal{M}_\infty$) into X had sufficiently small distortion relative to the closeness of Y to X. In the setting of [26, 25], there is a natural one-parameter family of normed spaces that tends to X, namely the spaces ℓ_p^k with $p \to 1$ or $p \to \infty$, respectively, and indeed the space Y is taken to be an appropriate member of this family. For a general normed space X, it is a priori unclear how to perturb it so as to implement this strategy. Moreover, the arguments of [26, 25] rely on additional special properties of the specific normed spaces in question that hinder their applicability to general normed spaces: The example of [26] is unsuited to the question that we study here because it was shown in [24] that in fact $\dim_D(\mathcal{M}_1) \lesssim \log n$ for some $D = O(1)$; and, the proof in [25] of the non-embeddability of $\mathcal{M}_\infty$ into Y is based on a theorem of Matoušek [31] whose proof relies heavily on the coordinate structure of $Y = \ell_p^k$. We shall overcome the former difficulty by using the complex interpolation method to perturb X, and we shall overcome the latter difficulty by invoking the theory of nonlinear spectral gaps.

Suppose that $(X, \|\cdot\|)$ is a finite-dimensional normed space. The perturbative step of our argument considers the Hilbert space H whose unit ball is an ellipsoid that is closest to the unit ball of X, i.e., a *distance ellipsoid* of X; see Section 2.2 below. We then use the complex interpolation method (see Section 2.4.3 below) to obtain a one-parameter family of normed spaces $\{[X_{\mathbb{C}}, H_{\mathbb{C}}]_\theta\}_{\theta\in[0,1]}$ that intertwines the complexifications (see Section 2.4.2 below) of X and H, respectively. These intermediate spaces will serve as a proxy for the one-parameter family $\{\ell_p^n\}_{p\in[1,\infty]}$ that was used in [25]. In order to see how they fit into this picture we briefly recall the argument of [25].

Suppose that $\mathsf{G} = ([n], E_{\mathsf{G}})$ is a $O(1)$-regular graph with $\lambda_2(\mathsf{G}) = 1 - \Omega(1)$ (i.e., an expander). In [25, Proposition 4.1] it was shown that for every $D \geqslant 1$ and $k \in \mathbb{N}$, if $([n], d_{\mathsf{G}})$ embeds with distortion D into ℓ_∞^k, then necessarily $k \gtrsim n^{c/D}$ for some universal constant $c > 0$. This is so because Matoušek proved in [31] that for any $p \in [1, \infty)$, any embedding of $([n], d_{\mathsf{G}})$ into ℓ_p incurs distortion at least $\eta(\log n)/p$, where $\eta > 0$ is a universal constant. The norms on ℓ_∞^k and $\ell_{\log k}^k$ are within a factor of e of each other, so it follows that $D \geqslant \eta(\log n)/(e \log k)$, i.e., $k \geqslant n^{\eta/(eD)}$.

The reason for the distortion lower bound of [31] that was used above is that [31] shows that there exists a universal constant $C > 0$ such that for every $p \geqslant 1$ we have

$$\forall\, t_1, \ldots, t_n \in \mathbb{R}, \qquad \frac{1}{n^2}\sum_{i=1}^n\sum_{j=1}^n |t_i - t_j|^p \leqslant \frac{(Cp)^p}{|E_{\mathsf{G}}|}\sum_{\{i,j\}\in E_{\mathsf{G}}} |t_i - t_j|^p. \tag{6}$$

The proof of (6) relies on the fact that the case $p = 2$ of (6) is nothing more than the usual Poincaré inequality that follows through elementary linear algebra from the fact that $\lambda_2(\mathsf{G})$ is bounded away from 1, combined with an extrapolation argument that uses elementary inequalities for real numbers (see also the expositions in [8, 42]). By summing (6) over coordinates we deduce that

$$\forall\, x_1, \ldots, x_n \in \ell_p, \qquad \left(\frac{1}{n^2}\sum_{i=1}^n\sum_{j=1}^n \|x_i - x_j\|_p^p\right)^{\frac{1}{p}} \lesssim p\left(\frac{1}{|E_{\mathsf{G}}|}\sum_{\{i,j\}\in E_{\mathsf{G}}} \|x_i - x_j\|_p^p\right)^{\frac{1}{p}}. \tag{7}$$

This implies that any embedding of $([n], d_{\mathsf{G}})$ into ℓ_p incurs average distortion at least a constant multiple of $(\log n)/p$ via the same reasoning as the one that preceded Corollary 2.

The reliance on coordinate-wise inequalities in the derivation of (7) is problematic when it comes to the need to treat a general finite-dimensional normed space $(X, |\cdot\|)$. This "scalar" way of reasoning also leads to the fact that in (7) the ℓ_p norm is raised to the power p. Since, even in the special case $X = \ell_p^k$, (7) is applied in the above argument when $p = \log \dim(X)$, this hinders our ability to deduce an estimate such as the conclusion (1) of Theorem 1.

To overcome this obstacle, we consider a truly nonlinear (quadratic) variant of (7) which is known as a *nonlinear spectral-gap inequality*. See Section 2.3 below for the formulation of this concept, based on a line of works in metric geometry that has been more recently investigated systematically in [34, 35, 40, 36]. Our main tool is a result of [40], which is quoted as Theorem 9 below. It provides an estimate in the spirit of (7) for n-tuples of vectors in each of the complex interpolation spaces $\{[X_{\mathbb{C}}, H_{\mathbb{C}}]_\theta\}_{\theta\in(0,1]}$, in terms of the parameter θ and the p-smoothness constant of the normed space $[X_{\mathbb{C}}, H_{\mathbb{C}}]_\theta$ (see Section 2.4.1 below for the relevant definition). We then implement the above perturbative strategy by estimating the closeness of X to a subspace of $[X_{\mathbb{C}}, H_{\mathbb{C}}]_\theta$, and optimizing over the auxiliary interpolation parameter θ.

While the result of [40] that we use here is substantial, we encourage readers to examine its proof rather than relying on it as a "black box," because we believe that this proof is illuminating and accessible to non-experts. Specifically, the proof in [40] of Theorem 9 below relies on Ball's notion of Markov type [5] p through the martingale method of [41], in combination with complex interpolation and a trick of V. Lafforgue that was used by Pisier in [45]. It is interesting to observe that here we use the fact that the bound that is obtained in [40] depends on the p-smoothness constant of $[X_{\mathbb{C}}, H_{\mathbb{C}}]_\theta$, but it contains no other dependence on p. Since in our final optimization over θ we take p to be very close to 1, we can't allow for an implicit dependence on p that is unbounded as $p \to 1$. Such a p-independent bound is indeed obtained in [40], but unlike the present application, it was a side issue in [40], where only the case $p = 2$ was used.

2.2 Distance ellipsoids

Recall that given $\mathsf{d} \in [1, \infty)$, a Banach space $(X, \|\cdot\|)$ is said to be d-isomorphic to a Hilbert space if it admits a scalar product $\langle\cdot,\cdot\rangle : X \times X \to \mathbb{R}$, such that if we denote its associated Hilbertian norm by $|x| = \sqrt{\langle x, x\rangle}$, then

$$\forall\, x \in X, \qquad |x| \leqslant \|x\| \leqslant \mathsf{d}|x|. \tag{8}$$

The (Banach–Mazur) *Euclidean distance* of X, denoted $\mathsf{d}_X \in [1, \infty)$, is then defined to be the infimum over those $\mathsf{d} \in [1, \infty)$ for which (8) holds true. If X is not d-isomorphic to a Hilbert space for any $\mathsf{d} \in [1, \infty)$, then we write $\mathsf{d}_X = \infty$. If X is finite-dimensional, then John's theorem [19] asserts that $\mathsf{d}_X \leqslant \sqrt{\dim(X)}$. By a standard compactness argument, if X is finite-dimensional, then the infimum in the definition of d_X is attained. In that case, the unit ball of the Hilbertian norm $|\cdot|$, i.e., the set $\{x \in X : |x| \leqslant 1\}$, is commonly called a *distance ellipsoid* of X.

2.3 Nonlinear spectral gaps

Suppose that $(\mathcal{M}, d_{\mathcal{M}})$ is a metric space, $n \in \mathbb{N}$ and $p \in (0, \infty)$. Following [35], the (reciprocal of) the *nonlinear spectral gap* with respect to $d_{\mathcal{M}}^p$ of a symmetric stochastic matrix

$\mathsf{A} = (a_{ij}) \in M_n(\mathbb{R})$, denoted $\gamma(\mathsf{A}, d_{\mathcal{M}}^p)$, is the smallest $\gamma \in (0,\infty)$ such that

$$\forall\, x_1,\ldots,x_n \in \mathcal{M}, \qquad \frac{1}{n^2}\sum_{i=1}^n\sum_{j=1}^n d_{\mathcal{M}}(x_i,x_j)^p \leqslant \frac{\gamma}{n}\sum_{i=1}^n\sum_{j=1}^n a_{ij} d_{\mathcal{M}}(x_i,x_j)^p.$$

We refer to [35] for an extensive discussion of this notion; it suffices to state here that the reason for this nomenclature is that if we denote the standard metric on the real line by $d_{\mathbb{R}}$ (i.e., $d_{\mathbb{R}}(s,t) = |s-t|$ for every $s,t \in \mathbb{R}$), then it is straightforward to check that $\gamma(\mathsf{A}, d_{\mathbb{R}}^2) = 1/(1-\lambda_2(\mathsf{A}))$.

In general, nonlinear spectral gaps can differ markedly from the usual (reciprocal of) the gap in the (linear) spectrum, though [40] is devoted to an investigation of various settings in which one can obtain comparison inequalities for nonlinear spectral gaps when the underlying metric is changed. Estimates on $\gamma(\mathsf{A}, d_{\mathcal{M}}^p)$ have a variety of applications in metric geometry, and here we establish their relevance to dimensionality reduction. Specifically, we shall derive below the following result, which will be shown to imply Theorem 1.

▶ **Theorem 6** (Nonlinear spectral gap for Hilbert isomorphs)**.** *Fix $n \in \mathbb{N}$ and a symmetric stochastic matrix $\mathsf{A} = (a_{ij}) \in M_n(\mathbb{R})$. Then for every normed space $(X, \|\cdot\|)$ with $\mathsf{d}_X < \infty$, we have*

$$\gamma\big(\mathsf{A}, \|\cdot\|^2\big) \lesssim \begin{cases} \frac{\mathsf{d}_X^2}{1-\lambda_2(\mathsf{A})} & \text{if } \ \mathsf{d}_X\sqrt{1-\lambda_2(\mathsf{A})} \leqslant e, \\ \left(\frac{\log\left(\mathsf{d}_X\sqrt{1-\lambda_2(\mathsf{A})}\right)}{1-\lambda_2(\mathsf{A})}\right)^2 & \text{if } \ \mathsf{d}_X\sqrt{1-\lambda_2(\mathsf{A})} > e. \end{cases} \tag{9}$$

Proof of Theorem 1 assuming Theorem 6. We claim that (9) implies the following simpler bound.

$$\gamma\big(\mathsf{A}, \|\cdot\|^2\big) \lesssim \left(\frac{\log\left(\mathsf{d}_X\sqrt{2}\right)}{1-\lambda_2(\mathsf{A})}\right)^2. \tag{10}$$

Indeed, if $\mathsf{d}_X\sqrt{1-\lambda_2(\mathsf{A})} > e$, then the right hand side of (10) is at least the right hand side of (9) due to the fact that, since A is symmetric and stochastic, $\lambda_2(\mathsf{A}) \geqslant -1$, so that $\sqrt{1-\lambda_2(\mathsf{A})} \leqslant \sqrt{2}$. On the other hand, if $\mathsf{d}_X\sqrt{1-\lambda_2(\mathsf{A})} \leqslant e$ then $\mathsf{d}_X^2/(1-\lambda_2(\mathsf{A})) \leqslant e^2/(1-\lambda_2(\mathsf{A}))^2$, which is at most a universal constant multiple of the right hand side of (10) because $\mathsf{d}_X \geqslant 1$.

By the definition of $\gamma\big(\mathsf{A}, \|\cdot\|^2\big)$, it follows from (10) that there exists a universal constant $\alpha > 0$ such that for every $x_1,\ldots,x_n \in X$ we have

$$\frac{1}{n^2}\sum_{i=1}^n\sum_{j=1}^n \|x_i-x_j\|^2 \leqslant \alpha\left(\frac{\log\left(\mathsf{d}_X\sqrt{2}\right)}{1-\lambda_2(\mathsf{A})}\right)^2 \cdot \frac{1}{n}\sum_{i=1}^n\sum_{j=1}^n a_{ij}\|x_i-x_j\|^2.$$

This estimate simplifies to give

$$\mathsf{d}_X \geqslant \frac{1}{\sqrt{2}}\exp\left(\frac{1-\lambda_2(\mathsf{A})}{\sqrt{\alpha n}}\left(\frac{\sum_{i=1}^n\sum_{j=1}^n \|x_i-x_j\|^2}{\sum_{i=1}^n\sum_{j=1}^n a_{ij}\|x_i-x_j\|^2}\right)^{\frac{1}{2}}\right). \tag{11}$$

The desired estimate (1) (with $C = 2/\sqrt{\alpha}$) now follows because $\mathsf{d}_X \leqslant \sqrt{\dim(X)}$ by [19]. ◀

▶ Remark 7. Suppose that $\mathsf{G} = ([n], E_{\mathsf{G}})$ is a Cayley graph of a finite group such that $\lambda_2(\mathsf{G}) = 1 - \Omega(1)$. The metric space $([n], d_{\mathsf{G}})$ embeds with distortion $\mathrm{diam}(\mathsf{G})$ into ℓ_2^{n-1} by considering any bijection between $[n]$ and the vertices of the n-simplex. There is therefore no

a priori reason why it wouldn't be possible to embed $([n], d_{\mathsf{G}})$ with distortion $O(1)$ into some normed space X whose Banach–Mazur distance from a Hilbert space is at least a sufficiently large multiple of $\mathrm{diam}(\mathsf{G})$. But this is not so if $\mathrm{diam}(\mathsf{G})$ is sufficiently large. Indeed, recalling Remark 5, it follows from (11) that any embedding of $([n], d_{\mathsf{G}})$ into X incurs distortion that is at least a universal constant multiple of $\mathrm{diam}(\mathsf{G})/\log(2\mathsf{d}_X)$. Thus, even if we allow d_X to be as large as $\mathrm{diam}(\mathsf{G})^{O(1)}$, then any embedding of $([n], d_{\mathsf{G}})$ into X incurs distortion that is at least a universal constant multiple of $\mathrm{diam}(\mathsf{G})/\log \mathrm{diam}(\mathsf{G})$. Also, if $\mathrm{diam}(\mathsf{G}) \gtrsim \log n$ (e.g., if G has bounded degree) then this means that any embedding of $([n], d_{\mathsf{G}})$ into X incurs distortion that is at least a universal constant multiple of $(\log n)/\log(2\mathsf{d}_X)$ and, say, even if we allow d_X to be as large as $(\log n)^{O(1)}$, then any embedding of $([n], d_{\mathsf{G}})$ into X incurs distortion that is at least a universal constant multiple of $(\log n)/\log\log n$.

2.4 Proof of Theorem 6

We have seen that in order to prove Theorem 1 it suffices to prove Theorem 6. In order to do so, we shall first describe several ingredients that appear in its proof.

2.4.1 Uniform convexity and smoothness

Suppose that $(X, \|\cdot\|)$ is a normed space and fix $p, q > 0$ satisfying $1 \leqslant p \leqslant 2 \leqslant q$. Following Ball, Carlen and Lieb [7], the p-smoothness constant of X, denoted $\mathcal{S}_p(X)$, is the infimum over those $S > 0$ such that

$$\forall\, x, y \in X, \qquad \|x+y\|^p + \|x-y\|^p \leqslant 2\|x\|^p + 2S^p\|y\|^p. \tag{12}$$

(If no such S exists, then define $\mathcal{S}_p(X) = \infty$.) By the triangle inequality we always have $\mathcal{S}_1(X) = 1$. The q-convexity constant of X, denoted $\mathcal{K}_q(X)$, is the infimum over those $K > 0$ such that

$$\forall\, x, y \in X, \qquad 2\|x\|^q + \frac{2}{K^q}\|y\|^q \leqslant \|x+y\|^q + \|x-y\|^q.$$

(As before, if no such K exists, then define $\mathcal{K}_q(X) = \infty$.) We refer to [7] for the relation of these parameters to more traditional moduli of uniform convexity and smoothness that appear in the literature. It is beneficial to work with the quantities $\mathcal{S}_p(X), \mathcal{K}_q(X)$ rather than the classical moduli because they are well-behaved with respect to basic operations, an example of which is the duality $\mathcal{K}_{p/(p-1)}(X^*) = \mathcal{S}_p(X)$, as shown in [7]. Another example that is directly relevant to us is their especially clean behavior under complex interpolation, as derived in Section 2.4.3 below.

2.4.2 Complexification

All of the above results were stated for normed spaces over the real numbers, but in the ensuing proofs we need to consider normed spaces over the complex numbers. We do so through the use of the standard notion of complexification. Specifically, for every normed space $(X, \|\cdot\|_X)$ over $\mathbb{R}$ one associates as follows a normed space $(X_{\mathbb{C}}, \|\cdot\|_{X_{\mathbb{C}}})$ over $\mathbb{C}$. The underlying vector space is $X_{\mathbb{C}} = X \times X$, which is viewed as a vector space over $\mathbb{C}$ by setting $(\alpha+\beta i)(x, y) = (\alpha x - \beta y, \beta x + \alpha y)$ for every $\alpha, \beta \in \mathbb{R}$ and $x, y \in X$. The norm on $X_{\mathbb{C}}$ is given by

$$\forall\, x, y \in X, \qquad \|(x, y)\|_{X_{\mathbb{C}}} = \left(\frac{1}{\pi}\int_0^{2\pi} \big\|(\cos\theta)x - (\sin\theta)y\big\|_X^2\, \mathrm{d}\theta\right)^{\frac{1}{2}}. \tag{13}$$

The normalization in (13) ensures that $x \mapsto (x,0)$ is an isometric embedding of X into $X_{\mathbb{C}}$. It is straightforward to check that for every $n \in \mathbb{N}$ and every symmetric stochastic matrix $\mathsf{A} \in M_n(\mathbb{R})$ we have $\gamma(\mathsf{A}, \|\cdot\|_X^2) = \gamma(\mathsf{A}, \|\cdot\|_{X_{\mathbb{C}}}^2)$. Also, $\mathcal{S}_2(X_{\mathbb{C}}) = \mathcal{S}_2(X)$ and $\mathcal{K}_2(X_{\mathbb{C}}) = \mathcal{K}_2(X)$. When $p \in (1,2)$ and $q \in (2,\infty)$ we have $\mathcal{S}_p(X_{\mathbb{C}}) \asymp \mathcal{S}_p(X)$ and $\mathcal{K}_q(X_{\mathbb{C}}) \asymp \mathcal{K}_q(X)$; if one were to allow the implicit constants in these asymptotic equivalences to depend on p, q then this follows from the results of [16, 15, 7], and the fact that these constants can actually be taken to be universal follows from carrying out the relevant arguments with more care, as done in [39, 35] (see specifically Lemma 6.3 and Corollary 6.4 of [35]). Finally, we have $\mathsf{d}_{X_{\mathbb{C}}} = \mathsf{d}_X$.

2.4.3 Complex interpolation

We very briefly recall Calderón's vector-valued complex interpolation method [13]; see Chapter 4 of the monograph [10] for an extensive treatment. A pair of complex Banach spaces $(Y, \|\cdot\|_Y), (Z, \|\cdot\|_Z)$ is said to be compatible if they are both linearly embedded into a complex linear space W with $Y + Z = W$. The space W is a complex Banach space under the norm $\|w\|_W = \inf\{\|y\|_Y + \|z\|_Z :\ y + z = w\}$. Let $\mathcal{F}(Y,Z)$ denote the space of all bounded continuous functions $\psi : \{\zeta \in \mathbb{C} :\ 0 \leqslant \Re(\zeta) \leqslant 1\} \to W$ that are analytic on the open strip $\{\zeta \in \mathbb{C} :\ 0 < \Re(\zeta) < 1\}$. To every $\theta \in [0,1]$ one associates a Banach space $[Y,Z]_\theta$ as follows. The underlying vector space is $\{\psi(\theta) :\ \psi \in \mathcal{F}(Y,Z)\}$, and the norm of $w \in [Y,Z]_\theta$ is given by $\|w\|_{[Y,Z]_\theta} = \inf_{\{\psi \in \mathcal{F}(Y,Z):\ \psi(\theta) = w\}} \max\{\sup_{t\in\mathbb{R}} \|\psi(ti)\|_Y, \sup_{t\in\mathbb{R}} \|\psi(1+ti)\|_Z\}$. This turns $[Y,Z]_\theta$ into a Banach space, and we have $[Y,Z]_0 = Y, [Y,Z]_1 = Z$. Also, $[Y,Y]_\theta = Y$ for every $\theta \in [0,1]$.

Calderón's vector-valued version [13] of the Riesz–Thorin theorem [48, 51] asserts that if $(Y, \|\cdot\|_Y), (Z, \|\cdot\|_Z)$ and $(U, \|\cdot\|_U), (V, \|\cdot\|_V)$ are two compatible pairs of complex Banach spaces and $T : Y \cap Z \to U \cap V$ is a linear operator that extends to a bounded linear operator from $(Y, \|\cdot\|_Y)$ to $(U, \|\cdot\|_U)$ and from $(Z, \|\cdot\|_Z)$ to $(V, \|\cdot\|_V)$, then the following operator norm bounds hold true.

$$\forall\, \theta \in [0,1], \qquad \|T\|_{[Y,Z]_\theta \to [U,V]_\theta} \leqslant \|T\|_{Y\to U}^{1-\theta} \|T\|_{Z\to V}^{\theta}. \tag{14}$$

The ensuing proof of Theorem 6 uses the interpolation inequality (14) four times (one of which is within the proof of a theorem that we shall quote from [40]; see Theorem 9 below). We shall now proceed to derive some preparatory estimates that will be needed in what follows.

For $p \geqslant 1$, a complex Banach space $(Z, \|\cdot\|_Z)$, and a weight $\omega : \{1,2\} \to [0,\infty)$ on the 2-point set $\{1,2\}$, we denote (as usual) by $L_p(\omega; Z)$ the space $Z \times Z$ equipped with the norm that is given by setting $\|(a,b)\|_{L_p(\omega;Z)}^p = \omega(1)\|a\|_Z^p + \omega(2)\|b\|_Z^p$ for every $a, b \in Z$.

If $(Y, \|\cdot\|_Y), (Z, \|\cdot\|_Z)$ is a compatible pair of complex Banach spaces then by Calderón's vector-valued version of Stein's interpolation theorem [49, Theorem 2] (see part(i) of §13.6 in [13] or Theorem 5.3.6 in [10]), for every $p, q \in [1,\infty]$, $\theta \in [0,1]$ and $\omega, \tau : \{1,2\} \to [0,\infty)$ we have

$$[L_p(\omega; Y), L_q(\tau; Z)]_\theta = L_r\Big(\omega^{\frac{1-\theta}{p}} \tau^{\frac{\theta}{q}}; [Y,Z]_\theta\Big), \quad \text{where} \quad r = \frac{pq}{\theta p + (1-\theta)q}. \tag{15}$$

The equality in (15) is in the sense of isometries, i.e., the norms on both sides coincide.

Suppose that $p_1, p_2 \in [1,2]$ and that the smoothness constants $\mathcal{S}_{p_1}(Y), \mathcal{S}_{p_2}(Z)$ are finite. Fix $S_1 > \mathcal{S}_{p_1}(Y)$ and $S_2 > \mathcal{S}_{p_2}(Z)$. Then by (12) we have

$$\forall\, y_1, y_2 \in Y, \qquad \|y_1 + y_2\|_Y^{p_1} + \|y_1 - y_2\|_Y^{p_1} \leqslant 2\|y_1\|_Y^{p_1} + 2S_1^{p_1}\|y_2\|_Y^{p_1}, \tag{16}$$

and

$$\forall\, z_1, z_2 \in Z, \qquad \|z_1+z_2\|_Z^{p_2} + \|z_1-z_2\|_Z^{p_2} \leqslant 2\|z_1\|_Z^{p_2} + 2S_2^{p_2}\|z_2\|_Z^{p_2}. \tag{17}$$

For every $S > 0$ and $p \geqslant 1$ define $\omega(S,p) : \{1,2\} \to (0,\infty)$ by $\omega(S,p)(1) = 2$ and $\omega(S,p)(2) = 2S^p$. Also, denote the constant function $\mathbf{1}_{\{1,2\}}$ by $\tau : \{1,2\} \to (0,\infty)$, i.e., $\tau(1) = \tau(2) = 1$. With this notation, if we consider the linear operator $T : (Y+Z)\times(Y+Z) \to (Y+Z)\times(Y+Z)$ that is given by setting $T(w_1,w_2) = (w_1+w_2, w_1-w_2)$ for every $w_1, w_2 \in Y+Z$, then

$$\|T\|_{L_{p_1}(\omega(S_1,p_1);Y)\to L_{p_1}(\tau;Y)} \overset{(16)}{\leqslant} 1 \qquad \text{and} \qquad \|T\|_{L_{p_2}(\omega(S_2,p_2);Z)\to L_{p_2}(\tau;Z)} \overset{(17)}{\leqslant} 1. \tag{18}$$

Denoting $r = p_1p_2/(\theta p_1 + (1-\theta)p_2)$, note that $\omega(S_1,p_1)^{(1-\theta)/r}\omega(S_2,p_2)^{\theta/r} = \omega(S_1^{1-\theta}S_2^{\theta}, r)$. Hence, by (15) we have $[L_{p_1}(\omega(S_1,p_1);Y), L_{p_2}(\omega(S_2,p_2);Z)]_\theta = L_r(\omega(S_1^{1-\theta}S_2^{\theta},r);[Y,Z]_\theta)$ and also $[L_{p_1}(\tau;Y);L_{p_2}(\tau;Z)]_\theta = L_r(\tau,[Y,Z]_\theta)$. In combination with (14) and (18), these identities imply that the norm of T as an operator from $L_r(\omega(S_1^{1-\theta}S_2^{\theta},r);[Y,Z]_\theta)$ to $L_r(\tau,[Y,Z]_\theta)$ is at most 1. In other words, every $w_1, w_2 \in [Y,Z]_\theta$ satisfy

$$\|w_1+w_2\|_{[Y,Z]_\theta}^r + \|w_1-w_2\|_{[Y,Z]_\theta}^r \leqslant 2\|w_1\|_{[Y,Z]_\theta}^r + 2\Big(S_1^{1-\theta}S_2^{\theta}\Big)^r\|w_2\|_{[Y,Z]_\theta}^r.$$

Since S_1 and S_2 can be arbitrarily close to $\mathcal{S}_{p_1}(Y)$ and $\mathcal{S}_{p_2}(Z)$, respectively, we conclude that

$$\mathcal{S}_{\frac{p_1p_2}{\theta p_1+(1-\theta)p_2}}\big([Y,Z]_\theta\big) \leqslant \mathcal{S}_{p_1}(Y)^{1-\theta}\mathcal{S}_{p_2}(Z)^{\theta}. \tag{19}$$

By an analogous argument, if $q_1, q_2 \geqslant 2$ and the convexity constants $\mathcal{K}_{q_1}(Y), \mathcal{K}_{q_2}(Z)$ are finite, then

$$\mathcal{K}_{\frac{q_1q_2}{\theta q_1+(1-\theta)q_2}}\big([Y,Z]_\theta\big) \leqslant \mathcal{K}_{q_1}(Y)^{1-\theta}\mathcal{K}_{q_2}(Z)^{\theta}. \tag{20}$$

▶ Remark 8. If one considers the traditional moduli of uniform convexity and smoothness (see e.g. [27] for the definitions), then interpolation statements that are analogous to (19), (20) are an old result of Cwikel and Reisner [14], with the difference that [14] involves implicit constants that depend on p_1, p_2, q_1, q_2. By [7], this statement of [14] yields the estimates (19), (20) with additional factors in the right hand side that depend on p_1, p_2, q_1, q_2. For our present purposes, i.e., for the proof of Theorem 6, it is important to obtain universal constants here. We believe that by carrying out the proofs in [14] with more care this could be achieved, but by working instead with the quantities $\mathcal{S}_p(\cdot), \mathcal{K}_q(\cdot)$ through the above simple (and standard) interpolation argument, we circumvented the need to do this and obtained the clean interpolation statements (19), (20).

Next, suppose that $(X, \|\cdot\|)$ is a Banach space over $\mathbb{R}$ with $\mathsf{d}_X < \infty$. Fix $\mathsf{d} > \mathsf{d}_X$ and a Hilbertian norm $|\cdot| : X \to [0,\infty)$ that satisfies (8). Denote by H the Hilbert space that is induced by $|\cdot|$. Consider the complexifications $X_{\mathbb{C}}$ and $H_{\mathbb{C}}$. Observe that by (13) and (8) for every $x, y \in X$ we have

$$\|(x,y)\|_{H_{\mathbb{C}}} = \sqrt{|x|^2+|y|^2} \quad \text{and} \quad \|(x,y)\|_{H_{\mathbb{C}}} \leqslant \|(x,y)\|_{X_{\mathbb{C}}} \leqslant \mathsf{d}\|(x,y)\|_{H_{\mathbb{C}}}. \tag{21}$$

Since $X_{\mathbb{C}}$ and $H_{\mathbb{C}}$ are isomorphic Banach space with the same underlying vector space (over $\mathbb{C}$), they are a compatible, and therefore for every $\theta \in [0,1]$ we can consider the complex interpolation space $[H_{\mathbb{C}}, X_{\mathbb{C}}]_\theta$. The formal identity operator $\mathsf{I}_{X\times X} : X\times X \to X\times X$ satisfies

$$\|\mathsf{I}_{X\times X}\|_{X_{\mathbb{C}}\to X_{\mathbb{C}}} \leqslant 1, \quad \|\mathsf{I}_{X\times X}\|_{H_{\mathbb{C}}\to H_{\mathbb{C}}} \leqslant 1, \quad \|\mathsf{I}_{X\times X}\|_{X_{\mathbb{C}}\to H_{\mathbb{C}}} \leqslant 1, \quad \|\mathsf{I}_{X\times X}\|_{H_{\mathbb{C}}\to X_{\mathbb{C}}} \leqslant \mathsf{d}. \tag{22}$$

The first two inequalities in (22) are tautological, and the final two inequalities in (22) are a consequence of the inequalities in (21). Hence,

$$\|\mathsf{I}_{X\times X}\|_{[X_{\mathbb{C}},H_{\mathbb{C}}]_\theta\to X_{\mathbb{C}}} = \|\mathsf{I}_{X\times X}\|_{[X_{\mathbb{C}},H_{\mathbb{C}}]_\theta\to[X_{\mathbb{C}},X_{\mathbb{C}}]_\theta} \overset{(14)}{\leqslant} \|\mathsf{I}_{X\times X}\|^{1-\theta}_{X_{\mathbb{C}}\to X_{\mathbb{C}}}\|\mathsf{I}_{X\times X}\|^{\theta}_{H_{\mathbb{C}}\to X_{\mathbb{C}}} \overset{(22)}{\leqslant} \mathsf{d}^\theta,$$

and

$$\|\mathsf{I}_{X\times X}\|_{X_{\mathbb{C}}\to[X_{\mathbb{C}},H_{\mathbb{C}}]_\theta} = \|\mathsf{I}_{X\times X}\|_{[X_{\mathbb{C}},X_{\mathbb{C}}]_\theta\to[X_{\mathbb{C}},H_{\mathbb{C}}]_\theta} \overset{(14)}{\leqslant} \|\mathsf{I}_{X\times X}\|^{1-\theta}_{X_{\mathbb{C}}\to X_{\mathbb{C}}}\|\mathsf{I}_{X\times X}\|^{\theta}_{X_{\mathbb{C}}\to H_{\mathbb{C}}} \overset{(22)}{\leqslant} 1.$$

These two estimates can be restated as follows.

$$\forall\, x,y\in X,\qquad \|(x,y)\|_{[X_{\mathbb{C}},H_{\mathbb{C}}]_\theta} \leqslant \|(x,y)\|_{X_{\mathbb{C}}} \leqslant \mathsf{d}^\theta\|(x,y)\|_{[X_{\mathbb{C}},H_{\mathbb{C}}]_\theta}. \tag{23}$$

In what follows, we will use crucially the following theorem, which relates nonlinear spectral gaps to complex interpolation and uniform smoothness; this result appears in [40] as Corollary 4.7.

▶ **Theorem 9.** *Suppose that $(\mathcal{H},\|\cdot\|_{\mathcal{H}})$ and $(Z,\|\cdot\|_Z)$ are a compatible pair of complex Banach spaces, with $(\mathcal{H},\|\cdot\|_{\mathcal{H}})$ being a Hilbert space. Suppose that $q\in[1,2]$ and $\theta\in(0,1]$. For every $n\in\mathbb{N}$ and every symmetric stochastic matrix $\mathsf{A}\in M_n(\mathbb{R})$ we have*

$$\gamma\left(\mathsf{A},\|\cdot\|^2_{[Z,\mathcal{H}]_\theta}\right) \lesssim \frac{\mathcal{S}_q\left([Z,\mathcal{H}]_\theta\right)^2}{\theta^{\frac{2}{q}}\left(1-\lambda_2(\mathsf{A})\right)^{\frac{2}{q}}}. \tag{24}$$

We note in passing that in [40] (specifically, in the statement of [40, Theorem 4.5]) we have the following misprint: (24) is stated there for the transposed interpolation space $[\mathcal{H},X]_\theta$ rather than the correct space $[X,\mathcal{H}]_\theta$ as above. This misprint is not confusing when one reads [40] in context rather the statement of [40, Theorem 4.5] in isolation (e.g., clearly (24) should not deteriorate as the interpolation space approaches the Hilbert space $\mathcal{H}$). Also, the proof itself in [40] deals with the correct interpolation space $[X,\mathcal{H}]_\theta$ throughout (see equation (4.14) in [40]).

2.4.4 Completion of the proof of Theorem 6

Since for every Banach space $(X,\|\cdot\|)$ we have $\mathcal{S}_1(X)=1$, Theorem 6 is the special case $p=1$ of the following more refined theorem.

▶ **Theorem 10.** *Fix $p\in[1,2]$ and suppose that $(X,\|\cdot\|)$ is a Banach space satisfying $\mathsf{d}_X<\infty$ and $\mathcal{S}_p(X)<\infty$. For every $n\in\mathbb{N}$ and every symmetric stochastic matrix $\mathsf{A}=(a_{ij})\in M_n(\mathbb{R})$, we have*

$$\gamma\big(\mathsf{A},\|\cdot\|^2\big) \lesssim \begin{cases} \frac{\mathsf{d}_X^2}{1-\lambda_2(\mathsf{A})} & \text{if } \ \mathsf{d}_X^p\left(1-\lambda_2(\mathsf{A})\right)^{1-\frac{p}{2}} \leqslant e\mathcal{S}_p(X)^p, \\ \frac{\mathcal{S}_p(X)^2}{\left(1-\lambda_2(\mathsf{A})\right)^{\frac{2}{p}}}\left(\log\left(\frac{\mathsf{d}_X^p\left(1-\lambda_2(\mathsf{A})\right)^{1-\frac{p}{2}}}{\mathcal{S}_p(X)^p}\right)\right)^{\frac{2}{p}} & \text{if } \ \mathsf{d}_X^p\left(1-\lambda_2(\mathsf{A})\right)^{1-\frac{p}{2}} \geqslant e\mathcal{S}_p(X)^p. \end{cases} \tag{25}$$

Proof. Fix $\mathsf{d}>\mathsf{d}_X$ and $\theta\in(0,1]$. Consider a Hilbertian norm $|\cdot|:X\to[0,\infty)$ that satisfies (8) and denote by H the Hilbert space that is induced by $|\cdot|$. As we explained in Section 2.4.2, the complexification $X_{\mathbb{C}}$ satisfies $\mathcal{S}_p(X_{\mathbb{C}})\asymp\mathcal{S}_p(X)$. Also, by the parallelogram identity, the complex Hilbert space $H_{\mathbb{C}}$ satisfies $\mathcal{S}_2(H_{\mathbb{C}})=1$. Hence, by (19) with $Y=X_{\mathbb{C}}$, $Z=H_{\mathbb{C}}$, $p_1=p$ and $p_2=2$,

$$\mathcal{S}_{\frac{2p}{\theta p+2(1-\theta)}}\left([X_{\mathbb{C}},H_{\mathbb{C}}]_\theta\right) \leqslant \mathcal{S}_p(X_{\mathbb{C}})^{1-\theta} \lesssim \mathcal{S}_p(X)^{1-\theta}.$$

We may therefore apply Theorem 9 with $q = (2p)/(\theta p + 2(1-\theta))$ to deduce that

$$\gamma\left(\mathsf{A}, \|\cdot\|^2_{[X_{\mathbb{C}},H_{\mathbb{C}}]_\theta}\right) \lesssim \frac{\mathcal{S}_p(X)^{2(1-\theta)}}{\theta^{\theta+\frac{2(1-\theta)}{p}}\left(1-\lambda_2(\mathsf{A})\right)^{\theta+\frac{2(1-\theta)}{p}}} \asymp \frac{\mathcal{S}_p(X)^{2(1-\theta)}}{\theta^{\frac{2}{p}}\left(1-\lambda_2(\mathsf{A})\right)^{\theta+\frac{2(1-\theta)}{p}}}. \tag{26}$$

By the definition of $\gamma\left(\mathsf{A}, \|\cdot\|^2_{[X_{\mathbb{C}},H_{\mathbb{C}}]_\theta}\right)$, for every $(x_1,y_1),\ldots,(x_n,y_n) \in X \times X$ we have

$$\frac{1}{n^2}\sum_{i=1}^n\sum_{j=1}^n \|(x_i-x_j,y_i-y_j)\|^2_{[X_{\mathbb{C}},H_{\mathbb{C}}]_\theta} \leqslant \frac{\gamma\left(\mathsf{A}, \|\cdot\|^2_{[X_{\mathbb{C}},H_{\mathbb{C}}]_\theta}\right)}{n}\sum_{i=1}^n\sum_{j=1}^n a_{ij}\|(x_i-x_j,y_i-y_j)\|^2_{[X_{\mathbb{C}},H_{\mathbb{C}}]_\theta}.$$

By (23), this implies that

$$\frac{1}{n^2}\sum_{i=1}^n\sum_{j=1}^n \|(x_i-x_j,y_i-y_j)\|^2_{X_{\mathbb{C}}} \leqslant \frac{\mathsf{d}^{2\theta}\gamma\left(\mathsf{A}, \|\cdot\|^2_{[X_{\mathbb{C}},H_{\mathbb{C}}]_\theta}\right)}{n}\sum_{i=1}^n\sum_{j=1}^n a_{ij}\|(x_i-x_j,y_i-y_j)\|^2_{X_{\mathbb{C}}}.$$

Due to (26) and because X is isometric to a subspace of $X_{\mathbb{C}}$, this implies that

$$\forall\,\theta\in(0,1], \qquad \gamma\big(\mathsf{A}, \|\cdot\|^2\big) \lesssim \frac{\mathsf{d}_X^{2\theta}\mathcal{S}_p(X)^{2(1-\theta)}}{\theta^{\frac{2}{p}}\left(1-\lambda_2(\mathsf{A})\right)^{\theta+\frac{2(1-\theta)}{p}}}. \tag{27}$$

If $\mathsf{d}_X^p\left(1-\lambda_2(\mathsf{A})\right)^{1-p/2} \leqslant e\mathcal{S}_p(X)^p$, then by substituting $\theta = 1$ into (27) we obtain the first range of (25). When $\mathsf{d}_X^p\left(1-\lambda_2(\mathsf{A})\right)^{1-p/2} > e\mathcal{S}_p(X)^p$ the following value of θ minimizes the right hand side of (27) and belongs to the interval $(0,1]$.

$$\theta_{\mathrm{opt}} \stackrel{\mathrm{def}}{=} \frac{1}{\log\left(\frac{\mathsf{d}_X^p\left(1-\lambda_2(\mathsf{A})\right)^{1-\frac{p}{2}}}{\mathcal{S}_p(X)^p}\right)}.$$

A substitution of θ_{opt} into (27) yields the second range of (25). ◀

Acknowledgements. I thank Alex Andoni and Ilya Razenshteyn for encouragement to work on the question that is solved here. I also thank Gideon Schechtman for helpful discussions.

References

1. Noga Alon, Peter Frankl, and Vojtech Rödl. Geometrical realization of set systems and probabilistic communication complexity. In *26th Annual Symposium on Foundations of Computer Science*, pages 277–280, 1985. doi:10.1109/SFCS.1985.30.
2. Noga Alon and Yuval Roichman. Random Cayley graphs and expanders. *Random Structures Algorithms*, 5(2):271–284, 1994. doi:10.1002/rsa.3240050203.
3. A. Andoni, A. Nikolov, I. Razenshteyn, and E. Waingarten. Approximate near neighbors for general symmetric norms. To appear in STOC 2017. Preprint, available at https://arxiv.org/pdf/1611.06222, 2016.
4. Juan Arias-de Reyna and Luis Rodríguez-Piazza. Finite metric spaces needing high dimension for Lipschitz embeddings in Banach spaces. *Israel J. Math.*, 79(1):103–111, 1992. doi:10.1007/BF02764804.

5 K. Ball. Markov chains, Riesz transforms and Lipschitz maps. *Geom. Funct. Anal.*, 2(2):137–172, 1992. doi:10.1007/BF01896971.

6 Keith Ball. The Ribe programme. *Astérisque*, (352):Exp. No. 1047, viii, 147–159, 2013. Séminaire Bourbaki. Vol. 2011/2012. Exposés 1043–1058.

7 Keith Ball, Eric A. Carlen, and Elliott H. Lieb. Sharp uniform convexity and smoothness inequalities for trace norms. *Invent. Math.*, 115(3):463–482, 1994. doi:10.1007/BF01231769.

8 Yair Bartal, Nathan Linial, Manor Mendel, and Assaf Naor. On metric Ramsey-type phenomena. *Ann. of Math. (2)*, 162(2):643–709, 2005. doi:10.4007/annals.2005.162.643.

9 S. Bates, W. B. Johnson, J. Lindenstrauss, D. Preiss, and G. Schechtman. Affine approximation of Lipschitz functions and nonlinear quotients. *Geom. Funct. Anal.*, 9(6):1092–1127, 1999. doi:10.1007/s000390050108.

10 Jöran Bergh and Jörgen Löfström. *Interpolation spaces. An introduction*. Springer-Verlag, Berlin-New York, 1976. Grundlehren der Mathematischen Wissenschaften, No. 223.

11 J. Bourgain. On Lipschitz embedding of finite metric spaces in Hilbert space. *Israel J. Math.*, 52(1-2):46–52, 1985. doi:10.1007/BF02776078.

12 J. Bourgain. The metrical interpretation of superreflexivity in Banach spaces. *Israel J. Math.*, 56(2):222–230, 1986. doi:10.1007/BF02766125.

13 A.-P. Calderón. Intermediate spaces and interpolation, the complex method. *Studia Math.*, 24:113–190, 1964.

14 Michael Cwikel and Shlomo Reisner. Interpolation of uniformly convex Banach spaces. *Proc. Amer. Math. Soc.*, 84(4):555–559, 1982. doi:10.2307/2044034.

15 T. Figiel. On the moduli of convexity and smoothness. *Studia Math.*, 56(2):121–155, 1976.

16 Tadeusz Figiel and Gilles Pisier. Séries aléatoires dans les espaces uniformément convexes ou uniformément lisses. *C. R. Acad. Sci. Paris Sér. A*, 279:611–614, 1974.

17 M. Fréchet. Sur quelques points du calcul fonctionnel. *Rend. Circ. Mat. Palermo*, 22:1–74, 1906. doi:10.1007/BF03018603.

18 Shlomo Hoory, Nathan Linial, and Avi Wigderson. Expander graphs and their applications. *Bull. Amer. Math. Soc. (N.S.)*, 43(4):439–561 (electronic), 2006. doi:10.1090/S0273-0979-06-01126-8.

19 Fritz John. Extremum problems with inequalities as subsidiary conditions. In *Studies and Essays Presented to R. Courant on his 60th Birthday, January 8, 1948*, pages 187–204. Interscience Publishers, Inc., New York, N. Y., 1948.

20 William B. Johnson and Joram Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. In *Conference in modern analysis and probability (New Haven, Conn., 1982)*, volume 26 of *Contemp. Math.*, pages 189–206. Amer. Math. Soc., Providence, RI, 1984. doi:10.1090/conm/026/737400.

21 William B. Johnson, Joram Lindenstrauss, and Gideon Schechtman. On Lipschitz embedding of finite metric spaces in low-dimensional normed spaces. In *Geometrical aspects of functional analysis (1985/86)*, volume 1267 of *Lecture Notes in Math.*, pages 177–184. Springer, Berlin, 1987. doi:10.1007/BFb0078145.

22 Nigel J. Kalton. The nonlinear geometry of Banach spaces. *Rev. Mat. Complut.*, 21(1):7–60, 2008. doi:10.5209/rev_REMA.2008.v21.n1.16426.

23 Subhash Khot and Assaf Naor. Nonembeddability theorems via Fourier analysis. *Math. Ann.*, 334(4):821–852, 2006. doi:10.1007/s00208-005-0745-0.

24 R. Krauthgamer, J. R. Lee, M. Mendel, and A. Naor. Measured descent: a new embedding method for finite metrics. *Geom. Funct. Anal.*, 15(4):839–858, 2005. doi:10.1007/s00039-005-0527-6.

25 James R. Lee, Manor Mendel, and Assaf Naor. Metric structures in L_1: dimension, snowflakes, and average distortion. *European J. Combin.*, 26(8):1180–1190, 2005. doi:10.1016/j.ejc.2004.07.002.

26 J. R. Lee and A. Naor. Embedding the diamond graph in L_p and dimension reduction in L_1. *Geom. Funct. Anal.*, 14(4):745–747, 2004. doi:10.1007/s00039-004-0473-8.

27 Joram Lindenstrauss and Lior Tzafriri. *Classical Banach spaces. II*, volume 97 of *Ergebnisse der Mathematik und ihrer Grenzgebiete [Results in Mathematics and Related Areas]*. Springer-Verlag, Berlin-New York, 1979. Function spaces.

28 Nathan Linial, Eran London, and Yuri Rabinovich. The geometry of graphs and some of its algorithmic applications. *Combinatorica*, 15(2):215–245, 1995. doi:10.1007/BF01200757.

29 Jiří Matoušek. Note on bi-Lipschitz embeddings into normed spaces. *Comment. Math. Univ. Carolin.*, 33(1):51–55, 1992.

30 Jiří Matoušek. On the distortion required for embedding finite metric spaces into normed spaces. *Israel J. Math.*, 93:333–344, 1996. doi:10.1007/BF02761110.

31 Jiří Matoušek. On embedding expanders into l_p spaces. *Israel J. Math.*, 102:189–197, 1997. doi:10.1007/BF02773799.

32 Jiří 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.

33 Manor Mendel and Assaf Naor. Markov convexity and local rigidity of distorted metrics. *J. Eur. Math. Soc. (JEMS)*, 15(1):287–337, 2013. doi:10.4171/JEMS/362.

34 Manor Mendel and Assaf Naor. Spectral calculus and Lipschitz extension for barycentric metric spaces. *Anal. Geom. Metr. Spaces*, 1:163–199, 2013. doi:10.2478/agms-2013-0003.

35 Manor Mendel and Assaf Naor. Nonlinear spectral calculus and super-expanders. *Publ. Math. Inst. Hautes Études Sci.*, 119:1–95, 2014. doi:10.1007/s10240-013-0053-2.

36 Manor Mendel and Assaf Naor. Expanders with respect to Hadamard spaces and random graphs. *Duke Math. J.*, 164(8):1471–1548, 2015. doi:10.1215/00127094-3119525.

37 J. Milnor. On the Betti numbers of real varieties. *Proc. Amer. Math. Soc.*, 15:275–280, 1964.

38 Assaf Naor. An introduction to the Ribe program. *Jpn. J. Math.*, 7(2):167–233, 2012. doi:10.1007/s11537-012-1222-7.

39 Assaf Naor. On the Banach-space-valued Azuma inequality and small-set isoperimetry of Alon-Roichman graphs. *Combin. Probab. Comput.*, 21(4):623–634, 2012. doi:10.1017/S0963548311000757.

40 Assaf Naor. Comparison of metric spectral gaps. *Anal. Geom. Metr. Spaces*, 2:1–52, 2014. doi:10.2478/agms-2014-0001.

41 Assaf Naor, Yuval Peres, Oded Schramm, and Scott Sheffield. Markov chains in smooth Banach spaces and Gromov-hyperbolic metric spaces. *Duke Math. J.*, 134(1):165–197, 2006. doi:10.1215/S0012-7094-06-13415-4.

42 Assaf Naor and Lior Silberman. Poincaré inequalities, embeddings, and wild groups. *Compos. Math.*, 147(5):1546–1572, 2011. doi:10.1112/S0010437X11005343.

43 Ilan Newman and Yuri Rabinovich. Hard metrics from Cayley graphs of abelian groups. *Theory Comput.*, 5:125–134, 2009. doi:10.4086/toc.2009.v005a006.

44 Mikhail I. Ostrovskii. *Metric embeddings. Bilipschitz and coarse embeddings into Banach spaces*, volume 49 of *De Gruyter Studies in Mathematics*. De Gruyter, Berlin, 2013. doi:10.1515/9783110264012.

45 Gilles Pisier. Complex interpolation between Hilbert, Banach and operator spaces. *Mem. Amer. Math. Soc.*, 208(978):vi+78, 2010. doi:10.1090/S0065-9266-10-00601-0.

46 Yuri Rabinovich. On average distortion of embedding metrics into the line. *Discrete Comput. Geom.*, 39(4):720–733, 2008. doi:10.1007/s00454-007-9047-5.

47 M. Ribe. On uniformly homeomorphic normed spaces. *Ark. Mat.*, 14(2):237–244, 1976.

48 Marcel Riesz. Sur les maxima des formes bilinéaires et sur les fonctionnelles linéaires. *Acta Math.*, 49(3-4):465–497, 1927. doi:10.1007/BF02564121.

49 Elias M. Stein. Interpolation of linear operators. *Trans. Amer. Math. Soc.*, 83:482–492, 1956.

50 René Thom. Sur l'homologie des variétés algébriques réelles. In *Differential and Combinatorial Topology (A Symposium in Honor of Marston Morse)*, pages 255–265. Princeton Univ. Press, Princeton, N.J., 1965.

51 G. O. Thorin. Convexity theorems generalizing those of M. Riesz and Hadamard with some applications. *Comm. Sem. Math. Univ. Lund [Medd. Lunds Univ. Mat. Sem.]*, 9:1–58, 1948.

Dynamic Geodesic Convex Hulls in Dynamic Simple Polygons*

Eunjin Oh[1] and Hee-Kap Ahn[2]

1 **Department of Computer Science and Engineering, POSTECH, Pohang, Korea**
`jin9082@postech.ac.kr`
2 **Department of Computer Science and Engineering, POSTECH, Pohang, Korea**
`heekap@postech.ac.kr`

Abstract

We consider the geodesic convex hulls of points in a simple polygonal region in the presence of non-crossing line segments (barriers) that subdivide the region into simply connected faces. We present an algorithm together with data structures for maintaining the geodesic convex hull of points in each face in a sublinear update time under the fully-dynamic setting where both input points and barriers change by insertions and deletions. The algorithm processes a mixed update sequence of insertions and deletions of points and barriers. Each update takes $O(n^{2/3}\log^2 n)$ time with high probability, where n is the total number of the points and barriers at the moment. Our data structures support basic queries on the geodesic convex hull, each of which takes $O(\text{polylog}\, n)$ time. In addition, we present an algorithm together with data structures for geodesic triangle counting queries under the fully-dynamic setting. With high probability, each update takes $O(n^{2/3}\log n)$ time, and each query takes $O(n^{2/3}\log n)$ time.

1998 ACM Subject Classification I.3.5 Computational Geometry and Object Modeling

Keywords and phrases Dynamic geodesic convex hull, dynamic simple polygons

Digital Object Identifier 10.4230/LIPIcs.SoCG.2017.51

1 Introduction

A set X of points in a (weakly) simple polygon P is *geodesically convex* if the shortest path between any two points of X with respect to P is contained in X. The *geodesic convex hull* of a set of points contained in a simple polygon, which was introduced by Sklansky et al. [18], is defined as the intersection of all geodesic convex sets containing the set.

Geodesic convex hulls have been widely used for a variety of applications including collision detection [1], robotics [10], and motion planning [19]. These algorithms assume that the input points and the region where the points lie are static, that is, all input elements remain the same over the time. However, in many real-world geometric applications, particularly those that run in real-time, input data may change over time–input data elements may be inserted or deleted. Therefore it is required to handle these changes to maintain the geodesic convex hull for dynamically changing input data efficiently.

In this paper, we consider the problem of maintaining the geodesic convex hulls of dynamic points in a dynamic simple polygonal region. In the problem, points are inserted and deleted in a simple polygonal region as well as the region changes by insertions and deletions of

* This work was supported by the NRF grant 2011-0030044 (SRC-GAIA) funded by the government of Korea.

non-crossing line segments (barriers) under the restriction that the dynamic line segments always subdivide the region into simply connected faces.

We also study data structures that support a *geodesic triangle counting query* under the dynamic environment. The geodesic triangle defined by three points contained in a simple polygon P is the geodesic convex hull of them. Given a set S of input points in P, a geodesic triangle counting query asks for the number of input points contained in the interior of the geodesic triangle defined by three query points. No algorithm or data structure is known for the dynamic geodesic triangle counting problem in a dynamic region while an algorithm for the Euclidean version of the problem is known [14].

Related work for dynamic geodesic convex hulls. The *convex hull* of a set S of n points in the plane is defined to be the intersection of all convex sets containing S. The first nontrivial algorithm for maintaining the convex hull under point insertions and deletions in the plane was given by Overmars and Leeuwen [16]. Their data structure supports each update in $O(\log^2 n)$ worst-case time, where n is the number of points they have at the moment. The data structure can answer several basic queries including finding the extreme point in a direction and finding tangent points. Each query takes $O(\log n)$ worst-case time. Later, the update time was improved to $O(\log n)$ amortized time [3].

Surprisingly, little has been known for maintaining geodesic convex hulls of dynamic points in a dynamic region, except the one by Ishaque and Tóth [12]. They considered the problem in a *semi-dynamic* setting where point insertions and barrier deletions are not allowed (a barrier is an edge of a simple polygon in their work). With this restriction, they observe that the geodesic convex hull of points contained in a connected face of the barrier arrangement gradually decreases. Moreover, the total number of changes to their convex hull representations is $O(n+m)$, where n is the number of points in the initial state and m is the number of barriers in the final state. With this observations, they presented an algorithm to maintain the geodesic convex hulls in $O((n+m)\log^2(n+m)\log n)$ total time. Their data structure supports several basic queries in $O(\text{polylog}\{n,m\})$ time.

A natural question one may ask is whether the geodesic convex hulls can be maintained with a sublinear update time in a more general setting where both insertions and deletions of points and barriers are allowed.

Related work for geodesic triangle counting. The simplex counting problem is a fundamental query problem which has been studied extensively in the literature [4, 6, 14]. For a set of n static points in $\mathbb{R}^d$, Chan [4] gave a near-optimal algorithm which achieves $O(n^{1-1/d})$ query time with high probability after $O(n\log n)$-time preprocessing using linear space. The dynamic version of this problem where insertions and deletions of points are allowed is *decomposable* in the sense that a query over $D \cup D'$ can be answered in constant time from the answers from D and D' for any pair of disjoint data sets D and D' [14]. Thus, we can obtain a dynamic data structure from a static data structure of this problem using the framework of Bentley and Saxe [2], or Overmars and Leeuwen [17].

The geodesic triangle counting problem in a simple polygon is a generalization of the simplex counting problem. In the problem, we are given three query points in a simple polygon and want to count all input points contained in the interior of the geodesic triangle defined by the query points. We do not know any previous work achieving nontrivial results on this problem. We consider this problem in the fully-dynamic setting where both input points and barriers change by insertions and deletions. The problem is not decomposable because the simple polygon changes in the course of updates. Therefore, it is unclear how to

apply the framework of Bentley and Saxe [2], or Overmars and Leeuwen [17] even if we have a data structure for the static version of the problem.

Our results. We present an algorithm and data structures to maintain the geodesic convex hulls in the fully-dynamic setting where both input points and barriers change by insertions and deletions. Each update can be processed in $O(n^{2/3}\log^2 n)$ amortized time with high probability, where n is the total number of points and barriers at the moment. In addition, we show that any data structure for maintaining all edges of the geodesic convex hull requires $\Omega(n^{1/3})$ update time. By maintaining the geodesic convex hulls, we can answer various basic queries in $O(\text{polylog}\, n)$ time in the worst case: line stabbing, inclusion, and tangent queries.

In this algorithm, we use a subprocedure to answer a geodesic triangle counting query under insertions and deletions of points and barriers. Each update can be processed in $O(n^{2/3}\log n)$ amortized time with high probability, and each query can be answered in $O(n^{2/3}\log n)$ time with high probability, where n is the total number of points and barriers at the moment. We believe that this algorithm is of independent interest.

All these algorithms are randomized in terms of their running times because we use the partition tree given by Chan [4]. We can obtain algorithms with deterministic running times by using the partition tree of Chazelle et al. [8] instead of the one of Chan [4]. In this case, the update times increase slightly while the query times remain the same.

1.1 Outline

We first present a data structure for a geodesic triangle counting query for static points contained in a static simple polygon P. This data structure consists of three levels. The first level is the geodesic triangulation of P obtained from the algorithm by Chazelle et al. [7]. In the second level, each geodesic triangle of the geodesic triangulation is associated with a balanced binary search tree with respect to the x-coordinates of the input points contained in the geodesic triangle. In the third level, each node of the balanced binary search tree is associated with a data structure for an Euclidean triangle counting query.

Given three query points which define the geodesic triangle ⅄ contained in P, we first find the nodes of the balanced binary search trees whose associated point sets contain input points contained in ⅄. For each such node, we find an Euclidean triangle $\triangle$ such that $S' \cap \triangle$ coincides with $S' \cap \text{⅄}$ for a set S' of the input points associated with the node. Then we use a query algorithm of an Euclidean triangle counting query with $\triangle$. This procedure takes $O(n^{1/2}\log n)$ time with high probability, where n is the total number of points and barriers.

We use this data structure to answer a geodesic triangle counting query for dynamic points contained in a dynamic simple polygon P. The key idea is that we reconstruct the data structure periodically, instead of updating it for each change. To be specific, we reconstruct the data structure after $n^{1/3}$ updates are made, where n is the total number of the points and barriers at the moment.

Given the geodesic triangle ⅄ defined by three query points, ⅄ may not be the geodesic convex hull of its three corners with respect to the barriers we had when the data structure was constructed, because the barrier set has changed after the (re)construction of the data structure. To overcome this difficulty, we decompose ⅄ into smaller geodesic triangles with respect to the barriers we had at the last time the data structure was constructed such that for each smaller geodesic triangle $\widetilde{\text{⅄}}$, there are $O(n^{1/3}\log n)$ nodes of the balanced binary search trees whose associated point sets contain some input points contained in $\widetilde{\text{⅄}}$. Using these properties, we apply the query algorithm for the static data structure with each smaller

geodesic triangle in the decomposition of $\triangle$. By careful analysis, we show that the query time is $O(n^{2/3} \log n)$ with high probability.

Then we use this dynamic structure for geodesic triangle counting queries and show how to maintain the geodesic convex hulls in the fully-dynamic setting. We observed that the total number of distinct edges that appear on the geodesic convex hulls is less than the total changes to their convex hull representations. Based on this observation, we compute some edges that may appear on the geodesic convex hull in advance when we construct the data structure of a geodesic triangle counting query.

To handle each update, we replace part of the convex hull with a chain of edges precomputed and maintained in the data structure. To find such a chain, we apply a geodesic triangle counting query. Then we can achieve $O(n^{2/3} \log^2 n)$ update time with high probability.

Due to lack of space, some of the proofs and details are omitted.

2 Preliminaries

Arrangements of Barriers. Let R be a sufficiently large rectangle. We consider the arrangement of a set B of non-crossing line segments, called barriers, contained in R. The set B is initially empty and changes by insertions and deletions of barriers.

The *intersection graph* of B is defined as follows. The graph has $|B| + 1$ nodes, one for each barrier in B and an additional node for R, where $|X|$ denotes the number of elements in a set X. Two nodes of the graph are connected by an edge if and only if their corresponding barriers (or the boundary of R) are adjacent: one endpoint of a barrier lies on the other barrier (or the boundary of R). We restrict the intersection graph to be connected at any time, which gives a certain constraint on barriers that we consider. This restriction was also assumed in the papers [5, 9, 11, 12]. Moreover, we require a barrier to have a constant number of adjacent barriers, that is, each node in the intersection graph has a constant degree, which was also assumed in the papers [5, 9, 11].

Each connected region of $R \setminus \cup_{b \in B} b$ together with its boundary forms a weakly simple polygon under the restriction. A single point may appear on the boundary of a weakly simple polygon more than once, but we treat them as distinct points. We call each connected region of $R \setminus \cup_{b \in B} b$ together with its boundary a *face* of the arrangement of B.

In the following, we assume that the intersection graph of B is a tree. Thus the arrangement of B consists of exactly one face at all times. We use $R(B)$ to denote the unique face (weakly simple polygon) of the arrangement. A more general case that the intersection graph of B has cycles can also be handled analogously in the same time and space since there are a constant amount of changes to the arrangement for each update. See Figure 1(a). We can get rid of this assumption by modifying our algorithm slightly.

Update Sequences. Initially, both S and B are empty. We are given updates for points and barriers one by one. An update is either an insertion or a deletion of a point or a barrier. With these updates, S and B change accordingly. We are to process each update before we receive the next update. Let $U = \langle u_1, u_2, \ldots \rangle$ be a mixed sequence of updates. Let S_i and B_i denote S and B, respectively, after u_i is processed for an index i. Let CH_i denote the geodesic convex hull of S_i with respect to B_i. Let n_i be the total complexity of S_i and B_i.

For two sets X and Y, we use $X \oplus Y$ to denote the symmetric difference between X and Y, that is, $(X \setminus Y) \cup (Y \setminus X)$. By definition, it holds that $|S_i \oplus S_j| + |B_i \oplus B_j| \leq |i - j|$. We sometimes use an index i of U to denote the interval between the time when we receive u_i and the time when we receive u_{i+1}. Time i indicates an arbitrary time in this interval.

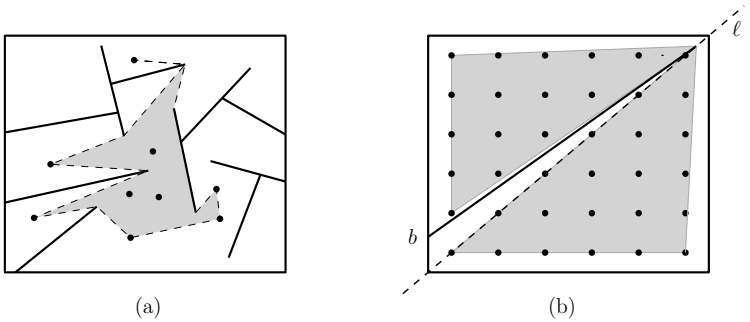


Figure 1 (a) The geodesic convex hull of points in a weakly simple polygon. (b) Every point on ℓ appears on the geodesic convex hull when barrier b is inserted.

Geodesic triangles and geodesic triangulations. For any two points x and y in $R(B)$, the *geodesic path* between x and y, denoted by $\pi_B(x, y)$ (or simply by $\pi(x, y)$), is the shortest path between x and y in $R(B)$. The boundary of the *geodesic triangle* defined by three points p_1, p_2 and p_3 in $R(B)$ consists of three geodesic paths $\pi_B(p_1, p_2)$, $\pi_B(p_2, p_3)$ and $\pi_B(p_3, p_1)$. The three vertices p_1, p_2 and p_3 are called the *corners* of the geodesic triangle. The interior of the geodesic triangle is bounded by three concave chains $\pi_B(p'_1, p'_2)$, $\pi_B(p'_2, p'_3)$, $\pi_B(p'_3, p'_1)$, where p'_i is the point such that $\pi_B(p_i, p'_i)$ is the maximal common path of $\pi_B(p_i, p_j)$ and $\pi_B(p_i, p_k)$ for three distinct indices i, j and k in $\{1, 2, 3\}$. We call the interior of a geodesic triangle the *deltoid* of it. We slightly abuse the term "deltoid" to denote a geodesic triangle excluding its boundary whose deltoid coincides with itself.

We sometimes mention a geodesic triangle (or deltoid) without specifying its corners. In this case, we mention it together with a barrier set B such that the boundary of the geodesic triangle consists of three geodesic paths with respect to B.

A *geodesic triangulation* of $R(B)$ is the decomposition of $R(B)$ into a number of interior-disjoint geodesic triangles with respect to B such that the union of the geodesic triangles coincides with $R(B)$. We say that a geodesic triangulation of $R(B)$ is *balanced* if any line segment that avoids the barriers in B intersects $O(\log |B|)$ geodesic triangles of the triangulation.

A lower bound for maintaining the geodesic convex hull. To maintain the geodesic convex hull at all times, we have to store the edges appearing on the boundary of the geodesic convex hull in the algorithm, which takes $\Omega(n^{4/3})$ time in total by the following lemma.

▶ **Lemma 1.** *For a mixed sequence of n updates of points and barriers, the number of distinct edges appearing on the boundary of the geodesic convex hull is $\Omega(n^{4/3})$.*

Proof. We prove this lemma using a lower bound example of a point-line incidence [15]. This lower bound example consists of a set $\mathcal{P}$ of N points and a set $\mathcal{L}$ of N lines such that there are $\Omega(N^{4/3})$ distinct point-line pairs (p, ℓ) with $p \in \ell$ for $p \in \mathcal{P}$ and $\ell \in \mathcal{L}$.

We construct a mixed sequence of n updates as follows. Let $N = n/3$. We first insert the points of $\mathcal{P}$ one by one. For a line $\ell \in \mathcal{L}$, there is a barrier with one endpoint on the boundary of R such that the insertion of the barrier makes all points of $\mathcal{P}$ lying on ℓ appear on the boundary of the geodesic convex hull. See Figure 1(b). We insert such a barrier, and then we remove it. We repeat this for every line in $\mathcal{L}$. Then we have $n/3$ point insertions, $n/3$ barrier insertions, and $n/3$ barrier deletions.

For the insertion of a barrier, the number of the new edges appearing on the geodesic convex hull is at least the number of the points of $\mathcal{P}$ lying on the line $\ell \in \mathcal{L}$ corresponding

to the barrier. Moreover, the new edges are contained in ℓ. Therefore, all such edges are distinct for every line in $\mathcal{L}$. This implies that the number of distinct edges of the geodesic convex hull is $\Omega(n^{4/3})$. ◀

Note that this lower bound example contains collinear points. We can perturb the points slightly in a certain way such that no three distinct points are collinear and the bound still holds. We omit details due to lack of space.

3 Dynamic Geodesic Triangle Range Queries

We are given three points c_1, c_2 and c_3 in $R(B_i)$ as a query for a geodesic triangle counting problem. We show how to compute the number of points of S_i lying in the deltoid of the geodesic triangle of c_1, c_2 and c_3 at time i.

3.1 Two Data Structures

We maintain two data structures: a geodesic-path data structure and an α-geodesic-triangulated subdivision. The first one is given by Goodrich and Tamassia [11]. With their structure, we can compute the geodesic path between any two query points in $R(B_i)$ represented as a balanced binary search tree in $O(\log^2 n_i)$ time.

The second one is our main data structure. At time i, an *α-geodesic-triangulated subdivision* is a balanced geodesic triangulation of $R(B_{i-\alpha})$ for some $0 \leq \alpha \leq i$ such that every geodesic triangle C in the triangulation is associated with a hierarchy of the partition trees given by Chan [4] on the point set $C \cap S_{i-\alpha}$, which will be described later.

The first level: balanced geodesic triangulation. We use the balanced geodesic triangulation $\mathcal{T}_i$ of $R(B_{i-\alpha})$ given by Chazelle at al. [7]. We call α the *inconsistency* of $\mathcal{T}_i$. The choice of α will be described in Section 3.2. Let $\widetilde{B}_i$ and $\widetilde{S}_i$ be $B_{i-\alpha}$ and $S_{i-\alpha}$, respectively. We call a deltoid in the geodesic triangulation a *cell* of $\mathcal{T}_i$. To make the description easier, we assume that no point of $\widetilde{S}_i$ lies on the boundary of any cell of $\mathcal{T}_i$. If it is not the case, we assume that a point of $\mathcal{T}_i$ lying on the common boundary of two cells of $\mathcal{T}_i$ belongs to exactly one of them to avoid overcounting.

The geodesic triangulation is for a *static* simple polygonal region. We do not make any change to this structure for updates until the inconsistency of the structure exceeds a certain level. Then we reconstruct it from scratch so that the structure has no inconsistency for the current set of barriers. Thus, $\widetilde{B}_i$ and $\widetilde{S}_i$ are B and S we had at the last time the data structures were constructed, respectively. Details will be described in Section 3.2.

▶ **Lemma 2.** *A deltoid with respect to B_i intersects $O((\alpha+1)\log|\widetilde{B}_i|)$ cells in the α-geodesic-triangulated subdivision. Moreover, they can be found in $O((\alpha+1)(\log^2|\widetilde{B}_i| + \log\alpha))$ time.*

The second and third levels: a hierarchy of partition trees. Imagine that we construct Chan's partition tree on $\widetilde{S}_i \cap C$ for every cell C of $\mathcal{T}_i$ that answers a Euclidean triangle counting query [4]. The partition tree requires $O(N)$ preprocessing time, $O(N)$ space, and $O(\sqrt{N})$ query time with high probability, where N is the number of input points.

▶ **Lemma 3.** *Let C be a cell of $\mathcal{T}_i$ and $\widetilde{\Delta}$ be a deltoid with respect to $\widetilde{B}_i$. We can compute the number of points in $(C \cap \widetilde{S}_i) \cap \widetilde{\Delta}$ in $O(n_C^{1/2})$ time with high probability, where $n_C = |C \cap \widetilde{S}_i|$.*

In our problem, a query is given as a deltoid with respect to B_i, not $\widetilde{B}_i$. Thus, we cannot apply the algorithm in Lemma 3 to the query deltoid directly. Instead, for a query deltoid $\triangle$, we construct a set $\widetilde{\mathcal{Q}}(C, \triangle)$ of pairwise disjoint deltoids with respect to $\widetilde{B}_i$ for each cell C of $\mathcal{T}_i$ such that the closures of the deltoids in $\widetilde{\mathcal{Q}}(C, \triangle)$ contain $C \cap \triangle$ in their union. We apply the algorithm in Lemma 3 to each deltoid in $\widetilde{\mathcal{Q}}(C, \triangle)$ and get the answer. The running time of this approach is $O(m_C n_C^{1/2})$, where $m_C = |\widetilde{\mathcal{Q}}(C, \triangle)|$ and $n_C = |C \cap \widetilde{S}_i|$.

To reduce the running time, instead of constructing Chan's partition tree on the set $C \cap \widetilde{S}_i$ for each cell C of $\mathcal{T}_i$, we construct a hierarchy of Chan's partition trees on $C \cap \widetilde{S}_i$ using a range tree. We construct a one-dimensional range tree (balanced binary search tree) on $C \cap \widetilde{S}_i$ with respect to the x-coordinates of the points. Each node v of the tree corresponds to a vertical slab $H(v)$. The root corresponds to the (degenerate) vertical slab $\mathbb{R}^2$. The left child and the right child of a node v correspond to the left vertical slab and the right vertical slab obtained from the partition of $H(v)$ with respect to the median x-coordinate of $\widetilde{S}_i$ contained in $C \cap H(v)$, respectively. For each node v of the range tree, we construct Chan's partition tree on the set $(C \cap H(v)) \cap \widetilde{S}_i$ (excluding the points of $\widetilde{S}_i$ lying on the right vertical side of $H(v)$) as the associated data structure of v.

In the query algorithm, we construct a set $\widetilde{\mathcal{Q}}(C, \triangle)$ of pairwise disjoint deltoids with respect to $\widetilde{B}_i$ satisfying Property ($\star$) such that the closures of the deltoids contains $C \cap \triangle$ in their union. This procedure is described in Section 3.3.1. Then we apply Lemma 4 as a subprocedure.

▶ **Property ($\star$).** *Any vertical line intersects $O(1)$ deltoids in $\widetilde{\mathcal{Q}}(C, \triangle)$ for every cell C in $\mathcal{T}_i$.*

▶ **Lemma 4.** *Given a deltoid $\triangle$ with respect to B_i, we can compute the number of points of $\widetilde{S}_i$ contained in $C \cap \triangle$ for a cell C of $\mathcal{T}_i$ in $O((m_C n_C)^{1/2} \log n_C)$ time with high probability, where $m_C = |\widetilde{\mathcal{Q}}(C, \triangle)|$ and $n_C = |C \cap \widetilde{S}_i|$.*

▶ **Lemma 5.** *We can construct the hierarchy of Chan's partition trees for every cell C of $\mathcal{T}_i$ in $O(|\widetilde{S}_i| \log(|\widetilde{B}_i| + |\widetilde{S}_i|))$ time in total with high probability.*

Maintaining the point-location data structure. We store one more piece of information in the second level of the α-geodesic-triangulated subdivision. For each cell C of $\mathcal{T}_i$, we maintain the dynamic point-location data structure of Chan and Nekrich [5] on the boundary of C and the barriers of $B_i \setminus \widetilde{B}_i$ intersecting C. Their data structure supports two types of queries: a point-location query and a vertical ray-shooting query. Each query takes $O(\log N (\log \log N)^2)$ time and each update takes $O(\log N \log \log N)$ time, where N is the complexity of the boundary of C and the barriers of $B_i \setminus \widetilde{B}_i$ intersecting C.

3.2 A Procedure for Updates

We do not make any change to the geodesic-triangulated subdivision for updates until the inconsistency α becomes larger than $n_i^{1/3}$. Then we reconstruct the subdivision from scratch so that the structure has no inconsistency, that is, we set $\widetilde{S}_i = S_i$ and $\widetilde{B}_i = B_i$ and construct $\mathcal{T}_i$ accordingly when the inconsistency becomes larger than $n_i^{1/3}$. We can reconstruct the geodesic-triangulated subdivision in $O(n_i^{2/3} \log n_i)$ amortized time.

For the other two data structures, we update them for each insertion and deletion of barriers: the point-location data structure for every cell C of $\mathcal{T}_i$ intersecting the barrier in $O(n_i^{1/3} \log^3 n_i (\log \log n_i))$ time, and the geodesic-path data structure in $O(\log^2 n_i)$ time.

▶ **Lemma 6.** *At any time i, the amortized time complexity for the reconstruction of the geodesic-triangulated subdivision is $O(n_i^{2/3} \log n_i)$ with high probability.*

3.3 A Procedure for Geodesic Triangle Counting Queries

Assume that we have an update u_i to process and we have the α-geodesic-triangulated subdivision $\mathcal{T}_i$ with $\alpha \leq n_i^{1/3}$. Note that $|\widetilde{B}_i| = O(n_i)$ and $|\widetilde{S}_i| = O(n_i)$. Given any three query points c_1, c_2 and c_3 in $R(B_i)$, we present an algorithm that returns the number of points of S_i contained in the deltoid $\triangle$ of the geodesic triangle defined by c_1, c_2, c_3 with respect to B_i in $O(n_i^{2/3} \log n_i)$ time. Recall that the geodesic triangulated subdivision is constructed on $R(\widetilde{B}_i)$ while $\triangle$ is a deltoid with respect to B_i. Thus we cannot apply the algorithm in Lemma 3 directly to $\triangle$. Instead, we decompose $\triangle$ into a number of deltoids with respect to $\widetilde{B}_i$ and apply the algorithm in Lemma 3 to each such deltoid.

The query algorithm consists of three steps. In the first step, we find all cells of $\mathcal{T}_i$ intersecting $\triangle$ in $O((\alpha+1)\log^2 n_i)$ time using Lemma 2. Let $\mathcal{C}(\triangle)$ be the set of such cells. In the second step described in Section 3.3.1, for each cell $C \in \mathcal{C}(\triangle)$, we construct a set $\widetilde{\mathcal{Q}}(C, \triangle)$ of pairwise disjoint deltoids satisfying Property $(\star)$ with respect to $\widetilde{B}_i$ whose closures contain $C \cap \triangle$ in their union. The total complexity of this set for every cell in $\mathcal{C}(\triangle)$ is $O(1+\alpha)$. Then, in the last step described in Section 3.3.2, we compute the number X_1 of the points of $\widetilde{S}_i$ contained in $\triangle$ using the set $\widetilde{\mathcal{Q}}(C, \triangle)$ for every cell $C \in \mathcal{C}(\triangle)$. In addition, we compute the numbers of points in $S_i \setminus \widetilde{S}_i$ and $\widetilde{S}_i \setminus S_i$ contained in $\triangle$, and denote them by X_2 and X_3, respectively. Then $X_1 + X_2 - X_3$ is the number of points of S_i contained in $\triangle$.

3.3.1 Constructing a Set of Deltoids with respect to $\widetilde{B}_i$

Let C be a cell in $\mathcal{C}(\triangle)$. In brief, we construct a set $\widetilde{\mathcal{Q}}(C, \triangle)$ of pairwise disjoint deltoids with respect to $\widetilde{B}_i$ as follows. We compute the vertical decomposition of C with respect to the barriers in $B_i \oplus \widetilde{B}_i$. Then we find all cells of the vertical decomposition intersecting $\triangle$. For each such cell, we compute the intersection of the cell with $\triangle$, which is the geodesic convex hull of at most six points with respect to $\widetilde{B}_i$. Then for each intersection, we obtain at most four deltoids from a geodesic triangulation of it. All such deltoids form $\widetilde{\mathcal{Q}}(C, \triangle)$.

Computing the vertical decomposition. Let $\mathcal{V}(C)$ be the set of the endpoints of the barriers of $B_i \oplus \widetilde{B}_i$ contained in the closure of C. We consider two vertical extensions from every endpoint in $\mathcal{V}(C)$ going to opposite directions until they hit a barrier in $B_i \oplus \widetilde{B}_i$ or the boundary of C. Note that no barrier in $\widetilde{B}_i$ intersects the interior of C. Thus, we can compute all extensions in $O(|\mathcal{V}(C)| \log n_i (\log\log n_i)^2)$ time in total using the point-location data structure associated with C that supports a vertical ray-shooting query.

The extensions together with the barriers in $B_i \setminus \widetilde{B}_i$ decompose C into $O(1+\alpha)$ cells. We call this decomposition the *vertical decomposition of* C. We dynamically maintain the arrangement of $B_i \oplus \widetilde{B}_i$ using the point-location data structure associated with C. Thus by computing all extensions, we can obtain the vertical decomposition of C.

Note that each cell (connected region) of the vertical decomposition contains at most four convex vertices on its boundary. Moreover, the closure of each cell is the geodesic convex hull of its four convex vertices with respect to both $\widetilde{B}_i$ and B_i.

Traversing the vertical decomposition. Then we find all cells of the vertical decomposition of C intersecting $\triangle$ by traversing the vertical decomposition of C. In the case that C is contained in $\triangle$, every cell in the vertical decomposition intersects $\triangle$. Thus we consider the case that the boundary of $\triangle$ intersects C.

We observe that we can compute the intersection of the boundary of C with the boundary of $\triangle$ while we compute the set $\mathcal{C}(\triangle)$. Then, starting from the intersection points, we traverse

the vertical decomposition of C along the boundary of $\triangle$. In this way, we can visit every cell of the vertical decomposition intersecting $\triangle$. Moreover, we visit only the cells intersecting $\triangle$.

Every cell of the decomposition intersecting $\triangle$ contains an endpoint of at most one barrier of $B_i \setminus \widetilde{B}_i$ on its boundary. Thus, during the traversal, we can find a neighboring cell of the vertical decomposition intersecting the boundary of $\triangle$ in $O(\log n_i)$ time. The running time of this procedure is bounded by the number of cells of the decomposition intersecting $\triangle$ times $O(\log n_i)$.

Computing the set of deltoids. For each cell of the vertical decomposition of C intersecting $\triangle$, consider the intersection of $\triangle$ with the cell. It is the geodesic convex hull of at most six points with respect to both $\widetilde{B}_i$ and B_i. By computing a geodesic triangulation of it with respect to $\widetilde{B}_i$, we can obtain at most four deltoids with respect to $\widetilde{B}_i$ from the intersection in $O(\log^2 n_i)$ time. Here, we do not explicitly compute the deltoids. Instead, we compute the corners of each deltoid using the geodesic-path data structure we maintain. Then we can compute the convex vertices of the deltoid of the geodesic triangle in $O(\log^2 n_i)$ time.

Let $\widetilde{\mathcal{Q}}(C, \triangle)$ be the set of all such deltoids obtained from all cells in the vertical decomposition of C intersecting $\triangle$. Note that the number of all cells of the decomposition of C intersecting $\triangle$ is asymptotically the same as the size of $\widetilde{\mathcal{Q}}(C, \triangle)$. The running time of this procedure is bounded by the size of $\widetilde{\mathcal{Q}}(C, \triangle)$ times $O(\log^2 n_i)$.

Analysis. For analysis, we prove the followings.

- The total number of deltoids in $\widetilde{\mathcal{Q}}(C, \triangle)$ for all cells $C \in \mathcal{C}(\triangle)$ is $O(\alpha + 1)$.
- The set $\widetilde{\mathcal{Q}}(C, \triangle)$ satisfies Property $(\star)$.

The first claim implies that the running time for computing $\widetilde{\mathcal{Q}}(C, \triangle)$ for all cells $C \in \mathcal{C}(\triangle)$ is $O((\alpha + 1) \log n_i (\log \log n_i)^2)$, which is dominated by the time for constructing the vertical decomposition for every cell in $\mathcal{C}(\triangle)$. The first and second claims give properties for $\widetilde{\mathcal{Q}}(C, \triangle)$ we want to achieve.

3.3.2 Computing the Number of Points in a Deltoid

We compute the number of points in $(C \cap \widetilde{\triangle}) \cap \widetilde{S}_i$ for every cell C in $\mathcal{C}(\triangle)$ and every deltoid $\widetilde{\triangle}$ of $\widetilde{\mathcal{Q}}(C, \triangle)$. Due to properties of $\widetilde{\mathcal{Q}}(C, \triangle)$, we can compute it by applying Lemma 4 in $O(n_i^{2/3} \log n_i)$ time. Then we handle the points in $S_i \oplus \widetilde{S}_i$ by deciding whether each of them is contained in $\triangle$ in $O(\log^2 n_i)$ time. We omit details due to lack of space.

▶ **Theorem 7.** *A geodesic triangle counting query can be answered in $O(n^{2/3} \log n)$ time with high probability under insertions and deletions of points and barriers, where n is the total number of the points and barriers at the moment. We process each update in $O(n^{2/3} \log n)$ amortized time with high probability using $O(n \log n)$ space.*

4 Maintaining the Geodesic Convex Hull

In this section, we present an algorithm together with three data structures including the data structures described in Section 3 to maintain the geodesic convex hull under insertions and deletions of points and barriers.

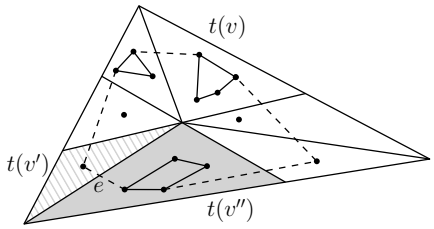


Figure 2 The nodes v' and v'' are children of the node v. For v, we compute the bridges (dashed segments). The edge e is the bridge of the convex hull of $P(v')$ and the convex hull of $P(v'')$.

4.1 A Triangle-Range Hull Tree

In addition to the data structures for a geodesic triangle query, we maintain a data structure, which we call a *triangle-range hull tree*. The triangle-range hull tree is constructed on the set $P_i = (S_i \cap \widetilde{S}_i) \cup V(B_i \cap \widetilde{B}_i)$, where $V(B)$ denotes the vertices of $R(B)$ for a set B of barriers. Note that $|P_i| = O(n_i)$. Given a set of points, this data structure allows us to compute the Euclidean convex hull of the points of P_i lying inside a query Euclidean triangle. In this section, we present an algorithm to construct the triangle-range hull tree and an algorithm to compute the Euclidean convex hull of the points contained in a query Euclidean triangle.

A partition tree of Chan. The triangle-range hull tree is a partition tree constructed on P_i containing additional information. There are several variants of a partition tree with different partitioning schemes. Among them, we use the partition tree given by Chan [4]. This is because in their scheme, the triangle $t(v)$ corresponding to a node v is subdivided into a constant number of *interior-disjoint* triangles each of which corresponds to a child of v. Each node v is associated with a point set $P(v) = t(v) \cap P_i$. Moreover, for any Euclidean triangle $\triangle$, the number of nodes v in T such that $t(v)$ intersects the boundary of $\triangle$ is $O(\sqrt{|P_i|})$.

Construction of the triangle-range hull tree. In our problem, we use the partition tree constructed on P_i to compute the Euclidean convex hull of points contained in a query Euclidean triangle. Recall that P_i consists of points from $S_i \cap \widetilde{S}_i$ and points from $V(B_i \cap \widetilde{B}_i)$. For a vertex p of the Euclidean convex hull of $P' \subseteq P_i$, we call the vertex of the Euclidean convex hull that comes first from p in clockwise order along the convex hull among all vertices from $S_i \cap \widetilde{S}_i$ (or, $V(B_i) \cap V(\widetilde{B}_i)$) the *$S$-neighbor* (or, *$B$-neighbor*) of p.

For every node v of the partition tree, we compute a part of the convex hull of $P(v)$ such that the partition tree supports the following operations for the convex hull of $P(v)$:

- **(O1)** Given two edges e_1 and e_2 of the convex hull, we can compute the number of edges of the convex hull lying from e_1 to e_2 in clockwise order in $O(\log n_i)$ time.
- **(O2)** For an integer j and an edge e of the convex hull, we can access the jth edge of the convex hull from e in clockwise order in $O(\log n_i)$ time.
- **(O3)** Given a vertex p of the convex hull, we can find the S-neighbor and B-neighbor of p in clockwise order along the convex hull in $O(\log n_i)$ time.

Let v be a node of T. Assume that for every descendant v' of v, we have already computed a part of the convex hull of $P(v')$ such that the partition tree supports all three operations. We show how to compute a part of the convex hull of $P(v)$ as follows. We compute a constant number of edges of the convex hull of $P(v)$ that do not appear on the convex hull of $P(v')$ for any child v' of v, which we call *bridges* for v. For illustration, see Figure 2. By property of the partition tree, $P(v)$ is the union of $P(v')$ for all children v' of v, and $t(v)$ contains

the union of the triangles $t(v')$ for all children v' of v. A bridge for v is an outer common tangent of two convex hulls of $P(v_1)$ and of $P(v_2)$ for two children v_1 and v_2 of v.

Kirkpatrick and Snoeyink [13] presented an algorithm to compute the outer common tangents of two given convex polygons with k vertices in $O(\log k)$ time once the vertices of each convex polygon are stored in an array. In our case, we need $O(\log k)$ time for accessing the jth edge of the convex hull of $P(v_1)$ (or $P(v_2)$) from a given edge for an integer j. Since a node of T has a constant number of children, we can compute all bridges for v in $O(\log^2 n_i)$ time. We maintain the bridges for v in clockwise order along the convex hull of $P(v)$. In addition, we compute information for the bridges to support operations O1, O2 and O3 for v. (We omit details due to lack of space.)

▶ **Lemma 8.** *The triangle-range hull tree supports operations O1, O2 and O3 for every node.*

To construct the triangle-range hull tree, we spend $O(\log^2 n_i)$ time for each node v, thus the running time for the construction is $O(n_i \log^2 n_i)$. Moreover, the number of bridges we compute additionally is asymptotically bounded by the number of edges of the partition tree. Thus, the size of the partition tree remains the same and we have the following lemma.

▶ **Lemma 9.** *The triangle-range hull tree can be constructed on P_i in $O(n_i \log^2 n_i)$ time with high probability. The size of the triangle-range hull tree is $O(n_i)$.*

Computation of the convex hull of points in a query Euclidean triangle. Let $\triangle$ be a Euclidean triangle. We compute the Euclidean convex hull of $P_i \cap \triangle$. To be specific, we compute a tree of size $O(\sqrt{n_i})$ supporting all three operations for the convex hull of $P_i \cap \triangle$. Each node of the tree is associated with a sequence of edges of the convex hull of $P_i \cap \triangle$.

The algorithm is similar to the one for triangle range searching. We start from the root of the triangle-range hull tree T. Let v be a node we just reached. Then we compute the convex hull of $P(u) \cap \triangle$ recursively for every child u of v. There are three possible cases: (1) $t(u)$ intersects the boundary of $\triangle$, (2) $t(u)$ is contained in the interior of $\triangle$, or (3) $t(u)$ does not intersect $\triangle$. For the first case, we search further the subtrees rooted at u recursively. For the second case, we already have the convex hull of $P(u) \cap \triangle$. This is because the subtree of T rooted at u supports the three operations for $P(u) = P(u) \cap \triangle$. For the third case, we do not search further the subtree rooted at u.

After computing the convex hull of $P(u) \cap \triangle$ for every child u of v, we compute the edges of the convex hull of $P(v) \cap \triangle$ which do not appear on the boundary of the convex hull of $P(u) \cap \triangle$ for any child u of v, which are bridges for v on the convex hull of $P(v) \cap \triangle$. Note that such a bridge is an outer common tangent of the convex hull of $P(u_1) \cap \triangle$ and the convex hull of $P(u_2) \cap \triangle$ for two children u_1 and u_2 of v. Since v has a constant number of children, there are a constant number of bridges for v. We compute them in $O(\log^2 n)$ time using the algorithm by Kirkpatrick and Snoeyink [13], and sort them in clockwise order along the convex hull of $P(v) \cap \triangle$. We also compute additional information to support operations O1, O2 and O3 for v. Finally, we reach leaf nodes v of T. Since $P(v)$ has a constant size, we compute the convex hull of all points of $P(v)$ lying inside $\triangle$ explicitly in constant time.

As a result of handling the query, we return the nodes of T we visited and the sequence of bridges for each such node. That is, the output of the query algorithm is a tree consisting of the nodes of T we visited each of which stores the sequence of bridges. Since we visited $O(\sqrt{n_i})$ nodes, the size of the output is $O(\sqrt{n_i})$. One difference of the partition tree and the output tree is that a leaf node of the output tree does not necessarily correspond to a constant number of points. It happens when $t(u)$ is contained in $\triangle$ for some node u of T. In

this case, the leaf node of the output tree points to its corresponding node of T. Thus we can apply all three operations on the leaf node of the output tree.

With a simple analysis, we can show that the running time of this query algorithm is $O(N \log^2 n_i)$, where N is the number of nodes v in T such that $t(v)$ intersects the boundary of $\triangle$. Therefore, the running time for our query algorithm is $O(\sqrt{n_i} \log^2 n_i)$.

▶ **Lemma 10.** *We can compute a tree of size $O(\sqrt{n_i})$ in $O(\sqrt{n_i} \log^2 n_i)$ time with high probability that supports the three operations for the convex hull of points of P contained in a query Euclidean triangle.*

Whenever we reconstruct the data structure for a geodesic triangle counting query, we also reconstruct the triangle-range hull tree. Additionally, we handle the deletion of a point p from $S \cup V(B)$ by removing it from the triangle-range hull tree in $O(\log^3 n_i)$ time.

4.2 Representation of the Geodesic Convex Hull

Ishaque and Tóth [12] showed that n updates may induce $\Omega(n^2)$ combinatorial changes in the geodesic convex hull. However, we observe that the total number of distinct edges of the geodesic convex hull under n updates is less than n^2. Based on this observation, we compute a number of chains in advance, which are geodesic paths stored in the geodesic-path data structure and the boundaries of convex hulls stored in the triangle-range hull tree. Then at any time i, we represent the geodesic convex hull CH_i of S_i with respect to B_i as a sequence consisting of subchains along the boundary of CH_i.

We maintain this sequence using a concatenable queue implemented by AVL-trees, and call it the *representation tree*. A concatenable queue is used also in [16] to represent the Euclidean convex hull. This data structure allows us to split a sequence, merge two sequences, insert or delete an element in $O(\log n_i)$ time. Each element in the representation tree corresponds to a subchain of the boundary of CH_i of one of the three types:

- **(T1)** A single edge of CH_i connecting two points in S_i
- **(T2)** A geodesic path with respect to B_i connecting two points in S_i
- **(T3)** A subchain of the convex hull of $P \cap \triangle$ for some Euclidean triangle $\triangle$

For a T1 element e, we simply store its corresponding edge to e. For an element of other types, instead of storing the subchain directly, we store information that supports the three operations for the element. Moreover, we store additional information to each node v (element) to support operations O1, O2 and O3 for CH_i.

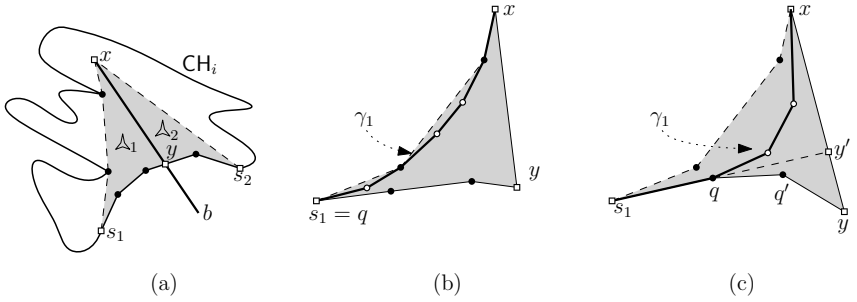
▶ **Lemma 11.** *The representation tree supports operations O1, O2 and O3 for CH_i.*

4.3 Procedures for Various Types of Queries

Lemma 11 allows us to answer various basic queries. Here, we show how to answer queries of three types, which were considered by Ishaque and Tóth [12]: a *line stabbing query*, an *inclusion query* and a *tangent query*.

For a line stabbing query with a line segment ℓ contained in $R(B_i)$, we want to find the intersection of ℓ with CH_i. For an inclusion query with a point p in $R(B_i)$, we want to determine whether or not p is contained in CH_i. For a tangent query with a point p lying outside of CH_i, we want to find the vertices v of CH_i where p is tangent to CH_i. Each query can be answered in polylogarithmic time.

▶ **Lemma 12.** *We can answer a line stabbing and inclusion query in $O(\log^2 n_i)$ time in the worst case. We can answer a tangent query in $O(\log^3 n_i)$ time in the worst case.*



(a) (b) (c)

Figure 3 (a) When b is inserted, we replace $\pi_{B_{i-1}}(s_1, s_2)$ with two convex chains in $\triangle_1$ and $\triangle_2$. (b), (c) We compute q such that the maximal common path of $\pi(s_1, y)$ and γ_1 is $\pi(s_1, q)$.

4.4 Procedures for Updates

We update the triangle-range hull tree when a point or a barrier is deleted. But we do not update it for insertions of points or barriers. Instead, we reconstruct it whenever we reconstruct the data structure for a geodesic triangle counting query. We also compute the geodesic convex hull of S with respect to B when we reconstruct them. After the reconstruction, the representation tree consists of T1 elements only.

In this paper, we present algorithms for processing an insertion of a barrier b. We omit details of the procedures for the other cases. At time i, we have the geodesic convex hull CH_{i-1}. We first check whether b intersects the interior of CH_{i-1} by applying the stabbing query with the line segment b. If b does not intersect the interior of CH_{i-1}, it holds that $\mathsf{CH}_i = \mathsf{CH}_{i-1}$. Otherwise, the intersection of b with the interior of CH_{i-1} consists of at most two connected components. (This happens when ℓ contains an edge of CH_{i-1}.) If it consists of two connected components, we consider them as two distinct barriers. So, in the following, we assume that the intersection of b with the interior of CH_{i-1} is connected.

Now we consider the intersection of b with the boundary of CH_{i-1}. The intersection consists of at most two points. We consider the case that the intersection is a single point y. The other case can be handled analogously.

Let s_1 and s_2 be the S-neighbors of y along the boundary of CH_{i-1} in clockwise and counterclockwise orders, and let x be the endpoint of b lying inside CH_{i-1}. See Figure 3(a). We can compute s_1 and s_2 in $O(\log n_i)$ time using operation O3. As the barrier b is inserted, some points of S_i lying in the interior of CH_{i-1} appear on the boundary of CH_i. Such points lie in the deltoid $\triangle$ of the geodesic triangle with three corners s_1, s_2 and x with respect to B_i. Moreover, we have the following observation.

▶ **Observation 13.** *CH_i is the geodesic convex hull of $(\mathsf{CH}_{i-1} \setminus \triangle) \cup (S_i \cap \triangle)$ w.r.t. B_i.*

Let $\triangle_1$ and $\triangle_2$ be the two deltoids such that their union including their common boundary xy is $\triangle$. Without loss of generality, we assume that $s_1 \in \triangle_1$ and $s_2 \in \triangle_2$. We use γ_t to denote the part of the boundary of the geodesic convex hull of $(S_i \cap \triangle_t) \cup \{s_t, x\}$ with respect to B_i that connects s_t and x (and is not $\pi(s_t, x)$) for $t = 1, 2$. See Figure 3(b). To obtain CH_i, we replace $\pi(s_1, s_2)$ with γ_1 and γ_2 in the representation tree of CH_{i-1}. We show how to compute γ_1 only. The polygonal chain γ_2 can be computed analogously. Then we show how to replace $\pi(s_1, s_2)$ with them in the representation tree.

The procedure for computing γ_1 consists of three steps. First, we find a smaller deltoid $\triangle' \subseteq \triangle_1$ with properties similar to ones of $\triangle_1$. Second, we find an Euclidean triangle $\triangle$ such that the part of the boundary of the Euclidean convex hull of $(S_i \cup V(B_i)) \cap \triangle$ from x

to q in clockwise order is exactly $\gamma_1 \setminus \pi(s_1, q)$, where q is the corner of $\triangle$ that does not lie on xy. Third, we compute the convex hull of $(S_i \cup V(B_i)) \cap \triangle$ using the triangle-range hull tree and update CH_i using this information.

Finding a smaller deltoid. Here, instead of considering $\mathbin{\text{⏢}}_1$, we choose a deltoid $\mathbin{\text{⏢}}' \subseteq \mathbin{\text{⏢}}_1$ satisfying the following properties:

- The boundary of $\mathbin{\text{⏢}}'$ consists of one (maximal) concave chain and two line segments.
- The boundary of the geodesic convex hull of $S_i \cap \mathbin{\text{⏢}}'$ excluding $\pi_{B_i}(x, q)$ is exactly $\gamma_1 \setminus \pi_{B_i}(s_1, q)$, where q is the corner of $\mathbin{\text{⏢}}'$ that does not lie on xy.

We observe that $\pi(s_1, y) \cap \gamma_1$ is connected. We find the point $q \in \pi(s_1, y)$ closest to y such that $\pi(s_1, q) \subseteq \gamma_1$. It is possible that $q = s_1$. See Figure 3(b) and (c). We can compute q in $O(n_i^{2/3} \log^2 n_i)$ time by applying binary search on $\pi(s_1, y)$ with a geodesic triangle counting query described in Section 3. We extend the edge of $\pi(s_1, q)$ incident to q until it hits xy. Let y' be the intersection of xy with the extension. We let $\mathbin{\text{⏢}}'$ be the geodesic triangle with corners q, y' and x. Then this geodesic triangle satisfies the properties we want to achieve.

Finding an Euclidean triangle $\triangle$. We choose the Euclidean triangle $\triangle$ whose three corners are the three corners of $\mathbin{\text{⏢}}'$. Since $\mathbin{\text{⏢}}'$ is a deltoid with respect to B_i, the line segment xq connecting x and q appears on the boundary of the Euclidean convex hull CH of $(S_i \cup V(B_i)) \cap \triangle$. Moreover, any point of $S_i \cup V(B_i)$ contained in the interior of the region bounded by $\pi(x, q)$ and xq does not appear on the boundary of CH. Thus the following holds.

▶ **Lemma 14.** *The polygonal chain $\gamma_1 \setminus \pi(s_1, q)$ coincides with the part of the boundary of the Euclidean convex hull of $(S_i \cup V(B_i)) \cap \triangle$ excluding xq.*

Computing the convex hull of $(S_i \cup V(B_i)) \cap \triangle$. Let $P = (\widetilde{S}_i \cap S_i) \cup V(\widetilde{B}_i \cap B_i)$. We first compute the convex hull of $P \cap \triangle$ using the triangle-range hull tree. Let CH be the convex hull. Then we consider each point p in $S_i \setminus \widetilde{S}_i$ one by one, and update CH to be the convex hull of CH and p. Each update takes $O(\log^2 n_i)$ time because we have to spend $O(\log n_i)$ time to access the jth vertex of CH for some index j. Finally, we compute the Euclidean convex hull of $(S_i \cup V(B_i)) \cap \triangle$.

▶ **Lemma 15.** *We can compute the Euclidean convex hull of $(S_i \cup V(B_i)) \cap \triangle$ in $O({n_i}^{1/2} \log^2 n_i)$ time with high probability.*

Now we have γ_1 and γ_2 consisting of $O({n_i}^{1/3})$ subchains belonging to T1, T2, or T3. We insert them to the representation tree in $O({n_i}^{1/3} \log n_i)$ time. Then we remove $\pi(s_1, s_2)$ from the representation tree. This takes $O(\log n_i)$ time since the representation tree is a concatenable queue supporting split operation.

Therefore, the running time for handling the insertion of a barrier is dominated by the running time for the second step, which takes $O(n_i^{2/3} \log^2 n_i)$ time.

▶ **Lemma 16.** *The geodesic convex hull of S_i with respect to $B_{i-1} \cup \{b\}$ for some barrier b can be computed in $O(n_i^{2/3} \log^2 n_i)$ time with high probability once we have CH_{i-1}.*

▶ **Theorem 17.** *We can update the geodesic convex hull in $O(n^{2/3} \log^2 n)$ amortized time with high probability under insertions and deletions of points and barriers, where n is the total number of the points and barriers at the moment. A line stabbing query, inclusion query, and tangent query can be answered in polylogarithmic time in the worst case.*

References

1. Julien Basch, Jeff Erickson, Leonidas J. Guibas, John Hershberger, and Li Zhang. Kinetic collision detection between two simple polygons. *Computational Geometry*, 27(3):211–235, 2004.
2. Jon Louis Bentley and James B. Saxe. Decomposable searching problems 1: Static-to-dynamic transformations. *Journal of Algorithms*, 1(4):297–396, 1980.
3. Gerth Stølting Brodal and Riko Jacob. Dynamic planar convex hull. In *Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science (FOCS 2002)*, pages 617–626, 2002.
4. Timothy M. Chan. Optimal partition trees. *Discrete & Computational Geometry*, 47(4):661–690, 2012.
5. Timothy M. Chan and Yakov Nekrich. Towards an optimal method for dynamic planar point location. In *Proceedings of the IEEE 56th Annual Symposium on Foundations of Computer Science (FOCS 2015)*, pages 390–409, 2015.
6. Bernard Chazelle. Lower bounds on the complexity of polytope range searching. *Journal of the American Mathematical Society*, 2(4):637–666, 1989.
7. Bernard Chazelle, Herbert Edelsbrunner, Michelangelo Grigni, Leonidas Guibas, John Hershberger, Micha Sharir, and Jack Snoeyink. Ray shooting in polygons using geodesic triangulations. *Algorithmica*, 12(1):54–68, 1994.
8. Bernard Chazelle, Micha Sharir, and Emo Welzl. Quasi-optimal upper bounds for simplex range searching and new zone theorems. *Algorithmica*, 8(1):407–429, 1992.
9. Yi-Jen Chiang, Franco P. Preparata, and Roberto 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.
10. Anurag Ganguli, Jorge Cortés, and Francesco Bullo. Multirobot rendezvous with visibility sensors in nonconvex environments. *IEEE Transactions on Robotics*, 25(2):340–352, 2009.
11. Michael T. Goodrich and Roberto Tamassia. Dynamic ray shooting and shortest paths in planar subdivisions via balanced geodesic triangulations. *Journal of Algorithms*, 23(1):51–73, 1997.
12. Mashhood Ishaque and Csaba D. Tóth. Relative convex hulls in semi-dynamic arrangements. *Algorithmica*, 68(2):448–482, 2014.
13. David Kirkpatrick and Jack Snoeyink. Computing common tangents without a separating line. In *Proceedings of the 4th International Workshop on Algorithms and Data Structures (WADS 1995)*, pages 183–193, 1995.
14. Jiří Matoušek. Efficient partition trees. *Discrete & Computational Geometry*, 8(3):315–334, 1992.
15. Jiří Matoušek. *Lectures on discrete geometry*. Springer Science & Business Media, 2013.
16. Mark H. Overmars and Jan van Leeuwen. Maintenance of configurations in the plane. *Journal of Computer and System Science*, 23(2):166–204, 1981.
17. Mark H. Overmars and Jan van Leeuwen. Worst-case optimal insertion and deletion methods for decomposable searching problem. *Information Processing Letters*, 12(4):168–173, 1981.
18. Jack Sklansky, Robert L. Chazin, and Bruce J. Hansen. Minimum-perimeter polygons of digitized silhouettes. *IEEE Transactions on Computers*, C-21(3):260–268, 1972.
19. Ileana Streinu. Pseudo-triangulations, rigidity and motion planning. *Discrete and Computational Geometry*, 34:587–635, 2005.

Voronoi Diagrams for a Moderate-Sized Point-Set in a Simple Polygon*

Eunjin Oh[1] and Hee-Kap Ahn[2]

1 **Department of Computer Science and Engineering, POSTECH, Pohang, Korea**
jin9082@postech.ac.kr
2 **Department of Computer Science and Engineering, POSTECH, Pohang, Korea**
heekap@postech.ac.kr

Abstract

Given a set of sites in a simple polygon, a geodesic Voronoi diagram partitions the polygon into regions based on distances to sites under the geodesic metric. We present algorithms for computing the geodesic nearest-point, higher-order and farthest-point Voronoi diagrams of m point sites in a simple n-gon, which improve the best known ones for $m \leq n/\operatorname{polylog} n$. Moreover, the algorithms for the nearest-point and farthest-point Voronoi diagrams are optimal for $m \leq n/\operatorname{polylog} n$. This partially answers a question posed by Mitchell in the Handbook of Computational Geometry.

1998 ACM Subject Classification I.3.5 Computational Geometry and Object Modeling

Keywords and phrases Simple polygons, Voronoi diagrams, geodesic distance

Digital Object Identifier 10.4230/LIPIcs.SoCG.2017.52

1 Introduction

The *geodesic distance* between any two points x and y contained in a simple polygon is the length of the shortest path contained in the polygon connecting x and y. A geodesic Voronoi diagram of a set S of m sites contained in a simple polygon P partitions P into regions based on distances to sites of S under the geodesic metric. The *geodesic nearest-point Voronoi diagram* of S partitions P into cells, exactly one cell per site, such that every point in a cell has the same nearest site of S under the geodesic metric. The higher-order Voronoi diagram, also known as the order-k Voronoi diagram, is a generalization of the nearest-point Voronoi diagram. For an integer k with $1 \leq k \leq m-1$, the *geodesic order-k Voronoi diagram* of S partitions P into cells, at most one cell per k-tuple of sites, such that every point in a cell has the same k nearest sites under the geodesic metric. Thus, the geodesic order-1 Voronoi diagram is the geodesic nearest-point Voronoi diagram. The geodesic order-$(m-1)$ Voronoi diagram is called the geodesic farthest-point Voronoi diagram. Hence, the *geodesic farthest-point Voronoi diagram* of S partitions P into cells, at most one cell per site, such that every point in a cell has the same farthest site under the geodesic metric.

In this paper, we study the problem of computing the geodesic nearest-point, higher-order and farthest-point Voronoi diagrams of a set S of m points contained in a simple n-gon P. Each edge of a geodesic Voronoi diagram is either a hyperbolic arc or a line segment consisting of points equidistant from two sites [2, 3, 11]. The boundary between any two neighboring cells of a geodesic Voronoi diagram is a chain of $O(n)$ edges. Each end vertex

* This work was supported by the NRF grant 2011-0030044 (SRC-GAIA) funded by the government of Korea.

of the boundary is of degree 1 or 3 under the assumption that no point in the polygon is equidistant from four distinct sites while every other vertex is of degree 2. There are $O(k(m-k))$ degree-3 vertices in the geodesic order-k Voronoi diagram of S [11]. Every degree-3 vertex is equidistant from three sites and is a point where three Voronoi cells meet. The number of degree-2 vertices is $\Theta(n)$ for the geodesic nearest-point Voronoi diagram and the geodesic farthest-point Voronoi diagram [2, 3]. For the geodesic order-k Voronoi diagram, it is known that the number of degree-2 vertices is $O(kn)$ [11], but this bound is not tight.

The first nontrivial algorithm for computing the geodesic nearest-point Voronoi diagram was given by Aronov in 1989 [2]. Their algorithm takes $O((n+m)\log^2(n+m))$ time. Later, Papadopoulou and Lee [15] improved the running time to $O((n+m)\log(n+m))$. However, there has been no progress since then while the best known lower bound of the running time remains to be $\Omega(n+m\log m)$. In fact, Mitchell posed a question whether this gap can be resolved in the Handbook of Computational Geometry [13, Chapter 27].

For the geodesic order-k Voronoi diagram, the first nontrivial algorithm was given by Liu and Lee [11] in 2013. Their algorithm works for a polygonal domain with holes and takes $O(k^2(n+m)\log(n+m))$ time. Thus, this algorithm also works for a simple polygon. They presented an asymptotically tight combinatorial complexity of the geodesic order-k Voronoi diagram for a polygonal domain with holes, which is $\Theta(k(m-k)+kn)$. However, it is not tight for a simple polygon: the geodesic order-$(m-1)$ Voronoi diagram of m points in a simple n-gon has complexity $\Theta(n+m)$ [3]. There is no bound better than the one by Liu and Lee known for the complexity of the geodesic order-k Voronoi diagram in a simple polygon.

For the geodesic farthest-point Voronoi diagram, the first nontrivial algorithm was given by Aronov et al. [3] in 1993, which takes $O((n+m)\log(n+m))$ time. While the best known lower bound is $\Omega(n+m\log m)$, there has been no progress until Oh et al. [14] presented a faster $O((n+m)\log\log n)$-time algorithm for the special case that all sites are on the boundary of the polygon in 2016. They also claimed that their algorithm can be extended to compute the geodesic farthest-point Voronoi diagram for any m points contained in a simple n-gon in $O(n\log\log n+m\log(n+m))$ time.[1]

Our results. Our main contributions are the algorithms for computing the nearest-point, higher-order and farthest-point Voronoi diagrams of m sites in a simple n-gon, which improve the best known ones for $m \leq n/\operatorname{polylog} n$. To be specific, we present

- an $O(n+m\log m\log^2 n)$-time algorithm for the nearest-point Voronoi diagram,
- an $O(k^2 m\log m\log^2 n+\min\{nk, n(m-k)\})$-time algorithm for the order-k Voronoi diagram, and
- an $O(n+m\log m+m\log^2 n)$-time algorithm for the farthest-point Voronoi diagram.

Moreover, our algorithms reduce the gaps of the running times towards the lower bounds. Our algorithm for the geodesic nearest-point Voronoi diagram is optimal for $m \leq n/\log^3 n$. Since the algorithm by Papadopoulou and Lee is optimal for $m \geq n$, our algorithm together with the one by Papadopoulou and Lee gives the optimal running time for computing the diagram, except for the case that $n/\log^3 n < m < n$.

Similarly, our algorithm for the geodesic farthest-point Voronoi diagram is optimal for $m \leq n/\log^2 n$. Since the algorithm by Aronov et al. [3] is optimal for $m \geq n$, our algorithm together with the one by Aronov et al. gives the optimal running time for computing the diagram, except for the case that $n/\log^2 n < m < n$. This answers the question posed by

[1] Details will be found in the journal version of their paper.

Mitchell on the geodesic nearest-point and farthest-point Voronoi diagrams, except for the short intervals of $n/\operatorname{polylog} n < m < n$ stated above.

For the geodesic order-k Voronoi diagram, we analyze an asymptotically tight combinatorial complexity of the diagram for a simple polygon, which is $\Theta(k(m-k)+\min\{nk, n(m-k)\})$.

Other contributions of this paper are the algorithms for computing the topological structures of the geodesic nearest-point, order-k and farthest-point Voronoi diagrams which take $O(m \log m \log^2 n)$, $O(k^2 m \log m \log^2 n)$ and $O(m \log m \log^2 n)$ time, respectively. These algorithms allow us to obtain a dynamic data structure for answering nearest or farthest point queries. In this problem, we are given a static simple n-gon P and a dynamic point set $S \subseteq P$. We are allowed to insert points to S and delete points from S. After processing updates, we are to find the point of S nearest (or farthest) from a query point efficiently. This data structure requires $O(\sqrt{m}\log(n+m))$ query time and $O(\sqrt{m}\log m \log^2 n)$ update time, where m is the number of points in S at the moment.

1.1 Outline

Our algorithms for computing the geodesic nearest-point, higher-order and farthest-point Voronoi diagrams are based on a *polygon-sweep paradigm*. For the geodesic nearest-point and higher-order Voronoi diagrams, we fix a point o on the boundary of the polygon and move another point x from o in clockwise direction along the boundary of the polygon. While x moves along the boundary, we compute the Voronoi diagram of sites contained in the subpolygon bounded by the shortest path between o and x and the part of the boundary of P from o to x in clockwise order. For the geodesic farthest-point Voronoi diagram, we sweep the polygon with a curve consisting of points equidistant from the geodesic center of the sites. The curve moves from the boundary towards the geodesic center. During the sweep, we gradually compute the diagram restricted to the region we have swept.

To achieve algorithms faster than the best known ones for $m \leq n/\operatorname{polylog} n$, we first compute the topological structure of a diagram instead of computing the diagram itself directly. The topological structure, which will be defined later, represents the adjacency of the Voronoi cells and has complexity smaller than the one of the complete Voronoi diagram. Once we have the topological structure, we can compute the complete Voronoi diagram in $O(T_1 + T_2 \log n)$ time, where T_1 is the combinatorial complexity of the complete Voronoi diagram and T_2 is the combinatorial complexity of the topological structure of the diagram.

We define four types of events where the topological structure changes. To handle each event, we compute a point equidistant from three points under the geodesic metric. There is no algorithm known for computing a point equidistant from three points efficiently, except an $O(n)$-time trivial algorithm. We present an $O(\log^2 n)$-time algorithm assuming that the data structure of Guibas and Hershberger [9] is constructed for P. This algorithm allows us to handle each event in $O(\operatorname{polylog}\{n, m\})$ time.

One application of an algorithm for computing the topological structure is a data structure for nearest (or farthest) point queries for a dynamic point set. To obtain this data structure, we apply the framework given by Bentley and Saxe [4]. We observe that we can find the Voronoi cell of the diagram containing a query point in $O(\log(n+m))$ time once we have the topological structure of the diagram.

2 Preliminaries

Let P be a simple n-gon and S be a set of m points in P. For ease of description, we use $\mathsf{VD}[S]$, k-$\mathsf{VD}[S]$ and $\mathsf{FVD}[S]$ (or simply VD, k-VD and FVD if they are understood in the

context) to denote the geodesic nearest-point, order-k and farthest-point Voronoi diagrams of S in P, respectively. We assume the *general position condition* that no vertex of P is equidistant from two distinct sites of S and no point of P is equidistant from four distinct sites of S. This was also assumed in previous work [3, 11, 15] on geodesic Voronoi diagrams. This condition can be obtained by applying a slight perturbation of the positions of sites [6].

Consider any three points x, y and z in P. We use $\pi(x, y)$ to denote the shortest path (geodesic path) between x and y contained in P, and $d(x, y)$ to denote the geodesic distance between x and y. Two geodesic paths $\pi(x, y)$ and $\pi(x, z)$ do not cross each other, but may overlap with each other. We call a point x' the *junction* of $\pi(x, y)$ and $\pi(x, z)$ if $\pi(x, x')$ is the maximal common path of $\pi(x, y)$ and $\pi(x, z)$. Refer to Figure 1(a).

Given a point $p \in P$ and a closed set $A \subseteq P$, we slightly abuse the notation $\pi(p, A)$ to denote the shortest path contained in P connecting p and a point in A. Similarly, we abuse the notation $d(p, A)$ to denote the length of $\pi(p, A)$. It holds that $d(p, A) \leq d(p, q) + d(q, A)$ for any two points $p, q \in P$ and any closed set $A \subseteq P$.

We say a set $A \subseteq P$ is *geodesically convex* if $\pi(x, y) \subseteq A$ for any two points x and y in A. The *geodesic convex hull* of S is the intersection of all geodesic convex sets containing S. The geodesic convex hull of a set of m points can be computed in $O(n + m \log(n + m))$ time [9].

The *geodesic center* of a simple polygon P is the point c that minimizes $\max_{p \in P} d(c, p)$. The center is unique [16] and can be computed in $O(n)$ time [1]. Similarly, the geodesic center of S can be defined as the point c that minimizes $\max_{s \in S} d(c, s)$. It is known that the geodesic center of a set S of m points in P coincides with the geodesic center of the geodesic convex hull of S [3]. Therefore, we can compute the center of S by computing the geodesic convex hull of S and its center. This takes $O(n + m \log(n + m))$ time in total.

Due to lack of space, some of the proofs are omitted. All missing proofs can be found in the full version of this paper.

3 Computing the Geodesic Center of Points in a Simple Polygon

We first present an $O(\log^2 n)$-time algorithm for computing the geodesic center of three points contained in P, assuming that we have the data structure of Guibas and Hershberger [9, 10]. This algorithm will be used as a subprocedure for computing the geodesic Voronoi diagrams.

3.1 Computing the Geodesic Center of Three Points

Let p_1, p_2 and p_3 be three points in P, and let c be the geodesic center of them. The geodesic convex hull of p_1, p_2, p_3 is bounded by $\pi(p_1, p_2), \pi(p_2, p_3)$, and $\pi(p_3, p_1)$. The geodesic convex hull may have complexity $\Omega(n)$, but its interior is bounded by at most three concave chains. This allows us to compute the geodesic center of it efficiently.

We first construct the data structure of Guibas and Hershberger [9, 10] for P that allows us to compute the geodesic distance between any two points in $O(\log n)$ time. To compute c, we compute the shortest paths $\pi(p_1, p_2), \pi(p_2, p_3)$, and $\pi(p_3, p_1)$. Each shortest path has a linear size, but we can compute them in $O(\log n)$ time using the data structure of Guibas and Hershberger. Then we find a convex t-gon with $t \leq 6$ containing c such that the geodesic path $\pi(x, p_i)$ has the same combinatorial structure for any point x in the t-gon for each $i = 1, 2, 3$. To find such a convex t-gon, we apply two-level binary search. Then we can compute c directly in constant time inside the t-gon.

The data structure given by Guibas and Hershberger. Guibas and Hershberger [9, 10] gave a data structure of linear size that enables us to compute the geodesic distance between

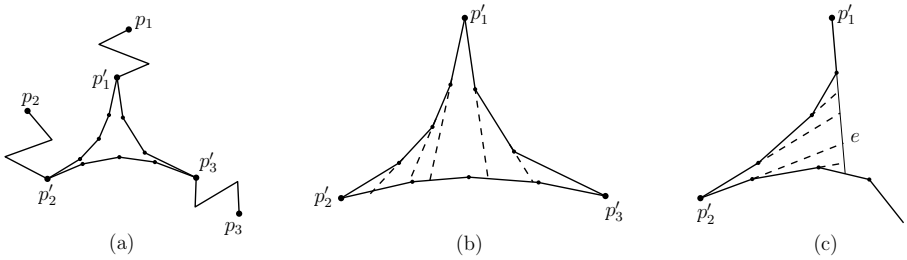


Figure 1 (a) p'_i is the junction of $\pi(p_i, p_j)$ and $\pi(p_i, p_k)$ for three distinct indices i, j and k in $\{1, 2, 3\}$. (b) The subdivision of $\triangle$ with respect to p'_1. (c) The subdivision of e with respect to p'_2.

any two query points lying inside P in $O(\log n)$ time. We call this structure the *shortest path data structure*. They showed that this data structure can be constructed in $O(n)$ time.

In the preprocessing, they compute a number of shortest paths such that for any two points p and q in P, the shortest path $\pi(p, q)$ consists of $O(\log n)$ subchains of precomputed shortest paths and $O(\log n)$ additional edges. In the query algorithm, they find such subchains and edges connecting them in $O(\log n)$ time. Then the query algorithm returns the shortest path between two query points represented as a binary tree of height $O(\log n)$ [10]. Therefore, we can apply binary search on the vertices of the shortest path between any two points.

Computing the geodesic center of three points: two-level binary search. Let $\triangle$ be the geodesic convex hull of p_1, p_2 and p_3. The geodesic center c of the three points is the geodesic center of $\triangle$ [3], thus is contained in $\triangle$. If the center lies on the boundary of $\triangle$, we can compute it in $O(\log n)$ time since it is the midpoint of two points. So, we assume that the center lies in the interior of $\triangle$. Let p'_i be the junction of $\pi(p_i, p_j)$ and $\pi(p_i, p_k)$ for three distinct indices i, j and k in $\{1, 2, 3\}$. See Figure 1(a).

We use the following lemmas to apply two-level binary search.

▶ **Lemma 1** ([9]). *We can compute the junctions p'_1, p'_2 and p'_3 in $O(\log n)$ time.*

▶ **Lemma 2** ([5]). *Given a point $p \in \triangle$ and a direction, we can find the first intersection point of the boundary of $\triangle$ with the ray from p in the direction in $O(\log n)$ time.*

The first level. Imagine that we subdivide $\triangle$ into $O(n)$ cells with respect to p'_1 by extending the edges of $\pi(p'_1, p'_2) \cup \pi(p'_1, p'_3)$ towards $\pi(p'_2, p'_3)$. See Figure 1(b). The extensions of the edges can be sorted in the order of their endpoints appearing along $\pi(p'_2, p'_3)$. Consider the subdivision of $\triangle$ by the extensions, and assume that we can determine which side of a given extension in $\triangle$ contains c in $T(n)$ time. Then we can compute the cell of the subdivision containing c in $O(T(n) \log n)$ time by applying binary search on the extensions. Note that any point x in the same cell has the same combinatorial structure of $\pi(x, p_1)$ (and $\pi(x, p'_1)$).

We also do this for p'_2 and p'_3. Then we have three cells whose intersection contains c. Let D be the intersection of these three cells. We can find D in constant time by the the fact that D is a convex polygon with at most six edges from extensions of the cells.

We do not subdivide $\triangle$ explicitly. Because we have $\pi(p'_1, p'_2)$ and $\pi(p'_1, p'_3)$ in binary trees of height $O(\log n)$, we can apply binary search on the extensions of the edges of the geodesic paths without subdividing $\triangle$ explicitly. In this case, during the binary search, we compute the extension of a given edge of $\pi(p'_1, p'_2) \cup \pi(p'_1, p'_3)$ using Lemma 2 in $O(\log n)$ time.

There is a vertex p on the boundary of $\triangle$ such that for any point x contained in D we have $d(p_1, x) = d(p_1, p) + \|p - x\|$, where $\|p - x\|$ is the Euclidean distance between p and x.

Moreover, we already have p from the computation of the cell containing c in the subdivision with respect to p'_1. The same holds for p_2 and p_3. Therefore, we can compute the point c that minimizes the maximum of $d(c, p_1), d(c, p_2)$ and $d(c, p_3)$ in constant time inside D.

Therefore, we have the following lemma.

▶ **Lemma 3.** *Assuming that we can determine which side of an extension in $\triangle$ contains c in $T(n)$ time, we can compute the geodesic center c in $O((T(n) + \log n) \log n)$ time.*

The second level. In the second level binary search, we determine which side of an extension e in $\triangle$ contains c. Without loss of generality, we assume that e comes from the subdivision with respect to p'_1. Then $\pi(p_1, x)$ has the same combinatorial structure for any point $x \in e$.

This subproblem was also considered in previous works on computing the geodesic center of a simple polygon [1, 16]. They first compute the point c_e in e that minimizes $\max_{p \in P} d(p, c_e)$, that is, the geodesic center of the polygon restricted to e. Based on c_e and its farthest point, Pollack et al. [16] presented a way to decide which side of e contains the geodesic center of the polygon in constant time. However, to compute c_e, they spend $O(n)$ time.

In our problem, we can do this in logarithmic time using the fact that the interior of $\triangle$ is bounded by at most three concave chains. By this fact, there are two possible cases: c_e is an endpoint of e, or c_e is equidistant from p_1 and p_i for $i = 2$ or 3. We compute the point on e equidistant from p_1 and p_2, and the point on e equidistant from p_1 and p_3. Then we find the point c_e among the two points and the two endpoints of e. In the following, we show how to compute the point on e equidistant from p_1 and p_2 if it exists. The point on e equidistant from p_1 and p_3 can be computed analogously.

Observe that e can be subdivided into $O(n)$ disjoint line segments by the extensions of the edges of $\pi(p'_2, v_1) \cup \pi(p'_2, v_2)$ towards e, where v_1 and v_2 are endpoints of e. See Figure 1(c). For any point x in the same line segment, $\pi(p_2, x)$ has the same combinatorial structure.

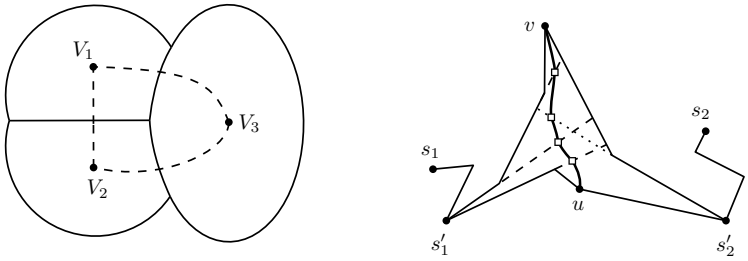
There is at most one point on e equidistant from p_1 and p_2. (For a proof, see the full paper.) Thus we can apply binary search on the line segments in the subdivision of e. As we did before, we do not subdivide e explicitly. Instead, we use the binary trees representing $\pi(p'_2, v_1)$ and $\pi(p'_2, v_2)$. For a point x in e, by comparing $d(p_1, x)$ and $d(p_2, x)$, we can determine which part of x on e contains the point equidistant from p_1 and p_2 in constant time. In this case, we can compute the extension from an edge towards e in constant time since e is a line segment. Thus, we complete the binary search in $O(\log n)$ time.

Therefore, we can compute c_e in $O(\log n)$ time and determine which side of e in $\triangle$ contains c in the same time using the method of Pollack et al [16]. The following lemma summarizes this section.

▶ **Lemma 4.** *Given any three points p_1, p_2 and p_3 contained in a simple n-gon P, the geodesic center of p_1, p_2 and p_3 can be computed in $O(\log^2 n)$ time after the shortest path data structure for P is constructed in linear time.*

Similarly, we can compute a point equidistant from any three points in P (or, two points and a line segment). They are used as subprocedures for computing the Voronoi diagrams. Note that the geodesic center of three points may not be equidistant from all of them. Moreover, there may be an infinite number of points equidistant from the three points (or, two points and a line segment). In this case, we compute the one closest to them.

▶ **Lemma 5.** *Given any three points contained in a simple n-gon, we can compute the closest equidistant point from them under the geodesic metric in $O(\log^2 n)$ time if it exists.*



(a) (b)

Figure 2 (a) The adjacency graph of a Voronoi diagram. (b) The number of edges in the common boundary of two adjacent Voronoi cells is bounded by the total complexity of $\pi(s_1', v)$, $\pi(s_1', u)$, $\pi(s_2', v)$ and $\pi(s_2', u)$, where u and v are endpoints of the common boundary.

▶ **Lemma 6.** *Given any two points and any line segment contained in a simple n-gon, we can compute the closest equidistant point from them under the geodesic metric in $O(\log^2 n)$ time if it exists.*

Combining the result in this subsection with the algorithms for computing the center of points in the plane [12] and computing the geodesic center of a simple polygon [16], we can compute the geodesic center of m points contained in P. (For details, see the full version.)

▶ **Theorem 7.** *The geodesic center of m points contained in a simple polygon with n vertices can be computed in $O(m \log m \log^2 n)$ time after the shortest path data structure for the simple polygon is constructed.*

4 Topological Structures of Voronoi Diagrams

In this section, we define the topological structure of Voronoi diagrams and show how to compute the complete Voronoi diagrams from their topological structures. The topological structure of a Voronoi diagram represents the adjacency of their Voronoi cells.

The common boundary of any two adjacent Voronoi cells is connected for the nearest-point and farthest-point Voronoi diagrams of point sites in a simple polygon [2, 3]. Similarly, it can be shown that this also holds for the higher-order Voronoi diagram of point sites in a simple polygon.

The topological structure of k-VD is defined as follows for $1 \leq k \leq m-1$. Imagine that we apply vertex suppression for every degree-2 vertex of the Voronoi diagram while preserving the topology of the Voronoi diagram. Vertex suppression of a vertex v of degree 2 is the operation of removing v and adding an edge connecting the two neighbors of v. We call the dual of this graph the *adjacency graph* of the Voronoi diagram. See Figure 2(a). It represents the topological structure of the Voronoi diagram. The adjacency graph is a planar graph with complexity $O(k(m-k))$, because the number of degree-1 and degree-3 vertices of k-VD is $O(k(m-k))$ [11].

Assume that we have the adjacency graph of the Voronoi diagram together with the exact positions of the degree-1 and degree-3 vertices of the diagram. Consider two Voronoi cells V_1 and V_2 which are adjacent to each other. Each Voronoi cell is defined by k sites, but any two adjacent Voronoi cells share $k-1$ sites. Let s_1 and s_2 be the two sites defining V_1 and V_2, respectively, which are not shared by them. There are two Voronoi vertices v and u defined by a triple (s_1, s_2, s) and a triple (s_1, s_2, s') of sites defining v and u, respectively, for some

sites s, s'. Each Voronoi edge in the common boundary of V_1 and V_2 is a part of the bisector of s_1 and s_2 lying between v and u. See Figure 2(b).

To compute the Voronoi edges in the common boundary of V_1 and V_2, we consider the geodesic paths $\pi(s'_i, v)$ and $\pi(s'_i, u)$ for $i = 1, 2$, where s'_i is the junction of $\pi(s_i, v)$ and $\pi(s_i, u)$. Then for any vertex x in $\pi(s'_1, v) \cup \pi(s'_1, u)$, there exists a point q in the bisector of s_1 and s_2 lying between v and u such that $\pi(s_1, q)$ and $\pi(s_1, x)$ have the same combinatorial structure. The same holds for s_2. Thus, the number of edges in the common boundary is bounded by the total complexity of $\pi(s'_1, v) \cup \pi(s'_1, u)$ and $\pi(s'_2, v) \cup \pi(s'_2, u)$. Thus, we may compute the geodesic paths explicitly and consider every edge of the geodesic paths.

Therefore, we can compute the common boundary of two adjacent Voronoi cells in time linear to its complexity plus $O(\log n)$. This leads to $O(T_1 + T_2 \log n)$ time for computing the complete Voronoi diagram from its topological structure, where T_1 is the complexity of the complete Voronoi diagram and T_2 is the complexity of the adjacency graph.

▶ **Lemma 8.** *We can compute the complete Voronoi diagram of m points in a simple polygon with n vertices in $O(T_1 + T_2 \log n)$ time once its adjacency graph, and the degree-1 and degree-3 vertices at their places are given, where T_1 is the complexity of the complete Voronoi diagram and T_2 is the complexity of the adjacency graph.*

Therefore, in the following, we focus on computing the topological structure of VD, k-VD and FVD, that is, the adjacency graphs of them with degree-3 vertices at their places.

5 The Geodesic Nearest-Point Voronoi Diagram

Fortune [7] presented an $O(m \log m)$-time algorithm to compute the nearest-point Voronoi diagram of m points in the plane by sweeping the plane with a horizontal line from top to bottom. During the sweep, they compute a part of the Voronoi diagram of sites lying above the horizontal line, which finally becomes the complete Voronoi diagram in the end of the sweep. To do this, they define two types of events and handle $O(m)$ events in total. Each event can be handled in $O(\log m)$ time, which leads to $O(m \log m)$ total running time.

In our case, we sweep the polygon with a geodesic path $\pi(o, x)$ for a fixed point o on the boundary of P and a point x moving along the boundary of P from o in clockwise direction. The point x is called the *sweep point*. If we compute all $O(n + m)$ degree-1, degree-2 and degree-3 vertices of the Voronoi diagram during the sweep, we may not achieve the running time better than $O((n + m) \log(n + m))$. The key to improve the running time is to compute the topological structure of the Voronoi diagram first which consists of the degree-1 and degree-3 vertices of the Voronoi diagram and the adjacency graph of Voronoi cells. Then we construct the complete Voronoi diagram, including degree-2 vertices, from its topological structure using Lemma 8.

Let o be an arbitrary point on ∂P, where ∂P denotes the boundary of P. Consider the sweep point x that moves from o along ∂P in clockwise order. We use $P(x)$ to denote the subpolygon of P bounded by $\pi(o, x)$ and the part of ∂P from o to x in clockwise order. Note that $P(x)$ is weakly simple. See Figure 3. As x moves along ∂P, $P(x)$ does not decrease. That is, $P(x_1) \subseteq P(x_2)$ for any two points x_1 and x_2 on ∂P such that x_1 comes before x_2 from o in clockwise order.

For a site $s \in P(x)$, let $R_s(x)$ be the region $\{p \in P(x) \mid d(p, s) \leq d(p, \pi(o, x))\}$. By definition, $R_s(x)$ does not decrease as x moves along ∂P in clockwise order.

▶ **Lemma 9.** *$R_s(x)$ is connected.*

Proof. To prove the lemma, we show that $\pi(p,s) \subseteq R_s(x)$ for any point $p \in R_s(x)$. This implies that $R_s(x)$ is connected because s is contained in $R_s(x)$ if $s \in P(x)$ by definition. Let p be a point in $R_s(x)$. Consider a point $r \in \pi(p,s)$. We have $d(r,s) = d(p,s) - d(p,r)$. Moreover, we have $d(p,\pi(o,x)) - d(p,r) \leq d(r,\pi(o,x))$. Since $p \in R_s(x)$, it holds that $d(p,s) \leq d(p,\pi(o,x))$. Thus, $d(r,s) \leq d(r,\pi(o,x))$. Therefore, r is in $R_s(x)$, and therefore, $R_s(x)$ is connected. ◀

We say that a subset A of P is *weakly monotone* with respect to a geodesic path π' if the intersection of $\pi(p,\pi')$ with A is connected for any point $p \in A$.

▶ **Lemma 10.** *The boundary of $R_s(x)$ consists of one polygonal chain of ∂P and a simple curve whose both endpoints lie on $\partial P(x)$ unless $s \in \pi(o,x)$. Moreover, the simple curve is weakly monotone with respect to $\pi(o,x)$.*

Proof. For any point $p \in R_s(x)$, the geodesic ray from p in direction opposite to the edge of $\pi(p,\pi(o,x))$ incident to p is contained in $R_s(x)$. This is because $d(r,\pi(o,x)) = d(p,\pi(o,x)) + d(p,r) \geq d(p,s) + d(p,r) \geq d(r,s)$ by triangle inequality for any point r in the geodesic ray. The lemma follows from this fact and Lemma 9. ◀

Now we consider the union of $R_s(x)$ for every site s in $P(x)$ and denote it by $R(x)$. The dashed region in Figure 3(a) is $R(x)$. Note that for any point $p \in P(x)$ and any site $s \in S$ lying outside of $P(x)$, it holds that $d(p,\pi(o,x)) < d(p,s)$. This implies that for any point $p \in R(x)$, the nearest site of p is in $P(x)$. Therefore, to compute VD restricted to $R(x)$, we do not need to consider the sites lying outside of $P(x)$.

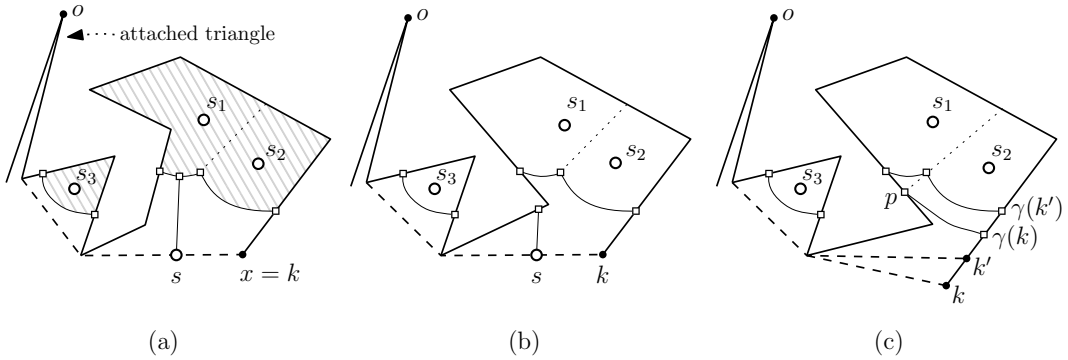
▶ **Corollary 11.** *The closure of $P(x) \setminus R(x)$ is weakly simple.*

▶ **Corollary 12.** *Each connected component of $R(x)$ consists of a polygonal chain of ∂P and a connected simple curve with endpoints on ∂P unless $\pi(o,x)$ contains some site. Moreover, the union of such curves is weakly monotone with respect to $\pi(o,x)$ at any time.*

We maintain the nearest-point Voronoi diagram of sites contained in $P(x)$ restricted to $R(x)$ while x moves along ∂P. However, after the sweep, $R(x)$ does not coincide with P. This means that we cannot obtain VD in this way. To solve this problem, we attach a long and very thin triangle to ∂P to make a bit larger simple polygon P', as illustrated in Figure 3, so that once we finish the sweep (when the sweep point returns back to o) in P' we have the complete Voronoi diagram in P. (The triangle has height of the diameter of P.)

The beach line and the breakpoints. There are $O(m)$ connected components of $R(x)$ each of whose boundaries consists of a polygonal chain of ∂P and a connected simple curve. The *beach line* is defined to be the union of $O(m)$ simple curves of $R(x)$. It has properties similar to those of the beach line of the Euclidean Voronoi diagram. It consists of $O(n+m)$ hyperbolic or linear arcs. We do not maintain them explicitly because the complexity of the sequence is too large for our purpose. Instead, we maintain the combinatorial structure of the beach line using $O(m)$ space as follows.

A point on the beach line is called a *breakpoint* if it is equidistant from two distinct sites. Each breakpoint moves and traces out some Voronoi edge as the sweep point moves along $\partial P'$. We represent each breakpoint symbolically. That is, a breakpoint is represented as the pair of sites equidistant from it. We say that such a pair *defines* the breakpoint. Given the pair defining a breakpoint and the position of the sweep point, we can find the exact position of the breakpoint using Lemma 6 in $O(\log^2 n)$ time.



(a) (b) (c)

Figure 3 (a) The site event defined by s with key k. Two (degenerate) breakpoints appear in $B(k)$. (b) Two (degenerate) endpoints appear in $B(k)$. (c) The vanishing event defined by the breakpoint of (s_1, s_2) with key k. The endpoint defined by s_1 and the breakpoint merge into the endpoint defined by s_2.

Additionally, we consider the endpoints of $O(m)$ connected curves of the beach line. We simply call them the *endpoints* of the beach line. For an endpoint p, there is the unique site s with $d(p, s) = d(p, \pi(o, x))$. We say that s *defines* p.

By Corollary 11 and Corollary 12, we can define the order of these breakpoints and these endpoints. Let $B(x) = \langle \beta_1, \ldots, \beta_{m'} \rangle$ be the sequence of the breakpoints and the endpoints of the beach line sorted in clockwise order along the boundary of $P(x) \setminus R(x)$ with $m' = O(m)$. We maintain $B(x)$ instead of all arcs on the beach line. As x moves along $\partial P'$, the sequence $B(x)$ changes.

While maintaining $B(x)$, we compute the adjacency graph of VD together with the degree-1 and degree-3 vertices of VD restricted to $R(x)$. (Recall that a cell of VD restricted to $R(x)$ is associated with a site in $P(x)$.) Specifically, when a new breakpoint defined by a pair (s, s') of sites is added to $B(x)$, we add an edge connecting the node for s and the node for s' into the adjacency graph.

Events. We have four types of events: site events, circle events, vanishing events, and merging events. Every event corresponds to a *key*, which is a point on the boundary of P'. We maintain the events with respect to their keys sorted in clockwise order from o along the boundary. Given a sorted sequence of ν events for $\nu \in \mathbb{N}$, we can insert a new event in $O(\log \nu)$ time since we are given each point on the boundary of P' together with the edge of P' where it lies. The event occurs when the sweep point x passes through the corresponding key. The sequence $B(x)$ changes only when x passes through the key of an event.

The definitions of the first two events are similar to the ones in Fortune's algorithm. Each site s in S defines a *site event*. The key k of the site event defined by s is the point on $\partial P'$ closest to o among the points x' with $s \in \pi(o, x')$. When the sweep point passes through k, the site s appears on the beach line. Moreover, new breakpoints defined by s and some other site s', or new endpoints defined by s appear on $B(x)$. See Figure 3(a) and (b).

The other events are defined by a pair of consecutive points in $B(x)$ or a single breakpoint in $B(x)$. Before the sweep point reaches the key of an event, the points in the pair defining the event may become non-consecutive, or the breakpoint defining the event may disappear from $B(x)$ due to the changes of $B(\cdot)$. In this case, we say that the event is *invalid*. Otherwise, we say that the event is *valid*.

A pair (β_1, β_2) of consecutive breakpoints in $B(x)$ defines a *circle event* if a point c equidistant from s_1, s_2 and s_3 exists, where (s_1, s_2) and (s_2, s_3) are two pairs of sites defining

β_1 and β_2, respectively. The key k of this circle event is the point on $\partial P'$ closest to o among the points x' with $d(c, \pi(o, x')) = d(c, s_1)(= d(c, s_2) = d(c, s_3))$. Assume that this event is valid when x passes through k. At that time, c appears on the beach line. Moreover, β_1 and β_2 disappear from the beach line, and a new breakpoint defined by (s_1, s_3) appears on the beach line. (One may think that β_1 and β_2 merge into the breakpoint defined by (s_1, s_3).)

Each breakpoint β in $B(x)$ defines a *vanishing event*. Let (s_1, s_2) be the pair of sites defining β. Consider two points on $\partial P'$ equidistant from s_1 and s_2. We observe that exactly one of the two points lies outside of $R(x)$. We denote it by p. See Figure 3(c). The key k of the vanishing event is the point on $\partial P'$ closest to o among the points x' with $d(p, \pi(o, x')) = d(p, s_1)$. Assume that this event is valid when x passes through k. Then, β traces out a Voronoi edge and reaches p, which is a degree-1 Voronoi vertex. Moreover, $B(x)$ changes accordingly as follows. Right before x reaches k, an endpoint defined by s_1 or s_2 is a neighbor of β in $B(x)$. (Otherwise, the beach line is not weakly monotone.) Without loss of generality, we assume that an endpoint of s_1 is a neighbor of β. When x reaches the key, this endpoint and β disappear from $B(x)$, and an endpoint defined by s_2 appears in $B(x)$. (One may think that this endpoint and β merge into the endpoint defined by s_2.)

A pair (β_1, β_2) of consecutive endpoints in $B(x)$ defines a *merging event* if β_1 and β_2 are endpoints of different connected curves of the beach line. Let s_1 and s_2 be the sites defining β_1 and β_2, respectively. Without loss of generality, assume that β_1 comes before β_2 from o in clockwise order. Let p be the (unique) point equidistant from s_1, s_2 that comes after β_1 and before β_2 from o along $\partial P'$ in clockwise order. The key k of the merging event is the point on $\partial P'$ closest to o among the points x' with $d(p, \pi(o, x')) = d(p, s_1)$. Assume that this event is valid when x passes through k. Then, as the sweep point x moves along $\partial P'$, β_1 and β_2 are closer and finally meet each other at p. This means that the two connected curves, one containing β_1 and the other containing β_2, merge into one connected curve. At this time, β_1 and β_2 merge into a breakpoint defined by (s_1, s_2).

5.1 An algorithm

Initially, $B(x) = B(o)$ is empty, so there is no circle, vanishing, or merging event. We compute the keys of all site events in advance. For each site, we compute the key corresponding to its event in $O(\log n)$ time [8].

As $B(x)$ changes, we obtain new pairs of consecutive breakpoints or endpoints, and new breakpoints. Then we compute the key of each event in $O(\log^2 n)$ time using the following lemmas and Lemma 5. In addition, as $B(x)$ changes, some event becomes invalid. Once an event becomes invalid, it does not become valid again. Thus we discard an event if it becomes invalid. Therefore, the number of events we have is $O(|B(x)|)$ at any time.

▶ **Lemma 13.** *Given a point p in P' and a distance $r \in \mathbb{R}$, the point on $\partial P'$ closest to o among the points x' with $d(p, \pi(o, x')) = r$ can be found in $O(\log^2 n)$ time.*

▶ **Lemma 14.** *For any two sites s_1 and s_2 in P', we can compute the two points equidistant from s_1 and s_2 lying on the boundary of P' in $O(\log^2 n)$ time.*

Handling site events. To handle a site event defined by a site s with key k, we do the following. By definition, the site s appears on the beach line when the sweep point x passes through k. Thus, two breakpoints defined by (s, s') for other sites s' (or two endpoints defined by s) appear on $B(k)$. See Figure 3(a) and (b). We find the positions of them in $B(x)$ and update $B(x)$ by adding them using Lemma 15.

▶ **Lemma 15.** *We can obtain $B(k)$ from $B(k')$ in $O(\log^2 n \log |B(k')|)$ time, where k' is the key previous to k.*

Adding two breakpoints or two endpoints to $B(x)$ makes a constant number of new pairs of consecutive breakpoints or endpoints, which define circle or merging events. This also makes a constant number of new vanishing events. We compute the key for each event in $O(\log^2 n)$ time and add the events to the event queue in $O(\log |B(x)|)$ time. Recall that the number of events we have is $O(|B(x)|)$ at any time. Therefore, each site event can be handled in $O(\log^2 n \log |B(x)|)$ time.

Handling the other events. Let k be the key of a circle, vanishing, or merging event. For a circle event, two breakpoints defining the event disappear from the beach line, and a new breakpoint appears. For a vanishing event, the breakpoint defining the event and its neighboring endpoint disappear from the beach line, and a new endpoint appears. For a merging event, two endpoints defining the event are replaced with a new breakpoint. In any case, we can update $B(x)$ in $O(\log |B(x)|)$ time.

After updating $B(x)$, we have a constant number of new pairs of consecutive breakpoints or endpoints, and a constant number of new breakpoints. We compute the keys of events defined by them in $O(\log^2 n)$ time.

Analysis. During the sweep, we handle $O(m)$ events in total. Each event can be processed in $O(\log^2 n \log m)$ time since the length of $B(x)$ is $O(m)$ at any time. Thus we have the following lemma and theorem.

▶ **Lemma 16.** *After computing the shortest path data structure for P' and the shortest path map for P' from the point o, we can compute the topological structure of VD in $O(m \log m \log^2 n)$ time using $O(m)$ space (excluding the two data structures).*

▶ **Theorem 17.** *Given a set of m point sites contained in a simple polygon with n vertices, we can compute the geodesic nearest-point Voronoi diagram of the sites in $O(n + m \log m \log^2 n)$ time using $O(n + m)$ space.*

6 The Geodesic Higher-order Voronoi Diagram

In this section, we first present an asymptotically tight combinatorial complexity of the geodesic higher-order Voronoi diagram of points. Then we present an algorithm to compute the diagram by applying the polygon-sweep paradigm introduced in Section 5. In the plane, Zavershynskyi and Papadopoulou [17] presented a plane-sweep algorithm to compute the higher-order Voronoi diagram. We use their approach together with our approach in Section 5. In the following, we assume that $1 \leq k \leq m - 1$.

6.1 The Complexity of the Diagram inside a Simple Polygon

Liu and Lee [11] presented an asymptotically tight complexity of the geodesic higher-order Voronoi diagram of points in a polygonal domain with holes, which is $\Theta(k(m - k) + nk)$. However, it is not tight for a simple polygon.

Figure 4 shows an example that the order-k Voronoi diagram has complexity of $\Omega(k(m - k) + \min\{nk, n(m - k)\})$. We construct a simple polygon P and a set of sites with respect to a reference point o. Let ℓ_1 be a sufficiently long horizontal line segment whose right endpoint is o and ℓ_2 be a sufficiently long line segment with a positive slope whose left endpoint is

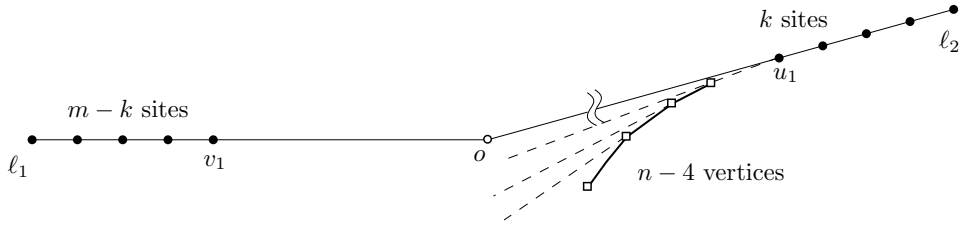


Figure 4 An example that the order-k Voronoi diagram has complexity of $\Omega(k(m-k) + \min\{nk, n(m-k)\})$.

o. (ℓ_1 and ℓ_2 are not parts of the simple polygon, but they are auxiliary line segments to locate the points.) We put $m-k$ sites on ℓ_1 such that the sites are sufficiently close to each other. Similarly, we put k sites on ℓ_2 such that sites are sufficiently close to each other and $\|v_{m-k} - o\| \ll \|u_1 - o\|$, where v_i's are sites on ℓ_1 sorted along ℓ_1 from o and u'_j's are sites on ℓ_2 sorted along ℓ_2 from o, for $i = 1, \ldots, m-k$ and $j = 1, \ldots, k$.

As a part of the boundary of P, we put a concave and y-monotone polygonal curve with $n-4$ vertices below ℓ_2 such that the highest point of the curve is sufficiently close to u_1. Then we put another four vertices of P sufficiently far from the sites such that P contains all sites and there is no simple polygon containing more Voronoi vertices than P contains.

In the full version, we show that k-VD of this example has complexity of $\Omega(k(m-k) + \min\{nk, n(m-k)\})$. Moreover, we show that k-VD for any simple polygon and any point set has complexity of $O(k(m-k) + \min\{nk, n(m-k)\})$.

▶ **Lemma 18.** *The geodesic order-k Voronoi diagram of m points inside a simple polygon with n vertices has complexity of* $\Theta(k(m-k) + \min\{nk, n(m-k)\})$.

6.2 Computing the Topological Structure of the Diagram

Due to lack of space, we present only an outline of our algorithm. For details, refer to the full version. The algorithm is similar to the one in Section 5. Recall that for a site $s \in P(x)$, the boundary of $R_s(x)$ consists of one polygonal chain of ∂P and a simple curve with endpoints on ∂P. We call the simple curve the *wave-curve* for s. The wave-curve for s is monotone with respect to $\pi(o, x)$.

We say that a point $p \in P(x)$ lies *above* a curve if $\pi(p, \pi(o, x))$ intersects the curve. We use $L_i(x)$ to denote the region of $P(x)$ consisting of all points lying above at least i wave-curves for sites contained in $P(x)$. The *ith-level* of the arrangement of wave-curves of sites contained in $P(x)$ is defined to be the boundary of $L_i(x)$ excluding ∂P. The ith-level for each i is weakly monotone with respect to $\pi(o, x)$ and consists of at most m simple curves with endpoints lying on ∂P.

k-VD$[S]$ restricted to $L_k(x)$ coincides with k-VD$[S \cap P(x)]$ restricted to $L_k(x)$. Therefore, we can obtain the topological structure of k-VD$[S]$ by maintaining the topological structure of the kth-level of the arrangement of wave-curves. In addition to this, we maintain the topological structures of the ith-levels of the arrangement for all $i < k$ to detect the changes to the topological structure of the kth-level.

There are four types of events; site events, circle events, vanishing events and merging events. The definitions of the event types are analogous to the ones in Section 5. We can handle the site and circle events in a way similar to the one for the Euclidean order-k Voronoi diagram given by Zavershynskyi and Papadopoulou [17]. We can handle the other events in a way similar to the one in Section 5.

▶ **Theorem 19.** *Given a set of m points contained in a simple polygon with n vertices, we can compute the geodesic order-k Voronoi of the points in $O(k^2 m \log m \log^2 n + \min\{nk, n(m-k)\})$ time using $O(n + km)$ space.*

7 The Geodesic Farthest-Point Voronoi Diagram

We present an outline of our algorithm for computing the topological structure of FVD. For details, refer to the full version. Aronov et al. [3] showed that FVD forms a tree whose root is the geodesic center c of the sites. They first compute c and the geodesic convex hull of the sites. Then they compute FVD restricted to the boundary of the polygon. Then they compute the edges of FVD towards the geodesic center by a *reverse geodesic sweep method*. We follow the framework of the algorithm by Aronov et al. [3], but we can achieve a faster algorithm for $m \leq n/\log^2 n$ by applying their approach to compute the topological structure of FVD.

In the reverse geodesic sweep, a *sweep line* is a simple curve consisting of points equidistant from c. Let B be the (circular) sequence of the sites that have their Voronoi cells on the sweep line in clockwise order. As an exception, in the initial state, we set B to be the sequence of the sites whose Voronoi cells appear on ∂P.

No site, vanishing, or merging event occurs during the sweep because FVD forms a tree. So we handle only circle events. Every triple of consecutive sites in B defines a circle event. The key defined by a triple (s_1, s_2, s_3) is $d(c, c')$ for the point c' equidistant from s_1, s_2 and s_3. We can compute the key of each triple of consecutive sites in B in $O(\log^2 n)$ time.

▶ **Lemma 20.** *We can compute the topological structure of FVD in $O(m \log m \log^2 n)$ or $O(n + m \log m + m \log^2 n)$ time after computing the shortest path data structure for P.*

▶ **Theorem 21.** *Given a set of m points contained in a simple polygon with n vertices, we can compute the geodesic farthest-point Voronoi diagram of the points in $O(n + m \log m + m \log^2 n)$ time using $O(n + m)$ space.*

8 Dynamic Data Structures for Nearest or Farthest Point Queries

We showed that the topological structure of a Voronoi diagram can be computed without considering the whole polygon. The topological structure can be used for data structures for answering nearest or farthest point queries for dynamic point sets.

We apply the framework given by Bentley and Saxe [4]. At all times, we maintain $O(\sqrt{m})$ disjoint sets each of which consists of $O(\sqrt{m})$ points. For each set, we compute the topological structure of the nearest-point (or farthest-point) Voronoi diagram of the points in the set.

For a nearest (or farthest) point query, we simply find the Voronoi cells containing the query point for $O(\sqrt{m})$ Voronoi diagrams. Then we have $O(\sqrt{m})$ candidates for the nearest (or farthest) point from the query point. We find the nearest (or farthest) point from the query point directly among them. Inside a simple polygon, the complexity of each complete Voronoi diagram is $O(n + \sqrt{m})$. Thus each update takes $O(n + \sqrt{m}\,\mathrm{polylog}\{n, m\})$ time if we maintain the complete Voronoi diagrams. This is why we compute the topological structures of the Voronoi diagrams.

In the full version of this paper, we show how to find the Voronoi cell containing a query point using the topological structure of a Voronoi diagram. The key idea is to approximate the Voronoi diagram into a polygonal subdivision of complexity $O(\sqrt{m})$ using its adjacency graph and the exact positions of degree-1 and degree-3 vertices.

▶ **Theorem 22.** *We can construct a data structure of size $O(n+m)$ that supports a nearest-point (or a farthest) query for point insertions and deletions. Each query time is $O(\sqrt{m}\log(n+m))$ and each update time is $O(\sqrt{m}\log m\log^2 n))$.*

References

1. Hee-Kap Ahn, Luis Barba, Prosenjit Bose, Jean-Lou De Carufel, Matias Korman, and Eunjin Oh. A linear-time algorithm for the geodesic center of a simple polygon. *Discrete & Computational Geometry*, 56(4):836–859, 2016.
2. Boris Aronov. On the geodesic Voronoi diagram of point sites in a simple polygon. *Algorithmica*, 4(1):109–140, 1989.
3. Boris Aronov, Steven Fortune, and Gordon Wilfong. The furthest-site geodesic Voronoi diagram. *Discrete & Computational Geometry*, 9(1):217–255, 1993.
4. Jon Louis Bentley and James B. Saxe. Decomposable searching problems 1: Static-to-dynamic transformations. *Journal of Algorithms*, 1(4):297–396, 1980.
5. Bernard Chazelle, Herbert Edelsbrunner, Michelangelo Grigni, Leonidas Guibas, John Hershberger, Micha Sharir, and Jack Snoeyink. Ray shooting in polygons using geodesic triangulations. *Algorithmica*, 12(1):54–68, 1994.
6. Herbert Edelsbrunner and Ernst Peter Mücke. Simulation of simplicity: A technique to cope with degenerate cases in geometric algorithms. *ACM Transactions on Graphics*, 9(1):66–104, 1990.
7. Steven Fortune. A sweepline algorithm for Voronoi diagrams. *Algorithmica*, 2(1):153–174, 1987.
8. Leonidas Guibas, John Hershberger, Daniel Leven, Micha Sharir, and Robert E. Tarjan. Linear-time algorithms for visibility and shortest path problems inside triangulated simple polygons. *Algorithmica*, 2(1):209–233, 1987.
9. Leonidas J. Guibas and John Hershberger. Optimal shortest path queries in a simple polygon. *Journal of Computer and System Sciences*, 39(2):126–152, 1989.
10. John Hershberger. A new data structure for shortest path queries in a simple polygon. *Information Processing Letters*, 38(5):231–235, 1991.
11. Chih-Hung Liu and D. T. Lee. Higher-order geodesic Voronoi diagrams in a polygonal domain with holes. In *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2013)*, pages 1633–1645, 2013.
12. Nimrod Megiddo. Linear-time algorithms for linear programming in $\mathbb{R}^3$ and related problems. *SIAM Journal on Computing*, 12(4):759–776, 1983.
13. Joseph S. B. Mitchell. Geometric shortest paths and network optimization. In *Handbook of Computational Geometry*, pages 633–701. Elsevier, 2000.
14. Eunjin Oh, Luis Barba, and Hee-Kap Ahn. The farthest-point geodesic voronoi diagram of points on the boundary of a simple polygon. In *Proceedings of the 32nd International Symposium on Computational Geometry (SoCG 2016)*, pages 56:1–56:15, 2016.
15. Evanthia Papadopoulou and D. T. Lee. A new approach for the geodesic Voronoi diagram of points in a simple polygon and other restricted polygonal domains. *Algorithmica*, 1998(4):319–352, 1998.
16. Richard Pollack, Micha Sharir, and Günter Rote. Computing the geodesic center of a simple polygon. *Discrete & Computational Geometry*, 4(6):611–626, 1989.
17. Maksym Zavershynskyi and Evanthia Papadopoulou. A sweepline algorithm for higher order voronoi diagrams. In *Proceedings of the 10th International Symposium on Voronoi Diagrams in Science and Engineering (ISVD 2013)*, pages 16–22, 2013.

A Quest to Unravel the Metric Structure Behind Perturbed Networks*†

Srinivasan Parthasarathy[1], David Sivakoff[2], Minghao Tian[3], and Yusu Wang[4]

1 **Computer Science and Engineering Department, The Ohio State University, Columbus, OH, USA**
srini@cse.ohio-state.edu
2 **Statistics and Mathematics Departments, The Ohio State University, Columbus, OH, USA**
dsivakoff@stat.osu.edu
3 **Computer Science and Engineering Department, The Ohio State University, Columbus, OH, USA**
tian.394@osu.edu
4 **Computer Science and Engineering Department, The Ohio State University, Columbus, OH, USA**
yusu@cse.ohio-state.edu

Abstract

Graphs and network data are ubiquitous across a wide spectrum of scientific and application domains. Often in practice, an input graph can be considered as an observed snapshot of a (potentially continuous) hidden domain or process. Subsequent analysis, processing, and inferences are then performed on this observed graph. In this paper we advocate the perspective that an observed graph is often a noisy version of some discretized 1-skeleton of a hidden domain, and specifically we will consider the following natural network model: We assume that there is a true graph G^* which is a certain proximity graph for points sampled from a hidden domain $\mathcal{X}$; while the observed graph G is an Erdös-Rényi type perturbed version of G^*.

Our network model is related to, and slightly generalizes, the much-celebrated small-world network model originally proposed by Watts and Strogatz. However, the main question we aim to answer is orthogonal to the usual studies of network models (which often focuses on characterizing / predicting behaviors and properties of real-world networks). Specifically, we aim to recover the metric structure of G^* (which reflects that of the hidden space $\mathcal{X}$ as we will show) from the observed graph G. Our main result is that a simple filtering process based on the *Jaccard index* can recover this metric within a multiplicative factor of 2 under our network model. Our work makes one step towards the general question of inferring structure of a hidden space from its observed noisy graph representation. In addition, our results also provide a theoretical understanding for Jaccard-Index-based denoising approaches.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems, G.2.2 Graph Theory

Keywords and phrases metric structure, Erdös-Rényi perturbation, graphs, doubling measure

Digital Object Identifier 10.4230/LIPIcs.SoCG.2017.53

* A full version of the paper is available at https://arxiv.org/abs/1703.05475.

† This work is in part supported by National Science Foundation under grants IIS-1550757 and CCF-1618247. SP and DS would like to acknowledge NSF grant #DMS: 1418265 for partially supporting this work. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

1 Introduction

Graphs and networks are ubiquitous across a wide spectrum of scientific and application domains. Analyzing various types of graphs and network data play a fundamental role in modern data science. In the past several decades, there has been a large amount of research studying various aspects of graphs, ranging from developing efficient algorithms to process graphs, to information retrieval and inference based on graph data.

In many cases, we can view an input graph as an observed (discrete) 1-skeleton of a (potentially continuous) hidden domain. Subsequent analysis, processing, and inferences are then performed on this observed graph, with the ultimate goal being to understand the hidden space where the graph is sampled from. Many beautiful generative models for graphs have been proposed [9, 20], aiming to understand this transition process from a hidden space to the observed 1-skeleton, and to facilitate further tasks performed on graphs.

One line of such generative graph models assumes that an observed network is obtained by adding random perturbation to a specific type of underlying "structured graph" (such as a grid or a ring). For example, the much-celebrated small-world model by Watts and Strogatz [26] generates a graph by starting with a k-nearest neighbor graph spanned by nodes regularly distributed along a ring. It then randomly "rewires" some of the edges connecting neighboring points to instead connect nodes possibly far away. Watts and Strogatz showed that this simple model can generate networks that possess features of both a random graph and a proximity graph, and display two important characteristics often seen in real networks: low diameter in shortest path metric and high clustering coefficients. There have since been many variants of this model proposed so as to generate networks with different properties, such as adding random edges in a distance-dependent manner [23, 15], or extending similar ideas to incorporate hierarchical structures in networks; e.g, [16, 25]. There have also been numerous studies on characterizing statistical summaries, such as the average path lengths or the degree distributions, of small-world like networks; e.g [5, 11]; see [24, 6] for surveys.

Our work. In this paper, we take the perspective that an observed graph can be viewed as a noisy snapshot of the discretized 1-skeleton of a hidden domain of interest, and propose the following network model: Assume that the hidden space that generates data is a "nice" measure μ supported on a compact metric space $\mathcal{X} = (X, d_X)$ (e.g, the uniform measure supported on an embedded smooth low-dimensional Riemannian manifold). Suppose that the data points V are sampled i.i.d from this measure μ, and the "true graph" G_r^* connecting them is the r-neighborhood graph spanned by V (i.e, two points u, v are connected if their distance $d_X(u, v) \leq r$). The observed graph G however is only a noisy version of the true proximity graph G_r^*, and we model this noise by an Erdös-Rényi (ER) type perturbation – each edge in the true graph G_r^* can be deleted with probability p, while a "short-cut" edge between two unconnected nodes u, v could be inserted to G with probability q.

To motivate this model, imagine in a social network a person typically makes friends with other persons that are close to herself in the unknown feature space modeled by our metric space $\mathcal{X}$. The distribution of people (graph nodes) is captured by the measure μ on $\mathcal{X}$. However, there are always (or may be even many) exceptions – friends could be established by chance, and two seemingly similar persons (say, close geographically and in tastes) may not develop friendship. Thus it is reasonable to model an observed social network G as an ER-type perturbation of the proximity graph G_r^* to account for such exceptions.

The general question we hope to address is how to recover various properties of the hidden domain $\mathcal{X}$ from the observed graph G. In this paper we investigate a specific problem: how

to recover the metric structure of G_r^* (induced by the shortest path distances in G_r^*) from the noisy observation G. As we show in Theorem 5, the metric structure of G_r^* "approximates" that of the hidden domain $\mathcal{X}$. Note that a few inserted "short-cuts" could significantly change the shortest path metric, one potential factor leading to the small-world phenomenon. Our main result is that a simple filtering procedure based on the so-called *Jaccard index* can recover the shortest path metric of G_r^* within a multiplicative factor of 2 (with high probabilities). We also provide some preliminary experimental results.

Remarks and discussion. The problem of recovering G_r^* from the observed graph G is different and orthogonal to the usual studies on similar network models: Those studies often focus on characterizing the graphs generated by such models and whether those characteristics match with real networks. We instead aim to recover metric structure of a hidden true graph G_r^* from a given graph G. There are different motivations for this task. For example, it could be that the true graph G_r^* is the real object of interest, and we wish to "denoise" the observed graph G to get a more accurate representation of G_r^*. Indeed, in [12], Godberg and Roth empirically show how to use small-world model to help remove false edges in protein-protein interaction (PPI) networks. See [4] for more examples.

Furthermore, even if the observed graph G is of interest itself, we may still want to recover information about the domain $\mathcal{X}$ where G is generated from. For example, suppose we are given two networks G_1 and G_2 modeling say the collaboration networks from two different disciplines, and our goal is to compare the hidden collaboration structures behind the two disciplines. Comparing the precise graph structures of observed graphs G_1 and G_2 could be misleading, as even if they are generated from the same hidden space $\mathcal{X}$, they could still look different due to the random generation process. It is more robust if we can compare the two hidden spaces generating them instead.

Finally, we remark that similar to the small-world network models, our model also overlays a random perturbation over a "structured" network. Indeed, our network model in some sense generalizes the small-world network model by Watts and Strogaz. Specifically, in the model by Watts and Strogaz (and some later variants), the underlying "structured" network is a ring (or lattice). In our case, we assume that graph nodes P are sampled from a measure μ and using the r-neighborhood proximity graph G_r^* to model this underlying "structured" network. This setup adds generality to our model: For example, it allows us to produce non-uniform and more complex degree distributions than those previously produced by starting with lattice vertices. At the same time, by putting conditions on the measure μ, it still gives us sufficient structure to relate G_r^* and G, as we will show in this paper. We also point out that the theoretical results hold for graphs across a range of density, where the number of edges could range from $\Theta(n \log n)$ to $\Theta(n^2)$.

All missing proofs due to lack space can be found in the full version [19].

2 Model for Perturbed Network

We now introduce a general model to generate an observed network G. Suppose we are given a compact geodesic metric space $\mathcal{X} = (X, d_X)$ [1][7]. Intuitively, we view an observed graph $G = (V, E)$ as a noisy 1-skeleton of $\mathcal{X}$, where graph nodes V of G are sampled from this

[1] A geodesic metric space is a metric space where any two points in it are connected by a path whose length equals the distance between them. Riemannian manifolds or compact sets in the Euclidean space are all geodesic metric spaces.

hidden metric space. More precisely, we will assume that V is sampled i.i.d. from a measure $\mu : X \to \mathbb{R}^+$ supported on X.

▶ **Definition 1** (Measure)**.** Given a topological space X, a *measure* μ on X is simply a function that maps every Borel subset B of X to a non-negative number $\mu(B)$ which is additive: that is the measure of a countable family of pairwise-disjoint Borel subsets of X equals the sum of their respective measures.

In this paper, a measure is always a *probability measure*, meaning that $\mu(X) = 1$. To provide sufficient structure to the observed graph G so that it is not completely arbitrary, we want to assert some reasonable conditions on μ. To this end, we consider doubling measures:

▶ **Definition 2** (Doubling measure [13])**.** Given a metric space $\mathcal{X} = (X, d_X)$, let $\mathrm{B}(x, r) \subset X$ denotes the open metric ball $\mathrm{B}(x, r) = \{y \in X \mid d_X(x, y) < r\}$. A measure μ on $\mathcal{X}$ is said to be *doubling* if balls have finite and positive measure and there is a constant $L = L(\mu)$ s.t. for all $x \in X$ and any $r > 0$, we have $\mu(\mathrm{B}(x, 2r)) \leq L \cdot \mu(\mathrm{B}(x, r))$. We call L the *doubling constant* and say μ is an L*-doubling measure*.

These conditions on the measure also implies conditions on the underlying space X supporting the measure. Specifically, it is known that any metric space supporting a doubling measure has to be doubling as well, with its doubling constant depending on that of the measure [13].

Network model. We now describe our network model. Given a compact metric space $\mathcal{X} = (X, d_X)$ and an L-doubling measure $\mu : X \to \mathbb{R}^+$ supported on X, let V be a set of n points sampled i.i.d. from μ. We assume that the *true graph* $G^*_r = (V, E^*)$ is the r-neighborhood graph for some parameter $r > 0$; that is, $E(G^*) = E^* = \{(u, v) \mid d_X(u, v) \leq r, u, v \in V\}$.

▶ **Definition 3.** The *observed graph* $G(r, p, q) = (V, E)$ is based on $G^*_r = (V, E^*)$, but with the following two types of random perturbations:

- p-deletion: For each edge $(u, v) \in E^*$, (u, v) is in the observed graph $G(r, p, q)$ with probability $1 - p$ (that is, an edge in E^* is deleted with probability p).
- q-insertion: For any pair of nodes $u, v \in V$ s.t. $(u, v) \notin E^*$, we have that $(u, v) \in E$ with probability q.

Intuitively, in our model, the observed network G is a random geometric graph sampled from the metric space $\mathcal{X}$ which then undergoes Erdös-Rényi type perturbation. In what follows, we often omit the parameters r, p, q from the notations G^*_r and $G(r, p, q)$, when their choices are clear from the context. Note that both G^* and G are unweighted graphs (that is, all edges have weight 1). We now equip each graph with its shortest path metric, and obtain two discrete metric space (V, d_{G^*}) and (V, d_G) induced by G^* and G, respectively.

Problem statement and main results. Adding short-cuts (via q-insertions) could significantly distort the shortest path metric in G^*. Our ultimate goal is to infer information about both $\mathcal{X}$ and μ where points are sampled from, through the study of the observed graph G. In this paper we aim to recover the metric structure of G^* (as a reflection of metric structure of $\mathcal{X}$) from G. Specifically, we show that a simple filtering process based on the so-called Jaccard index can remove sufficient "bad edges" in G so as to recover the shortest path metric of G^* up to a factor of 2 w.h.p.

▶ **Definition 4** (Jaccard index)**.** Given an arbitrary graph G, let $N_G(u)$ denote the set of neighbors of u in G (i.e. nodes connected to $u \in V(G)$ by edges in $E(G)$). Given any edge

$(u, v) \in E(G)$, the *Jaccard index* $\rho_{u,v}$ of this edge is defined as

$$\rho_{u,v}(G) = \frac{|N_G(u) \cap N_G(v)|}{|N_G(u) \cup N_G(v)|}. \tag{1}$$

We remark that Jaccard index is a popular way to measure similarity between a pair of nodes connected by an edge in a graph [17], and has been commonly used in practice for denoising and sparsification purposes [22, 21]. Our results provide a theoretical understanding for such empirical Jaccard-based denoising approaches.

The main result is stated in Theorem 13. To show how this is established, we show two results on the influence of the shortest path under the p-deletion (Theorem 9) and under the q-insertion (Theorem 12), respectively. The proof for Theorem 13 combines the ideas for proofs of these two results.

Metric structures for G_r^* versus for $\mathcal{X}$. Our main results recover the shortest path metric for G_r^* approximately. In some sense, the metric of a proximity graph provides an approximation of that of X, the domain where input graph nodes are sampled from; see e.g, [1, 8] for the case where X is a smooth Riemannian manifold embedded in Euclidean space.

We make this relationship precise for our setting as follows. The proof of this result is rather standard (see e.g, the proof of Theorem 5.2 of [8]) and can be found in the full version [19].

▶ **Theorem 5.** *Let (X, d_X) be a compact geodesic metric space and μ a doubling measure supported on X. Let V_n be a set of n points sampled i.i.d. from μ, and G_r^* the r-neighborhood graph constructed on V_n (each edge in G_r^* has equal weight 1) with the associated shortest path metric $d_{G_r^*}$. For any sample V_n, consider the distance between $r \cdot d_{G_r^*}$ ($d_{G_r^*}$ scaled by r) and d_X restricted to the sample V_n; that is,*

$$\|r \cdot d_{G_r^*} - d_X|_{V_n}\|_\infty := \max_{v,v' \in V_n} |r \cdot d_{G_r^*}(v, v') - d_X(v, v')|.$$

Then we have that for a fixed r, $\limsup_{n\to\infty} \|r \cdot d_{G_r^*} - d_X|_{V_n}\|_\infty \le r$ almost surely.

3 Recovering the shortest path metric of G^*

To illustrate the main idea, we first consider the deletion-only and insertion-only perturbation of the true graph G^* in Sections 3.1 and 3.2, respectively. As we will see below, the main difficulty lies in handling insertions (short-cuts). We then combine the two cases and present our main result, Theorem 13. First, we describe one (natural) assumption on r that we will use later in all our statements.

Note that as r tends to 0, the corresponding r-neighborhood graph may be very sparse, and a sparse graph G_r^* is quite sensitive to random deletions and insertions. We would like to consider r in a range where is meaningful. We make the following assumption, asserting a lower-bound on the mass contained inside any metric ball of radius $r/2$:

[Assumption-R]: The parameter r is large enough such that for any $x \in X$, $\mu(\mathrm{B}(x, \frac{r}{2})) \ge \mathrm{s}$ where s satisfies $\mathrm{s} \ge \frac{12 \ln n}{n-2} (= \Omega(\frac{\ln n}{n}))$.

Intuitively, r is large enough such that with high probability each vertex v in G_r^* has degree $\Omega(\ln n)$. Note that requiring r to be large enough to have an $\Omega(\ln n/n)$ lower bound on the measure of any metric ball is natural. For example, for a random geometric graph $G(r, n)$ constructed as the r-neighborhood graph for points i.i.d. sampled from a uniform

measure on a Euclidean cube, asymptotically this is the same requirement so as to make sure that the resulting r-neighborhood graph is connected with high probability [20].

The proof of the following observation is simple and can be found in [19].

▶ **Lemma 6.** *Under Assumption-R, with probability at least $1-n^{-5/3}$, all vertices in G_r^* have more than $\frac{s(n-1)}{3} > 4\ln n$ neighbors.*

Since μ is a doubling measure, any two neighbors (u,v) in the r-neighborhood graph G_r^* would share many neighbors. Specifically, if (u,v) is an edge in G_r^*, that is, $d_X(u,v) \leq r$, then $\mathrm{B}(u,r) \cap \mathrm{B}(v,r)$ must contain a metric ball of radius $r/2$ (say centered at midpoint z of a shortest path connecting u to v in X; see Figure 1 (a)). Thus by a similar argument as the proof of Lemma 6, we obtain the following bound on the number of common neighbors between the nodes u, v if edge $(u,v) \in G_r^*$.

▶ **Corollary 7.** *Assume that the graph nodes V of G_r^* are sampled i.i.d from an L-doubling measure μ supported on a compact geodesic metric space (X, d_X). Then under Assumption-R, with probability at least $1-n^{-2/3}$, any two neighbors $(u,v) \in G_r^*$ have $\frac{s(n-1)}{3} > 4\ln n = \Omega(\ln n)$ number of common neighbors.*

3.1 Deletion only

In this case, we assume that we remove each edge in G^* independently with probability p to obtain an observed empirical graph $\widehat{G}$. Our goal is to relate the shortest path metrics d_{G^*} of G^* and $d_{\widehat{G}}$ of $\widehat{G}$ respectively. Deletion-only means that shortest path distances in $\widehat{G}$ are larger than those in G^*. Since any two nodes u, v connected in G^* share sufficient number ($\Omega(\ln n)$) of common neighbors, intuitively, removing even a constant fraction of edges in G^* can still guarantee that w.h.p. u and v will still have some common neighbors left, and thus u and v can be connected through that common neighbor by a path of length 2 in $\widehat{G}$. Hence overall, w.h.p. the distortion in shortest path distance is at most by a factor of 2.

▶ **Definition 8.** Let G and G' be two graphs spanned on the same set of nodes V, and equipped with graph shortest path metric d_G and $d_{G'}$, respectively. By $d_G \leq c d_{G'}$, we mean that for any two nodes $u, v \in V$, we have that $d_G(u,v) \leq c d_{G'}(u,v)$. We say that $d_{G'}$ is *a c-approximation of d_G* if $\frac{1}{c} d_G \leq d_{G'} \leq c d_G$.

▶ **Theorem 9** (Random deletion)**.** *Let V be n points sampled i.i.d. from a probability measure $\mu : X \to \mathbb{R}^+$ supported on a compact metric space (X, d_X). Let G^* be the r-neighborhood graph for V; and $\widehat{G}$ a graph obtained by removing each edge in G^* independently with probability p. Under Assumption-R and for $p < \frac{1}{2} e^{-\frac{9\ln n}{s(n-1)}}$, we have with probability at least $1 - \frac{1}{n^{\Omega(1)}}$, the shortest path metric $d_{\widehat{G}}$ is a 2-approximation of the shortest path metric d_{G^*}.*

Specifically, since $s > \frac{12\ln n}{n-1}$, the statement holds for $p < \frac{1}{2e^{3/4}}$. As s becomes larger, the upper bound on p gets closer to $1/2$.

Proof. For a node $u \in V$, let $N_{G^*}(u)$ and $N_{\widehat{G}}(u)$ denote the set of neighbors of u in graph G^* and graph $\widehat{G}$, respectively.

Since deletion cannot decrease the length of shortest paths, we have $d_{G^*} \leq d_{\widehat{G}}$. We now show that $d_{\widehat{G}} \leq 2 d_{G^*}$.

Consider $(u,v) \in E(G^*)$: Assume that they share $k_{u,v}$ number of common neighbors; that is, $k_{u,v} = |N_{G^*}(u) \cap N_{G^*}(v)|$. The probability that $N_{\widehat{G}}(u) \cap N_{\widehat{G}}(v) = \emptyset$ (i.e, u and v have no common neighbor in graph $\widehat{G}$) is thus $(2p)^{k_{u,v}}$.

On the other hand, by Corollary 7, with probability at least $1 - n^{-2/3}$ we have that $k_{u,v} \geq s(n-1)/3$ for all $(u,v) \in E(G^*)$. By applying the law of total probability, it then follows that the probability that there exists any $(u,v) \in E(G^*)$ with $N_{\widehat{G}}(u) \cap N_{\widehat{G}}(v) = \emptyset$ is at most: $n^{-2/3} + n^2(2p)^{s(n-1)/3} < n^{-2/3} + n^2(e^{-3\ln n}) < n^{-1/3}$, where we plug in the bound on p to derive the first inequality.

Hence with probability at least $1 - n^{-1/3}$, we have that for all edges $(u,v) \in E(G^*)$, their distance in $\widehat{G}$ satisfies $d_{\widehat{G}}(u,v) \leq 2$ (via one of their common neighbor in $N_{\widehat{G}}(u) \cap N_{\widehat{G}}(v)$). This in turn implies that with probability at least $1 - n^{-1/3}$, for any path $\pi = \langle v_1, \ldots, v_m \rangle$ in G^* with length m, we can find a path of length at most $2m$ in $\widehat{G}$ to connect v_1 to v_m (as each edge (v_i, v_{i+1}) in π corresponds to a path of length at most 2 in $\widehat{G}$). If u and v are disconnected in G^*, then obviously they are still disconnected in $\widehat{G}$. Hence, for any two $u, v \in V$, $d_{\widehat{G}}(u,v) \leq 2d_{G^*}(u,v)$, and the theorem follows. ◀

3.2 Insertion only

Now assume that the observed graph $\widehat{G}$ is generated from the true graph G^* where all edges in G^* also exist in $\widehat{G}$, and for any $u, v \in V$ with $(u,v) \neq E(G^*)$, we have $(u,v) \in E(\widehat{G})$ with probability q. In this case, the shortest path metric can be significantly altered in $d_{\widehat{G}}$. Hence to recover the metric d_{G^*}, instead of operating on $\widehat{G}$ directly, we will construct another graph $\tilde{G}$ from $\widehat{G}$, so that its shortest path metric $d_{\tilde{G}}$ approximates d_{G^*}.

We propose the following Jaccard-Index-based filtering process, which we call a τ-Jaccard filtering, as it uses a parameter τ. (Recall the definition of Jaccard index in Def. 4). We represent the output filtered (denoised) graph as $\tilde{G}_\tau$:

τ-Jaccard filtering: Given graph $\widehat{G}$, for each edge $(u,v) \in E(\widehat{G})$, we insert the edge (u,v) into $E(\tilde{G}_\tau)$ if and only if $\rho_{u,v}(\widehat{G}) \geq \tau$. That is, $V(\tilde{G}_\tau) = V(\widehat{G})$ and $E(\tilde{G}_\tau) := \{(u,v) \in E(\widehat{G}) \mid \rho_{u,v}(\widehat{G}) \geq \tau\}$.

Below we first show that w.h.p., all "good" edges in the true r-neighborhood graph G^* will have a large Jaccard index, so that they will be kept in $\tilde{G}_\tau$ after a τ-Jaccard filtering procedure with appropriate τ.

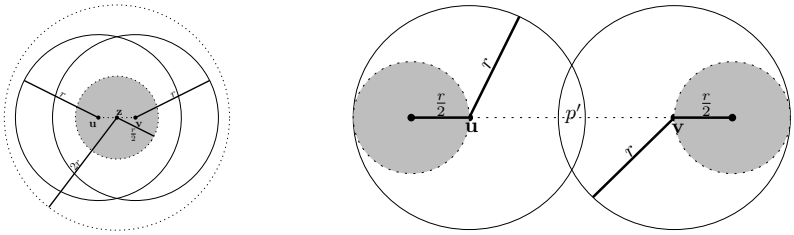
▶ **Lemma 10.** *Let V be a set of n points sampled i.i.d. from an L-doubling probability measure μ supported on a compact geodesic metric space $\mathcal{X} = (X, d_X)$. If Assumption-R holds and $q \leq cs$, then for $\forall \tau \leq \frac{1}{(6+\frac{1}{\ln n}+12c)L^2}$, we have with probability at least $1 - n^{-2/3}$, that $\rho_{u,v}(\widehat{G}) \geq \tau$ for* all *pairs of nodes $u, v \in V$ with $d_X(u,v) \leq r$.*

For example, if $c = \frac{1}{2}$ (i.e, $q \leq \frac{s}{2}$), then the bound on $\rho_{u,v}$ holds for $\tau \leq \frac{1}{13L^2}$. Note c may not be a constant and can depend on n; as c increases, the upper bound on τ decreases.

Proof. Consider a fixed pair of nodes $u, v \in V$, and let $F = F(u,v)$ be the event that $d_X(u,v) \leq r$. Set $\alpha_* = |N_{G^*}(u) \cap N_{G^*}(v)|$ to be the number of common neighbors of u and v in G^*. Let $\beta = |N_{\widehat{G}}(u) \cup N_{\widehat{G}}(v)|$ denote the total number of neighbors of u and v in the perturbed graph $\widehat{G}$.

Since $\widehat{G}$ can have only more edges than G^*, $|N_{\widehat{G}}(u) \cap N_{\widehat{G}}(v)| \geq |N_{G^*}(u) \cap N_{G^*}(v)| = \alpha_*$ and thus $\rho_{u,v}(\widehat{G}) \geq \frac{\alpha_*}{\beta}$. In what follows, we prove that $\frac{\alpha_*}{\beta} \geq \tau \cdot I_F$ (which implies that $\rho_{u,v}(\widehat{G}) \geq \tau \cdot I_F$) with probability at least $1 - 2n^{-8/3}$. (Here, we use I_A to denote the indicator random variable of the event A, and the conventions that $\rho_{u,v}(\widehat{G}) = 0$ if $(u,v) \notin \widehat{G}$ and $0/0 = 0$.)

Note that α_* is a random variable, which equals the number of (i.i.d. sampled) points from $V - \{u,v\}$ that fall in the region $\mathrm{B}(u,r) \cap \mathrm{B}(v,r)$. That is, conditional on u and v, α_*



(a) (b)

Figure 1 In these figures, we draw metric balls as Euclidean balls just for illustration purpose. (a) illustrates the bound $p_{\alpha*} \geq \mu(\mathrm{B}(z, r/2))$ which follows from $\mathrm{B}(z, r/2) \subseteq \mathrm{B}(u, r) \cap \mathrm{B}(v, r)$. (b) Key observation for Lemma 11: as $d_X(u, v) > r$, we have that the region $[B(u, r) \cup B(v, r)] \setminus [B(u, r) \cap B(v, r)]$ contains at least two metric balls, each of radius $r/2$.

is drawn from a binomial distribution $Bin(n-2, p_{\alpha_*})$ with $p_{\alpha_*} = \mu(\mathrm{B}(u, r) \cap \mathrm{B}(v, r))$, and the conditional expectation of α_* given u and v is $\delta_{\alpha_*} = (n-2) \cdot p_{\alpha_*}$.

Now observe that, conditional on u and v, the random variable $\beta - 2$ (see footnote[2]) has distribution $Bin(n-2, p_\beta)$ with $p_\beta = p_{\beta_*} + (1-p_{\beta_*})(2q-q^2)$, where $p_{\beta_*} = \mu(\mathrm{B}(u, r) \cup \mathrm{B}(v, r))$. Indeed, observe that, conditional on u and v, points contributing to β can be generated as follows. Let $U = \mathrm{B}(u, r) \cup \mathrm{B}(v, r)$. Independently, for each $i = 1, \ldots, n-2$, we draw a point x_i randomly from μ and we also perform an independent coin flip for this point, with probability of heads equal to $1 - (1-q)^2 = 2q - q^2$. This quantity is the probability for a point *outside* U to be connected to either u or v under edge-insertion probability q. We set the indicator variable $y_i = 1$ iff either $x_i \in U$, or $x_i \notin U$ and the i^{th} coin flip is heads. Conditional on u and v, the resulting $n-2$ indicator random variables $y_1, \ldots, y_{n-2}$ are i.i.d. with $\mathrm{P}[y_i = 1 \mid u, v] = p_{\beta_*} + (1-p_{\beta_*})(2q-q^2) = p_\beta$. Therefore, given u and v, the distribution of $\beta - 2 = \sum y_i$ is $Bin(n-2, p_\beta)$. The conditional expectation of β given u and v, denoted δ_β, satisfies

$$(n-2) \cdot p_{\beta_*} \leq \delta_\beta = (n-2) \cdot p_\beta + 2 \leq (n-2) \cdot p_{\beta_*} + (n-2) \cdot 2q + 2. \tag{2}$$

Let us for now assume that $\frac{c_1 \delta_{\alpha*}}{c_2 \delta_\beta} \geq \tau I_F$ a.s. for constants $c_1 = 1 - \sigma_1$ and $c_2 = 1 + \sigma_2$ with $0 < \sigma_1 < 1$ and $0 < \sigma_2$ to be set shortly.

If $d_X(u, v) \leq r$, then $\mathrm{B}(u, r) \cap \mathrm{B}(v, r)$ contains at least one metric ball of radius $r/2$ (say $\mathrm{B}(z, r/2)$ with z being the mid-point of a shortest path between u and v in $\mathcal{X}$; see Figure 1 (a)).

Hence by Assumption-R, on the event $d_X(u, v) \leq r$, we have

$$\delta_{\alpha_*} \geq (n-2) \cdot \mu(\mathrm{B}(z, r/2)) \geq (n-2) \cdot \frac{12 \ln n}{n-2} = 12 \ln n.$$

Similarly, using (2), the conditional expectation of β satisfies

$$\delta_\beta \geq (n-2) \cdot p_{\beta_*} \geq (n-2) \cdot \mu(\mathrm{B}(u, r)) \geq 12 \ln n. \tag{3}$$

We now set $\sigma_1 = 2/3$ and $\sigma_2 = 1$. It then follows from Chernoff bounds that

$$\mathrm{P}[\alpha_* < c_1 \delta_{\alpha_*} \mid u, v, F] + \mathrm{P}[\beta > c_2 \delta_\beta \mid u, v] \leq e^{-\frac{\sigma_1^2}{2} \delta_{\alpha_*}} + e^{-\frac{\sigma_2}{3} \delta_\beta} \leq n^{-\frac{8}{3}} + n^{-4}.$$

[2] The subtraction of 2 in $\beta - 2$ accounts for points u and v, which are in $N_{\widehat{G}}(u) \cup N_{\widehat{G}}(v)$. Similarly, in the binomial distribution we will have only $n-2$, accounting for points in $V - \{u, v\}$.

Taking expectation of the above with respect to u and v gives

$$\mathrm{P}[\alpha_* < c_1\delta_{\alpha_*} \mid F] + \mathrm{P}[\beta > c_2\delta_\beta] \leq 2n^{-\frac{8}{3}}. \tag{4}$$

On the other hand, since $\frac{\alpha_*}{\beta} \geq 0$, we have

$$\begin{aligned}\mathrm{P}[\frac{\alpha_*}{\beta} < \tau I_F] \leq \mathrm{P}[\frac{\alpha_*}{\beta} < \tau \mid (\alpha_* \geq c_1\delta_{\alpha_*}) \wedge (\beta \leq c_2\delta_\beta) \wedge F] \\ + \mathrm{P}[(\{\alpha_* < c_1\delta_{\alpha_*}\} \vee \{\beta > c_2\delta_\beta\}) \wedge F].\end{aligned} \tag{5}$$

Since we assumed that $\frac{c_1\delta_{\alpha_*}}{c_2\delta_\beta} \geq \tau I_F$, if $\alpha_* \geq c_1\delta_{\alpha_*}$ and $\beta \leq c_2\delta_\beta$ and $d_X(u,v) \leq r$, then we have $\frac{\alpha_*}{\beta} \geq \frac{c_1\delta_{\alpha_*}}{c_2\delta_\beta} \geq \tau$. This means that

$$\mathrm{P}[\frac{\alpha_*}{\beta} < \tau \mid (\alpha_* \geq c_1\delta_{\alpha_*}) \wedge (\beta \leq c_2\delta_\beta) \wedge F] = 0.$$

Hence the first term in the right-hand side of (5) is 0. Together with (4), and recalling $\rho_{u,v}(\widehat{G}) \geq \frac{\alpha_*}{\beta}$, we have

$$\mathrm{P}[\rho_{u,v}(\widehat{G}) < \tau I_F] \leq \mathrm{P}[\frac{\alpha_*}{\beta} < \tau I_F] \leq 2n^{-\frac{8}{3}}.$$

By the union bound, the probability that $\rho_{u,v}(\widehat{G}) \geq \tau$ for all pairs of nodes $u, v \in V$ such that $d_X(u,v) \leq r$ is thus at least $1 - \frac{1}{2}n^2(2n^{-\frac{8}{3}}) = 1 - n^{-\frac{2}{3}}$.

Finally, we need to verify that $\frac{c_1\delta_{\alpha_*}}{c_2\delta_\beta} = \frac{\delta_{\alpha_*}}{6\delta_\beta} \geq \tau I_F$ holds for a.e. u and v. This holds automatically if $d_X(u,v) > r$, so assume $d_X(u,v) \leq r$. Recall that $\delta_\beta \leq (n-2)\cdot p_{\beta_*} + (n-2)\cdot 2q + 2$ by (2). Since $q \leq c\mathrm{s}$, we have $(n-2)2q \leq 2(n-2)c\mathrm{s}$. On the other hand, by Assumption-R, $p_{\beta_*} \geq \mu(\mathrm{B}(u,r)) \geq \mathrm{s}$, hence $2(n-2)q \leq 2(n-2)c\cdot p_{\beta_*}$. Combining this with the fact that $(n-2)p_{\beta_*} \geq 12\ln n$ from (3) (which also implies that $2 \leq \frac{(n-2)p_{\beta_*}}{6\ln n}$), it then follows that

$$\frac{\delta_{\alpha_*}}{6\delta_\beta} \geq \frac{\delta_{\alpha_*}}{6((n-2)(1+\frac{1}{6\ln n})p_{\beta_*} + 2(n-2)c\cdot p_{\beta_*})} = \frac{p_{\alpha_*}}{p_{\beta_*}} \cdot \frac{1}{6 + \frac{1}{\ln n} + 12c}. \tag{6}$$

Now let z be the midpoint of a geodesic connecting u and v; see Figure 1 (a). Observe that $p_{\alpha_*} \geq \mu(\mathrm{B}(z, r/2))$, $p_{\beta_*} \leq \mu(\mathrm{B}(z,2r))$ and since μ is L-doubling, we have:

$$p_{\beta_*} \leq \mu(\mathrm{B}(z,2r)) \leq L\mu(\mathrm{B}(z,r)) \leq L^2\mu(\mathrm{B}(z,r/2)) \leq L^2 p_{\alpha_*}. \tag{7}$$

Combining equations (6) and (7), we have that if $\tau \leq \frac{1}{(6+\frac{1}{\ln n}+12c)L^2}$, then $\frac{\delta_{\alpha_*}}{6\delta_\beta} \geq \tau$ is satisfied. This proves the lemma. ◀

Discussion on the bounds of parameters. Lemma 10 implies that, with high probability, we will not remove any good edges if the doubling constant L of the measure is at most $O(\frac{1}{\sqrt{\tau}})$ and the insertion probability is small ($q \leq cs$). The requirement that $L = O(\frac{1}{\sqrt{\tau}})$ is rather mild; we now inspect the requirement $q \leq cs$: Since sn lower-bounds the degree of a node in the true graph G^* (by Lemma 6), it is reasonable that the insertion probability q is required to be small compared to s; as otherwise, the "noise" (inserted edges) will overwhelm the signal (original edges). Furthermore, it is important to note that c is not necessarily a constant – it can depend on n, but as c increases, the upper bound of the admissible range for parameter τ decreases.

The following result complements Lemma 10 by stating that for insertion probability $q \le cs$, all "really bad" edges in $\widehat{G}$ will have small Jaccard index, and thus will be removed by our τ-filtering process.

In particular, we define an edge $(u,v) \in E(\widehat{G}) \setminus E(G^*)$ in the observed graph $\widehat{G}$ to be *really-bad* if $N_{G^*}(u) \cap N_{G^*}(v) = \emptyset$. Note that $(u,v) \notin E(G^*)$ is equivalent to $d_X(u,v) > r$.

▶ **Lemma 11.** *Let V be a set of n points sampled i.i.d. from an L-doubling probability measure μ supported on a compact geodesic metric space $\mathcal{X} = (X, d_X)$. If Assumption-R holds and $q \le cs$, then $\forall \tau \ge (c+2)q + 2(c+2)\sqrt{\frac{\ln n}{s(n-2)}}$, we have with probability at least $1 - n^{-2}$, $\rho_{u,v}(\widehat{G}) < \tau$ for* all *pairs of nodes $u, v \in V$ such that (u,v) is really-bad.*

For example, if $c = 1$ and $s \cdot n = \omega(\ln n)$, then the condition on τ is that $\tau \ge 3q + o(1)$.

Proof. Consider a fixed pair of nodes $u, v \in V$, and let $F = F(u,v)$ be the event that $N_{G^*}(u) \cap N_{G^*}(v) = \emptyset$ and $d_X(u,v) > r$. Let $\alpha = |N_{\widehat{G}}(u) \cap N_{\widehat{G}}(v)|$,

$$\alpha_I = \left|\{x \in N_{G^*}(u) \cup N_{G^*}(v) \,:\, x \text{ is connected to both } u \text{ and } v \text{ in } \widehat{G}\}\right|, \text{ and}$$
$$\alpha_o = \left|\{x \notin N_{G^*}(u) \cup N_{G^*}(v) \,:\, x \text{ is connected to both } u \text{ and } v \text{ in } \widehat{G}\}\right|.$$

Then we have $\alpha = \alpha_I + \alpha_o$. Set $\beta_* = |N_{G^*}(u) \cup N_{G^*}(v)|$, so we have $|N_{\widehat{G}}(u) \cup N_{\widehat{G}}(v)| \ge \beta_* + \alpha_o =: \beta$. It is easy to see that

$$\rho_{u,v}(\widehat{G}) = \frac{\alpha}{|N_{\widehat{G}}(u) \cup N_{\widehat{G}}(v)|} \le \frac{\alpha}{\beta_* + \alpha_o} = \frac{\alpha}{\beta}.$$

We aim to show that with very high probability $\frac{\alpha}{\beta} I_F < \tau$, which implies that $\rho_{u,v}(\widehat{G}) I_F < \tau$.

First, we claim that, conditional on the locations of u and v and the event F, the distribution of α is $Bin(n-2, p_\alpha)$ with $p_\alpha = \frac{p_{\beta_*} - p'}{1-p'} q + \frac{1-p_{\beta_*}}{1-p'} q^2$, where $p_{\beta_*} = \mu(B(u,r) \cup B(v,r))$ and $p' = \mu(B(u,r) \cap B(v,r))$. We also claim that the conditional distribution of β given u, v and F is $Bin(n-2, p_\beta)$ with $p_\beta = \frac{p_{\beta_*} - p'}{1-p'} + \frac{1-p_{\beta_*}}{1-p'} q^2$. Details in [19].

If $d_X(u,v) > r$, the region $[B(u,r) \cup B(v,r)] \setminus [B(u,r) \cap B(v,r)]$ contains at least two disjoint metric balls of radius $r/2$; see Figure 1(b). Therefore, $p_{\beta_*} - p' \ge 2\mu(B(\frac{r}{2})) \ge 2s$. The conditional expectation of α given u, v and F, denoted by $\delta_\alpha (= (n-2)p_\alpha)$, satisfies:

$$(n-2)\frac{p_{\beta_*} - p'}{1-p'} q \le \delta_\alpha = (n-2)[\frac{p_{\beta_*} - p'}{1-p'} q + \frac{1-p_{\beta_*}}{1-p'} q^2] \le (1 + \frac{c}{2})(n-2)\frac{p_{\beta_*} - p'}{1-p'} q, \quad (8)$$

where the last inequality follows from $q \le cs \le c \cdot \frac{p_{\beta_*} - p'}{2}$. The conditional expectation of β given u, v and F, denoted δ_β, satisfies

$$\delta_\beta = (n-2)p_\beta \ge (n-2)\frac{p_{\beta_*} - p'}{1-p'}. \quad (9)$$

Let us now assume that $\frac{c_1 \delta_\alpha}{c_2 \delta_\beta} I_F \le \tau$ a.s. for $c_1 = 1 + \epsilon$ and some constant $c_2 = 1 - \sigma$ with $\epsilon = \frac{2}{q}\sqrt{\frac{\ln n}{s(n-2)}}$ and some $0 < \sigma < 1$ to be set later.

If $q \le 2\sqrt{\frac{\ln n}{s(n-2)}}$, then we have $\epsilon \ge 1$. In this case, combining Chernoff bounds with (8) and the fact that $p_{\beta_*} - p' \ge 2\mu(B(\frac{r}{2})) \ge 2s$ obtained earlier, we have

$$\begin{aligned} \mathrm{P}[\alpha \ge (1+\epsilon)\delta_\alpha \mid u, v, F] &\le e^{-\frac{\epsilon}{3}\delta_\alpha} = e^{-\frac{2}{3q}\sqrt{\frac{\ln n}{s(n-2)}}\delta_\alpha} \le e^{-\frac{2}{3q}\sqrt{\frac{\ln n}{s(n-2)}}(n-2)\frac{p_{\beta_*} - p'}{1-p'} q} \\ &\le e^{-\frac{4}{3}\sqrt{(n-2)(\ln n)s}} \le e^{-\frac{4}{3}\sqrt{(n-2)(\ln n)\frac{12\ln n}{n-2}}} \le n^{-4}. \end{aligned} \quad (10)$$

Otherwise, we have $q > 2\sqrt{\frac{\ln n}{s(n-2)}}$, so $0 < \epsilon < 1$. In this case, by Chernoff bounds

$$\begin{aligned}\mathrm{P}[\alpha \geq (1+\epsilon)\delta_\alpha \mid u, v, F] \leq e^{-\frac{1}{2}\epsilon^2\delta_\alpha} &\leq e^{-2\frac{\ln n}{s(n-2)}\frac{1}{q^2}(n-2)\frac{p_{\beta_*}-p'}{1-p'}q} \\ &= e^{-2\frac{\ln n(p_{\beta_*}-p')}{sq}} \leq e^{-2\ln n\cdot\frac{2s}{sq}} \leq n^{-4}. \end{aligned} \tag{11}$$

On the other hand, by Chernoff bounds, we have $\mathrm{P}[\beta \leq c_2\delta_\beta \mid u, v, F] \leq e^{-\frac{\sigma^2}{2}\delta_\beta}$. Note that $\delta_\beta \geq (n-2)\cdot\frac{p_{\beta_*}-p'}{1-p'} \geq (n-2)\cdot 2s \geq 24\ln n$. We now set $\sigma = 1/2$ so $c_2 = 1-\sigma = 1/2$. By taking expectation with respect to u and v, we have

$$\mathrm{P}[\alpha \geq c_1\delta_\alpha \mid F] + \mathrm{P}[\beta \leq c_2\delta_\beta \mid F] \leq 2n^{-4}. \tag{12}$$

Since $\tau > 0$, we have that

$$\mathrm{P}[\frac{\alpha}{\beta}I_F \geq \tau] \leq \mathrm{P}[\frac{\alpha}{\beta} \geq \tau \mid (\alpha < c_1\delta_\alpha) \wedge (\beta > c_2\delta_\beta) \wedge F]\ \mathrm{P}[\{(\alpha \geq c_1\delta_\alpha) \vee (\beta \leq c_2\delta_\beta)\} \wedge F]. \tag{13}$$

Under our assumption that $\frac{c_1\delta_\alpha}{c_2\delta_\beta}I_F \leq \tau$ a.s., if $\alpha < c_1\delta_\alpha$, $\beta > c_2\delta_\beta$ and $d_X(u,v) > r$, then $\frac{\alpha}{\beta} < \frac{c_1\delta_\alpha}{c_2\delta_B} \leq \tau$. Therefore, the first term on the right side of (13) is $\mathrm{P}[\frac{\alpha}{\beta} \geq \tau \mid (\alpha < c_1\delta_\alpha) \wedge (\beta > c_2\delta_\beta) \wedge F] = 0$. It then follows from (12) that:

$$\mathrm{P}[\frac{\alpha}{\beta}I_F \geq \tau] \leq \mathrm{P}[(\alpha \geq c_1\delta_\alpha) \vee (\beta \leq c_2\delta_\beta) \mid F] \leq 2n^{-4}$$

Since $\rho_{u,v}(\widehat{G}) \leq \frac{\alpha}{\beta}$, we have $\mathrm{P}[\rho_{u,v}(\widehat{G})I_F \geq \tau] \leq \mathrm{P}[\frac{\alpha}{\beta}I_F \geq \tau] \leq 2n^{-4}$. By union bound, the probability that $\rho_{u,v}(\widehat{G}) < \tau$ for all pairs of nodes $u, v \in V$ satisfying the required conditions is thus at least $1 - \frac{1}{2}n^2(2n^{-4}) = 1 - n^{-2}$.

Finally, for the above argument to hold, we need the assumption $\frac{c_1\delta_\alpha}{c_2\delta_\beta}I_F \leq \tau$ to be satisfied uniformly for all u and v. This comes from the choices and conditions of our parameters; see [19] for details. The lemma then follows. ◀

The above result implies that after Jaccard filtering, although there still may be some extra edges remaining in $\tilde{G}_\tau$, each such edge (u,v) is not really-bad. In fact, $N_{G^*}(u) \cap N_{G^*}(v) \neq \emptyset$ for each such extra remaining edge (u,v), implying that $d_{G^*}(u,v) \leq 2$. This, combined with Lemma 10, essentially leads to the following result. To simplify our statement, we assume $sn = \omega(\ln n)$ in the following result; a more complicated form can be obtained without this assumption (similar to the statement in Lemma 11).

▶ **Theorem 12** (Random Insertion). *Let V be a set of n points sampled i.i.d. from an L-doubling measure $\mu : X \to \mathbb{R}^+$ supported on a compact metric space (X, d_X). Let G^* be the resulting r-neighborhood graph for V; and $\widehat{G}$ a graph obtained by inserting each edge not in G^* independently with probability q. Let $\tilde{G}_\tau$ be the graph after τ-Jaccard filtering of $\widehat{G}$. Then, if Assumption-R holds, $q \leq cs$ and $sn = \omega(\ln n)$, then for $\forall \frac{1}{(6+\frac{1}{\ln n}12c)L^2} \geq \tau \geq (c+2)q + o(1)$, with high probability the shortest path distance metric $d_{\tilde{G}_\tau}$ satisfies: $\frac{1}{2}d_{G^*} \leq d_{\tilde{G}_\tau} \leq d_{G^*}$; that is, $d_{\tilde{G}_\tau}$ is a 2-approximation for d_{G^*} with high probability.*

Proof. Define $\mathcal{E}_1$ to be the event when all the edges in G^* are present in $\tilde{G}_\tau$. By Lemma 10, event $\mathcal{E}_1$ happens with probability at least $1 - n^{-2/3}$. Hence with at least this probability, $d_{\tilde{G}_\tau} \leq d_{G^*}$. We now prove the lower bound for $d_{\tilde{G}_\tau}$.

Let $\mathcal{E}_2$ be the event where for all edges $(u,v) \in E(\tilde{G}_\tau) \setminus E(G^*)$, (u,v) is not really-bad. Lemma 11 says that event $\mathcal{E}_2$ happens with probability at least $1 - n^{-2}$. To this end, observe that if an edge (u,v) is not really-bad, then we have that $d_{G^*}(u,v) \leq 2$ as $N_{G^*}(u) \cap N_{G^*}(v) \neq \emptyset$; specifically, there is a path $u \to w \to v$ connecting u and v through some $w \in N_{G^*}(u) \cap N_{G^*}(v)$.

In what follows, assume both events $\mathcal{E}_1$ and $\mathcal{E}_2$ happen – as discussed above, this assumption holds with high probability due to Lemmas 10 and 11.

Now consider two points $u, v \in V$. First, suppose that u, v are connected in $\tilde{G}_\tau$. Let $\pi = \langle u_0 = u, u_1, \ldots, u_s = v \rangle$ be a shortest path between them in $\tilde{G}_\tau$. Consider each edge (u_i, u_{i+1}) in the shortest path π in $\tilde{G}_\tau$. Either $(u_i, u_{i+1}) \in E(G^*)$, in which case we set $\hat{\pi}(u_i, u_{i+1}) = (u_i, u_{i+1})$. Otherwise if $(u_i, u_{i+1}) \notin E(G^*)$, then (u_i, u_{i+1}) is not really-bad due to event $\mathcal{E}_2$, meaning that $d_{G^*}(u_i, u_{i+1}) \leq 2$. Hence we can find a path $\hat{\pi}(u_i, u_{i+1}) \subset G^*$ of length at most two to connect u_i and u_{i+1} in G^*. Putting these two together, we can construct a path $\hat{\pi} = \hat{\pi}(u_0, u_1) \circ \hat{\pi}(u_1, u_2) \circ \cdots \circ \hat{\pi}(u_{s-1}, u_s)$ connecting $u = u_0$ to $v = u_s$ in G^*. Clearly, this path has length at most $2s$. Hence, for any $u, v \in V$, we have that $d_{G^*}(u,v) \leq 2 d_{\tilde{G}_\tau}(u,v)$ if (u,v) is connected in $\tilde{G}_\tau$.

If u and v are not connected in $\tilde{G}_\tau$, then they are not connected in G^* either; because if there is a path connecting them in G^*, then the same path is present in $\tilde{G}_\tau$ as event $\mathcal{E}_1$ holds. Putting everything together, we then have that with high probability, for any $u, v \in V$, $d_{G^*}(u,v) \leq 2 d_{\tilde{G}_\tau}(u,v)$; that is $d_{\tilde{G}_\tau} \geq \frac{1}{2} d_{G^*}$. The theorem then follows. ◀

4 Combined case

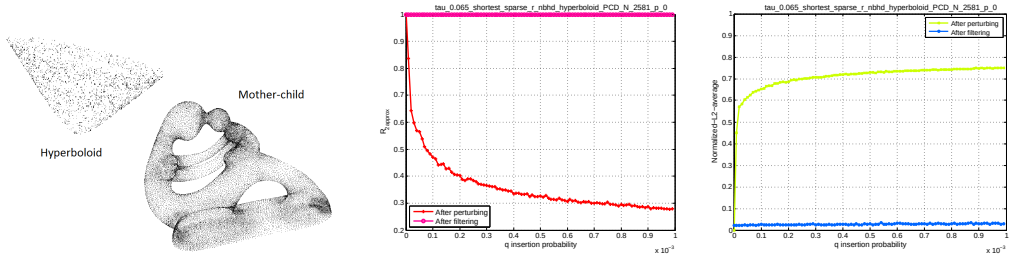
The arguments used in Sections 3.1 and 3.2 can be modified to prove our main result when the observed graph $\widehat{G} = G(r,p,q)$ is generated via the network model described in Definition 3 that includes both edge deletion and insertion. The proof can be found in [19].

▶ **Theorem 13.** *Let V be a set of n points sampled i.i.d. from an L-doubling measure $\mu : X \to \mathbb{R}^+$ supported on a compact metric space (X, d_X). Let G^* be the resulting r-neighborhood graph for V; and $\widehat{G}$ a graph obtained by the network model $G(r,p,q)$ described in Definition 3. Let $\tilde{G}_\tau$ be the graph after τ-Jaccard filtering of $\widehat{G}$. Then, if Assumption-R holds, $p \leq \frac{1}{4}$, $q \leq \min\{\frac{1}{8}, cs\}$ and $sn = \omega(\ln n)$, then for any τ such that $\frac{(1-p)^2}{(10+\frac{5}{3\ln n}+20c)L^2} \geq \tau \geq \frac{(c+2)q}{1-p} + o(1)$, with high probability the shortest path distance metric $d_{\tilde{G}_\tau}$ is a 2-approximation of the shortest path metric d_{G^*} of the true graph G^*.*

Extension to local doubling measure. We can relax the L-doubling condition of the measure μ where points are sampled from to a *local doubling* condition, where the L-doubling property is only required to hold for metric balls of small radius. Specifically,

▶ **Definition 14** ((R_0, L_{R_0})-doubling measure)**.** Given a metric space $\mathcal{X} = (X, d_X)$, a measure μ on $\mathcal{X}$ is said to be (R_0, L_{R_0})-*doubling* if balls have finite and positive measure and there is a constant L_{R_0} s.t. for all $x \in X$ and any $0 < R \leq R_0$, we have $\mu(\mathrm{B}(x, 2R)) \leq L_{R_0} \cdot \mu(\mathrm{B}(x,R))$.

All our results hold for (R_0, L_{R_0})-doubling measure, as long as the parameter r generating the true graph G^*_r satisfies $r < R_0$. The proofs follow the same argument as those for L-doubling measure almost verbatim, and thus are omitted.



(a) (b) (c)

Figure 2 (a) $2.5K$ points sampled from a hyperboloid surface and $24K$ points sampled from mother-child model. (b) Comparison of 2-approximation rate $R_{2approx}$ as insertion probability (x-axis) increases. Top curve is after Jaccard-filtering, while bottom one is for perturbed graph without filtering. (c) Normalized L_2-average error with top curve being the one without filtering, and the bottom one (with significantly lower error) for after Jaccard-filtering. These plots are for hyperboloid case.

5 Some empirical results

We provide some proof-of-principle results to show the effectiveness of the Jaccard filtering process. See [19] for a complete version. There are two sets of experiments.

Synthetic datasets with ground truth. In this experiment we seek to demonstrate that the Jaccard filtering approach works in a robust manner as predicted by our theoretical results. In particular, we start with the following two measures: $\mu_1 : S_1 \to \mathbb{R}^+$ is the "quasi-uniform" measure on the hyperboloid S_1 specified by $x^2 + y^2 - z^2 = 1$ [2]; and $\mu_2 : S_2 \to \mathbb{R}^+$ is a non-uniform measure on the mother-and-child geometric model S_2, where the measure is proportional to the local feature size at each point. For each μ_i, we sample n points V i.i.d and build an r-neighborhood graph (we will specify choice of r later). See Figure 2 (a) for illustration of input samples. This gives rise to a ground-truth neighborhood graph G^*_r. We next generate a set of observed graph $G_{p,q}$, varying the deletion probability (p) and insertion probability (q). Using a fixed parameter τ, we perform τ-Jaccard filtering for each $G_{p,q}$ to obtain a filtered graph $\widehat{G}^{\tau}_{p,q}$. To measure the difference between two metrics D and D', we use two types of error to be introduced shortly. But first, note that since we delete edges, the connectivity of the graph may change. Assume that $D_{i,j} = \infty$ if the two corresponding points p_i and p_j are not connected in the graph. Note that if $D_{i,j} = \infty$ and $D'_{i,j} = \infty$, the relationship $\frac{1}{2}D_{i,j} \le D'_{i,j} \le 2D_{i,j}$ still hold.

- **2-approximation rate $R_{2approx}$** is defined by

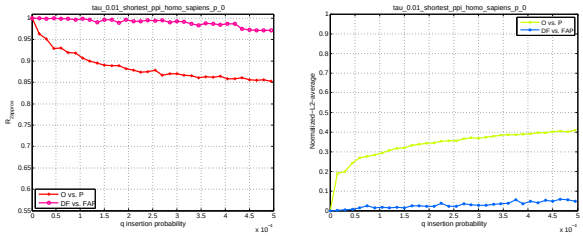
$$R_{2approx}(D, D') = \frac{|\{(i,j), 1 \le i < j \le n \mid \frac{1}{2}D_{i,j} \le D'_{i,j} \le 2D_{i,j}\}|}{n(n-1)/2}.$$

  In other words, $R_{2approx}$ is the ratio of "good" pairwise distances from D' that 2-approximate those in D.

We also consider L_2 type error. To avoid the cases that $D_{i,j}$ is not comparable with $D'_{i,j}$, we collect the following *good-index set*

$I_{good}(D, D') = \{(i,j), 1 \le i < j \le n \mid \text{either } (D_{i,j} < \infty) \wedge (D'_{i,j} < \infty); \text{ or } (D_{i,j} = \infty) \wedge (D'_{i,j} = \infty)\}$.

- **Normalized L_2-average error $\delta_N(D, D')$** is intuitively the root-mean-squared (RMS) error $\delta(D, D')$ normalized by the normalized L_2-norm of D. More specifically,



$R_{2approx}$ normalized L_2

Figure 3 "O vs. P" is the error rate between D_G and D_{G_q}; while "DP vs. FAP" is between D_{G^τ} and $D_{G_q^\tau}$.

$$\delta(D, D') = \sqrt{\frac{\sum_{(i,j)\in I_{good}} (D_{i,j} - D'_{i,j})^2}{|I_{good}|}};\ \delta_N(D, D') = \frac{\delta(D, D')}{\sqrt{\frac{1}{|\{i<j, D_{i,j}<\infty\}|} \sum_{i<j, D_{i,j}<\infty} D_{i,j}^2}}.$$

Let D_G denote the shortest path metric induced by a graph G. In Figure 2 (b), we compare the 2-approximation rate $R_{2approx}(D_{G^*}, D_{G_q})$ for the sequence of observed graphs G_q for increasing insertion probability q, with $R_{2approx}(D_{G^*}, D_{\widehat{G}_q^\tau})$s for the sequence of filtered graph $\widehat{G}_q^\tau$; while the comparison of the normalized L_2 error $\delta_N(D_{G^*}, D_{G_q})$ versus $\delta_N(D_{G^*}, D_{\widehat{G}_q^\tau})$s for increasing qs is shown in Figure 2 (c). These plots are for the hyperboloid model; those for mother-child model are in [19]. The deletion probability is fixed at $p = 0$, as our experiments show (also matching our theoretical results) that the shortest path metric is rather stable against deletion for a large range of deletion probability. As we can see, randomly inserting edges distorts the shortest path metrics (with low 2-approximation rate and high normalized L_2 error for G_qs). However, our Jaccard-index filtering process restores the metric not only w.r.t 2-approximation rate (which is predicted by our theoretical results), but also w.r.t normalized L_2 error. In this experiment, we choose r (to build the r-neighborhood graph) to be twice of the average distance from a point to its 10-th nearest neighbor in P. The resulting graph for hyperboloid has about 2.5K nodes and 38K edges. Examples where the graphs are much denser are given in [19].

Real network without ground truth. For a given real network G, we can consider it as an observed graph. However, we do not know how this network is generated and there is no ground truth graph G^*. Nevertheless, we carry out the following experiments to indirectly infer the effectiveness of Jaccard-filtering.

Specifically, given G, we gradually add random $(p = 0, q)$-perturbation to it, and compare the shortest path metric D_{G_q} of the perturbed graph G_q with the metric D_G of input network G; q is the insertion probability. Next, we perform τ-Jaccard filtering for all these graphs G and G_qs to obtain G^τ and G_q^τ respectively, and then compare the shortest path metric $D_{G_q^\tau}$ for filtered graphs G_q^τ with D_{G^τ} of G^τ. See Figure 3, where the input is a protein-protein interaction network [14] (6327 nodes and 147547 edges). The distance metric becomes more stable after Jaccard-filtering. More discussions and experiments are in [19], including an example of co-authorships network [18] where Jaccard-filtering shows even bigger improvement.

6 Concluding remarks

Our paper represents one step towards unraveling the structure of the space where data are sampled from. There are many interesting problems along this direction, including how to generalize our network model to better model real networks. We describe one direction here: Our current work recovers the shortest path metric of the hidden graph G. However, there are other common metrics induced from G, such as the diffusion distance metric. In fact, for dense random graphs, say graphs generated from a graphon [10] (including stochastic block models), the spectral structure of such random graphs are stable. This may imply that diffusion distances could also be stable against random perturbations even without any filtering process. Note that such graphs have $\Theta(n^2)$ number edges asymptotically. However, for sparse graphs (which our model could generate), empirically we observe that diffusion distances are not stable under random perturbations. It would be interesting to see whether the Jaccard filtering process (or other filtering procedure) could recover diffusion distances with theoretical guarantees. (Interestingly, we have observed that empirically, Jaccard filtering can recover diffusion distance as well in our experiments.) Finally, it would be interesting to explore whether the analysis and ideas for network models from our paper could be used to create a practical wormhole detector in wireless networks, akin to Ban et al's local connectivity tests based on $[\alpha, \beta]$-rings [3].

Acknowledgement. The authors thank Samory Kpotufe for the pointer to the local version of L-doubling measure.

References

1 Morteza Alamgir and Ulrike V. Luxburg. Shortest path distance in random k-nearest neighbor graphs. In *29th Intl. Conf. Machine Learning (ICML)*, pages 1031–1038, 2012.

2 D. Asta and C. Shalizi. Geometric network comparisons. In *31st Annu. Conf. Uncertainty in AI (UAI)*, 2015.

3 Xiaomeng Ban, Rik Sarkar, and Jie Gao. Local connectivity tests to identify wormholes in wireless networks. In *12th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, MobiHoc'11, pages 13:1–13:11. ACM, 2011.

4 HS Bhadauria and ML Dewal. Efficient denoising technique for CT images to enhance brain hemorrhage segmentation. *Journal of digital imaging*, 25(6):782–791, 2012.

5 Bela Bollobás and Fan R. K. Chung. The diameter of a cycle plus a random matching. *SIAM Journal on discrete mathematics*, 1(3):328–333, 1988.

6 Béla Bollobás and Oliver M. Riordan. Mathematical results on scale-free random graphs. *Handbook of graphs and networks: from the genome to the internet*, pages 1–34, 2003.

7 Martin R Bridson and André Haefliger. *Metric spaces of non-positive curvature*, volume 319. Springer Science & Business Media, 2011.

8 Frédéric Chazal, Leonidas J Guibas, Steve Y. Oudot, and Primoz Skraba. Persistence-based clustering in riemannian manifolds. *Journal of the ACM (JACM)*, 60(6):41, 2013.

9 Rick Durrett. *Random Graph Dynamics*, volume 20. Cambridge University Press, 2006.

10 Justin Eldridge, Mikhail Belkin, and Yusu Wang. Graphons, mergeons, and so on! In *Advances in Neural Information Processing Systems*, pages 2307–2315, 2016.

11 Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. On power-law relationships of the internet topology. *ACM SIGCOMM Computer Comm. Review*, 29(4):251–262, 1999.

12 Debra S. Goldberg and Frederick P. Roth. Assessing experimentally derived interactions in a small world. *Proceedings of the National Academy of Sciences*, 100(8):4372–4376, 2003.

13 Juha Heinonen. *Lectures on analysis on metric spaces*. Springer Science & Business Media, 2012.

14 G. Joshi-Tope, Marc Gillespie, Imre Vastrik, Peter D'Eustachio, Esther Schmidt, Bernard de Bono, Bijay Jassal, G. R. Gopinath, G. R. Wu, Lisa Matthews, et al. Reactome: a knowledgebase of biological pathways. *Nucleic acids research*, 33(suppl 1):D428–D432, 2005.

15 Jon Kleinberg. The small-world phenomenon: An algorithmic perspective. In *Proc. 32nd. ACM Symp. Theory Computing*, pages 163–170. ACM, 2000.

16 Jon Kleinberg. Small-world phenomena and the dynamics of information. In *Advances in Neural Information Processing Systems (NIPS)*, pages 431–438. 2002.

17 Elizabeth A. Leicht, Petter Holme, and Mark E. J. Newman. Vertex similarity in networks. *Physical Review E*, 73(2):026120, 2006.

18 Tiancheng Lou and Jie Tang. Mining structural hole spanners through information diffusion in social networks. In *WWW'13*, 2013.

19 S. Parthasarathy, D. Sivakoff, M. Tian, and Y. Wang. A quest to unravel the metric structure behind perturbed networks. *ArXiv e-prints*, March 2017. arXiv:1703.05475.

20 Mathew Penrose. *Random geometric graphs*. Oxford University Press, 2003.

21 Venu Satuluri, Srinivasan Parthasarathy, and Yiye Ruan. Local graph sparsification for scalable clustering. In *ACM SIGMOD Intl. Conf. Management Data*, pages 721–732, 2011.

22 A. Singer and H.-T. Wu. Two-dimensional tomography from noisy projections taken at unknown random directions. *SIAM journal on imaging sciences*, 6(1):136–175, 2013.

23 H. F. Song and X.-J. Wang. Simple, distance-dependent formulation of the Watts-Strogatz model for directed and undirected small-world networks. *Phys. Rev. E*, 90:062801, 2014.

24 Xiao Fan Wang and Guanrong Chen. Complex networks: small-world, scale-free and beyond. *Circuits and Systems Magazine, IEEE*, 3(1):6–20, 2003.

25 Duncan J. Watts, Peter Sheridan Dodds, and M. E. J. Newman. Identity and search in social networks. *Science*, 296(5571):1302–1305, 2002. doi:10.1126/science.1070120.

26 Duncan J. Watts and Steven H. Strogatz. Collective dynamics of 'small-world'networks. *nature*, 393(6684):440–442, 1998.

From Crossing-Free Graphs on Wheel Sets to Embracing Simplices and Polytopes with Few Vertices

Alexander Pilz*[1], Emo Welzl[2], and Manuel Wettstein[3]

1 **Department of Computer Science, ETH Zürich, Zürich, Switzerland**
pilza@inf.ethz.ch
2 **Department of Computer Science, ETH Zürich, Zürich, Switzerland**
emo@inf.ethz.ch
3 **Department of Computer Science, ETH Zürich, Zürich, Switzerland**
mw@inf.ethz.ch

Abstract

A set $P = H \cup \{w\}$ of $n+1$ points in the plane is called a *wheel set* if all points but w are extreme. We show that for the purpose of counting crossing-free geometric graphs on P, it suffices to know the so-called *frequency vector* of P. While there are roughly 2^n distinct order types that correspond to wheel sets, the number of frequency vectors is only about $2^{n/2}$.

We give simple formulas in terms of the frequency vector for the number of crossing-free spanning cycles, matchings, w-embracing triangles, and many more. Based on these formulas, the corresponding numbers of graphs can be computed efficiently.

Also in higher dimensions, wheel sets turn out to be a suitable model to approach the problem of computing the *simplicial depth* of a point w in a set H, i.e., the number of simplices spanned by H that contain w. While the concept of frequency vectors does not generalize easily, we show how to apply similar methods in higher dimensions. The result is an $O(n^{d-1})$ time algorithm for computing the simplicial depth of a point w in a set H of n d-dimensional points, improving on the previously best bound of $O(n^d \log n)$.

Configurations equivalent to wheel sets have already been used by Perles for counting the faces of high-dimensional polytopes with few vertices via the Gale dual. Based on that we can compute the number of facets of the convex hull of $n = d + k$ points in general position in $\mathbb{R}^d$ in time $O(n^{\max\{\omega, k-2\}})$ where $\omega \approx 2.373$, even though the asymptotic number of facets may be as large as n^k.

1998 ACM Subject Classification G.2.1 Combinatorics, F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases geometric graph, wheel set, simplicial depth, Gale transform, polytope

Digital Object Identifier 10.4230/LIPIcs.SoCG.2017.54

1 Introduction

Computing the number of crossing-free straight-line drawings of certain graph classes (e.g., triangulations, spanning trees, etc.) on a planar point set is a well-known problem in computational and discrete geometry. While for point sets in convex position many of these numbers have simple closed formulas, it seems difficult to compute them efficiently for a

* A.P. is supported by an Erwin Schrödinger fellowship, Austrian Science Fund (FWF): J-3847-N35.

given general point set, or to provide tight upper and lower bounds. In this work, we provide means for solving these problems for a special class of point sets which we call wheel sets.

Let $P = H \cup \{w\}$ be a set of $n+1$ points in the plane. Unless stated otherwise, P is assumed to be in general position and the points in H are assumed to be extreme (i.e., part of the boundary of the convex hull of P). P is in *convex position* if all points including w are extreme, and P is a *wheel set* if all points except w are extreme. If P is either of them, then we call it a *conowheel set*. We denote by P_{con} a concrete set in convex position (say, the vertex set of a regular $(n+1)$-gon) and by P_{bar} a *barely-in wheel set* (i.e., H is the vertex set of a regular n-gon and w is sufficiently close to an edge e of the n-gon in such a way that w is in the interior of every triangle spanned by e and a third point of H).

The numbers of triangulations and pseudo-triangulations on wheel sets [24], as well as perfect matchings [26], have been studied before. Our work generalizes these approaches. Wheel sets have also been used to represent vectors in the investigation of high-dimensional polytopes with few vertices; already in the 1960s, Perles counted the number of combinatorially different wheel sets (as reported by Grünbaum [15]). In the terminology of modern discrete geometry, these correspond to the different order types of wheel sets.

Order types. The *order type* of a point set P is a combinatorial description that assigns an orientation (either clockwise or counterclockwise) to every ordered triple of points. Two point sets are said to have the same order type if there exists a bijection between the two sets that preserves these orientations [13]. We follow the practice of considering two point sets to have the same order type if there exists a bijection that reverses all orientations.

Many combinatorial properties of a point set can be recovered from its order type. In particular, the order type determines whether two segments with endpoints in P cross, and whether a given point in P is extreme. It is not hard to see that all sets in convex position have the same order type. However, the same is not true for wheel sets.

▶ **Theorem 1.** *The number of distinct order types of conowheel sets of size $n+1$ is*[1]

$$\frac{1}{4n} \sum_{2 \nmid k \mid n} \varphi(k) 2^{n/k} + 2^{\lfloor (n-3)/2 \rfloor} = \Theta(2^n/n) \ .$$

The above formula has been obtained first by Perles (as stated, without proof, in [15, Chapter 6.3]) for the number of simplicial polytopes with few vertices, and we explain the connection to wheel sets in Section 4. Perles also counted the number of equivalent so-called *distended standard forms of Gale diagrams*, which basically correspond to wheel sets with different order types. In Section 2 we describe this correspondence.

Frequency vectors. While the order type of a point set determines the set of crossing-free geometric graphs on it, we show in Section 3 that we can rely on the following, coarser classification when only considering wheel sets.

Let $P = H \cup \{w\}$ be a conowheel set and let $h \in H$ be arbitrary. Let $\mathsf{l}(h)$ denote the number of points strictly to the left of the directed line going from w to h, and let $\mathsf{r}(h)$ denote the number of points strictly to the right of that line. The *frequency vector* of P is the vector $F(P) = (F_0, F_1, \ldots, F_{n-1})$ where F_i is the number of points $h \in H$ satisfying $|\,\mathsf{l}(h) - \mathsf{r}(h)| = i$. Consider the following examples for $n = 7$.

$$F(P_{\mathsf{con}}) = (1,0,2,0,2,0,2) \qquad F(P_{\mathsf{bar}}) = (1,0,2,0,4,0,0)$$

[1] Here, $\varphi(k)$ denotes Euler's totient function, which counts the integers coprime to k that are at most k.

Note that the frequency vector can be computed in $O(n \log n)$ time by radially sorting H around w. It is also clear that the order type determines the frequency vector. However, the opposite is not true. In Section 2, we give a complete characterization of frequency vectors, which allows us to conclude the following.

▶ **Theorem 2.** *For any $n \geq 1$, the number of frequency vectors realizable by a conowheel set over $n+1$ points is exactly $2^{\lceil n/2 \rceil - 1}$.*

Given that the number of frequency vectors is significantly smaller than the number of order types, it is unclear how much the frequency vector reveals about a conowheel set. However, we will show that for the purpose of counting crossing-free structures it is both sufficient and necessary.

Moreover, there is again a connection to simplicial polytopes with few vertices. In Section 4, we show that the number of frequency vectors is equal to the number of f-vectors of polytopes in d-space with at most $d+3$ vertices (including the empty polytope). The latter has been calculated by Linusson [17] using a sophisticated counting of so-called M-sequences.

Geometric graphs. A *geometric graph* on P is a graph with vertex set P and edges drawn as straight segments between the corresponding endpoints, and it is *crossing-free* if no two edges intersect in their respective relative interiors.

There exists a vast literature that is concerned with counting these crossing-free structures on specific point sets or proving extremal upper and lower bounds [3, 27, 28, 29]. One comparatively simple case is if P is in convex position. In that case, counting triangulations is a classical problem that goes back to Euler, and it gives rise to the famous Catalan numbers. For many other families of graphs (such as perfect matchings and spanning trees), simple closed formulas can be obtained as well [7, 11, 22].

Randall et al. [24] were the first to consider geometric graphs on wheel sets. They found the extremal configurations for triangulations and pseudo-triangulations by using an argument that involves continuously moving the extra point w. The case of perfect matchings has been studied by Ruiz-Vargas and Welzl [26]. The next theorem is a generalization of a result from their paper.

In the following, let $\mathcal{G}$ be a set of abstract (unlabeled) graphs with $n+1$ vertices, and let $\mathsf{nb}_{\mathcal{G}}(P)$ denote the number of crossing-free geometric graphs on P which are isomorphic to a graph in $\mathcal{G}$. In other words, $\mathsf{nb}_{\mathcal{G}}(P)$ is the number of non-crossing straight-line embeddings of graphs in $\mathcal{G}$ on P.

▶ **Theorem 3.** *thmggraph Let $\mathcal{G}$ be arbitrary, and let $P = H \cup \{w\}$ be a conowheel set of size $n+1$. Then, $\mathsf{nb}_{\mathcal{G}}(P)$ depends only on the frequency vector $F(P) = (F_0, F_1, \ldots, F_{n-1})$. More concretely,*

$$\mathsf{nb}_{\mathcal{G}}(P) = \gamma_n - \frac{1}{2} \sum_{h \in H} \lambda_{\mathsf{l}(h),\mathsf{r}(h)} = \sum_{k=0}^{n-1} F_k \Lambda_k \ ,$$

where γ_n and $\lambda_{l,r} = \lambda_{r,l}$ are integers and Λ_k are rationals depending on $\mathcal{G}$.

While the latter formula in the above theorem makes the dependency on the frequency vector more obvious, the former will turn out to be more natural. The latter formula follows from the former simply by putting $\Lambda_k = \gamma_n/n + 1/2 \cdot \lambda_{(n+k-1)/2,(n-k-1)/2}$.

We give just one example here, which at the same time makes the connection to the later parts of the paper. Let $\mathcal{G} = \{K_4^{\cdots}\}$, where $K_4^{\cdots}$ is obtained by adding $n-3$ additional isolated

vertices to the complete graph K_4. The following formula is obtained alongside Theorem 3 in Section 3.

$$\mathsf{nb}_{\mathcal{G}}(P) = \binom{n}{3} - \frac{1}{2}\sum_{h\in H}\left(\binom{\mathsf{l}(h)}{2} + \binom{\mathsf{r}(h)}{2}\right) \qquad \text{for } \mathcal{G} = \{K_4^{\cdots}\}\,. \tag{1}$$

Observe that all non-crossing embeddings of $K_4^{\cdots}$ on a given conowheel set $P = H \cup \{w\}$ have the following property. One of the vertices of the underlying K_4 is mapped to the point w, while the other three vertices are mapped to three points which form a triangle that contains the extra point w in its interior. We thus get a rather simple formula for the number of *w-embracing triangles* (i.e., point triples in H whose convex hull contains w). Note that the set of w-embracing triangles does not change if we replace a point $p \in H$ by a point p' on the ray starting at w and passing through p; for counting w-embracing triangles, the approach for conoweel sets thus generalizes to arbitrary point sets. In other words, the formula in equation (1) also counts the number of w-embracing triangles in an arbitrary point set H in general position. We note that the algorithm which counts w-embracing triangles in [25] is essentially an implementation of equation (1).

Higher dimensions. The concept of conowheel sets can be generalized to arbitrary dimensions, where we may again consider sets with at most one non-extreme point. However, even for counting w-embracing tetrahedra in 3-space, the ideas from the proof of Theorem 3 do not generalize. Nevertheless, in Section 4 we give a generalization of equation (1). From that we obtain improved time bounds for computing the number of w-embracing simplices or, in other words, the *simplicial depth* of a point w (as defined in [18]).

▶ **Theorem 4.** *thmsimpldepth Let $d \geq 3$ be fixed and let H be a set of n points in $\mathbb{R}^d$. Then, the simplicial depth of a point w in H can be computed in $O(n^{d-1})$ time.*

Again, this result is stated for arbitrary sets H and not for wheel sets only, as for the simplicial depth only the position relative to w is relevant. We further note that the algorithm generalizes to counting all k-element subsets of H whose convex hull contains w.

The simplicial depth of a point has attracted considerable attention as a measure of data depth. Several authors describe the calculation of the simplicial depth of a point in the plane [12, 16, 25]. $O(n^2)$ and $O(n^4)$ time algorithms for 3- and 4-space, respectively, are provided by Cheng and Ouyang [6], who also point out flaws in previous algorithms in 3-space. Our result improves over the previously best known general $O(n^d \log n)$ time algorithm for points in constant dimension d [2]. For arbitrary dimensions, the problem is known to be $\#P$-complete and $W[1]$-hard [2].

The work by Perles aimed at counting the number of facets of high-dimensional simplicial polytopes with few vertices. Via the Gale dual, this number corresponds to the number of simplices embracing the origin in a dual point set. In Section 4, we show how to compute the number of facets of the convex hull of $d + k$ points in general position in $\mathbb{R}^d$ in time $O(n^{\max\{\omega, k-2\}})$ (with $O(n^\omega)$ being an upper bound for matrix multiplication).

2 Order Types and Frequency Vectors

The purpose of this section is to give an explanation for Table 1. The latter contains the numbers of distinct order types and frequency vectors corresponding to conowheel sets of size $n + 1$. For completeness, we have also included the corresponding numbers if equivalence over order types is defined to not include reflections.

Table 1 Number of order types and frequency vectors of conowheel sets over $n+1$ points.

n	Order Types with reflection	Order Types w/o reflection	Freq. Vectors	n	Order Types with reflection	Order Types w/o reflection	Freq. Vectors
1	1	1	1	7	9	10	8
2	1	1	1	8	12	16	8
3	2	2	2	9	23	30	16
4	2	2	2	10	34	52	16
5	4	4	4	11	63	94	32
6	5	6	4	12	102	172	32

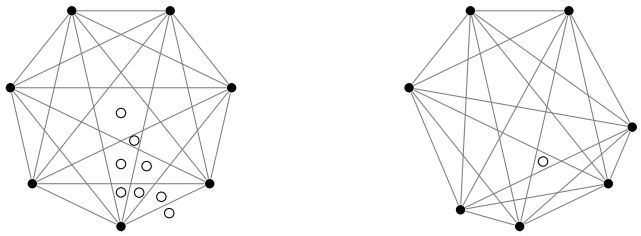


Figure 1 All order types of conowheel sets for $n = 7$. The extra point w is drawn in white.

Order types. Given a set H of $n = 7$ points forming the vertex set of a regular heptagon, there are 8 conowheel sets $P = H \cup \{w\}$ with distinct order types that can be obtained by adding an extra point w, see the left hand side of Figure 1. Notice the discrepancy with the number 9 displayed in Table 1. The ninth and last order type can be obtained by first deforming H as illustrated on the right hand side of Figure 1. This necessary deformation of H seems to complicate matters significantly, but only at first sight.

▶ **Theorem 1** (restated). *The number of distinct order types of conowheel sets of size $n+1$ is*[2]

$$\frac{1}{4n}\sum_{2\nmid k\mid n}\varphi(k)2^{n/k}+2^{\lfloor(n-3)/2\rfloor}=\Theta(2^n/n)\ .$$

The asymptotic estimate is explained by taking the dominant summand with $k = 1$. The exact formula was first obtained by Perles (cf. [15, Chapter 6.3]) as the number of combinatorially different simplicial polytopes in dimension $n-3$ with at most n vertices.

The formula was also obtained in the context of counting the number of 2-colored self-dual necklaces with $2n$ beads with mirrored necklaces identified [5, 23]. These are binary (say, black and white) circular sequences of length $2n$ such that elements at distance n (i.e., opposing beads) are distinct (i.e., if one is black the other must be white, and vice versa). The correspondence between simplicial polytopes and necklaces has been shown by Montellano-Ballesteros and Strausz [21] using Radon complexes, unaware of Perles' result.

A proof of Theorem 1 can be given by using a simple bijection to necklaces. We note that a similar (and slightly simpler) formula is known if mirrored necklaces are not identified. Naturally, that formula also counts order types of conowheel sets without reflection.

[2] Here, $\varphi(k)$ denotes Euler's totient function, which counts the integers coprime to k that are at most k.

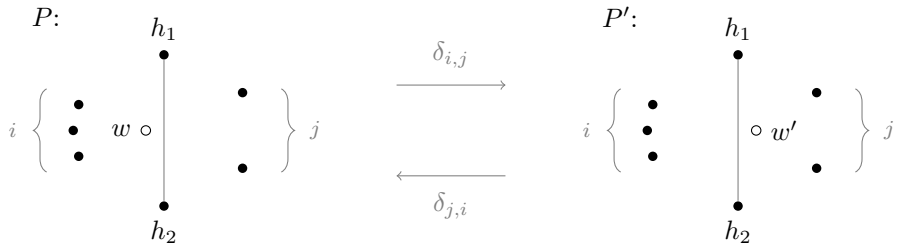


Figure 2 Moving the extra point over the segment h_1h_2 for the case $i = 3$ and $j = 2$.

Frequency vectors. The following lemma gives a characterization of frequency vectors. The proof is by letting a line rotate about the extra point w, and by observing how it dissects the point set H during the process. Full details can be found in [26].

▶ **Lemma 5.** *$F = (F_0, F_1, \ldots, F_{n-1}) \in \mathbb{N}^n$ is the frequency vector of a conowheel set $P = H \cup \{w\}$ of size $n+1$, i.e., $F = F(P)$, if and only if*

(i) $\sum_{k=0}^{n-1} F_k = n$,
(ii) $F_k = 0$ *for all* $k \equiv_2 n$,
(iii) F_k *is even for all* $k \geq 1$, *and*
(iv) *if* $F_k \neq 0$ *and* $k \geq 2$, *then* $F_{k-2} \neq 0$.

With this characterization, it is not hard to determine the number of frequency vectors.

▶ **Theorem 2** (restated). *For any $n \geq 1$, the number of frequency vectors realizable by a conowheel set over $n+1$ points is exactly $2^{\lceil n/2 \rceil - 1}$.*

Proof. For $n = 1$ and $n = 2$ the formula evaluates to 1, which is consistent with the fact that there is only one respective order type for either two or three points. For larger n, we give a proof by induction for odd n, and note that even n can be handled analogously.

So let $n = m + 2 \geq 3$ be odd. Using Lemma 5, we can characterize the set of frequency vectors that are realizable by $n+1$ points by saying that it contains all vectors $F = (F_0, F_1, \ldots, F_{n-1})$ which have one of the following two mutually exclusive forms.

- $F_0 = 1$, $F_1 = 0$, and $(F_2 - 1, F_3, F_4, \ldots, F_{n-1})$ is any frequency vector realizable by $m+1$ points.
- $F_0 \geq 3$ is odd, $F_{n-2} = F_{n-1} = 0$, and $(F_0 - 2, F_1, F_2, \ldots, F_{n-3})$ is any frequency vector realizable by $m+1$ points.

If $2^{\lceil m/2 \rceil - 1}$ is the number of frequency vectors realizable by $m+1$ points, the corresponding number for $n+1$ points is thus $2 \cdot 2^{\lceil m/2 \rceil - 1} = 2^{\lceil n/2 \rceil - 1}$. ◀

3 Geometric Graphs

Recall that $\mathsf{nb}_{\mathcal{G}}(P)$ is the number of crossing-free geometric graphs on a point set P that are isomorphic to a graph in the family $\mathcal{G}$. Consider now two conowheel sets $P = H \cup \{w\}$ and $P' = H \cup \{w'\}$ which can be transformed into each other by moving the extra point over the segment between $h_1, h_2 \in H$ (and without encountering any other collinearities). The situation is illustrated in Figure 2. Assume that on the w-side of the segment h_1h_2 there are i points of H, and on the w'-side there are j points of H (thus, $i + j = n - 2$). Let $\delta_{i,j}$ be the increment of $\mathsf{nb}_{\mathcal{G}}$ when going from P to P', i.e., $\delta_{i,j} = \mathsf{nb}_{\mathcal{G}}(P') - \mathsf{nb}_{\mathcal{G}}(P)$.

▶ **Lemma 6.** *For every $\mathcal{G}$, $\delta_{i,j}$ is well-defined, i.e., its value depends only on i, j and $\mathcal{G}$, and not on the exact placement of H or the location where the extra point traverses the segment between h_1 and h_2.*

Proof. All geometric graphs that do not contain the edge $\{h_1, h_2\}$ are not affected by the mutation, i.e., they are crossing-free on P if and only if they are crossing-free on P'. Therefore, $\delta_{i,j}$ is equal to the number of crossing-free geometric graphs on P' containing $\{h_1, h_2\}$ minus the number of crossing-free geometric graphs on P containing $\{h_1, h_2\}$. For the following reasons, these numbers only depend on i, j and $\mathcal{G}$.

In the case of P, on the w-side we have $i+3$ points (including h_1 and h_2) in a barely-in configuration, for which there exists a unique order type. On the opposite side we have $j+2$ points (including h_1 and h_2) in convex position, for which there also exists a unique order type. Because of the presence of the edge $\{h_1, h_2\}$ any other edges must be completely contained in one of the two sides, and the claim follows. In the case of P', an analogous argument works. ◀

Example, embracing triangles. Consider the case $\mathcal{G} = \{K_4^{\cdots}\}$. Observe that any crossing-free embedding of $K_4^{\cdots}$ on P uses w as the inner vertex of the underlying K_4. Furthermore, if the embedding uses the edge $\{h_1, h_2\}$, which implies that h_1 and h_2 are outer vertices of K_4, then any one of the i points on the left hand side can be used as the third outer vertex of K_4. This gives exactly i crossing-free embeddings of $K_4^{\cdots}$ on P which use the edge $\{h_1, h_2\}$. Similarly, we get j for the corresponding number of embeddings on P'. Therefore, $\delta_{i,j} = j - i$ for $\mathcal{G} = \{K_4^{\cdots}\}$.

▶ **Theorem 3** (restated)**.** *Let $\mathcal{G}$ be arbitrary, and let $P = H \cup \{w\}$ be a conowheel set of size $n+1$. Then, $\mathsf{nb}_{\mathcal{G}}(P)$ depends only on the frequency vector $F(P) = (F_0, F_1, \ldots, F_{n-1})$. More concretely,*

$$\mathsf{nb}_{\mathcal{G}}(P) = \gamma_n - \frac{1}{2}\sum_{h\in H} \lambda_{\mathsf{l}(h),\mathsf{r}(h)} = \sum_{k=0}^{n-1} F_k \Lambda_k \ ,$$

where γ_n and $\lambda_{l,r} = \lambda_{r,l}$ are integers and Λ_k are rationals depending on $\mathcal{G}$.

Proof. We proceed by choosing the numbers $\lambda_{l,r}$ such that the validity of the formula is preserved under continuous motion of P, and then choose γ_n such that it holds for some starting configuration. To be more concrete, we allow continuous motion of P where all points are allowed to move if P is in convex position, and only w is allowed to move if P is a wheel set. At discrete moments in time we allow collinearity of three points, the one in the middle being w. In this way any two conowheel sets can be transformed into each other.

Let now P and P' be as in Figure 2. Note that the values $\mathsf{l}(h)$ and $\mathsf{r}(h)$ do not change for any $h \in H \setminus \{h_1, h_2\}$ when going from P to P'. For h_1 and h_2 the corresponding values are

P :	$\mathsf{l}(h_1) = \mathsf{r}(h_2) = i$	$\mathsf{r}(h_1) = \mathsf{l}(h_2) = j+1$
P' :	$\mathsf{l}(h_1) = \mathsf{r}(h_2) = i+1$	$\mathsf{r}(h_1) = \mathsf{l}(h_2) = j$

We therefore preserve the validity of the formula as long as the numbers $\lambda_{l,r} = \lambda_{r,l}$ are chosen in such a way that the following equality holds.

$$\delta_{i,j} = \frac{1}{2}(\lambda_{i,j+1} + \lambda_{j+1,i}) - \frac{1}{2}(\lambda_{i+1,j} + \lambda_{j,i+1}) = \lambda_{i,j+1} - \lambda_{i+1,j} \ .$$

The definition $\lambda_{l,r} := \delta_{n-2,0} + \delta_{n-3,1} + \cdots + \delta_{l,r-1} + c_n$ satisfies this constraint. Moreover, the assumed symmetry $\lambda_{l,r} = \lambda_{r,l}$ follows from the skew-symmetry $\delta_{i,j} = -\delta_{j,i}$. Note that

$l + r = n - 1$ always, and that c_n is an arbitrary integer independent of l and r (for the proof to go through one could simply fix $c_n = 0$).

Finally, γ_n is chosen in such a way that the formula holds for some conowheel set. The most natural choice for "anchoring" the formula is a set in convex position.

$$\gamma_n := \mathsf{nb}_{\mathcal{G}}(P_{\mathsf{con}}) + \frac{1}{2} \sum_{l,r\colon l+r=n-1} \lambda_{l,r} \qquad \blacktriangleleft$$

Computing the frequency vector can be done in $O(n \log n)$ time. Given the values Λ_k, computing the number $\mathsf{nb}_{\mathcal{G}}(P)$ of embeddings then requires only $O(n)$ additional arithmetic operations.

Example continued, embracing triangles. We already derived $\delta_{i,j} = j - i$ for $\mathcal{G} = \{K_4^{\cdots}\}$. This now gives rise to

$$\lambda_{l,r} = \delta_{n-2,0} + \delta_{n-3,1} + \cdots + \delta_{l,r-1} + c_n = \sum_{j=0}^{r-1} j - \left(\binom{n-1}{2} - \sum_{i=0}^{l-1} i \right) + c_n = \binom{l}{2} + \binom{r}{2} \ ,$$

if we choose $c_n = \binom{n-1}{2}$. It can be checked that $\mathsf{nb}_{\mathcal{G}}(P_{\mathsf{con}}) = 0$. Hence,

$$\gamma_n = \mathsf{nb}_{\mathcal{G}}(P_{\mathsf{con}}) + \frac{1}{2} \sum_{l,r\colon l+r=n-1} \lambda_{l,r} = 0 + \frac{1}{2} \sum_{l=0}^{n-1} \binom{l}{2} + \frac{1}{2} \sum_{r=0}^{n-1} \binom{r}{2} = \binom{n}{3} \ .$$

After putting everything together we obtain the exact formula displayed earlier in equation (1).

3.1 Further Examples

We call the following two simple applications "insensitive" since the number of crossing-free embeddings is the same on all wheel sets, but may be different for sets in convex position.

Spanning cycles. Consider the case where $\mathcal{G}$ contains a cycle over $n + 1$ vertices. Then, we have $\delta_{i,j} = 0$ because no crossing-free spanning cycle can use the edge $\{h_1, h_2\}$. That is, unless $i = 0$ or $j = 0$, in which case we have $\delta_{0,n-2} = -\delta_{n-2,0} = n - 1$. For anchoring we calculate $\mathsf{nb}_{\mathcal{G}}(P_{\mathsf{con}}) = 1$. It follows that all wheel sets over $n + 1$ points admit n crossing-free spanning cycles (which can easily be seen directly).

Spanning paths. If $\mathcal{G}$ contains a path over $n + 1$ vertices we also get $\delta_{i,j} = 0$ unless $i = 0$ or $j = 0$, but for a different reason. On P there are $2 \cdot 2^i \cdot 2^{j-1}$ crossing-free embeddings that use the edge $\{h_1, h_2\}$, since there are 2 choices for deciding which one of h_1 and h_2 is connected to the left hand side, 2^i choices for completing the left hand side to a path and 2^{j-1} choices for completing the right hand side to a path. Likewise, on P' there are $2 \cdot 2^{i-1} \cdot 2^j$ embeddings, which is the same number. Using a similar argument it can be checked that

$$\delta_{0,n-2} = -\delta_{n-2,0} = 2 \cdot (2^{n-2} + \sum_{k=1}^{n-2} 2^{k-1} \cdot 2^{n-2-k}) - 2 \cdot 2^{n-3} = (n-1)2^{n-2} \ .$$

For anchoring we compute $\mathsf{nb}_{\mathcal{G}}(P_{\mathsf{con}}) = (n+1)2^{n-2}$, implying $\mathsf{nb}_{\mathcal{G}}(P) = n2^{n-1}$ for any wheel set P.

The following two applications are "sensitive" in the sense that different wheel sets in general have different numbers of crossing-free embeddings. The running example with embracing triangles also is of this kind.

Matchings. Let $\mathcal{G} = \mathcal{M}$, the set of (not necessarily perfect) matchings over $n+1$ vertices. It is known that $\mathsf{nb}_{\mathcal{M}}(P_{\mathsf{con}}) = M_{n+1} := \sum_{k=0}^{\lfloor (n+1)/2 \rfloor} \binom{n+1}{2k} C_k$, the $(n+1)$-th Motzkin number [22]. It is thus easy to compute $\delta_{i,j} = M_i M_{j+1} - M_{i+1} M_j$ since, as always, we only have to worry about embeddings that use the edge $\{h_1, h_2\}$. This gives $\lambda_{l,r} = M_l M_r$ and $\gamma_n = M_{n+1} + \frac{1}{2} \sum_{l,r} M_l M_r$. After simplifying[3], we obtain the following formula.

$$\mathsf{nb}_{\mathcal{G}}(P) = \frac{3M_{n+1} - M_n}{2} - \frac{1}{2} \sum_{h \in H} M_{\mathsf{l}(h)} M_{\mathsf{r}(h)} \qquad \text{for } \mathcal{G} = \mathcal{M} \tag{2}$$

Spanning trees. Let $\mathcal{G} = \mathcal{ST}$, the set of all trees over $n+1$ vertices. We will make use of the fact that $\mathsf{nb}_{\mathcal{ST}}(P_{\mathsf{con}}) = T_{n+1} := \frac{1}{2n+1}\binom{3n}{n}$ [7, 11]. Furthermore, we will use the hypergeometric identity $\sum_{k=0}^{i} T_{k+1} T_{i-k+1} = \frac{1}{i+1}\binom{3i+1}{i}$. To compute $\delta_{i,j}$, consider the set P as in Figure 2. In order to complete the left hand side into a spanning tree, we have to build two disjoint trees rooted at h_1 and h_2, respectively. There are 2 choices for assigning w either to the upper or the lower tree, and there are $i+1$ choices for distributing the i points on the left among the two trees. Indeed, the latter claim holds because the k out of i points assigned to h_1, say, have to appear consecutively with h_1 on the convex hull as otherwise we are forced to create a crossing. Once the points have been distributed, we are left with two point sets of size $k+1$ and $i-k+2$ in convex position. For completing the right hand side into a spanning tree, a simpler argument can be used without the additional complication of w. Moreover, the set P' can be handled analogously.

$$\begin{aligned}
\delta_{i,j} &= 2 \sum_{k=0}^{j} T_{k+1} T_{j-k+2} \cdot \sum_{k=0}^{i} T_{k+1} T_{i-k+1} - 2 \sum_{k=0}^{i} T_{k+1} T_{i-k+2} \cdot \sum_{k=0}^{j} T_{k+1} T_{j-k+1} \\
&= 2 \left(\frac{2}{j+2} \binom{3j+3}{j} \cdot \frac{1}{i+1} \binom{3i+1}{i} - \frac{2}{i+2} \binom{3i+3}{i} \cdot \frac{1}{j+1} \binom{3j+1}{j} \right)
\end{aligned}$$

For this application, we do not know if a simple closed form expression for $\lambda_{l,r}$ exists. Still, note that if one were to compute $\mathsf{nb}_{\mathcal{ST}}(P)$, the numbers $\delta_{i,j}$ can be summed up in linear time and the value of γ_n can be computed on the fly for any given n.

Other applications. Observe that, for example, a geometric triangulation of P_{con} can be embedded as a plane graph on P_{bar}. However, this embedding is no longer a triangulation (i.e., a tessellation of $\mathsf{CH}(P_{\mathsf{bar}})$ into triangles). Hence, there is no natural choice of $\mathcal{G}$ such that $\mathsf{nb}_{\mathcal{G}}(P)$ is the number of triangulations of any conowheel set P. It turns out that there are several other families of geometric graphs (pseudotriangulations, crossing-free convex partitions, etc.) whose quantity on a conowheel set P cannot be expressed easily in the form $\mathsf{nb}_{\mathcal{G}}(P)$, but for which the continuous motion argument is still applicable and leads to similarly simple formulas. All that is required is a specialized version of Lemma 6.

Furthermore, we note that Theorem 3 generalizes to crossing-free embeddings of hypergraphs, where "crossing-free" means that the convex hulls of any two hyperedges intersect in an at most 1-dimensional set.

[3] A crossing-free matching on P_{con} either leaves w unmatched (M_n possibilities) or it matches w with one of the other n points ($\sum_{l,r} M_l M_r$ possibilities). Hence, as required, $M_{n+1} = M_n + \sum_{l,r} M_l M_r$.

3.2 Wheel Sets and the Rectilinear Crossing Number

Even though conowheel sets and the associated frequency vectors seem like a very specific set of objects, they do occur naturally in more general settings. Consider for example an arbitrary set $\tilde{P}$ of $n+1$ points in general position and let $\square$ and $\triangle$ be the number of 4-element subsets of $\tilde{P}$ in convex and in non-convex position, respectively.[4] Let $p \in \tilde{P}$ be any point. We can construct a conowheel set $P = H \cup \{w\}$ containing $w = p$ and, for every $q \in \tilde{P} \setminus \{p\}$, the point h which lies on the intersection of the ray from p to q and a fixed circle centered at w (as done also, e.g., in [16]). That is, P is simply a representation of the local sequence of p in $\tilde{P}$ in terms of conowheel sets; see [14]. Further observe that a triangle spanned by points in $\tilde{P}$ contains p iff its image in P contains w. Hence, $\mathsf{nb}_{K_4^{\cdots}}(P)$ is the number of such triangles, which is given by equation (1). We thus obtain $\triangle$ by a summation over all points p in $\tilde{P}$. Since $\square + \triangle = \binom{n+1}{4}$, we also obtain $\square$. We note that this can be transformed into equations from [1, 19, 30] that give $\square$ in terms of the number of j-edges (i.e., directed edges spanned by $\tilde{P}$ with exactly j points of $\tilde{P}$ to their left).

To sum up, we can associate a frequency vector with every point of a given point set, and this set of frequency vectors determines the value of $\square$. Unfortunately, this simple approach does not generalize to counting other types of graphs; there are examples of point sets with the same set of frequency vectors but a different number of triangulations.

4 Higher Dimensions: Origin-embracing Simplices

As already noted in the introduction, the concept of conowheel sets can be generalized to higher dimensions. However, already in $\mathbb{R}^3$ we face certain challenges. For example, the number of tetrahedralizations of $n+1$ points in convex position in $\mathbb{R}^3$ does not only depend on n, in contrast to the 2-dimensional case. Even when considering simpler structures, like the set of w-embracing tetrahedra, the ideas from Section 3 do not generalize. (Intuitively, our argument of w passing over a segment does not work in 3-space, as it can pass "behind" a triangle.) Still, we can use similar techniques to obtain improved time bounds for computing the simplicial depth of a point w.

Oriented simplices. Given a set T of d affinely independent points in $\mathbb{R}^d$, its convex hull $\mathsf{CH}(T)$ is a $(d-1)$-simplex and its affine hull is a hyperplane. We want to be able to refer to the two sides of this hyperplane by identifying a positive and a negative side. For that consider a sequence $p_1p_2\dots p_d$ of d affinely independent points. The affine hull of $\{p_1, p_2, \dots, p_d\}$ can be described as the set of points q with $\sigma(q, p_1p_2\dots p_d) = 0$, where $\sigma(q, p_1p_2\dots p_d) := \det(p_1 - q, p_2 - q, \dots, p_d - q)$. We call the set of points q with $\sigma(q, p_1p_2\dots p_d) > 0$ the *positive side of* $p_1p_2\dots p_d$, and the set of points q with $\sigma(q, p_1p_2\dots p_d) < 0$ the *negative side of* $p_1p_2\dots p_d$. Note that for $d = 2$, the positive side of p_1p_2 is the set of points left of the line through p_1 and p_2, directed from p_1 to p_2. Also note that the positive side of p_1p_2 coincides with the negative side of p_2p_1. For $d \geq 3$, distinct orderings of the given d points may have identical positive sides; this happens iff the orderings can be obtained from each other by an even number of transpositions. An *oriented* $(d-1)$*-simplex* is a sequence $p_1p_2\dots p_d$ of d affinely independent points, with its associated $(d-1)$-simplex $\mathsf{CH}(\{p_1, p_2, \dots, p_d\})$, and its associated positive and negative side as defined above. Two such oriented $(d-1)$-simplices

[4] The number $\square$ is also the number of crossings of the complete geometric graph on $\tilde{P}$, a quantity that has obtained special attention in connection with the so-called *rectilinear crossing number of* K_n (i.e., the smallest number of crossings in a straight line drawing of the complete graph in the plane).

are defined to be equivalent if their sequences can be obtained from each other by an even number of transpositions (e.g., $p_1p_2p_3 \equiv p_3p_1p_2 \equiv p_2p_3p_1$ and $p_3p_2p_1 \equiv p_1p_3p_2 \equiv p_2p_1p_3$).

Via oriented simplices, the concept of order types generalizes to arbitrary dimensions; the order type of a set $P = H \cup \{w\}$ of $n+1$ points in $\mathbb{R}^d$ determines the set of points on the positive side of the oriented $(d-1)$-simplex $wh_1 \dots h_{d-1}$, for each $(d-1)$-tuple in H. Let $\mathsf{l}(h_1 \dots h_{d-1})$ be the number of these points, and let $\mathsf{r}(h_1 \dots h_{d-1}) = n-d+1-\mathsf{l}(h_1 \dots h_{d-1})$. We can thus define the frequency vector $F(P) = (F_0, F_1, \dots, F_{n-d+1})$ by letting F_i denote the number of tuples $\rho \in H^{d-1}$ s.t. $|\,\mathsf{l}(\rho) - \mathsf{r}(\rho)| = i$. However, already for $d = 3$ it turns out that this frequency vector does not always determine the number of w-embracing tetrahedra, i.e., the number of subsets of $d+1$ points of H whose convex hull contains w.

4.1 Embracing sets

Here, we generalize the notion of embracing triangles to larger sets. Consider a set $P = H \cup \{w\}$ of $n+1$ points in $\mathbb{R}^d$. A subset $A \subseteq H$ is a *w-embracing k-set* if $w \in \mathsf{CH}(A)$ and $|A| = k$. For simplicity, we will usually consider $w = \mathbf{0}$ and call A an *embracing k-set*.

Let us quickly return to dimension $d = 2$ and consider a conowheel set $P = H \cup \{w\}$. As follows, we can show that the number of embracing k-sets is determined by the frequency vector of P for any k, and not just for $k = 3$ as seen earlier in equation (1). Indeed, since H is in general position, for every non-embracing k-set $A \subseteq H$ there exists a unique point $h \in A$ such that $\mathsf{CH}(A)$ is in the closed halfplane to the left of the directed segment wh. Observe that for any choice of $h \in H$ we can construct $\binom{\mathsf{l}(h)}{k-1}$ such non-embracing k-sets, and thus we get a generalization of equation (1). (This direct approach to counting non-embracing triangles was, e.g., also used in [25].) Interestingly, the reverse is also true.

▶ **Lemma 7.** *The sequence $(\mathsf{embr}_k)_{k=3}^n$, where embr_k is the number of embracing k-sets in a conowheel set $P = H \cup \{w\}$ of size $n+1$, determines the frequency vector of P.*

Proof. Let $E = (\mathsf{embr}_k)_{k=3}^n$. Consider the vector $L = (L_j)_{j=1}^{n-1}$ where L_j is the number of points $h \in H$ with $\mathsf{l}(h) = j$. Clearly, L determines the frequency vector of P. It thus suffices to show that E determines L.

The number embr_k of embracing k-sets is

$$\mathsf{embr}_k = \binom{n}{k} - \sum_{h \in H} \binom{\mathsf{l}(h)}{k-1} \ ,$$

which may be rewritten as

$$\binom{n}{k} - \mathsf{embr}_k = \sum_{j=1}^{n-1} L_j \binom{j}{k-1} \ .$$

Observe that the above equation also holds for $k = 2$. We can thus define a vector $E' = (e_i)_{i=1}^{n-1}$ with $e_i = \binom{n}{i+1} - \mathsf{embr}_{i+1}$ and a square matrix $A = (a_{i,j})_{i,j=1}^{n-1}$ with $a_{i,j} = \binom{j}{i}$, such that

$$E' = AL \ .$$

Since the matrix A is unitriangular (i.e., triangular and without zeroes on the diagonal) it has an inverse, from which we conclude that E' determines L. ◀

▶ **Corollary 8.** *Let P and P' be two conowheel sets. Then $\mathsf{nb}_{\mathcal{G}}(P) = \mathsf{nb}_{\mathcal{G}}(P')$ for any graph class $\mathcal{G}$ if and only if $F(P) = F(P')$.*

Proof. We already know from Theorem 3 that the frequency vector determines the number of embeddings. For the other direction, we reconstruct the number of embracing k-sets by appropriately choosing the graph classes $\mathcal{G}$. After that, the frequency vector is determined by Lemma 7.

The number of embracing triangles is equal to the number of embeddings of $K_4^{\cdots}$ and therefore, by our assumption, the same for both P and P'. We now simply generalize to k-wheels, i.e., $\mathcal{G}$ contains a cycle with k vertices, each adjacent to one additional vertex. All that is left to observe is that the number of distinct embeddings of such a k-wheel on a conowheel set is the same as the number of embracing k-sets. ◀

Note that, for arbitrary point sets, we can compute the number of crossing-free embeddings of such k-wheels in polynomial time: For $k = 3$, this number is equal to the number of crossing-free embeddings of $K_4^{\cdots}$, which can be obtained from the frequency vector associated with each point, see Section 3.2. For $k \geq 4$, we distinguish the cases where the geometric embedding of a k-wheel has only three vertices on the unbounded cell and where it has more. The latter case can be dealt with by computing the number of embracing k-sets for each point. The former can be obtained by computing the number of triangles with j points in the interior and multiplying this number by $3\binom{j}{k-2}$. This is because for every vertex of such a triangle, a path of $k-2$ points inside this triangle in radial order around that vertex gives a k-wheel with the triangle as the unbounded cell. For all j, the corresponding number of triangles can be obtained in $O(n^3)$ time [4, 10].

Unfortunately, generalizing the above approach of counting embracing k-sets to higher dimensions fails already in 3-space. Indeed, consider the set of non-embracing tetrahedra for a set $H \subset \mathbb{R}^3$ in convex position. Observe now that any such tetrahedron has either three or four edges that form a "tangent" plane through w.

Instead, consider a partition $B \dot\cup W = H$ defined by a plane ϕ through w that is disjoint from H. Then, the set of non-embracing k-sets consists of those completely in B and W, respectively, and those intersected by ϕ. For the latter, consider the intersection of $\mathsf{CH}(A)$ of such a set A with ϕ. There is again a single "tangent" point $t = pq \cap \phi$ such that $\mathsf{CH}(A) \cap \phi$ is on one side of the line wt on ϕ. Hence, the number of embracing k-sets in 3-space is

$$\mathsf{embr}_k = \binom{n}{k} - \binom{|B|}{k} - \binom{|W|}{k} - \sum_{pq \in B \times W} \binom{\mathsf{l}(pq)}{k-2} \ .$$

We can generalize this approach in the following way.

▶ **Lemma 9.** *Let H be a set of n points in $\mathbb{R}^d$, with $H \cup \{\mathbf{0}\}$ in general position, and let ψ be a generic 2-flat containing the origin. Let* $\mathrm{proj} : \mathbb{R}^d \to \mathbb{R}^{d-2}$ *be a projection that maps all of ψ to $\mathbf{0} \in \mathbb{R}^{d-2}$. Then, the number of embracing k-sets in H is*

$$\mathsf{embr}_k(\mathrm{proj}(H)) - \frac{1}{2} \sum_{\substack{\rho \in \binom{H}{d-1} \\ \mathsf{CH}(\rho) \cap \psi \neq \emptyset}} \left(\binom{\mathsf{l}(\rho)}{k-d+1} + \binom{\mathsf{r}(\rho)}{k-d+1} \right) \ .$$

Proof. Clearly, any embracing k-set is also an embracing k-set in the projection, so we only have to subtract the number of non-embracing k-sets which happen to be embracing in the projection. Let A be such a set. Since $\mathbf{0} \in \mathrm{proj}(\mathsf{CH}(A))$, we have $\mathsf{CH}(A) \cap \psi \neq \emptyset$. In the 2-dimensional subspace defined by ψ, there is a unique point t on the boundary of $\mathsf{CH}(A) \cap \psi$ such that $\mathsf{CH}(A) \cap \psi$ is in the left closed halfplane defined by $\mathbf{0}t$ (recall that we assume general position). Since ψ is generic, t is the intersection of ψ with a $(d-2)$-simplex defined by a tuple ρ of $d-1$ points of A, and the oriented $(d-1)$-simplex $\mathbf{0}\rho$ has all points of $A \setminus \rho$

either on its positive or negative side. We are thus counting each such non-embracing k-set twice (for the left and the right "tangent"), and the claim follows. ◀

With the previous lemma at hand, it is now a simple task to give a proof of our main computational result.

▶ **Theorem 4** (restated). *Let $d \geq 3$ be fixed and let H be a set of n points in $\mathbb{R}^d$. Then, the simplicial depth of a point w in H can be computed in $O(n^{d-1})$ time.*

Proof. The proof of Lemma 9 is constructive (apart from the choice of ψ, which can be done arbitrarily using the techniques in [8]). Whether a $(d-1)$-simplex intersects ψ can be computed in $O(d^d)$ time. It thus remains to compute the values of $\mathsf{I}(\rho)$ for the $(d-1)$-tuples ρ. While a brute-force approach would take $O(n^d)$ time for this operation, we can actually consider the points of H as vectors representing points in the $(d-1)$-dimensional projective plane. We compute the dual hyperplane arrangement [9] in $O(n^{d-1})$ time, which allows us to extract the values of $\mathsf{I}(\rho)$ within the same time bounds (as also discussed in [9]). ◀

Note that the part whose running time depends on d is computing the values of $\mathsf{I}(\rho)$. After $O(n^{d-1})$ time, we can produce a vector indicating the number of $(d-1)$-simplices intersecting ψ with a certain number of points on their positive side. Using this vector, we only have to sum up over $O(n)$ binomial coefficients in each recursion step to obtain the number of embracing k-sets.

4.2 Polytopes with few vertices

Through the so-called Gale transform (cf. [31, 32, 33]), origin-embracing triangles are in correspondence to facets ($(n-4)$-faces) of simplicial $(n-3)$-polytopes with at most n vertices. More generally, subsets of size i containing the origin in their convex hull correspond to $(n-i-1)$-faces. Therefore, some of our results connect to such simplicial d-dimensional polytopes with at most $d+3$ vertices (number of frequency vectors, number of order types, computation of number of embracing triangles, etc.) and thus to known results in that context.

Gale duality. For $n > d$, we call a matrix $A \in \mathbb{R}^{n\times d}$ *legal* if A has full rank d and $A^\top \mathbf{1}_n = \mathbf{0}_d$. Let $S_A = (p_1, p_2, \ldots, p_n)$ be the sequence of points in $\mathbb{R}^d$ with the coordinates of p_i obtained from the ith row of A. Legal thus means that S_A is not contained in a hyperplane and that the origin is the centroid of S_A. For legal matrices $A \in \mathbb{R}^{n\times d}$ and $B \in \mathbb{R}^{n\times n-d-1}$, we call B an *orthogonal dual* of A, in symbols $A \perp B$, if $A^\top B = \mathbf{0}_{d\times(n-d-1)}$. S_B is called a *Gale dual (Gale transform, Gale diagram) of* S_A.[5] In other words, if $A \perp B$ then all column vectors of B are orthogonal to all column vectors of A; together with the legal condition, this means that the column vectors of B span the space of all vectors orthogonal to the column vectors of A and to $\mathbf{1}_n$, i.e., it is a basis of the null space of the vector space spanned by the columns of $(A\mathbf{1}_n)$. (The matrix $(A\mathbf{1}_n)$ is the matrix A with an extra column of 1's.)

▶ **Proposition 10** ([20, p. 111]). *Let $A \perp B$ with $S_A = (p_1, p_2, \ldots, p_n)$, $S_B = (p_1^*, p_2^*, \ldots, p_n^*)$. For every $I \subseteq [n]$, the set $\{p_i \mid i \in I\}$ is contained in a facet of $\mathsf{CH}(S_A)$ iff $\{p_i^* \mid i \notin I\}$ is embracing.*

[5] Following [32], we add the requirement that the origin is the centroid, in contrast to, e.g., [20, Chapter 5.6].

For a d-dimensional polytope $\mathcal{P}$, the *f-vector of $\mathcal{P}$* is defined as $f(\mathcal{P}) = (f_{-1}, f_0, \ldots, f_{d-1})$, where $f_i(\mathcal{P})$ is the number of i-dimensional faces (i-faces) of $\mathcal{P}$ (the empty face is the unique (-1)-face, 0-faces are vertices, 1-faces are edges, …, $(d-1)$-faces are facets). A property of the Gale dual is that the points in S_A are in general position iff the rows in B are linearly independent [20, p. 111]. Thus, if $S := \{p_1, p_2, \ldots, p_n\}$ is a set of n points in general position, $\mathcal{P} := \mathsf{CH}(S)$, and Q is the set $\{p_1^*, p_2^*, \ldots, p_n^*\}$, then $f_i(\mathcal{P})$ equals the number of embracing $(n-i-1)$-sets in Q.[6] Computing the f-vector can thus be done by computing the Gale dual and by using Proposition 10.

▶ **Proposition 11.** *Given a legal matrix $A \in \mathbb{R}^{n \times d}$, an orthogonal dual can be computed in time $O(n^\omega)$, where ω is the exponent for matrix multiplication over $\mathbb{R}$.*

▶ **Corollary 12.** *For a set S of $n = d + k$ points in general position in $\mathbb{R}^d$, the number of facets of the simplicial polytope $\mathsf{CH}(S)$ can be computed in time $O(n^{k-2})$ for $k \geq 5$ and in $O(n^\omega)$ otherwise, where ω is the exponent for matrix multiplication over $\mathbb{R}$.*

Note that the asymptotic number of facets may be as large as n^k. A generalization of Corollary 12 to sets not necessarily in general position is possible for $k = 3$. Our efficient computation of the number of embracing k-sets thus lets us obtain not only the f-vector of a polytope, but of course related vectors like the h- and the g-vector. We finally draw the connection between order types of conowheel sets and the combinatorial structure of simplicial d-polytopes with $d + 3$ vertices.

▶ **Proposition 13.** *The family of embracing triangles of a conowheel set $P = H \cup \{w\}$ determines the order type of P.*

It is therefore no coincidence that the number obtained in Theorem 1 is the same as the one obtained by Perles for the number of simplicial d-polytopes with $d + 3$ vertices (see [15, Chapter 6.3]). Also, the number of f-vectors of polytopes with at most $d + 3$ vertices, as obtained by Linusson [17], equals the number of frequency vectors by Lemma 7; via the Gale dual, we thus obtain a direct proof for the number of these f-vectors, as desired by Linusson. Doing so for $d + 4$ vertices, however, remains an open problem.

References

1. Bernardo M. Ábrego and Silvia Fernández-Merchant. A lower bound for the rectilinear crossing number. *Graphs Combin.*, 21(3):293–300, 2005. doi:10.1007/s00373-005-0612-5.
2. Peyman Afshani, Donald R. Sheehy, and Yannik Stein. Approximating the simplicial depth. *CoRR*, abs/1512.04856, 2015.
3. Oswin Aichholzer, Thomas Hackl, Clemens Huemer, Ferran Hurtado, Hannes Krasser, and Birgit Vogtenhuber. On the number of plane geometric graphs. *Graphs Combin.*, 23:67–84, 2007. doi:10.1007/s00373-007-0704-5.
4. Esther M. Arkin, Samir Khuller, and Joseph S. B. Mitchell. Geometric knapsack problems. *Algorithmica*, 10(5):399–427, 1993. doi:10.1007/BF01769706.
5. Andries E. Brouwer. The enumeration of locally transitive tournaments. Technical Report ZW 138/80, Mathematisch Centrum, Amsterdam, 1980.
6. Andrew Y. Cheng and Ming Ouyang. On algorithms for simplicial depth. In *Proc. 13th Canadian Conference on Computational Geometry*, pages 53–56, 2001.

[6] Note that linear independence of the rows of B does not assure general position of Q, but projecting Q to the unit circle does, as no two points are collinear with $\mathbf{0}$.

7 Serge Dulucq and Jean-Guy Penaud. Cordes, arbres et permutations. *Discr. Math.*, 117(1):89–105, 1993. doi:10.1016/0012-365X(93)90326-O.

8 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.

9 Herbert Edelsbrunner, Joseph O'Rourke, and Raimund Seidel. Constructing arrangements of lines and hyperplanes with applications. *SIAM J. Comput.*, 15(2):341–363, 1986. doi:10.1137/0215024.

10 David Eppstein, Mark H. Overmars, Günter Rote, and Gerhard J. Woeginger. Finding minimum area k-gons. *Discr. Comput. Geom.*, 7:45–58, 1992. doi:10.1007/BF02187823.

11 Philippe Flajolet and Marc Noy. Analytic combinatorics of non-crossing configurations. *Discr. Math.*, 204(1-3):203–229, 1999. doi:10.1016/S0012-365X(98)00372-0.

12 Joseph Gil, William L. Steiger, and Avi Wigderson. Geometric medians. *Discr. Math.*, 108(1-3):37–51, 1992. doi:10.1016/0012-365X(92)90658-3.

13 Jacob E. Goodman and Richard Pollack. Multidimensional sorting. *SIAM J. Comput.*, 12(3):484–507, 1983.

14 Jacob E. Goodman and Richard Pollack. Semispaces of configurations, cell complexes of arrangements. *J. Combin. Theory Ser. A*, 37(3):257–293, 1984.

15 Branko Grünbaum. *Convex Polytopes*. Springer, 2nd edition, 2003.

16 Samir Khuller and Joseph S. B. Mitchell. On a triangle counting problem. *Inf. Process. Lett.*, 33(6):319–321, 1990. doi:10.1016/0020-0190(90)90217-L.

17 Svante Linusson. The number of M-sequences and f-vectors. *Combinatorica*, 19(2):255–266, 1999. doi:10.1007/s004930050055.

18 R. Y. Liu. On a notion of data depth based on random simplices. *Annals of Statistics*, 18:405–414, 1990.

19 László Lovász, Katalin Vesztergombi, Uli Wagner, and Emo Welzl. Convex quadrilaterals and k-sets. In *Towards a Theory of Geometric Graphs*, pages 139–148. AMS, Providence, 2004.

20 Jiří Matoušek. *Lectures on Discrete Geometry*. Springer, 2002.

21 Juan José Montellano-Ballesteros and Ricardo Strausz. Counting polytopes via the Radon complex. *J. Comb. Theory, Ser. A*, 106(1):109–121, 2004. doi:10.1016/j.jcta.2004.01.005.

22 Theodore S. Motzkin. Relations between hypersurface cross ratios, and a combinatorial formula for partitions of a polygon, for permanent preponderance, and for non-associative products. *Bull. Amer. Math. Soc.*, 54(4):352–360, 1948.

23 Edgar M. Palmer and Robert W. Robinson. Enumeration of self-dual configurations. *Pacific J. Math.*, 110(1):203–221, 1984.

24 Dana Randall, Günter Rote, Francisco Santos, and Jack Snoeyink. Counting triangulations and pseudo-triangulations of wheels. In *Proc. 13th Canadian Conference on Computational Geometry*, pages 149–152, 2001.

25 Peter J. Rousseeuw and Ida Ruts. Bivariate location depth. *J. Royal Stat. Soc. Ser. C*, 45(4):516–526, 1996.

26 Andres J. Ruiz-Vargas and Emo Welzl. Crossing-free perfect matchings in wheel point sets. Unpublished manuscript, September 2015.

27 Micha Sharir and Adam Sheffer. Counting triangulations of planar point sets. *Electr. J. Combin.*, 18(1), 2011.

28 Micha Sharir, Adam Sheffer, and Emo Welzl. Counting plane graphs: Perfect matchings, spanning cycles, and Kasteleyn's technique. *J. Comb. Theory Ser. A*, 120(4):777–794, 2013. doi:10.1016/j.jcta.2013.01.002.

29 Micha Sharir and Emo Welzl. On the number of crossing-free matchings, cycles, and partitions. *SIAM J. Comput.*, 36(3):695–720, 2006. doi:10.1137/050636036.

30 Uli Wagner. On the rectilinear crossing number of complete graphs. In *Proc. 14th Annual Symposium on Discrete Algorithms*, pages 583–588. ACM/SIAM, 2003.

31 Uli Wagner and Emo Welzl. A continuous analogue of the Upper Bound Theorem. *Discr. Comput. Geom.*, 26(2):205–219, 2001. doi:10.1007/s00454-001-0028-9.

32 Emo Welzl. Entering and leaving j-facets. *Discr. Comput. Geom.*, 25(3):351–364, 2001. doi:10.1007/s004540010085.

33 Günter M. Ziegler. *Lectures on Polytopes*. Springer, 1995.

Approximate Range Counting Revisited*†

Saladi Rahul

Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN, USA
sala0198@umn.edu

Abstract

We study range-searching for colored objects, where one has to count (approximately) the number of colors present in a query range. The problems studied mostly involve orthogonal range-searching in two and three dimensions, and the dual setting of rectangle stabbing by points. We present optimal and near-optimal solutions for these problems. Most of the results are obtained via reductions to the approximate uncolored version, and improved data-structures for them. An additional contribution of this work is the introduction of nested shallow cuttings.

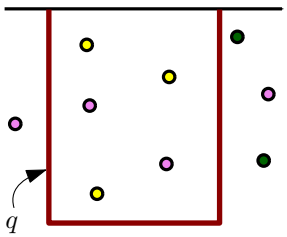
1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases geometric data structures, range searching, rectangle stabbing, approximate counting, colors

Digital Object Identifier 10.4230/LIPIcs.SoCG.2017.55

1 Introduction

Let S be a set of n geometric objects in $\mathbb{R}^d$ which are segregated into disjoint groups (i.e., *colors*). Given a query $q \subseteq \mathbb{R}^d$, a color c *intersects (or is present in)* q if any object in S of color c intersects q, and let k be the number of colors of S present in q.



In the *approximate colored range-counting problem*, the task is to preprocess S into a data structure, so that for a query q, one can efficiently report the *approximate* number of colors present in q. Specifically, return any value in the range $[(1-\varepsilon)k, (1+\varepsilon)k]$, where $\varepsilon \in (0,1)$ is a pre-specified parameter.

Colored range searching and its related problems have been studied before [8, 10, 11, 12]. They are known as GROUP-BY queries in the database literature. A popular variant is the *colored orthogonal range searching* problem: S is a set of n colored points in $\mathbb{R}^d$, and q is an axes-parallel rectangle. As a motivating example for this problem, consider the following query: "How many countries have employees aged between X_1 and X_2 while earning annually

* The full version of this work with the title "Approximate Range Counting Revisited" can be found on https://arxiv.org/abs/1512.01713v3.

† This research was partly supported by a Doctoral Dissertation Fellowship (DDF) from the Graduate School of University of Minnesota.

more than Y rupees?". An employee is represented as a colored point $(age, salary)$, where the color encodes the country, and the query is the axes-parallel rectangle $[X_1, X_2] \times [Y, \infty)$.

1.1 Previous work and background.

In the *standard* approximate range counting problem there are no colors. One is interested in the approximate number of objects intersecting the query. Specifically, if k is the number of objects of S intersecting q, then return a value in the range $[(1-\varepsilon)k, (1+\varepsilon)k]$.

General reduction to companion problems. Aronov and Har-Peled [3], and Kaplan, Ramos and Sharir [9] presented general techniques to answer approximate range counting queries. In both instances, the authors reduce the task of answering an approximate counting query, into answering a few queries in data-structures solving an easier *(companion)* problem. Aronov and Har-Peled's companion problem is the emptiness query, where the goal is to report whether $|S \cap q| = 0$. Specifically, assume that there is a data structure of size $S(n)$ which answers the emptiness query in $O(Q(n))$ time. Aronov and Har-Peled show that there is a data structure of size $O(S(n) \log n)$ which answers the approximate counting query in $O(Q(n) \log n)$ time (for simplicity we ignore the dependency on ε). Kaplan *et al.*'s companion problem is the range-minimum query, where each object of S has a weight associated with it and the goal is to report the object in $S \cap q$ with the minimum weight.

Even though the reductions of [3] and [9] seem different, there is an interesting discussion in Section 6 of [3] about the underlying "sameness" of both techniques.

Levels. Informally, for a set S of n objects, a *t-level* of S is a surface such that if a point q lies above (resp., on/below) the surface, then the number of objects of S containing q is $> t$ (resp., $\leq t$). Range counting can be reduced in some cases to deciding the level of a query point. Unfortunately, the complexity of a single level is not well understood. For example, for hyperplanes in the plane, the t-level has super-linear complexity $\Omega(n2^{\sqrt{\log t}})$ in the worst-case (the known upper bound is $O(nt^{1/3})$ and closing the gap is a major open problem). In particular, the prohibitive complexity of such levels makes them inapplicable for the approximate range counting problem, where one shoots for linear (or near-linear) space data-structures.

Shallow cuttings A *t-level shallow cutting* is a set of simple cells, that lies strictly below the $2t$-level, and their union covers all the points below (and on) the t-level. For many geometric objects in two and three dimensions, such t-shallow cuttings have $O(n/t)$ cells. Using such cuttings leads to efficient data-structures for approximate range counting. Specifically, one uses binary search on a "ladder" of approximate levels (realized via shallow cuttings) to find the approximation.

For halfspaces in $\mathbb{R}^3$, Afshani and Chan [1] avoid doing the binary search and find the two consecutive levels in optimal $O(\log \frac{n}{k})$ expected time. Later, Afshani, Hamilton and Zeh [2] obtained a worst-case optimal solution for many geometric settings. Interestingly, their results hold in the pointer machine model, the I/O-model and the cache-oblivious model. However, in the word-RAM model their solution is not optimal and the query time is $\Omega(\log\log U + (\log\log n)^2)$.

Specific problems. Approximate counting for orthogonal range searching in $\mathbb{R}^2$ was studied by Nekrich [11], and Chan and Wilkinson [5] in the word-RAM model. In this setting, the

input set is points in $\mathbb{R}^2$ and the query is a rectangle in $\mathbb{R}^2$. A hyper-rectangle in $\mathbb{R}^d$ is $(d+k)$-*sided* if it is bounded on both sides in k out of the d dimensions and unbounded on one side in the remaining $d-k$ dimensions. Nekrich [11] presented a data structure for approximate colored 3-sided range searching in $\mathbb{R}^2$, where the input is points and the query is a 3-sided rectangle in $\mathbb{R}^2$. However, it has an approximation factor of $(4+\varepsilon)$, whereas we are interested in obtaining a tighter approximation factor of $(1+\varepsilon)$. To the best of our knowledge, this is the only work directly addressing an approximate colored counting query.

1.2 Motivation

Avoiding expensive counting structures. A search problem is decomposable if given two disjoint sets of objects S_1 and S_2, the answer to $F(S_1 \cup S_2)$ can be computed in constant time, given the answers to $F(S_1)$ and $F(S_2)$ separately. This property is widely used in the literature for counting in standard problems (going back to the work of Bentley and Saxe [4] in the late 1970s). For colored counting problems, however, $F(\cdot)$ is not decomposable. If $F(S_1)$ (resp. $F(S_2)$) has n_1 (resp. n_2) colors, then this information is insufficient to compute $F(S_1 \cup S_2)$, as they might have common colors.

As a result, for *many exact* colored counting queries the known space and query time bounds are expensive. For example, for colored orthogonal range searching problem in $\mathbb{R}^d$, existing structures use $O(n^d)$ space to achieve polylogarithmic query time [10]. Any substantial improvement in the preprocessing time *and* the query time would lead to a substantial improvement in the best exponent of matrix multiplication [10] (which is a major open problem). Similarly, counting structures for colored halfspace counting in $\mathbb{R}^2$ and $\mathbb{R}^3$ [8] are expensive.

Instead of an exact count, if one is willing to settle for an approximate count, then this work presents a data structure with $O(n \text{ polylog } n)$ space and $O(\text{polylog } n)$ query time.

Approximate counting in the speed of emptiness. In an emptiness query, the goal is to decide if $S \cap q$ is empty. The approximate counting query is at least as hard as the emptiness query: When $k = 0$ and $k = 1$, no error is tolerated. Therefore, a natural goal while answering approximate range counting queries is to match the bounds of its corresponding *emptiness query*.

1.3 Our results and techniques

1.3.1 Specific problems

The focus of the paper is building data structures for approximate colored counting queries, which exactly match or *almost* match the bounds of their corresponding emptiness problem.

1.3.1.1 3-sided rectangle stabbing in 2d and related problems

In the colored interval stabbing problem, the input is n colored intervals with endpoints in $[\![U]\!] = \{1, \ldots, U\}$, and the query is a point in $[\![U]\!]$. We present a linear-space data structure which answers the approximate counting query in $O(\log \log U)$ time. The new data structure can be used to handle some geometric settings in 2d: the *colored dominance search* (the input is a set of n points, and the query is a 2-sided rectangle) and the *colored* 3*-sided rectangle stabbing* (the input is a set of n 3-sided rectangles, and the query is a point). The results are summarized in Table 1.

Table 1 A summary of the results obtained for several approximate colored counting queries. To avoid clutter, the $O(\cdot)$ symbol and the dependency on ε is not shown in the space and the query time bounds. For the second column in the table, the first entry is the input and the second entry is the query. For each of the results column in the table, the first entry is the space occupied by the data structure and the second entry is the time taken to answer the query. WR denotes the word-RAM model and PM denotes the pointer machine model.

Dime--nsion	Input, Query	New Results	Previous Approx. Counting Results	Exact Counting Results	Model
1	intervals, point	S: n, Q: $\log\log U$	S: n, Q: $\log\log U + (\log\log n)^2$	S: n, Q: $\log\log U + \log_w n$	WR
2	points, 2-sided rectangle				
2	3-sided rectangles, point	Theorem 1			
2	points, 3-sided rectangle	S: n, Q: $\log n$ Theorem 15(A)	S: $n\log^2 n$, Q: $\log^2 n$	not studied	PM
2	points, 4-sided rectangle	S: $n\log n$, Q: $\log n$ Theorem 15(B)	S: $n\log^3 n$, Q: $\log^2 n$	S: $n^2\log^6 n$, Q: $\log^7 n$ Kaplan *et al.* [10]	PM
3	points, 3-sided rectangle	S: $n\log^* n$, Q: $\log n \cdot \log\log n$ Theorem 7	S: $n\log^2 n$, Q: $\log^2 n$	not studied	PM

1.3.1.2 Range searching in $\mathbb{R}^2$

The input is a set of n colored points in the plane. For 3-sided query rectangles, an *optimal* solution (in terms of n) for approximate counting is obtained. For 4-sided query rectangles, an *almost-optimal* solution for approximate counting is obtained. The size of our data structure is off by a factor of $\log\log n$ w.r.t. its corresponding emptiness structure which occupies $O(n\frac{\log n}{\log\log n})$ space and answers the emptiness query in $O(\log n)$ time [6]. The results are summarized in Table 1.

1.3.1.3 Dominance search in $\mathbb{R}^3$

The input is a set of n colored points in $\mathbb{R}^3$ and the query is a 3-sided rectangle in $\mathbb{R}^3$ (i.e., an octant). An almost-optimal solution is obtained requiring $O(n\log\log n)$ space and $O(\log n)$ time to answer the approximate counting query.

1.3.2 General reductions

We present two general reductions for solving approximate colored counting queries by reducing them to "easy" companion queries.

Reduction-I (Reporting + C-approximation). In the first reduction a colored approximate counting query is answered using two companion structures: (a) *reporting structure* (its objective is to report the k colors), and (b) *C-approximation structure* (its objective is to

report any value z s.t. $k \in [z, Cz]$, where C is a constant). Significantly, unlike previous reductions [3, 9], there is *no asymptotic loss* of efficiency in space and query time bounds w.r.t. to the two companion problems.

Reduction-II (Only Reporting). The second reduction is a modification of the Aronov and Har-Peled [3] reduction. We present the reduction for the following reasons: (A) Unlike reduction-I, this reduction is "easier" to use since it uses only the reporting structure and avoids the C-approximation structure, and (B) the analysis of Aronov and Har-Peled is slightly complicated because of their insistence on querying emptiness structures. We show that by using reporting structures the analysis becomes simpler. This reduction is useful when the reporting query is not significantly costlier than the emptiness query. The full version of this work will describe this reduction and its applications.

1.3.3 Our techniques

The results are obtained via a non-trivial combination of several techniques. For example, (a) new reductions from colored problems to standard problems, (b) obtaining a linear-space data structure by performing random sampling on a super-linear-size data structure, (c) refinement of path-range trees of Nekrich [11] to obtain an optimal data structure for C-approximation of colored 3-sided range search in $\mathbb{R}^2$, and (d) *random sampling on colors* to obtain the two general reductions.

In addition, we introduce *nested shallow cuttings* for 3-sided rectangles in 2d. The idea of using a hierarchy of cuttings (or samples) is, of course, not new. However, for this specific setting, we get a hierarchy where there is no penalty for the different levels being compatible with each other. Usually, cells in the lower levels have to be clipped to cells in the higher levels of the hierarchy, leading to a degradation in performance. In our case, however, cells of the lower levels are fully contained in the cells of the level above it.

1.3.3.1 Paper organization

In Section 2, we present a solution to the colored 3-sided rectangle stabbing in 2d problem. In Section 3 we present a solution to the colored dominance search in $\mathbb{R}^3$ problem. In Section 4, the first general reduction is presented. In Section 5, the application of the first reduction to colored orthogonal range search in $\mathbb{R}^2$ problem is shown. Most of the proofs have been omitted and can be found in the full version.

2 3-sided Rectangle Stabbing in 2d

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

▶ **Theorem 1.** *Consider the following three colored geometric settings:*

1. ***Colored interval stabbing in 1d**, where the input is a set S of n colored intervals in one-dimension and the query q is a point. The endpoints of the intervals and the query point lie on a grid $[\![U]\!]$.*
2. ***Colored dominance search in 2d**, where the input is a set S of n colored points in 2d and the query q is a quadrant of the form $[q_x, \infty) \times [q_y, \infty)$. The input points and the point (q_x, q_y) lie on a grid $[\![U]\!] \times [\![U]\!]$.*
3. ***Colored 3-sided rectangle stabbing in 2d**, where the input is a set S of n colored 3-sided rectangles in 2d and the query q is a point. The endpoints of the rectangles and the query point lie on a grid $[\![U]\!] \times [\![U]\!]$.*

Then there exists an $O_\varepsilon(n)$ size word-RAM data structure which can answer an approximate counting query for these three settings in $O_\varepsilon(\log\log U)$ time. The notation $O_\varepsilon(\cdot)$ hides the dependency on ε.

Our strategy for proving this theorem is the following: In Subsection 2.1, we present a transformation of these three colored problems to the *standard* 3-sided rectangle stabbing in 2d problem. Then in Subsection 2.2, we construct nested shallow cuttings and use them to solve the standard 3-sided rectangle stabbing in 2d problem.

2.1 Transformation to a standard problem

From now on the focus will be on colored 3-sided rectangle stabbing in 2d problem, since the geometric setting of (1) and (2) in Theorem 1 are its special cases. We present a transformation of the colored 3-sided rectangle stabbing in 2d problem to the *standard* 3-sided rectangle stabbing in 2d problem.

Let $S_c \subseteq S$ be the set of 3-sided rectangles of a color c. In the preprocessing phase, we perform the following steps: (1) Construct a union of the rectangles of S_c. Call it $\mathcal{U}(S_c)$. (2) The vertices of $\mathcal{U}(S_c)$ include original vertices of S_c and some new vertices. Perform a *vertical decomposition* of $\mathcal{U}(S_c)$ by shooting a vertical ray upwards from every *new* vertex of $\mathcal{U}(S_c)$ till it hits $+\infty$. This leads to a decomposition of $\mathcal{U}(S_c)$ into $\Theta(|S_c|)$ pairwise-disjoint 3-sided rectangles. Call these new set of rectangles $\mathcal{N}(S_c)$.

Given a query point q, we can make the following two observations:

- If $S_c \cap q = \emptyset$, then $\mathcal{N}(S_c) \cap q = \emptyset$.
- If $S_c \cap q \neq \emptyset$, then exactly one rectangle in $\mathcal{N}(S_c)$ is stabbed by q.

Let $\mathcal{N}(S) = \bigcup_{\forall c} \mathcal{N}(S_c)$, and clearly, $|\mathcal{N}(S)| = O(n)$. Therefore, the colored 3-sided rectangle stabbing in 2d problem on S has been reduced to the *standard* 3-sided rectangle stabbing in 2d problem on $\mathcal{N}(S)$.

2.2 Standard 3-sided rectangle stabbing in 2d

In this subsection we will prove the following lemma.

▶ **Lemma 2** (Standard 3-sided rectangle stabbing in 2d)**.** *In this geometric setting, the input is a set S of n uncolored 3-sided rectangles of the form $[x_1, x_2] \times [y, \infty)$, and the query q is a point. The endpoints of the rectangles lie on a grid $[\![U]\!] \times [\![U]\!]$. There exists a data structure of size $O_\varepsilon(n)$ which can answer an approximate counting query in $O_\varepsilon(\log\log U)$ time.*

By a standard rank-space reduction, the rectangles of S can be projected to a $[\![2n]\!] \times [\![n]\!]$ grid: Let S_x (resp., S_y) be the list of the $2n$ vertical (resp., n horizontal) sides of S in increasing order of their $x-$ (resp., $y-$) coordinate value. Then each rectangle $r = [x_1, x_2] \times [y, \infty) \in S$ is projected to a rectangle $[rank(x_1), rank(x_2)] \times [rank(y), \infty)$, where $rank(x_i)$ (resp., $rank(y)$) is the index of x_i (resp., y) in the list S_x (resp., S_y). Given a query point $q \in [\![U]\!] \times [\![U]\!]$, we can use the van Emde Boas structure to perform a predecessor search on S_x and S_y in $O(\log\log U)$ time to find the position of q on the $[\![2n]\!] \times [\![n]\!]$ grid. Now we will focus on the new setting and prove the following result.

▶ **Lemma 3.** *For the standard 3-sided rectangle stabbing in 2d problem, consider a setting where the rectangles have endpoints lying on a grid $[\![2n]\!] \times [\![n]\!]$. Then there exists a data structure of size $O_\varepsilon(n)$ which can answer the approximate counting query in $O_\varepsilon(1)$ time.*

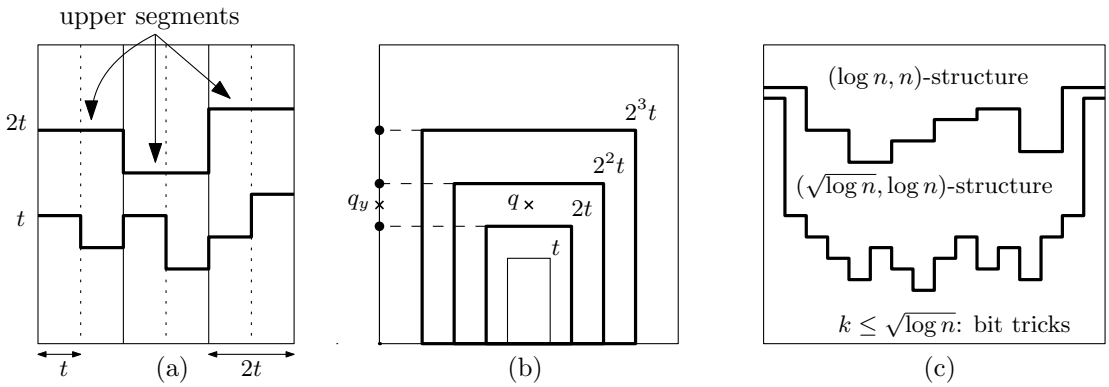


Figure 1 (a) A portion of the t-level and $2t$-level is shown. Notice that by our construction, each cell in the t-level is contained inside a cell in the $2t$-level. (b) A cell in the t-level and the set $\mathcal{C}_r$ associated with it. (c) A high-level summary of our data structure.

2.2.1 Nested shallow cuttings

To prove Lemma 3, we will first construct shallow cuttings for 3-sided rectangles in 2d. Unlike the general class of shallow cuttings, the shallow cuttings we construct for 3-sided rectangles will have a stronger property of cells in the lower level lying completely inside the cells of a higher level.

▶ **Lemma 4.** *Let S be a set of 3-sided rectangles (of the form $[x_1, x_2] \times [y, \infty)$) whose endpoints lie on a $[\![2n]\!] \times [\![n]\!]$ grid. A* t-level shallow cutting *of S produces a set $\mathcal{C}$ of interior-disjoint 3-sided rectangles/cells of the form $[x_1, x_2] \times (-\infty, y]$. There exists a set $\mathcal{C}$ with the following three properties:*

1. $|\mathcal{C}| = 2n/t$.
2. *If q does not lie inside any of the cell in $\mathcal{C}$, then $|S \cap q| \geq t$.*
3. *Each cell in $\mathcal{C}$ intersects at most $2t$ rectangles of S.*

Proof. Refer to the full version. ◀

▶ **Observation 5** (Nested Property)**.** *Let t and i be integers. Consider a t-level and a $2^i t$-level shallow cutting. By our construction, each cell in $2^i t$-level contains exactly 2^i cells of the t-level. More importantly, each cell in the t-level is contained inside a single cell of $2^i t$-level (see Figure 1(a)).*

2.2.2 Data structure

Now we will use nested shallow cuttings to find a constant-factor approximation for the 3-sided rectangle stabbing in 2d problem. In [2], the authors show how to convert a constant-factor approximation into a $(1+\varepsilon)$-approximation for this geometric setting. The solution is based on (t, t')-*level-structure* and $(\leq \sqrt{\log n})$-*level shared table.*

2.2.2.1 (t, t')-level structure

Let i, t and t' be integers s.t. $t' = 2^i t$. If $q(q_x, q_y)$ lies between the t-level and the t'-level cutting of S, then a (t, t')-level-structure will answer the approximate counting query in $O(1)$ time and occupy $O\left(n + \frac{n}{t} \log t'\right)$ space.

Structure. Construct a shallow cutting of S for levels $2^j t, \forall j \in [0, i]$. For each cell, say r, in the t-level we do the following: Let $\mathcal{C}_r$ be the set of cells from the $2^1 t, 2^2 t, 2^3 t, \ldots, 2^i t$-level, which contain r (Observation 5 guarantees this property). Now project the upper segment of each cell of $\mathcal{C}_r$ onto the y-axis (each segment projects to a point). Based on the y-coordinates of these $|\mathcal{C}_r|$ projected points build a fusion-tree [7]. Since there are $O(n/t)$ cells in the t-level and $|\mathcal{C}_r| = O(\log t')$, the total space occupied is $O(\frac{n}{t} \log t')$. See Figure 1(b).

Query algorithm. Since $q_x \in [\![2n]\!]$, it takes $O(1)$ time to find the cell r of the t-level whose x-range contains q_x. If the predecessor of q_y in $\mathcal{C}_r$ belongs to the $2^j t$-level, then $2^j t$ is a constant-factor approximation of k. The predecessor query also takes $O(1)$ time.

2.2.2.2 $(\leq \sqrt{\log n})$-level shared table

Suppose q lies in a cell in the $\sqrt{\log n}$-level shallow cutting of S. Then constructing the $(\leq \sqrt{\log n})$-level shared table will answer the exact counting query in $O(1)$ time. We will need the following lemma.

▶ **Lemma 6.** *For a cell c in the $\sqrt{\log n}$-level shallow cutting of S, its* conflict list S_c *is the set of rectangles of S intersecting c. Although the number of cells in the $\sqrt{\log n}$-level is $O\left(\frac{n}{\sqrt{\log n}}\right)$, the number of combinatorially "different" conflict lists is merely $O(\sqrt{n})$.*

Proof. Refer to the full version. ◀

Shared table. Construct a $\sqrt{\log n}$-level shallow cutting of S. For each cell c, perform a rank-space reduction of its conflict list S_c. Collect the combinatorially different conflict lists. On each conflict list, the number of combinatorially different queries will be only $O(|S_c|^2) = O(\log n)$. In a lookup table, for each pair of (S_c, q) we store the exact value of $|S_c \cap q|$. The total number of entries in the lookup table is $O(n^{1/2} \log n)$.

Query algorithm. Given a query $q(q_x, q_y)$, the following three $O(1)$ time operations are performed: (a) Find the cell c in the $\sqrt{\log n}$-level which contains q. If no such cell is found, then stop the query and conclude that $k \geq \sqrt{\log n}$. (b) Otherwise, perform a rank-space reduction on q_x and q_y to map it to the $[\![2|S_c|]\!] \times [\![|S_c|]\!]$ grid. Since, $|S_c| = O(\sqrt{\log n})$, we can build fusion trees [7] on S_c to perform the rank-space reduction in $O(1)$ time. (c) Finally, search for (S_c, q) in the lookup table and report the exact count.

2.2.2.3 Final structure

At first thought, one might be tempted to construct a $(0, n)$-level-structure. However, that would occupy $O(n \log n)$ space. The issue is that the (t, t')-level structure requires super-linear space for small values of t. Luckily, the $(\leq \sqrt{\log n})$-level shared table will efficiently handle the small values of t.

Therefore, the strategy is to construct the following: (a) a $(\leq \sqrt{\log n})$-level shared table, (b) a $(\sqrt{\log n}, \log n)$-level-structure, and (c) a $(\log n, n)$-level-structure. Now, the space occupied by all the three structures will be $O(n)$. See Figure 1(c) for a summary of our data structure.

3 Colored Dominance Search in $\mathbb{R}^3$

▶ **Theorem 7.** *In the colored dominance search in $\mathbb{R}^3$ problem, the input set S is n colored points in $\mathbb{R}^3$ and the query q is a point. Then there is a pointer machine data structure of size $O_\varepsilon(n \log^* n)$ which can answer an approximate colored counting query in $O_\varepsilon(\log n \cdot \log\log n)$ time. The notation $O_\varepsilon(\cdot)$ hides the dependency on ε.*

The strategy to prove this theorem is the following. First, we reduce the colored dominance search in $\mathbb{R}^3$ problem to a *standard* problem of 5-sided rectangle stabbing in $\mathbb{R}^3$. Then in the remaining section we solve the standard 5-sided rectangle stabbing in $\mathbb{R}^3$ problem.

3.1 Reduction to 5-sided rectangle stabbing in $\mathbb{R}^3$

In this subsection we present a reduction of colored dominance search in $\mathbb{R}^3$ problem to the standard 5-sided rectangle stabbing in $\mathbb{R}^3$ problem. Let S be a set of n colored points lying in $\mathbb{R}^3$. Let $S_c \subseteq S$ be the set of points of color c, and $p_1, p_2, \ldots, p_t$ be the points of S_c in decreasing order of their z-coordinate value. With each point $p_i(p_{ix}, p_{iy}, p_{iz})$, we associate a region ϕ_i in $\mathbb{R}^3$ which satisfies the following invariant: a point (x, y, z) belongs to ϕ_i if and only if in the region $[x, +\infty) \times [y, +\infty) \times [z, +\infty)$ the point of S_c with the largest z-coordinate is p_i. The following assignment of regions ensures the invariant:

- $\phi_1 = (-\infty, p_{1x}] \times (-\infty, p_{1y}] \times (-\infty, p_{1z}]$
- $\phi_i = (-\infty, p_{ix}] \times (-\infty, p_{iy}] \times (-\infty, p_{iz}] \setminus \bigcup_{j=1}^{i-1} \phi_j, \forall i \in [2, |S_c|]$.

By our construction, each region ϕ_i is unbounded in the negative z-direction. Each region ϕ_i is broken into disjoint 5-sided rectangles via *vertical decomposition* in the xy-plane. The vertical decomposition ensures that the total number of disjoint rectangles generated is bounded by $O(|S_c|)$. Now we can observe that (i) if a color c has at least one point inside q, then exactly one of its transformed rectangle will contain q, and (ii) if a color c has no point inside q, then none of its transformed rectangles will contain q. Therefore, the colored dominance search in $\mathbb{R}^3$ has been transformed to the standard 5-sided rectangle stabbing query.

3.2 Initial strcuture

▶ **Lemma 8.** *In the standard 5-sided rectangle stabbing in $\mathbb{R}^3$ problem, the input is a set S of n 5-sided rectangles in $\mathbb{R}^3$ and the query q is a point. Then there exists a pointer machine data structure of size $O_\varepsilon(n \log\log n)$ which can answer an approximate counting query in $O_\varepsilon(\log n \cdot \log\log n)$ time.*

The rest of the subsection is devoted to proving this lemma.

Recursion tree. Define a parameter $t = \log_{1+\varepsilon} n$. We will assume that the 5-sided rectangles are unbounded along the z-axis. Consider the projection of the rectangles of S on to the xy-plane and impose an orthogonal $[\![2\sqrt{\frac{n}{t}}]\!] \times [\![2\sqrt{\frac{n}{t}}]\!]$ grid such that each horizontal and vertical slab contains the projections of $\sqrt{nt}$ sides of S. Call this the root of the recursion tree. Next, for each vertical and horizontal slab, we recurse on the rectangles of S which are *sent* to that slab. At each node of the recursion tree, if we have m rectangles in the subproblem, then t is changed to $\log_{1+\varepsilon} m$ and the grid size changes to $[\![2\sqrt{\frac{m}{t}}]\!] \times [\![2\sqrt{\frac{m}{t}}]\!]$. We stop the recursion when a node has less than c rectangles, for a suitably large constant c.

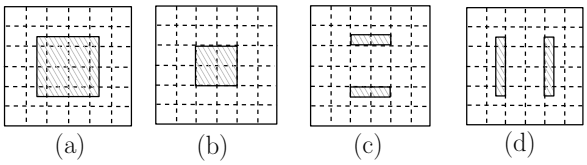


Figure 2

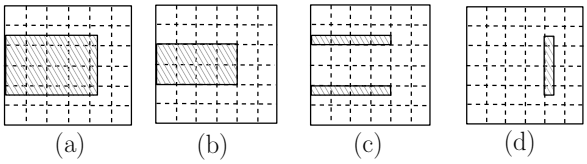


Figure 3

Assignment of rectangles. For a node in the tree, the intersection of every pair of horizontal and vertical grid line defines a *grid point*. Each rectangle of S is assigned to $O_\varepsilon(\log \log n)$ nodes in the tree. The assignment of a rectangle to a node is decided by the following three cases:

Case-I. The xy-projection of a rectangle intersects none of the grid points, i.e., it lies completely inside one of the row slab or/and the column slab. Then the rectangle is not assigned to this node, but sent to the child node corresponding to the row or column the rectangle lies in.

Case-II. The xy-projection of a rectangle r intersects at least one of the grid points. Let c_l and c_r be the leftmost and the rightmost column of the grid intersected by r. Similarly, let r_b and r_t be the bottommost and the topmost row of the grid intersected by r.

Then the rectangle is broken into at most five disjoint pieces: a *grid rectangle*, which is the bounding box of all the grid points lying inside r (see Figure 2(b)), two *column rectangles*, which are the portions of r lying in column c_l and c_r (see Figure 2(d)), and two *row rectangles*, which are the remaining portion of the rectangle r lying in row r_b and r_t (see Figure 2(c)). The grid rectangle is *assigned* to the node. Note that each column rectangle (resp., row rectangle) is now a 4-sided rectangle in $\mathbb{R}^3$ w.r.t. the column (resp., row) it lies in, and is sent to its corresponding child node.

Case-III. The xy-projection of a 4-*sided rectangle* r intersects at least one of the grid points. Without loss of generality, assume that the 4-sided rectangle r is unbounded along the negative x-axis. Then the rectangle is broken into at most four disjoint pieces: a *grid rectangle,* as shown in Figure 3(b), one *column rectangle*, as shown in Figure 3(d), and two *row rectangles*, as shown in Figure 3(c). The grid rectangle and the two row rectangles are *assigned* to the node. Note that the two row rectangles are now 3-sided rectangles in $\mathbb{R}^3$ w.r.t. their corresponding rows (unbounded in one direction along $x-$, $y-$ and $z-$axis). The column rectangle is sent to its corresponding child node. Analogous partition is performed for 4-sided rectangles which are unbounded along positive x-axis, positive y-axis and negative y-axis.

▶ **Observation 9.** *A rectangle of S gets assigned to at most four nodes at each level in the recursion tree.*

Proof. Consider a rectangle $r \in S$. If r falls under Case-II, then its grid rectangle is assigned to the node. Note that r can fall under Case-II only once, since each of its four row and column rectangles are now effectively 4-sided rectangles. Let r' be one of these row or column rectangles. If r' falls under Case-III at a node, then it gets assigned there. However, this time exactly *one* of the broken portion of r' will be sent to the child node. Therefore, there can be at most four nodes at each level where rectangle r (and broken portions of r) can get assigned. ◀

Data structures at each node. We build two types of structures at each node in the tree.

Structure-I. A rectangle r' is *higher* than rectangle r'' if r' has a larger span than r'' along z-direction. For each cell c of the grid, based on the rectangles which completely cover c, we construct a *sketch* as follows: select the rectangle with the $(1+\varepsilon)^0, (1+\varepsilon)^1, (1+\varepsilon)^2, \ldots$-th largest span. For a given cell, the size of the sketch will be $O(\log_{1+\varepsilon} m)$.

Structure-II. For a given row or column in the grid, let $\hat{S}$ be the 3-sided rectangles in $\mathbb{R}^3$ assigned to it. We build the linear-size structure of [2] on $\hat{S}$, which will return a $(1+\varepsilon)$-approximation of $|\hat{S} \cap q|$ in $O_\varepsilon(\log n)$ time. This structure is built for each row and column slab.

Space analysis. Consider a node in the recursion tree with m rectangles. There will be $\left(2\sqrt{\frac{m}{t}}\right) \times \left(2\sqrt{\frac{m}{t}}\right) = 4\frac{m}{t}$ cells at this node. The space occupied by structure-I will be $O\left(\frac{m}{t} \cdot \log_{1+\varepsilon} m\right) = O(m)$. The space occupied by structure-II will be $O(m)$. Using Observation 9, the total space occupied by all the nodes at a particular level will be $O(n)$. Since the height of the recursion tree is $O_\varepsilon(\log\log n)$, the total space occupied is $O_\varepsilon(n \log\log n)$.

Query algorithm. Given a query point q, we start at the root node. At each visited node, the following three steps are performed:

1. *Query structure-I.* Locate the cell c on the grid containing q. Scan the sketch of cell c to return a $(1+\varepsilon)$-approximation of the number of rectangles which cover c and contain q. This takes $O_\varepsilon(\log m)$ time.
2. *Query structure-II.* Next, query structure-II of the horizontal and the vertical slab containing q, to find a $(1+\varepsilon)$-approximation of the 3-sided rectangles containing q. This takes $O_\varepsilon(\log m)$ time.
3. *Recurse.* Finally, we recurse on the horizontal and the vertical slab containing q.

The final output is the *sum* of the count returned by all the nodes queried.

Query time analysis. Let $Q(n)$ denote the overall query time. Then

$$Q(n) = 2Q(\sqrt{nt}) + O_\varepsilon(\log n), t = \log_{1+\varepsilon} n.$$

This solves to $Q(n) = O_\varepsilon(\log n \cdot \log\log n)$. This finishes the proof of Lemma 8.

3.3 Final structure

▶ **Lemma 10.** *In the standard 5-sided rectangle stabbing in $\mathbb{R}^3$ problem, the input is a set S of n 5-sided rectangles in $\mathbb{R}^3$ and the query q is a point. Then there exists a pointer machine data structure of size $O_\varepsilon(n \log^* n)$ which can solve an approximate counting problem in $O_\varepsilon(\log n \cdot \log\log n)$ time.*

Refer to the full version for a proof of this lemma.

4 Reduction-I: Reporting + C-approximation

Our first reduction states that given a colored reporting structure and a colored C-approximation structure, one can obtain a colored $(1+\varepsilon)$-approximation structure with no additional loss of efficiency. We need a few definitions before stating the theorem. A geometric setting is *polynomially bounded* if there are only $n^{O(1)}$ possible outcomes of $S \cap q$, over all possible values of q. For example, in $1d$ orthogonal range search on n points, there are only $\Theta(n^2)$ possible outcomes of $S \cap q$. A function $f(n)$ is *converging* if $\sum_{i=0}^{t} n_i = n$, then $\sum_{i=0}^{t} f(n_i) = O(f(n))$. For example, it is easy to verify that $f(n) = n \log n$ is converging.

▶ **Theorem 11.** *For a colored geometric setting, assume that we are given the following two structures:*

- *a colored reporting structure of $\mathcal{S}_{rep}(n)$ size which can solve a query in $O(\mathcal{Q}_{rep}(n) + \kappa)$ time, where κ is the output-size, and*
- *a colored C-approximation structure of $\mathcal{S}_{capp}(n)$ size which can solve a query in $O(\mathcal{Q}_{capp}(n))$ time.*

We also assume that: (a) $\mathcal{S}_{rep}(n)$ and $\mathcal{S}_{capp}(n)$ are converging, and (b) the geometric setting is polynomially bounded. Then we can obtain a $(1+\varepsilon)$-approximation using a structure that requires $\mathcal{S}_{\varepsilon app}(n)$ space and $\mathcal{Q}_{\varepsilon app}(n)$ query time, such that

$$\mathcal{S}_{\varepsilon app}(n) = O(\mathcal{S}_{rep}(n) + \mathcal{S}_{capp}(n)) \tag{1}$$

$$\mathcal{Q}_{\varepsilon app}(n) = O\left(\mathcal{Q}_{rep}(n) + \mathcal{Q}_{capp}(n) + \varepsilon^{-2} \cdot \log n\right). \tag{2}$$

4.1 Refinement Structure

The goal of a refinement structure is to convert a constant-factor approximation of k into a $(1+\varepsilon)$-approximation of k.

▶ **Lemma 12** (Refinement structure)**.** *Let $\mathcal{C}$ be the set of colors in set S, and $\mathcal{C} \cap q$ be the set of colors in $\mathcal{C}$ present in q. For a query q, assume we know that:*

- *$k = |\mathcal{C} \cap q| = \Omega(\varepsilon^{-2} \log n)$, and*
- *$k \in [z, Cz]$, where z is an integer.*

Then there is a refinement structure of size $O\left(\mathcal{S}_{rep}\left(\frac{\varepsilon^{-2} n \log n}{z}\right)\right)$ which can report a value $\tau \in [(1-\varepsilon)k, (1+\varepsilon)k]$ in $O(\mathcal{Q}_{rep}(n) + \varepsilon^{-2} \log n)$ time.

The following lemma states that sampling colors (instead of input objects) is a useful approach to build the refinement structure.

▶ **Lemma 13.** *Consider a query q which satisfies the two conditions stated in Lemma 12. Let c_1 be a sufficiently large constant and c be another constant s.t. $c = \Theta(c_1 \log e)$. Choose a random sample R where each color in $\mathcal{C}$ is picked independently with probability $M = \frac{c_1 \varepsilon^{-2} \log n}{z}$. Then with probability $1 - n^{-c}$ we have $\left|k - \frac{|R \cap q|}{M}\right| \leq \varepsilon k$.*

Proof. Refer to the full version. ◀

▶ **Lemma 14** (Finding a suitable R)**.** *Pick a random sample R as defined in Lemma 13. Let n_R be the number of objects of S whose color belongs to R. We say R is* suitable *if it satisfies the following two conditions:*

- *$\left|k - \frac{|R \cap q|}{M}\right| \leq \varepsilon k$ for all queries which have $k = \Omega(\varepsilon^{-2} \log n)$.*
- *$n_R \leq 10nM$. This condition is needed to bound the size of the data structure.*

A suitable R always exists.

Proof. Refer to the full version. ◀

Refinement structure and query algorithm

In the preprocessing stage pick a random sample $R \subseteq \mathcal{C}$ as stated in Lemma 13. If the sample R is *not suitable*, then discard R and re-sample, till we get a suitable sample. Based on all the objects of S whose color belongs to R, build a colored reporting structure. Given a query q, the colored reporting structure is queried to compute $|R \cap q|$. We report $\tau \longleftarrow (|R \cap q|/M)$ as the final answer. The query time is bounded by $O(\mathcal{Q}_{rep}(n) + \varepsilon^{-2} \log n)$, since by Lemma 13, $|R \cap q| \leq (1+\varepsilon) \cdot kM = O(\varepsilon^{-2} \log n)$. This finishes the description of the refinement structure.

4.2 Overall solution

Data structure

The data structure consists of the following three components:

1. *Reporting structure.* Based on the set S we build a colored reporting structure. This occupies $O(\mathcal{S}_{rep}(n))$ space.
2. *$\sqrt{C}$-approximation structure.* Based on the set S we build a $\sqrt{C}$-approximation structure. The choice of $\sqrt{C}$ will become clear in the analysis. This occupies $O(\mathcal{S}_{capp}(n))$ space.
3. *Refinement structures.* Build the refinement structure of Lemma 12 for the values $z = (\sqrt{C})^i \cdot \varepsilon^{-2} \log n, \forall i \in \left[0, \log_{\sqrt{C}} \left(\lceil \varepsilon^2 n \rceil\right)\right]$. The total size of all the refinement structures will be $\sum O\left(\mathcal{S}_{rep}(nM)\right) = O(\mathcal{S}_{rep}(n))$, since $\mathcal{S}_{rep}(\cdot)$ is converging and $\sum nM = O(n)$. Note that our choice of z ensures that the size of the data structure is independent of ε.

Query algorithm

The query algorithm performs the following steps:

1. Given a query object q, the colored reporting structure reports the colors present in $S \cap q$ till all the colors have been reported or $\varepsilon^{-2} \log n + 1$ colors have been reported. If the first event happens, then the exact value of k is reported. Otherwise, we conclude that $k = \Omega(\varepsilon^{-2} \log n)$. This takes $O(\mathcal{Q}_{rep}(n) + \varepsilon^{-2} \log n)$ time.
2. If $k > \varepsilon^{-2} \log n$, then
   a. First, query the $\sqrt{C}$-approximation structure. Let k_a be the $\sqrt{C}$-approximate value returned s.t. $k \in [k_a, \sqrt{C} k_a]$. This takes $O(\mathcal{Q}_{capp}(n))$ time.
   b. Then query the refinement structure with the largest value of z s.t. $z \leq k_a \leq \sqrt{C} z$. It is trivial to verify that $k \in [z, Cz]$. This takes $O(\mathcal{Q}_{rep}(n) + \varepsilon^{-2} \log n)$ time.

5 Colored Orthogonal Range Search in $\mathbb{R}^2$

To illustrate an application of Reduction-I, we study the approximate colored counting query for orthogonal range search in $\mathbb{R}^2$. We only prove Theorem 15(1) here. Refer to the full version for the proof of Theorem 15(2).

▶ **Theorem 15.** *Consider the following two problems:*

1. **Colored 3-sided range search in $\mathbb{R}^2$.** *In this setting, the input set S is n colored points in $\mathbb{R}^2$ and the query q is a 3-sided rectangle in $\mathbb{R}^2$. There is a data structure of $O(n)$ size which can answer the approximate colored counting query in $O(\varepsilon^{-2} \log n)$ time. This pointer machine structure is optimal in terms of n.*
2. **Colored 4-sided range search in $\mathbb{R}^2$.** *In this setting, the input set S is n colored points in $\mathbb{R}^2$ and the query q is a 4-sided rectangle in $\mathbb{R}^2$. There is a data structure of $O(n \log n)$ size which can answer the approximate colored counting query in $O(\varepsilon^{-2} \log n)$ time.*

5.1 Colored 3-sided range search in $\mathbb{R}^2$

We use the framework of Theorem 11 to prove the result of Theorem 15(1). For this geometric setting, a colored reporting structure with $\mathcal{S}_{rep} = n$ and $\mathcal{Q}_{rep} = \log n$ is already known [12]. The path-range tree of Nekrich [11] gives a $(4+\varepsilon)$-approximation, but it requires super-linear space. The C-approximation structure presented in this subsection is a refinement of the path-range tree for the pointer machine model.

▶ **Lemma 16.** *For the colored 3-sided range search in $\mathbb{R}^2$ problem, there is a C-approximation structure which requires $O(n)$ space and answers a query in $O(\log n)$ time.*

We prove Lemma 16 in the rest of this subsection. Our solution is based on an interval tree and we will need the following fact about it.

▶ **Lemma 17.** *Using interval trees, a query on $(3+t)$-sided rectangles in $\mathbb{R}^3$ can be broken down into $O(\log n)$ queries on $(2+t)$-sided rectangles in $\mathbb{R}^3$. Here $t \in [1,3]$.*

Proof. Refer to the full version. ◀

5.1.1 Initial structure

▶ **Lemma 18.** *For the colored 3-sided range search in $\mathbb{R}^2$ problem, there is a 2-approximation structure which requires $O(n)$ space and answers a query in $O(\log^3 n)$ time.*

Proof. By a simple exercise, the colored 3-sided range search in $\mathbb{R}^2$ can be reduced to the colored dominance search in $\mathbb{R}^3$. Therefore, using the reduction of Subsection 3.1 the colored 3-sided range search in $\mathbb{R}^2$ also reduces to standard 5-sided rectangle stabbing problem (for brevity, call it 5-sided RSP).

There is a simple linear-size data structure which reports in $O(\log^3 n)$ time a 2-approximation for the 5-sided RSP: By inductively applying Lemma 17 twice, we can decompose 5-sided RSP to $O(\log^2 n)$ 3-sided RSPs. For 3-sided RSP, there is a linear-size structure of which reports a 2-approximation in $O(\log n)$ time [2]. By using this structure the 5-sided RSP can be solved in $O(\log^3 n)$ time. ◀

5.1.2 Final structure

Now we will present the optimal C-approximation structure of Lemma 16.

Structure. Sort the points of S based on their x-coordinate value and divide them into buckets containing $\log^2 n$ consecutive points. Based on the points in each bucket, build a D-structure which is an instance of Lemma 18. Next, build a height-balanced binary search tree $\mathcal{T}$, where the buckets are placed at the leaves from left to right based on their ordering along the x-axis. Let v be a proper ancestor of a leaf node u and let $\Pi(u,v)$ be the path from u to v (excluding u and v). Let $S_l(u,v)$ be the set of points in the subtrees rooted at nodes that are left children of nodes on the path $\Pi(u,v)$ but not themselves on the path. Similarly, let $S_r(u,v)$ be the set of points in the subtrees rooted at nodes that are right children of nodes on the path $\Pi(u,v)$ but not themselves on the path. For each pair (u,v), let $S'_l(u,v)$ (resp., $S'_r(u,v)$) be the set of points that each have the highest y-coordinate value among the points of the same color in $S_l(u,v)$ (resp., $S_r(u,v)$).

Finally, for each pair (u,v), construct a *sketch*, $S''_l(u,v)$, by selecting the $2^0, 2^1, 2^2, \ldots$-th highest y-coordinate point in $S'_l(u,v)$. A symmetric construction is performed to obtain $S''_r(u,v)$. The number of (u,v) pairs is bounded by $O((n/\log^2 n) \times (\log n)) = O(n/\log n)$ and hence, the space occupied by all the $S''_l(u,v)$ and $S''_r(u,v)$ sets is $O(n)$.

Query algorithm. To answer a query $q = [x_1, x_2] \times [y, \infty)$, we first determine the leaf nodes u_l and u_r of $\mathcal{T}$ containing x_1 and x_2, respectively. If $u_l = u_r$, then we query the D-structure corresponding to the leaf node and we are done. If $u_l \neq u_r$, then we find the node v which is the least common ancestor of u_l and u_r. The query is now broken into four sub-queries: First, report the approximate count in the leaves u_l and u_r by querying the D-structure of u_l with $[x_1, \infty) \times [y, \infty)$ and the D-structure of u_r with $(-\infty, x_2] \times [y, \infty)$. Next, scan the list $S''_r(u_l, v)$ (resp., $S''_l(u_r, v)$) to find a 2-approximation of the number of colors of $S_r(u_l, v)$ (resp., $S_l(u_r, v)$) present in q.

The final answer is the sum of the count returned by the four sub-queries. The time taken to find u_l, u_r and v is $O(\log n)$. Querying the leaf structures takes $O((\log(\log^2 n))^3) = O(\log n)$ time. The time taken for scanning the lists $S''_r(u_l, v)$ and $S''_l(u_r, v)$ is $O(\log n)$. Therefore, the overall query time is bounded by $O(\log n)$. Since each of the four sub-queries give a 2-approximation, overall we get a 8-approximation.

References

1. Peyman Afshani and Timothy M. Chan. On approximate range counting and depth. *Discrete & Computational Geometry*, 42(1):3–21, 2009.
2. Peyman Afshani, Chris H. Hamilton, and Norbert Zeh. A general approach for cache-oblivious range reporting and approximate range counting. *Computational Geometry: Theory and Applications*, 43(8):700–712, 2010.
3. Boris Aronov and Sariel Har-Peled. On approximating the depth and related problems. *SIAM Journal of Computing*, 38(3):899–921, 2008.
4. Jon Louis Bentley and James B. Saxe. Decomposable searching problems I: Static-to-dynamic transformation. *Journal of Algorithms*, 1(4):301–358, 1980.
5. Timothy M. Chan and Bryan T. Wilkinson. Adaptive and approximate orthogonal range counting. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 241–251, 2013.
6. Bernard Chazelle. Filtering search: A new approach to query-answering. *SIAM Journal of Computing*, 15(3):703–724, 1986.
7. Michael L. Fredman and Dan E. Willard. Surpassing the information theoretic bound with fusion trees. *Journal of Computer and System Sciences (JCSS)*, 47(3):424–436, 1993.
8. Prosenjit Gupta, Ravi Janardan, and Michiel H. M. Smid. Computational geometry: Generalized intersection searching. In *Handbook of Data Structures and Applications*. 2004.
9. Haim Kaplan, Edgar Ramos, and Micha Sharir. Range minima queries with respect to a random permutation, and approximate range counting. *Discrete & Computational Geometry*, 45(1):3–33, 2011.
10. Haim Kaplan, Natan Rubin, Micha Sharir, and Elad Verbin. Counting colors in boxes. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 785–794, 2007.
11. Yakov Nekrich. Efficient range searching for categorical and plain data. *ACM Transactions on Database Systems (TODS)*, 39(1):9, 2014.
12. Qingmin Shi and Joseph JáJá. Optimal and near-optimal algorithms for generalized intersection reporting on pointer machines. *Information Processing Letters (IPL)*, 95(3):382–388, 2005.

Coloring Curves That Cross a Fixed Curve*

Alexandre Rok[1] and Bartosz Walczak[2]

1 **Department of Mathematics, Ben-Gurion University of the Negev, Be'er Sheva, Israel**
rok@math.bgu.ac.il

2 **Department of Theoretical Computer Science, Faculty of Mathematics and Computer Science, Jagiellonian University, Kraków, Poland**
walczak@tcs.uj.edu.pl

Abstract

We prove that for every integer $t \geqslant 1$, the class of intersection graphs of curves in the plane each of which crosses a fixed curve in at least one and at most t points is χ-bounded. This is essentially the strongest χ-boundedness result one can get for this kind of graph classes. As a corollary, we prove that for any fixed integers $k \geqslant 2$ and $t \geqslant 1$, every k-quasi-planar topological graph on n vertices with any two edges crossing at most t times has $O(n \log n)$ edges.

1998 ACM Subject Classification I.3.5 Computational Geometry and Object Modeling

Keywords and phrases String graphs, χ-boundedness, k-quasi-planar graphs

Digital Object Identifier 10.4230/LIPIcs.SoCG.2017.56

1 Introduction

Overview

A *curve* is a homeomorphic image of the real interval $[0, 1]$ in the plane. The *intersection graph* of a family of curves has these curves as vertices and the intersecting pairs of curves as edges. Combinatorial and algorithmic aspects of intersection graphs of curves, known as *string graphs*, have been attracting researchers for decades. A significant part of this research has been devoted to understanding classes of string graphs that are χ-*bounded*, which means that every graph G in the class satisfies $\chi(G) \leqslant f(\omega(G))$ for some function $f\colon \mathbb{N} \to \mathbb{N}$, where $\chi(G)$ and $\omega(G)$ denote the chromatic number and the clique number (the maximum size of a clique) of G, respectively. Recently, Pawlik et al. [24, 25] proved that the class of all string graphs is not χ-bounded. However, all known constructions of string graphs with small clique number and large chromatic number require a lot of freedom in placing curves around in the plane.

What restrictions on placement of curves lead to χ-bounded classes of intersection graphs? McGuinness [19, 20] proposed studying families of curves that cross a fixed curve *exactly once*. This initiated a series of results culminating in the proof that the class of intersection graphs of such families is indeed χ-bounded [26]. By contrast, the class of intersection graphs of curves each crossing a fixed curve *at least once* is equal to the class of all string graphs and therefore is not χ-bounded. We prove an essentially farthest possible generalization of the former result, allowing curves to cross the fixed curve *at least once and at most t times*, for any bound t.

▶ **Theorem 1.** *For every integer $t \geqslant 1$, the class of intersection graphs of curves each crossing a fixed curve in at least one and at most t points is χ-bounded.*

* Alexandre Rok was partially supported by Israel Science Foundation grant 1136/12. Bartosz Walczak was partially supported by National Science Center of Poland grant 2015/17/D/ST1/00585.

Additional motivation for Theorem 1 comes from its application to bounding the number of edges in so-called k-quasi-planar graphs, which we discuss at the end of this introduction.

Context

Chromatic number of intersection graphs of geometric objects has been investigated since the 1960s. In a seminal paper, Asplund and Grünbaum [3] proved that intersection graphs of axis-parallel rectangles in the plane satisfy $\chi = O(\omega^2)$ and conjectured that for every integer $d \geqslant 1$, there is a function $f_d\colon \mathbb{N} \to \mathbb{N}$ such that intersection graphs of axis-parallel boxes in $\mathbb{R}^d$ satisfy $\chi \leqslant f_d(\omega)$. However, a few years later, a surprising construction due to Burling [5] showed that there are triangle-free intersection graphs of axis-parallel boxes in $\mathbb{R}^3$ with arbitrarily large chromatic number. Since then, the upper bound of $O(\omega^2)$ and the trivial lower bound of $\Omega(\omega)$ on the maximum possible chromatic number of a rectangle intersection graph have been improved only in terms of multiplicative constants [11, 13].

Another classical example of a χ-bounded class of geometric intersection graphs is provided by circle graphs—intersection graphs of chords of a fixed circle. Gyárfás [10] proved that circle graphs satisfy $\chi = O(\omega^2 4^\omega)$. The best known upper and lower bounds on the maximum possible chromatic number of a circle graph are $O(2^\omega)$ [14] and $\Omega(\omega \log \omega)$ [12].

McGuinness [19, 20] proposed investigating the problem when much more general geometric shapes are allowed but the way how they are arranged in the plane is restricted. In [19], he proved that the class of intersection graphs of L-shapes crossing a fixed horizontal line is χ-bounded. Families of L-shapes in the plane are *simple*, which means that any two members of the family intersect in at most one point. McGuinness [20] also showed that triangle-free intersection graphs of simple families of curves each crossing a fixed line in exactly one point have bounded chromatic number. Further progress in this direction was made by Suk [27], who proved that simple families of x-monotone curves crossing a fixed vertical line give rise to a χ-bounded class of intersection graphs, and by Lasoń et al. [17], who reached the same conclusion without assuming that the curves are x-monotone. Finally, in [26], we proved that the class of intersection graphs of curves each crossing a fixed line in exactly one point is χ-bounded. These results remain valid if the fixed straight line is replaced by a fixed curve [28].

The class of string graphs is not χ-bounded. Pawlik et al. [24, 25] presented a construction of triangle-free intersection graphs of segments (or geometric shapes of various other kinds) with chromatic number growing as fast as $\Theta(\log \log n)$ with the number of vertices n. It was further generalized to a construction of string graphs with clique number ω and chromatic number $\Theta_\omega((\log \log n)^{\omega-1})$ [16]. The best known upper bound on the chromatic number of string graphs in terms of the number of vertices is $(\log n)^{O(\log \omega)}$, proved by Fox and Pach [8] using a separator theorem for string graphs due to Matoušek [18]. For intersection graphs of segments or, more generally, x-monotone curves, an upper bound of the form $\chi = O_\omega(\log n)$ follows from the above-mentioned result in [27] or [26] via recursive halving. Upper bounds of the form $\chi = O_\omega((\log \log n)^{f(\omega)})$ (for some function $f\colon \mathbb{N} \to \mathbb{N}$) are known for very special classes of string graphs: rectangle overlap graphs [15, 16] and subtree overlap graphs [16]. The former still allow the triangle-free construction with $\chi = \Theta(\log \log n)$ and the latter the construction with $\chi = \Theta_\omega((\log \log n)^{\omega-1})$.

Quasi-planarity

A *topological graph* is a graph with a fixed curvilinear drawing in the plane. For $k \geqslant 2$, a *k-quasi-planar graph* is a topological graph with no k pairwise crossing edges. In particular, a 2-quasi-planar graph is just a planar graph. It is conjectured that k-quasi-planar graphs with

n vertices have $O_k(n)$ edges [4, 23]. For $k = 2$, this asserts a well-known property of planar graphs. The conjecture is also verified for $k = 3$ [2, 22] and $k = 4$ [1], but it remains open for $k \geqslant 5$. Best known upper bounds on the number of edges in a k-quasi-planar graph are $n(\log n)^{O(\log k)}$ in general [7, 8], $O_k(n \log n)$ for the case of x-monotone edges [29], $O_k(n \log n)$ for the case that any two edges intersect at most once [28], and $2^{\alpha(n)^\nu} n \log n$ for the case that any two edges intersect in at most t points, where α is the inverse Ackermann function and ν depends on k and t [28]. We apply Theorem 1 to improve the last bound to $O_{k,t}(n \log n)$.

▶ **Theorem 2.** *Every k-quasi-planar topological graph G on n vertices such that any two edges of G intersect in at most t points has at most $\mu_{k,t} n \log n$ edges, where $\mu_{k,t}$ depends only on k and t.*

The proof follows the same line as the proof in [28] for the case $t = 1$ (see Section 3).

2 Proof of Theorem 1

Setup

Let $\mathbb{N}$ denote the set of positive integers. Graph-theoretic terms applied to a family of curves $\mathcal{F}$ have the same meaning as applied to the intersection graph of $\mathcal{F}$. In particular, the *chromatic number* of $\mathcal{F}$, denoted by $\chi(\mathcal{F})$, is the minimum number of colors in a *proper coloring* of $\mathcal{F}$ (a coloring that distinguishes pairs of intersecting curves), and the *clique number* of $\mathcal{F}$, denoted by $\omega(\mathcal{F})$, is the maximum size of a *clique* in $\mathcal{F}$ (a set of pairwise intersecting curves in $\mathcal{F}$).

▶ **Theorem 1** (rephrased)**.** *For every $t \in \mathbb{N}$, there is a non-decreasing function $f_t \colon \mathbb{N} \to \mathbb{N}$ with the following property: for any fixed curve c_0, every family $\mathcal{F}$ of curves each intersecting c_0 in at least one and at most t points satisfies $\chi(\mathcal{F}) \leqslant f_t(\omega(\mathcal{F}))$.*

A point p is a *proper crossing* of curves c_1 and c_2 if c_1 passes from one side to the other side of c_2 in a sufficiently small neighborhood of p. From now on, without significant loss of generality, we make the following implicit assumption: any two distinct curves that we consider intersect in finitely many points, and each of their intersection points is a proper crossing. There is one exception to the latter condition: a curve c may have an endpoint on another curve if this is required by the definition of c (like for 1-curves defined below).

Initial reduction

We start by reducing Theorem 1 to a somewhat simpler and more convenient setting. We fix a horizontal line in the plane and call it the *baseline*. The upper half-plane bounded by the baseline is denoted by H^+. A *1-curve* is a curve in H^+ that has one endpoint on the baseline and does not intersect the baseline in any other point. Intersection graphs of 1-curves are known as *outerstring graphs* and form a χ-bounded class of graphs—this result, due to the authors, is the starting point of the proof of Theorem 1.

▶ **Theorem 3** ([26])**.** *There is a non-decreasing function $f_0 \colon \mathbb{N} \to \mathbb{N}$ such that every family $\mathcal{F}$ of 1-curves satisfies $\chi(\mathcal{F}) \leqslant f_0(\omega(\mathcal{F}))$.*

An *even-curve* is a curve that has both endpoints above the baseline and intersects the baseline in at least two points (this is an even number, by the proper crossing assumption). For $t \in \mathbb{N}$, a *$2t$-curve* is an even-curve that intersects the baseline in exactly $2t$ points. The *basepoint* of a 1-curve s is the endpoint of s on the baseline. A *basepoint* of an even-curve c

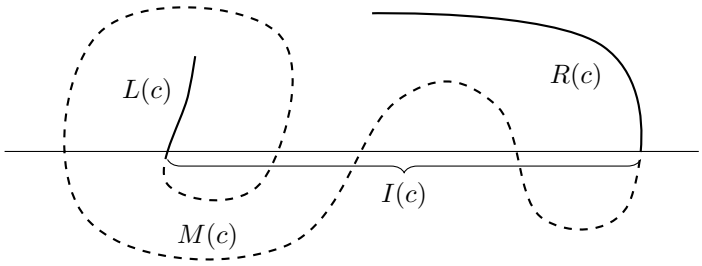


Figure 1 $L(c)$, $R(c)$, $M(c)$ (all the dashed part), and $I(c)$ for a 6-curve c.

is an intersection point of c with the baseline. Every even-curve c determines two 1-curves—the two parts of c from an endpoint to the closest basepoint. They are called the 1-*curves of* c and denoted by $L(c)$ and $R(c)$ so that the basepoint of $L(c)$ lies to the left of the basepoint of $R(c)$ on the baseline (see Figure 1). A family $\mathcal{F}$ of even-curves is an LR-*family* if every intersection between two curves $c_1, c_2 \in \mathcal{F}$ is an intersection between $L(c_1)$ and $R(c_2)$ or between $L(c_2)$ and $R(c_1)$. The main effort in this paper goes to proving the following statement on LR-families of even-curves.

▶ **Theorem 4.** *There is a non-decreasing function $f\colon \mathbb{N} \to \mathbb{N}$ such that every LR-family $\mathcal{F}$ of even-curves satisfies $\chi(\mathcal{F}) \leqslant f(\omega(\mathcal{F}))$.*

Theorem 4 makes no assumption on the maximum number of intersection points of an even-curve with the baseline. We derive Theorem 1 from Theorem 4 in two steps, first proving the following lemma, and then showing that Theorem 1 is essentially a special case of it.

▶ **Lemma 5.** *For every $t \in \mathbb{N}$, there is a non-decreasing function $f_t\colon \mathbb{N} \to \mathbb{N}$ such that every family $\mathcal{F}$ of $2t$-curves no two of which intersect on or below the baseline satisfies $\chi(\mathcal{F}) \leqslant f_t(\omega(\mathcal{F}))$.*

Proof of Lemma 5 from Theorem 4. The proof goes by induction on t. Let f_0 and f be the functions claimed by Theorem 3 and Theorem 4, respectively, and let $f_t(k) = f_{t-1}^2(k)f(k)$ for $t \geqslant 1$ and $k \in \mathbb{N}$. We establish the base case for $t = 1$ and the induction step for $t \geqslant 2$ simultaneously. Namely, fix an integer $t \geqslant 1$, and let $\mathcal{F}$ be as in the statement of the lemma. For every $2t$-curve $c \in \mathcal{F}$, enumerate the endpoints and basepoints of c as $p_0(c), \dots, p_{2t+1}(c)$ in their order along c so that $p_0(c)$ and $p_1(c)$ are the endpoints of $L(c)$ while $p_{2t}(c)$ and $p_{2t+1}(c)$ are the endpoints of $R(c)$. Build two families of curves $\mathcal{F}_1$ and $\mathcal{F}_2$ putting the part of c from $p_0(c)$ to $p_{2t-1}(c)$ to $\mathcal{F}_1$ and the part of c from $p_2(c)$ to $p_{2t+1}(c)$ to $\mathcal{F}_2$ for every $c \in \mathcal{F}$. If $t = 1$, then $\mathcal{F}_1$ and $\mathcal{F}_2$ are families of 1-curves. If $t \geqslant 2$, then $\mathcal{F}_1$ and $\mathcal{F}_2$ are equivalent to families of $2(t-1)$-curves, because the curve in $\mathcal{F}_1$ or $\mathcal{F}_2$ obtained from a $2t$-curve $c \in \mathcal{F}$ can be shortened a little at $p_{2t-1}(c)$ or $p_2(c)$, respectively, losing that basepoint but no intersection points with other curves. Therefore, by Theorem 3 or the induction hypothesis, we have $\chi(\mathcal{F}_k) \leqslant f_{t-1}(\omega(\mathcal{F}_k)) \leqslant f_{t-1}(\omega(\mathcal{F}))$ for $k \in \{1, 2\}$. For $c \in \mathcal{F}$ and $k \in \{1, 2\}$, let $\phi_k(c)$ be the color of the curve obtained from c in an optimal proper coloring of $\mathcal{F}_k$. Every subfamily of $\mathcal{F}$ on which ϕ_1 and ϕ_2 are constant is an LR-family and therefore, by Theorem 4 and monotonicity of f, has chromatic number at most $f(\omega(\mathcal{F}))$. We conclude that $\chi(\mathcal{F}) \leqslant \chi(\mathcal{F}_1)\chi(\mathcal{F}_2)f(\omega(\mathcal{F})) \leqslant f_{t-1}^2(\omega(\mathcal{F}))f(\omega(\mathcal{F})) = f_t(\omega(\mathcal{F}))$. ◀

A *closed curve* is a homeomorphic image of a unit circle in the plane. For a closed curve γ, the Jordan curve theorem asserts that the set $\mathbb{R}^2 \setminus \gamma$ consists of two connected components: one bounded, denoted by $\operatorname{int}\gamma$, and one unbounded, denoted by $\operatorname{ext}\gamma$.

Proof of Theorem 1 from Theorem 4. We elect to present this proof in an intuitive rather than rigorous way. Let $\mathcal{F}$ be a family of curves each intersecting c_0 in at least one and at most t points. Let γ_0 be a closed curve surrounding c_0 very closely so that γ_0 intersects every curve in $\mathcal{F}$ in exactly $2t$ points (winding if necessary to increase the number of intersections) and all endpoints of curves in $\mathcal{F}$ and intersection points of pairs of curves in $\mathcal{F}$ lie in $\operatorname{ext}\gamma_0$. We "invert" $\operatorname{int}\gamma_0$ with $\operatorname{ext}\gamma_0$ to obtain an equivalent family of curves $\mathcal{F}'$ and a closed curve γ_0' with the same properties except that all endpoints of curves in $\mathcal{F}'$ and intersection points of pairs of curves in $\mathcal{F}'$ lie in $\operatorname{int}\gamma_0'$. It follows that some part of γ_0' lies in the unbounded component of $\mathbb{R}^2 \setminus \bigcup \mathcal{F}'$. We "cut" γ_0' there and "unfold" it into the baseline, transforming $\mathcal{F}'$ into an equivalent family $\mathcal{F}''$ of $2t$-curves all endpoints of which and intersection points of pairs of which lie above the baseline. The "equivalence" of $\mathcal{F}$, $\mathcal{F}'$, and $\mathcal{F}''$ means in particular that the intersection graphs of $\mathcal{F}$, $\mathcal{F}'$, and $\mathcal{F}''$ are isomorphic, so the theorem follows from Lemma 5 (and thus Theorem 4). ◀

A statement analogous to Theorem 4 fails for families of objects each consisting of two 1-curves only, without the "middle part" connecting them. Specifically, we define a *double-curve* as a set $X \subset H^+$ that is a union of two disjoint 1-curves, denoted by $L(X)$ and $R(X)$ so that the basepoint of $L(X)$ lies to the left of the basepoint of $R(X)$, and we call a family $\mathcal{X}$ of double-curves an *LR-family* if every intersection between two double-curves $X_1, X_2 \in \mathcal{X}$ is an intersection between $L(X_1)$ and $R(X_2)$ or between $L(X_2)$ and $R(X_1)$.

▶ **Theorem 6.** *For every $\zeta \in \mathbb{N}$, there is a triangle-free LR-family of double-curves $\mathcal{X}$ such that $\chi(\mathcal{X}) \geqslant \zeta$.*

The proof of Theorem 6 is an easy adaptation of the construction from [24, 25]. We omit the details. The rest of this section is devoted to the proof of Theorem 4.

Overview of the proof of Theorem 4

Recall the assertion of Theorem 4: the LR-families of even-curves are χ-bounded. The proof is quite long and technical, so we find it useful to provide a high-level overview of its structure. The proof will be presented via a series of reductions. First, we will reduce Theorem 4 to the following statement (Lemma 7): the LR-families of 2-curves are χ-bounded. This statement will be proved by induction on the clique number. Specifically, we will prove the following as the induction step: if every LR-family of 2-curves $\mathcal{F}$ with $\omega(\mathcal{F}) \leqslant k-1$ satisfies $\chi(\mathcal{F}) \leqslant \xi$, then every LR-family of 2-curves $\mathcal{F}$ with $\omega(\mathcal{F}) \leqslant k$ satisfies $\chi(\mathcal{F}) \leqslant \zeta$, where ζ is a constant depending only on k and ξ. The only purpose of the induction hypothesis is to infer that if $\omega(\mathcal{F}) \leqslant k$ and $c \in \mathcal{F}$, then the family of 2-curves in $\mathcal{F} \setminus \{c\}$ that intersect c has chromatic number at most ξ. For notational convenience, LR-families of 2-curves with the latter property will be called *ξ-families*. We will thus reduce the problem to the following statement (Lemma 9): the ξ-families are χ-bounded, where the χ-bounding function depends on ξ.

We will deal with ξ-families via a series of technical lemmas of the following general form: every ξ-family with chromatic number large enough contains a specific configuration of curves. Two kinds of such configurations are particularly important: (a) a large clique, and (b) a 2-curve c and a subfamily $\mathcal{F}'$ with large chromatic number such that the basepoints of the 2-curves in $\mathcal{F}'$ lie between the basepoints of c. In the core of the argument are the proofs that

- every ξ-family with chromatic number large enough contains (a) or (b) (Lemma 16),
- assuming the above, every ξ-family with chromatic number large enough contains (a).

Combined, they complete the argument. Since the two proofs are almost identical, we introduce one more reduction—to *(ξ, h)-families* (Lemma 15). A (ξ, h)-family is just a ξ-family that satisfies an additional technical condition sufficient to carry both proofs at once.

More notation and terminology

Let $\prec$ denote the left-to-right order of points on the baseline ($p_1 \prec p_2$ means that p_1 is to the left of p_2). For convenience, we also use the notation $\prec$ for curves intersecting the baseline ($c_1 \prec c_2$ means that every basepoint of c_1 is to the left of every basepoint of c_2) and for families of such curves ($\mathcal{C}_1 \prec \mathcal{C}_2$ means that $c_1 \prec c_2$ for any $c_1 \in \mathcal{C}_1$ and $c_2 \in \mathcal{C}_2$). For a family $\mathcal{C}$ of curves intersecting the baseline (even-curves or 1-curves) and two 1-curves x and y, let $\mathcal{C}(x,y) = \{c \in \mathcal{C} \colon x \prec c \prec y\}$ or $\mathcal{C}(x,y) = \{c \in \mathcal{C} \colon y \prec c \prec x\}$ depending on whether $x \prec y$ or $y \prec x$. For a family $\mathcal{C}$ of curves intersecting the baseline and a segment I on the baseline, let $\mathcal{C}(I)$ denote the family of curves in $\mathcal{C}$ with all basepoints on I.

For an even-curve c, let $M(c)$ denote the subcurve of c connecting the basepoints of $L(c)$ and $R(c)$, and let $I(c)$ denote the segment on the baseline connecting the basepoints of $L(c)$ and $R(c)$ (see Figure 1). For a family $\mathcal{F}$ of even-curves, let $L(\mathcal{F}) = \{L(c) \colon c \in \mathcal{F}\}$, $R(\mathcal{F}) = \{R(c) \colon c \in \mathcal{F}\}$, and $I(\mathcal{F})$ denote the minimal segment on the baseline that contains $I(c)$ for every $c \in \mathcal{F}$.

A *cap-curve* is a curve in H^+ that has both endpoints on the baseline and does not intersect the baseline in any other point. For a cap-curve γ, it follows from the Jordan curve theorem that the set $H^+ \setminus \gamma$ consists of two connected components: one bounded, denoted by $\operatorname{int}\gamma$, and one unbounded, denoted by $\operatorname{ext}\gamma$. Any two cap-curves one with endpoints p_1, q_1 and the other with endpoints p_2, q_2 such that $p_1 \prec p_2 \prec q_1 \prec q_2$ intersect in an odd number of points.

Reduction to LR-families of 2-curves

We will reduce Theorem 4 to the following statement on LR-families of 2-curves, which is essentially a special case of Theorem 4.

▶ **Lemma 7.** *There is a non-decreasing function $f \colon \mathbb{N} \to \mathbb{N}$ such that every LR-family $\mathcal{F}$ of 2-curves satisfies $\chi(\mathcal{F}) \leqslant f(\omega(\mathcal{F}))$.*

A *component* of a family of 1-curves $\mathcal{S}$ is a connected component of $\bigcup \mathcal{S}$ (the union of all curves in $\mathcal{S}$). The following easy but powerful observation reuses an idea from [17, 20, 27].

▶ **Lemma 8.** *For every LR-family of even-curves $\mathcal{F}$, if $\mathcal{F}^\star$ is the family of curves $c \in \mathcal{F}$ such that $L(c)$ and $R(c)$ lie in distinct components of $L(\mathcal{F}) \cup R(\mathcal{F})$, then $\chi(\mathcal{F}^\star) \leqslant 4$.*

Proof. Let G be an auxiliary graph where the vertices are the components of $L(\mathcal{F}) \cup R(\mathcal{F})$ and the edges are the pairs V_1V_2 of components such that there is a curve $c \in \mathcal{F}^\star$ with $L(c) \subseteq V_1$ and $R(c) \subseteq V_2$ or $L(c) \subseteq V_2$ and $R(c) \subseteq V_1$. Since $\mathcal{F}$ is an LR-family, the curves in $\mathcal{F}^\star$ cannot intersect "outside" the components of $L(\mathcal{F}) \cup R(\mathcal{F})$. It follows that G is planar and thus 4-colorable. Fix a proper 4-coloring of G, and assign the color of a component V to every curve $c \in \mathcal{F}^\star$ with $L(c) \subseteq V$. For any $c_1, c_2 \in \mathcal{F}^\star$, if $L(c_1)$ and $R(c_2)$ intersect, then $L(c_1)$ and $R(c_2)$ lie in the same component V_1 while $L(c_2)$ lies in a component V_2 such that V_1V_2 is an edge of G, so c_1 and c_2 are assigned distinct colors. The coloring of $\mathcal{F}^\star$ is therefore proper. ◀

Proof of Theorem 4 from Lemma 7. We show that $\chi(\mathcal{F}) \leqslant f(\omega(\mathcal{F})) + 4$, where f is the function claimed by Lemma 7. We have $\mathcal{F} = \mathcal{F}_1 \cup \mathcal{F}_2$, where $\mathcal{F}_1 = \{c \in \mathcal{F} \colon L(c)$ and $R(c)$ lie in the same component of $L(\mathcal{F}) \cup R(\mathcal{F})\}$ and $\mathcal{F}_2 = \{c \in \mathcal{F} \colon L(c)$ and $R(c)$ lie in distinct components of $L(\mathcal{F}) \cup R(\mathcal{F})\}$. Lemma 8 yields $\chi(\mathcal{F}_2) \leqslant 4$. It remains to show that $\chi(\mathcal{F}_1) \leqslant f(\omega(\mathcal{F}))$.

Let $c_1, c_2 \in \mathcal{F}_1$. We claim that the intervals $I(c_1)$ and $I(c_2)$ are nested or disjoint. Suppose they are not. For $\varepsilon > 0$ and a component V of $L(\mathcal{F}) \cup R(\mathcal{F})$, let V^ε denote the ε-neighborhood of V in H^+. We assume that ε is small enough so that the sets V^ε for all

components V of $L(\mathcal{F}) \cup R(\mathcal{F})$ and the curves $M(c)$ for all $c \in \mathcal{F}_1$ are pairwise disjoint (except at common basepoints). For $k \in \{1,2\}$, since $L(c_k)$ and $R(c_k)$ belong to the same component V_k of $L(\mathcal{F}) \cup R(\mathcal{F})$, there is a cap-curve $\gamma_k \subset V_k^\varepsilon$ that connects the basepoints of $L(c_k)$ and $R(c_k)$. We can assume without loss of generality that γ_1 and γ_2 intersect in a finite number of points and each of their intersection points is a proper crossing (this is why we take $\gamma_k \subset V_k^\varepsilon$ instead of $\gamma_k \subseteq V_k$). Since $I(c_1)$ and $I(c_2)$ are neither nested nor disjoint, the basepoints of $L(c_2)$ and $R(c_2)$ lie one in $\operatorname{int}\gamma_1$ and the other in $\operatorname{ext}\gamma_1$, so γ_1 and γ_2 intersect in an odd number of points. For $k \in \{1,2\}$, let $\tilde{\gamma}_k$ be the closed curve obtained as the union of γ_k and $M(c_k)$. It follows that $\tilde{\gamma}_1$ and $\tilde{\gamma}_2$ intersect in an odd number of points and each of their intersection points is a proper crossing, which is a contradiction.

Transform $\mathcal{F}_1$ into a family of 2-curves $\mathcal{F}_1'$ replacing the part $M(c)$ of every curve $c \in \mathcal{F}_1$ by the lower semicircle connecting the endpoints of $M(c)$. These semicircles are pairwise disjoint (because $I(c_1)$ and $I(c_2)$ are nested or disjoint for any $c_1, c_2 \in \mathcal{F}_1$), so $\mathcal{F}_1'$ is an LR-family with intersection graph isomorphic to that of $\mathcal{F}_1$. Lemma 7 yields $\chi(\mathcal{F}_1) = \chi(\mathcal{F}_1') \leqslant f(\omega(\mathcal{F}_1')) \leqslant f(\omega(\mathcal{F}))$. ◀

Reduction to ξ-families

For $\xi \in \mathbb{N}$, a *ξ-family* is an LR-family of 2-curves $\mathcal{F}$ with the following property: for every 2-curve $c \in \mathcal{F}$, the family of 2-curves in $\mathcal{F} \setminus \{c\}$ that intersect c has chromatic number at most ξ. We reduce Lemma 7 to the following statement on ξ-families.

▶ **Lemma 9.** *For any $\xi, k \in \mathbb{N}$, there is a constant $\zeta \in \mathbb{N}$ such that every ξ-family $\mathcal{F}$ with $\omega(\mathcal{F}) \leqslant k$ satisfies $\chi(\mathcal{F}) \leqslant \zeta$.*

Proof of Lemma 7 from Lemma 9. Let $f(1) = 1$. For $k \geqslant 2$, let $f(k)$ be the constant claimed by Lemma 9 such that every $f(k-1)$-family $\mathcal{F}$ with $\omega(\mathcal{F}) \leqslant k$ satisfies $\chi(\mathcal{F}) \leqslant f(k)$. Let $k = \omega(\mathcal{F})$, and proceed by induction on k to prove $\chi(\mathcal{F}) \leqslant f(k)$. Clearly, if $k = 1$, then $\chi(\mathcal{F}) = 1$. For the induction step, assume $k \geqslant 2$. For every $c \in \mathcal{F}$, the family of 2-curves in $\mathcal{F} \setminus \{c\}$ that intersect c has clique number at most $k-1$ and therefore, by the induction hypothesis, has chromatic number at most $f(k-1)$. That is, $\mathcal{F}$ is an $f(k-1)$-family, and the definition of f yields $\chi(\mathcal{F}) \leqslant f(k)$. ◀

Dealing with ξ-families

First, we establish the following special case of Lemma 9.

▶ **Lemma 10.** *For every $\xi \in \mathbb{N}$, every ξ-family $\mathcal{F}$ with $\bigcap_{c\in\mathcal{F}} I(c) \neq \emptyset$ satisfies $\chi(\mathcal{F}) \leqslant 4\xi+4$.*

The proof of Lemma 10 is essentially the same as the proof of Lemma 19 in [28]. We need the following elementary lemma, which was also used in various forms in [17, 19, 20, 26, 27].

▶ **Lemma 11** (McGuinness [19, Lemma 2.1])**.** *Let G be a graph, $\prec$ be a total order on the vertices of G, and $\alpha, \beta \in \mathbb{N}$. If $\chi(G) > (2\beta+2)\alpha$, then G has an induced subgraph H such that $\chi(H) > \alpha$ and $\chi(G(u,v)) > \beta$ for every edge uv of H. In particular, if $\chi(G) > 2\beta+2$, then G has an edge uv with $\chi(G(u,v)) > \beta$. Here, $G(u,v)$ denotes the subgraph of G induced on the vertices strictly between u and v in the order $\prec$.*

Proof of Lemma 10. Suppose $\chi(\mathcal{F}) > 4\xi+4$. Since $\bigcap_{c\in\mathcal{F}} I(c) \neq \emptyset$, the 2-curves in $\mathcal{F}$ can be enumerated as $c_1, \ldots, c_{|\mathcal{F}|}$ so that $L(c_1) \prec \cdots \prec L(c_{|\mathcal{F}|}) \prec R(c_{|\mathcal{F}|}) \prec \cdots \prec R(c_1)$. Apply Lemma 11 to the intersection graph of $\mathcal{F}$ and the order $c_1, \ldots, c_{|\mathcal{F}|}$ to obtain two indices $i, j \in \{1, \ldots, |\mathcal{F}|\}$ such that the 2-curves c_i and c_j intersect and $\chi\big(\{c_{i+1}, \ldots, c_{j-1}\}\big) > 2\xi+1$.

Assume $L(c_i)$ and $R(c_j)$ intersect; the argument for the other case is analogous. There is a cap-curve $\gamma \subseteq L(c_i) \cup R(c_j)$ connecting the basepoints of $L(c_i)$ and $R(c_j)$. Every curve intersecting γ intersects c_i or c_j. Since $\mathcal{F}$ is a ξ-family, the 2-curves in $\{c_{i+1}, \dots, c_{j-1}\}$ that intersect c_i have chromatic number at most ξ, and so do those that intersect c_j. Every 2-curve $c_k \in \{c_{i+1}, \dots, c_{j-1}\}$ not intersecting γ satisfies $L(c_k) \subset \operatorname{int}\gamma$ and $R(c_k) \subset \operatorname{ext}\gamma$, so these 2-curves are pairwise disjoint. We conclude that $\chi\big(\{c_{i+1}, \dots, c_{j-1}\}\big) \leqslant 2\xi + 1$, which is a contradiction. ◀

Lemma 11 easily implies that every family of 2-curves $\mathcal{F}$ with $\chi(\mathcal{F}) > (2\beta+2)^2\alpha$ contains a subfamily $\mathcal{H}$ with $\chi(\mathcal{H}) > \alpha$ such that $\chi(\mathcal{F}(L(c_1), L(c_2))) > \beta$ and $\chi(\mathcal{F}(R(c_1), R(c_2))) > \beta$ for any two intersecting 2-curves $c_1, c_2 \in \mathcal{H}$. This is considerably strengthened by the following lemma. Its proof extends the idea used in [19] for the proof of Lemma 11.

▶ **Lemma 12.** *For every $\xi \in \mathbb{N}$, there is a function $f\colon \mathbb{N}\times\mathbb{N} \to \mathbb{N}$ with the following property: for any $\alpha, \beta \in \mathbb{N}$ and every ξ-family $\mathcal{F}$ with $\chi(\mathcal{F}) > f(\alpha, \beta)$, there is a subfamily $\mathcal{H} \subseteq \mathcal{F}$ such that $\chi(\mathcal{H}) > \alpha$ and $\chi(\mathcal{F}(x, y)) > \beta$ for any two intersecting 1-curves $x \in R(\mathcal{H})$ and $y \in L(\mathcal{H})$.*

Proof. Let $f(\alpha, \beta) = (2\beta + 12\xi + 20)\alpha$. Let $\mathcal{F}$ be a ξ-family with $\chi(\mathcal{F}) > f(\alpha, \beta)$. Construct a sequence of points $p_0 \prec \cdots \prec p_{m+1}$ on the baseline with the following properties:

- the points $p_0, \dots, p_{m+1}$ are distinct from all basepoints of 2-curves in $\mathcal{F}$,
- p_0 lies to the left of and p_{m+1} lies to the right of all basepoints of 2-curves in $\mathcal{F}$,
- $\chi(\mathcal{F}(p_i p_{i+1})) = \beta + 1$ for $0 \leqslant i \leqslant m-1$, and $\chi(\mathcal{F}(p_m p_{m+1})) \leqslant \beta + 1$.

This is done greedily by first choosing p_1 so that $\chi(\mathcal{F}(p_0 p_1)) = \beta+1$, then choosing p_2 so that $\chi(\mathcal{F}(p_1 p_2)) = \beta + 1$, and so on. For $0 \leqslant i \leqslant j \leqslant m$, let $\mathcal{F}_{i,j} = \{c \in \mathcal{F}\colon p_i \prec L(c) \prec p_{i+1}$ and $p_j \prec R(c) \prec p_{j+1}\}$. In particular, $\mathcal{F}_{i,i} = \mathcal{F}(p_i p_{i+1})$ for $0 \leqslant i \leqslant m$. Since $\mathcal{F} = \bigcup_{0\leqslant i\leqslant j\leqslant m} \mathcal{F}_{i,j}$, at least one of the following holds:

$$\chi\big(\textstyle\bigcup_{i=0}^{m} \mathcal{F}_{i,i}\big) > (2\beta+2)\alpha, \quad \chi\big(\textstyle\bigcup_{i=0}^{m-1} \mathcal{F}_{i,i+1}\big) > (12\xi+12)\alpha, \quad \chi\big(\textstyle\bigcup_{i=0}^{m-2}\bigcup_{j=i+2}^{m} \mathcal{F}_{i,j}\big) > 6\alpha.$$

In each case, we will find a subfamily $\mathcal{H} \subseteq \mathcal{F}$ such that any two intersecting 1-curves $x \in R(\mathcal{H})$ and $y \in L(\mathcal{H})$ satisfy $x \in R(\mathcal{F}_{i,j})$ and $y \in L(\mathcal{F}_{r,s})$, where $0 \leqslant i \leqslant j \leqslant m$, $0 \leqslant r \leqslant s \leqslant m$, and $|j - r| \geqslant 2$. Then, $\chi(\mathcal{F}(x, y)) \geqslant \chi(\mathcal{F}(p_{\max(j,r)-1} p_{\max(j,r)})) = \beta + 1$, as required.

Suppose $\chi\big(\bigcup_{i=0}^{m} \mathcal{F}_{i,i}\big) > (2\beta+2)\alpha$. We have $\chi(\mathcal{F}_{i,i}) \leqslant \beta + 1$ for $0 \leqslant i \leqslant m$. Color the 2-curves in every $\mathcal{F}_{i,i}$ properly using the same set of $\beta + 1$ colors on $\mathcal{F}_{i,i}$ and $\mathcal{F}_{r,r}$ whenever $i \equiv r \pmod 2$, thus using $2\beta + 2$ colors in total. It follows that $\chi(\mathcal{H}) > \alpha$ for some family $\mathcal{H} \subseteq \bigcup_{i=0}^{m} \mathcal{F}_{i,i}$ of 2-curves of the same color. To conclude, for any two intersecting 1-curves $x \in R(\mathcal{H})$ and $y \in L(\mathcal{H})$, we have $x \in R(\mathcal{F}_{i,i})$ and $y \in L(\mathcal{F}_{r,r})$ for some distinct indices $i, r \in \{0, \dots, m\}$ with $i \equiv r \pmod 2$ and thus $|i - r| \geqslant 2$.

Now, suppose $\chi\big(\bigcup_{i=0}^{m-1} \mathcal{F}_{i,i+1}\big) > (12\xi+12)\alpha$. By Lemma 10, we have $\chi(\mathcal{F}_{i,i+1}) \leqslant 4\xi + 4$ for $0 \leqslant i \leqslant m-1$. Color the 2-curves in every $\mathcal{F}_{i,i+1}$ properly using the same set of $4\xi + 4$ colors on $\mathcal{F}_{i,i+1}$ and $\mathcal{F}_{r,r+1}$ whenever $i \equiv r \pmod 3$, thus using $12\xi + 12$ colors in total. It follows that $\chi(\mathcal{H}) > \alpha$ for some family $\mathcal{H} \subseteq \bigcup_{i=0}^{m-1} \mathcal{F}_{i,i+1}$ of 2-curves of the same color. To conclude, for any two intersecting 1-curves $x \in R(\mathcal{H})$ and $y \in L(\mathcal{H})$, we have $x \in R(\mathcal{F}_{i,i+1})$ and $y \in L(\mathcal{F}_{r,r+1})$ for some distinct indices $i, r \in \{0, \dots, m-1\}$ with $i \equiv r \pmod 3$ and thus $|i + 1 - r| \geqslant 2$.

Finally, suppose $\chi\big(\bigcup_{i=0}^{m-2}\bigcup_{j=i+2}^{m} \mathcal{F}_{i,j}\big) > 6\alpha$. It follows that $\chi\big(\bigcup_{i\in I}\bigcup_{j=i+2}^{m} \mathcal{F}_{i,j}\big) > 3\alpha$, where $I = \{i \in \{0, \dots, m-2\}\colon i \equiv 0 \pmod 2\}$ or $I = \{i \in \{0, \dots, m-2\}\colon i \equiv 1 \pmod 2\}$. Consider an auxiliary graph G with vertex set I and edge set $\{ij\colon i, j \in I,\ i < j,$ and $\mathcal{F}_{i,j-1} \cup \mathcal{F}_{i,j} \neq \emptyset\}$. Since no two 2-curves in $\mathcal{F}$ cross below the baseline, G has no two edges $i_1 j_1$ and $i_2 j_2$ such that $i_1 < i_2 < j_1 < j_2$. In particular, G is an outerplanar graph, and

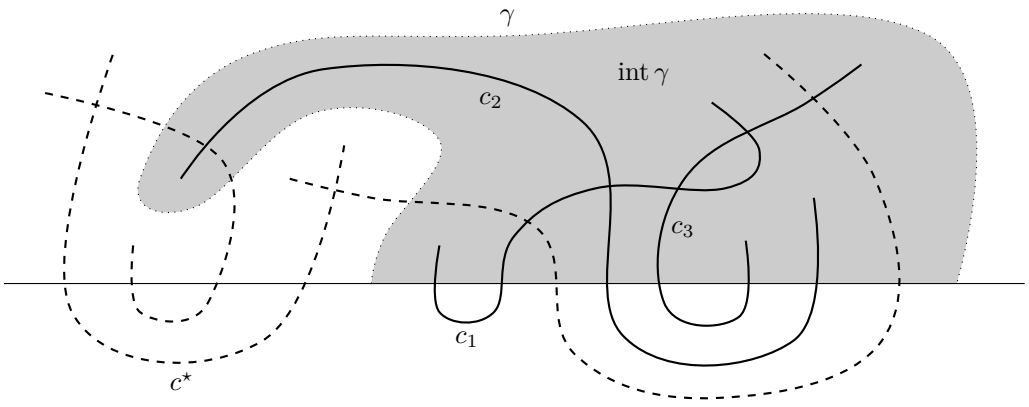


Figure 2 Illustration for Lemma 14: $\mathcal{G} = \{c_1, c_2, c_3\}$.

thus $\chi(G) \leqslant 3$. Fix a proper 3-coloring of G, and use the color of i on every 2-curve in $\bigcup_{j=i+2}^{m} \mathcal{F}_{i,j}$ for every $i \in I$. It follows that $\chi(\mathcal{H}) > \alpha$ for some family $\mathcal{H} \subseteq \bigcup_{i\in I}\bigcup_{j=i+2}^{m} \mathcal{F}_{i,j}$ of 2-curves of the same color. To conclude, for any two intersecting 1-curves $x \in R(\mathcal{H})$ and $y \in L(\mathcal{H})$, we have $x \in R(\mathcal{F}_{i,j})$ and $y \in L(\mathcal{F}_{r,s})$ for some indices $i, r \in I$, $j \in \{i+2, \ldots, m\}$, and $s \in \{r+2, \ldots, m\}$ such that $j \notin \{r-1, r\}$ (otherwise ir would be an edge of G), $j \neq r+1$ (otherwise two 2-curves, one from $\mathcal{F}_{i,r+1}$ and one from $\mathcal{F}_{r,s}$, would cross below the baseline), and thus $|j - r| \geqslant 2$. ◀

It is proved in [26] that for every family of 1-curves $\mathcal{S}$, there are a cap-curve γ and a subfamily $\mathcal{U} \subseteq \mathcal{S}$ with $\chi(\mathcal{U}) \geqslant \frac{1}{2}\chi(\mathcal{S})$ such that every 1-curve in $\mathcal{U}$ is contained in $\operatorname{int}\gamma$ and intersects some 1-curve in $\mathcal{S}$ that intersects $\operatorname{ext}\gamma$. The proof follows an idea from [10], used subsequently also in [17, 19, 20, 21, 27], where $\mathcal{U}$ is chosen as one of the sets of 1-curves at a fixed distance from an appropriately chosen 1-curve in the intersection graph of $\mathcal{S}$, and γ is a cap-curve surrounding $\mathcal{U}$ very closely. However, this method fails to imply an analogous statement for 2-curves. We will need a more powerful tool—part of the recent series of works on induced subgraphs that must be present in graphs with sufficiently large chromatic number.

▶ **Theorem 13** (Chudnovsky, Scott, Seymour [6, Theorem 1.8]). *There is a function $f\colon \mathbb{N} \to \mathbb{N}$ with the following property: for every $\alpha \in \mathbb{N}$, every string graph G with $\chi(G) > f(\alpha)$ contains a vertex v such that $\chi(G_v^2) > \alpha$, where G_v^2 denotes the subgraph of G induced on the vertices within distance at most 2 from v.*

The special case of Theorem 13 for triangle-free intersection graphs of curves any two of which intersect in at most one point was proved earlier by McGuinness [21, Theorem 5.3].

▶ **Lemma 14** (see Figure 2). *For every $\xi \in \mathbb{N}$, there is a function $f\colon \mathbb{N} \to \mathbb{N}$ with the following property: for every $\alpha \in \mathbb{N}$ and every ξ-family $\mathcal{F}$ with $\chi(\mathcal{F}) > f(\alpha)$, there are a cap-curve γ and a subfamily $\mathcal{G} \subseteq \mathcal{F}$ with $\chi(\mathcal{G}) > \alpha$ such that every 2-curve $c \in \mathcal{G}$ satisfies $L(c), R(c) \subset \operatorname{int}\gamma$ and intersects some 2-curve in $\mathcal{F}$ that intersects $\operatorname{ext}\gamma$.*

Proof. Let $f(\alpha) = f_1(3\alpha+5\xi+5)$, where f_1 is the function claimed by Theorem 13. Let $\mathcal{F}$ be a ξ-family with $\chi(\mathcal{F}) > f(\alpha)$. It follows that there is a 2-curve $c^\star \in \mathcal{F}$ such that the family of curves within distance at most 2 from $c^\star$ in the intersection graph of $\mathcal{F}$ has chromatic number greater than $3\alpha + 5\xi + 5$. For $k \in \{1, 2\}$, let $\mathcal{F}_k$ be the 2-curves in $\mathcal{F}$ at distance exactly k from $c^\star$ in the intersection graph of $\mathcal{F}$. Since $\chi(\{c^\star\} \cup \mathcal{F}_1 \cup \mathcal{F}_2) > 3\alpha + 5\xi + 5$ and $\chi(\mathcal{F}_1) \leqslant \xi$ (because $\mathcal{F}$ is a ξ-family), we have $\chi(\mathcal{F}_2) > 3\alpha+4\xi+4$. We have $\mathcal{F}_2 = \mathcal{G}_1 \cup \mathcal{G}_2 \cup \mathcal{G}_3 \cup \mathcal{G}_4$, where

$\mathcal{G}_1 = \{c \in \mathcal{F}_2 \colon L(c) \prec R(c) \prec L(c^\star) \prec R(c^\star)\}$, $\mathcal{G}_2 = \{c \in \mathcal{F}_2 \colon L(c^\star) \prec L(c) \prec R(c) \prec R(c^\star)\}$, $\mathcal{G}_3 = \{c \in \mathcal{F}_2 \colon L(c^\star) \prec R(c^\star) \prec L(c) \prec R(c)\}$, $\mathcal{G}_4 = \{c \in \mathcal{F}_2 \colon L(c) \prec L(c^\star) \prec R(c^\star) \prec R(c)\}$. Since $\chi(\mathcal{F}_2) > 3\alpha + 4\xi + 4$ and $\chi(\mathcal{G}_4) \leqslant 4\xi + 4$ (by Lemma 10), we have $\chi(\mathcal{G}_k) > \alpha$ for some $k \in \{1, 2, 3\}$. Since neither basepoint of $c^\star$ lies on $I(\mathcal{G}_k)$, there is a cap-curve γ with $L(c^\star), R(c^\star) \subset \operatorname{ext}\gamma$ and $L(c), R(c) \subset \operatorname{int}\gamma$ for all $c \in \mathcal{G}_k$. The lemma follows with $\mathcal{G} = \mathcal{G}_k$. ◀

Reduction to (ξ, h)-families

For $\xi \in \mathbb{N}$ and a function $h \colon \mathbb{N} \to \mathbb{N}$, a (ξ, h)-*family* is a ξ-family $\mathcal{F}$ with the following additional property: for every $\alpha \in \mathbb{N}$ and every subfamily $\mathcal{G} \subseteq \mathcal{F}$ with $\chi(\mathcal{G}) > h(\alpha)$, there is a subfamily $\mathcal{H} \subseteq \mathcal{G}$ with $\chi(\mathcal{H}) > \alpha$ such that every 2-curve in $\mathcal{F}$ with a basepoint on $I(\mathcal{H})$ has both basepoints on $I(\mathcal{G})$. We will prove the following lemma.

▶ **Lemma 15.** *For any $\xi, k \in \mathbb{N}$ and any function $h \colon \mathbb{N} \to \mathbb{N}$, there is a constant $\zeta \in \mathbb{N}$ such that every (ξ, h)-family $\mathcal{F}$ with $\omega(\mathcal{F}) \leqslant k$ satisfies $\chi(\mathcal{F}) \leqslant \zeta$.*

The notion of a (ξ, h)-family and Lemma 15 provide a convenient abstraction of what is needed to prove the next lemma and then to prove Lemma 9 with the use of the next lemma.

▶ **Lemma 16.** *For any $\xi, k \in \mathbb{N}$, there is a function $f \colon \mathbb{N} \to \mathbb{N}$ such that for every $\alpha \in \mathbb{N}$, every ξ-family $\mathcal{F}$ with $\omega(\mathcal{F}) \leqslant k$ and $\chi(\mathcal{F}) > f(\alpha)$ contains a 2-curve c with $\chi(\mathcal{F}(I(c))) > \alpha$.*

Proof of Lemma 16 from Lemma 15. Let $h_\alpha \colon \mathbb{N} \ni \beta \mapsto \beta + 2\alpha + 2 \in \mathbb{N}$, and let $f(\alpha)$ be the constant claimed by Lemma 15 such that every (ξ, h_α)-family $\mathcal{F}$ with $\omega(\mathcal{F}) \leqslant k$ satisfies $\chi(\mathcal{F}) \leqslant f(\alpha)$. Let $\mathcal{F}$ be a ξ-family with $\omega(\mathcal{F}) \leqslant k$ and $\chi(\mathcal{F}(I(c))) \leqslant \alpha$ for every $c \in \mathcal{F}$. It is enough to show that $\mathcal{F}$ is a (ξ, h_α)-family. To this end, consider a subfamily $\mathcal{G} \subseteq \mathcal{F}$ with $\chi(\mathcal{G}) > h_\alpha(\beta)$ for some $\beta \in \mathbb{N}$. Take $\mathcal{G}_L, \mathcal{G}_R \subseteq \mathcal{G}$ so that $L(\mathcal{G}_L) \prec L(\mathcal{G} \setminus \mathcal{G}_L)$, $\chi(\mathcal{G}_L) = \alpha + 1$, $R(\mathcal{G} \setminus \mathcal{G}_R) \prec R(\mathcal{G}_R)$, and $\chi(\mathcal{G}_R) = \alpha + 1$. Let $\mathcal{H} = \mathcal{G} \setminus (\mathcal{G}_L \cup \mathcal{G}_R)$. It follows that $\chi(\mathcal{H}) \geqslant \chi(\mathcal{G}) - 2\alpha - 2 > \beta$. If there is a 2-curve $c \in \mathcal{F}$ with one basepoint on $I(\mathcal{H})$ and the other basepoint not on $I(\mathcal{G})$, then $\mathcal{G}_L \subseteq \mathcal{F}(I(c))$ or $\mathcal{G}_R \subseteq \mathcal{F}(I(c))$, so $\chi(\mathcal{F}(I(c))) \geqslant \alpha + 1$, which is a contradiction. Therefore, every 2-curve in $\mathcal{F}$ with a basepoint on $I(\mathcal{H})$ has both basepoints on $I(\mathcal{G})$. This shows that $\mathcal{F}$ is a (ξ, h_α)-family. ◀

Proof of Lemma 9 from Lemma 15. Let h be the function claimed by Lemma 16 for ξ and k. Let $\mathcal{F}$ be a ξ-family with $\omega(\mathcal{F}) \leqslant k$. In view of Lemma 15, it is enough to show that $\mathcal{F}$ is a (ξ, h)-family. To this end, consider a subfamily $\mathcal{G} \subseteq \mathcal{F}$ with $\chi(\mathcal{G}) > h(\alpha)$ for some $\alpha \in \mathbb{N}$. Lemma 16 yields a 2-curve $c \in \mathcal{G}$ such that $\chi(\mathcal{G}(I(c))) > \alpha$. Every 2-curve in $\mathcal{F}$ with a basepoint on $I(c)$ has both basepoints on $I(c)$, otherwise it would cross c below the baseline. Therefore, the condition of a (ξ, h)-family is satisfied with $\mathcal{H} = \mathcal{G}(I(c))$. ◀

Dealing with (ξ, h)-families

The rest of the proof is inspired from the ideas in [26]. A family of 1-curves $\mathcal{S}$ *supports* a family of 2-curves $\mathcal{F}$ if every 2-curve in $\mathcal{F}$ intersects some 1-curve in $\mathcal{S}$. A *skeleton* is a pair $(\gamma, \mathcal{U})$ such that γ is a cap-curve and $\mathcal{U}$ is a family of pairwise disjoint 1-curves each of which has one endpoint (other than the basepoint) on γ and all the remaining part in $\operatorname{int}\gamma$ (see Figure 3). For a family of 1-curves $\mathcal{S}$, a skeleton $(\gamma, \mathcal{U})$ is an $\mathcal{S}$-*skeleton* if every 1-curve in $\mathcal{U}$ is a subcurve of some 1-curve in $\mathcal{S}$. A skeleton $(\gamma, \mathcal{U})$ *supports* a family of 2-curves $\mathcal{F}$ if every 2-curve $c \in \mathcal{F}$ satisfies $L(c), R(c) \subset \operatorname{int}\gamma$ and intersects some 1-curve in $\mathcal{U}$.

▶ **Lemma 17.** *For every function $h \colon \mathbb{N} \to \mathbb{N}$, there is a function $f \colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ such that for any $\alpha, \beta \in \mathbb{N}$, every (ξ, h)-family $\mathcal{F}$ with $\chi(\mathcal{F}) > f(\alpha, \beta)$ contains one of the following configurations:*

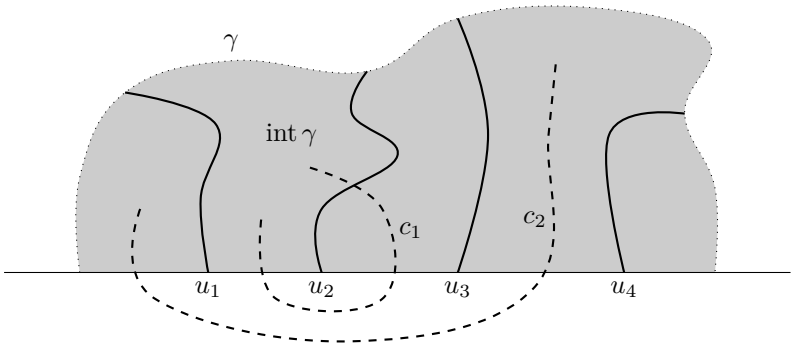


Figure 3 A skeleton $\big(\gamma, \{u_1, u_2, u_3, u_4\}\big)$, which supports c_1 but not c_2.

- *a subfamily $\mathcal{G} \subseteq \mathcal{F}$ with $\chi(\mathcal{G}) > \alpha$ supported by an $L(\mathcal{F})$-skeleton or an $R(\mathcal{F})$-skeleton,*
- *a subfamily $\mathcal{H} \subseteq \mathcal{F}$ with $\chi(\mathcal{H}) > \beta$ supported by a family of 1-curves $\mathcal{S}$ with $\mathcal{S} \subseteq L(\mathcal{F})$ or $\mathcal{S} \subseteq R(\mathcal{F})$ such that $s \prec \mathcal{H}$ or $\mathcal{H} \prec s$ for every 1-curve $s \in \mathcal{S}$.*

Proof. Let $f(\alpha, \beta) = f_1(2\alpha + h(2\beta) + 4)$, where f_1 is the function claimed by Lemma 14. Apply Lemma 14 to obtain a cap-curve γ and a subfamily $\mathcal{G} \subseteq \mathcal{F}$ with $\chi(\mathcal{G}) > 2\alpha + h(2\beta) + 4$ such that every 2-curve $c \in \mathcal{G}$ satisfies $L(c), R(c) \subset \operatorname{int}\gamma$ and intersects some 2-curve in $\mathcal{F}_{\mathrm{ext}}$. Here and further on, $\mathcal{F}_{\mathrm{ext}}$ denotes the family of 2-curves in $\mathcal{F}$ that intersect $\operatorname{ext}\gamma$. Let $\mathcal{U}_L$ be the 1-curves that are subcurves of 1-curves in $L(\mathcal{F})$, have one endpoint (other than the basepoint) on γ, and have all the remaining part in $\operatorname{int}\gamma$. Let $\mathcal{U}_R$ be the 1-curves that are subcurves of 1-curves in $R(\mathcal{F})$, have one endpoint (other than the basepoint) on γ, and have all the remaining part in $\operatorname{int}\gamma$. Thus $(\gamma, \mathcal{U}_L)$ is an $L(\mathcal{F})$-skeleton, and $(\gamma, \mathcal{U}_R)$ is an $R(\mathcal{F})$-skeleton. Let $\mathcal{G}_L$ be the 2-curves in $\mathcal{G}$ that intersect some 1-curve in $\mathcal{U}_L$, and let $\mathcal{G}_R$ be those that intersect some 1-curve in $\mathcal{U}_R$. If $\chi(\mathcal{G}_L) > \alpha$ or $\chi(\mathcal{G}_R) > \alpha$, then the first conclusion of the lemma holds. Thus assume $\chi(\mathcal{G}_L) \leqslant \alpha$ and $\chi(\mathcal{G}_R) \leqslant \alpha$. Let $\mathcal{G}' = \mathcal{G} \setminus (\mathcal{G}_L \cup \mathcal{G}_R)$. It follows that $\chi(\mathcal{G}') \geqslant \chi(\mathcal{G}) - 2\alpha > h(2\beta) + 4$.

By Lemma 8, the 2-curves $c \in \mathcal{G}'$ such that $L(c)$ and $R(c)$ lie in distinct components of $L(\mathcal{G}') \cup R(\mathcal{G}')$ have chromatic number at most 4. Therefore, there is a component V of $L(\mathcal{G}') \cup R(\mathcal{G}')$ such that $\chi(\mathcal{G}'_V) \geqslant \chi(\mathcal{G}') - 4 > h(2\beta)$, where $\mathcal{G}'_V = \{c \in \mathcal{G}' \colon L(c), R(c) \subseteq V\}$. There is a cap-curve $\nu \subseteq V$ connecting the two endpoints of the segment $I(\mathcal{G}'_V)$. Suppose there is a 2-curve $c \in \mathcal{F}_{\mathrm{ext}}$ with both basepoints on $I(\mathcal{G}'_V)$. If $L(c)$ intersects $\operatorname{ext}\gamma$, then the part of $L(c)$ from the basepoint to the first intersection point with γ, which is a 1-curve in $\mathcal{U}_L$, must intersect ν (as $\nu \subseteq V \subset \operatorname{int}\gamma$) and thus a curve in $\mathcal{G}'$ (as V is a component of $\mathcal{G}'$). Thus $\mathcal{G}' \cap \mathcal{G}_L \neq \emptyset$, which is a contradiction. An analogous contradiction is reached if $R(c)$ intersects $\operatorname{ext}\gamma$. This shows that no curve in $\mathcal{F}_{\mathrm{ext}}$ has both basepoints on $I(\mathcal{G}'_V)$.

Since $\mathcal{F}$ is a (ξ, h)-family and $\chi(\mathcal{G}'_V) > h(2\beta)$, there is a subfamily $\mathcal{H}' \subseteq \mathcal{G}'_V$ such that $\chi(\mathcal{H}') > 2\beta$ and every 2-curve in $\mathcal{F}$ with a basepoint on $I(\mathcal{H}')$ has the other basepoint on $I(\mathcal{G}'_V)$. This and the above imply that no curve in $\mathcal{F}_{\mathrm{ext}}$ has a basepoint on $I(\mathcal{H}')$. Since every curve in $\mathcal{H}'$ intersects some curve in $\mathcal{F}_{\mathrm{ext}}$, we have $\mathcal{H}' = \mathcal{H}_L \cup \mathcal{H}_R$, where $\mathcal{H}_L$ are the 2-curves in $\mathcal{H}'$ that intersect some 1-curve in $L(\mathcal{F}_{\mathrm{ext}})$ and $\mathcal{H}_R$ are those that intersect some 1-curve in $R(\mathcal{F}_{\mathrm{ext}})$. Since $\chi(\mathcal{H}') > 2\beta$, we conclude that $\chi(\mathcal{H}_L) > \beta$ or $\chi(\mathcal{H}_R) > \beta$ and thus the second conclusion of the lemma holds with $(\mathcal{H}, \mathcal{S}) = (\mathcal{H}_L, L(\mathcal{F}_{\mathrm{ext}}))$ or $(\mathcal{H}, \mathcal{S}) = (\mathcal{H}_R, R(\mathcal{F}_{\mathrm{ext}}))$. ◀

▶ **Lemma 18.** *For every function $h \colon \mathbb{N} \to \mathbb{N}$, there is a function $f \colon \mathbb{N} \to \mathbb{N}$ such that for every $\alpha \in \mathbb{N}$, every (ξ, h)-family $\mathcal{F}$ with $\chi(\mathcal{F}) > f(\alpha)$ contains a subfamily $\mathcal{G} \subseteq \mathcal{F}$ with $\chi(\mathcal{G}) > \alpha$ supported by an $L(\mathcal{F})$-skeleton or an $R(\mathcal{F})$-skeleton.*

Proof. Let $f(\alpha) = f_1(\alpha, f_1(\alpha, f_1(\alpha, 4\xi)))$, where f_1 is the function claimed by Lemma 17. Suppose to the contrary that no such subfamily $\mathcal{G}$ exists. Let $\mathcal{F}_0 = \mathcal{F}$. Apply Lemma 17 three times to obtain families $\mathcal{F}_1$, $\mathcal{F}_2$, $\mathcal{F}_3$, $\mathcal{S}_1$, $\mathcal{S}_2$, and $\mathcal{S}_3$ with the following properties:

- $\mathcal{F} = \mathcal{F}_0 \supseteq \mathcal{F}_1 \supseteq \mathcal{F}_2 \supseteq \mathcal{F}_3$,
- for $i \in \{1,2,3\}$, we have $\mathcal{S}_i \subseteq L(\mathcal{F}_{i-1})$ or $\mathcal{S}_i \subseteq R(\mathcal{F}_{i-1})$, $\mathcal{F}_i$ is supported by $\mathcal{S}_i$, and $s \prec \mathcal{F}_i$ or $\mathcal{F}_i \prec s$ for every 1-curve $s \in \mathcal{S}_i$.
- $\chi(\mathcal{F}_1) > f_1(\alpha, f_1(\alpha, 4\xi))$, $\chi(\mathcal{F}_2) > f_1(\alpha, 4\xi)$ and $\chi(\mathcal{F}_3) > 4\xi$.

There are indices $i, j \in \{1,2,3\}$ with $i < j$ such that $\mathcal{S}_i$ and $\mathcal{S}_j$ are of the same "type": either $\mathcal{S}_i \subseteq L(\mathcal{F}_{i-1})$ and $\mathcal{S}_j \subseteq L(\mathcal{F}_{j-1})$ or $\mathcal{S}_i \subseteq R(\mathcal{F}_{i-1})$ and $\mathcal{S}_j \subseteq R(\mathcal{F}_{j-1})$. Assume for the rest of the proof that $\mathcal{S}_i \subseteq R(\mathcal{F}_{i-1})$ and $\mathcal{S}_j \subseteq R(\mathcal{F}_{j-1})$; the argument for the other case is analogous.

Let $\mathcal{S}_\prec = \{s \in \mathcal{S}_j \colon s \prec \mathcal{F}_j\}$, $\mathcal{S}_\succ = \{s \in \mathcal{S}_j \colon \mathcal{F}_j \prec s\}$, $\mathcal{F}_\prec$ be the 2-curves in $\mathcal{F}_j$ that intersect some 1-curve in $\mathcal{S}_\prec$, and $\mathcal{F}_\succ$ be those that intersect some 1-curve in $\mathcal{S}_\succ$. Thus $\mathcal{F}_\prec \cup \mathcal{F}_\succ = \mathcal{F}_j$. This and $\chi(\mathcal{F}_j) \geqslant \chi(\mathcal{F}_3) > 4\xi$ yield $\chi(\mathcal{F}_\prec) > 2\xi$ or $\chi(\mathcal{F}_\succ) > 2\xi$. Assume for the rest of the proof that $\chi(\mathcal{F}_\prec) > 2\xi$; the argument for the other case is analogous.

Let $\mathcal{S}_\prec^{\min}$ be an inclusion-minimal subfamily of $\mathcal{S}_\prec$ with the property that $\mathcal{S}_\prec^{\min}$ still supports $\mathcal{F}_\prec$. Let $s^\star$ be the 1-curve in $\mathcal{S}_\prec^{\min}$ with rightmost basepoint, and let $\mathcal{F}_\prec^\star = \{c \in \mathcal{F}_\prec \colon L(c)$ intersects $s^\star\}$. Since $\mathcal{F}$ is a ξ-family, we have $\chi(\mathcal{F}_\prec^\star) \leqslant \xi$. By the choice of $\mathcal{S}_\prec^{\min}$, there exists a 2-curve $c^\star \in \mathcal{F}_\prec^\star$ disjoint from every 1-curve in $\mathcal{S}_\prec^{\min}$ other than $s^\star$. Since $\mathcal{F}_\prec$ is supported by $\mathcal{S}_i$, there is a 1-curve $s_i \in \mathcal{S}_i$ that intersects $L(c^\star)$. We show that every 2-curve in $\mathcal{F}_\prec \setminus \mathcal{F}_\prec^\star$ intersects s_i.

Let $c \in \mathcal{F}_\prec \setminus \mathcal{F}_\prec^\star$, and let s be a 1-curve in $\mathcal{S}_\prec^{\min}$ that intersects $L(c)$. Thus $s \neq s^\star$, by the definition of $\mathcal{F}_\prec^\star$. There is a cap-curve $\gamma \subseteq L(c) \cup s$. Since $s \prec s^\star \prec L(c)$ and $s^\star$ intersects neither s nor $L(c)$, we have $s^\star \subset \operatorname{int}\gamma$. Since $L(c^\star)$ intersects $s^\star$ but neither s nor $L(c)$, we also have $L(c^\star) \subset \operatorname{int}\gamma$. Since $s_i \prec \mathcal{F}_i$ or $\mathcal{F}_i \prec s_i$, the basepoint of s_i lies in $\operatorname{ext}\gamma$. Therefore, since s_i intersects $L(c^\star)$, the 1-curve s_i must enter $\operatorname{int}\gamma$ through a point on $L(c)$. This shows that every 2-curve in $\mathcal{F}_\prec \setminus \mathcal{F}_\prec^\star$ intersects s_i. This and the assumption that $\mathcal{F}$ is a ξ-family yield $\chi(\mathcal{F}_\prec \setminus \mathcal{F}_\prec^\star) \leqslant \xi$. We conclude that $\chi(\mathcal{F}_\prec) \leqslant \chi(\mathcal{F}_\prec^\star) + \chi(\mathcal{F}_\prec \setminus \mathcal{F}_\prec^\star) \leqslant 2\xi$, which is a contradiction. ◀

A *chain* of length n is a sequence $\big((a_1,b_1),\dots,(a_n,b_n)\big)$ of pairs of 2-curves such that

- for $1 \leqslant i \leqslant n$, the 1-curves $R(a_i)$ and $L(b_i)$ intersect,
- for $2 \leqslant i \leqslant n$, the basepoints of $R(a_i)$ and $L(b_i)$ lie between the basepoints of $R(a_{i-1})$ and $L(b_{i-1})$, and $L(a_i)$ intersects $R(a_1),\dots,R(a_{i-1})$ or $R(b_i)$ intersects $L(b_1),\dots,L(b_{i-1})$.

▶ **Lemma 19.** *For every $\xi \in \mathbb{N}$ and every function $h\colon \mathbb{N} \to \mathbb{N}$, there is a function $f\colon \mathbb{N} \to \mathbb{N}$ such that for every $n \in \mathbb{N}$, every (ξ,h)-family $\mathcal{F}$ with $\chi(\mathcal{F}) > f(n)$ contains a chain of length n.*

Proof (see Figure 4). We define the function f by induction. Let $f(1) = 1$; if $\chi(\mathcal{F}) > 1$, then $\mathcal{F}$ contains two intersecting 2-curves, which form a chain of length 1. For the induction step, fix $n \geqslant 1$, and assume that every (ξ,h)-family $\mathcal{H}$ with $\chi(\mathcal{H}) > f(n)$ contains a chain of length n. Let $\beta = f_1\big(f(n), h(2\xi) + 4\xi + 2\big)$ and $f(n+1) = f_2(f_2(f_2(\beta)))$, where f_1 is the function claimed by Lemma 12 and f_2 is the function claimed by Lemma 18. Let $\mathcal{F}$ be a (ξ,h)-family with $\chi(\mathcal{F}) > f(n+1)$. We claim that $\mathcal{F}$ contains a chain of length $n+1$.

Let $\mathcal{F}_0 = \mathcal{F}$. Apply Lemma 18 three times to find families of 2-curves $\mathcal{F}_1$, $\mathcal{F}_2$, $\mathcal{F}_3$ and skeletons $(\gamma_1,\mathcal{U}_1)$, $(\gamma_2,\mathcal{U}_2)$, $(\gamma_3,\mathcal{U}_3)$ with the following properties:

- $\mathcal{F} = \mathcal{F}_0 \supseteq \mathcal{F}_1 \supseteq \mathcal{F}_2 \supseteq \mathcal{F}_3$,
- for $i \in \{1,2,3\}$, $(\gamma_i,\mathcal{U}_i)$ is an $L(\mathcal{F}_{i-1})$-skeleton or an $R(\mathcal{F}_{i-1})$-skeleton supporting $\mathcal{F}_i$,
- $\chi(\mathcal{F}_1) > f_2(f_2(\beta))$, $\chi(\mathcal{F}_2) > f_2(\beta)$, and $\chi(\mathcal{F}_3) > \beta$.

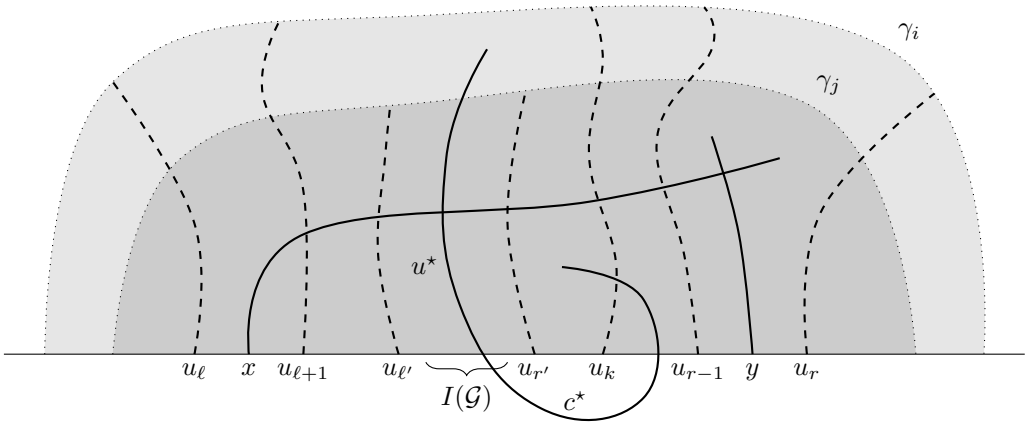


Figure 4 Illustration for the proof of Lemma 19.

There are two indices $i, j \in \{1, 2, 3\}$ with $i < j$ such that the skeletons $(\gamma_i, \mathcal{U}_i)$ and $(\gamma_j, \mathcal{U}_j)$ are of the same "type": either an $L(\mathcal{F}_{i-1})$-skeleton and an $L(\mathcal{F}_{j-1})$-skeleton or an $R(\mathcal{F}_{i-1})$-skeleton and an $R(\mathcal{F}_{j-1})$-skeleton. Assume for the rest of the proof that $(\gamma_i, \mathcal{U}_i)$ is an $L(\mathcal{F}_{i-1})$-skeleton and $(\gamma_j, \mathcal{U}_j)$ is an $L(\mathcal{F}_{j-1})$-skeleton; the argument for the other case is analogous.

By Lemma 12, since $\chi(\mathcal{F}_j) \geqslant \chi(\mathcal{F}_3) > \beta$, there is a subfamily $\mathcal{H} \subseteq \mathcal{F}_j$ such that $\chi(\mathcal{H}) > f(n)$ and $\chi(\mathcal{F}_j(x, y)) > h(2\xi) + 4\xi + 2$ for any two intersecting 1-curves $x, y \in L(\mathcal{H}) \cup R(\mathcal{H})$. Since $\chi(\mathcal{H}) > f(n)$, the family $\mathcal{H}$ contains a chain $\big((a_1, b_1), \ldots, (a_n, b_n)\big)$ of length n. Let x and y be the 1-curves $R(a_n)$ and $L(b_n)$ assigned so that $x \prec y$. By the definition of a chain, x and y intersect, and therefore $\chi(\mathcal{F}_j(x, y)) > h(2\xi) + 4\xi + 2$.

Enumerate the 1-curves in $\mathcal{U}_i$ as $u_1, \ldots, u_m$ so that $u_1 \prec \cdots \prec u_m$, where $m = |\mathcal{U}_i|$. Assume $u_1 \prec x \prec y \prec u_m$ for simplicity (adjusting the proof to the general case is straightforward). There are indices ℓ and r with $1 \leqslant \ell < r \leqslant m$, $u_\ell \prec x \prec u_{\ell+1}$, and $u_{r-1} \prec y \prec u_r$. Let $\mathcal{F}_j^L = \{c \in \mathcal{F}_j \colon x \prec L(c) \prec u_{\ell+1}\}$ and $\mathcal{F}_j^R = \{c \in \mathcal{F}_j \colon u_{r-1} \prec R(c) \prec y\}$. It follows that $\mathcal{F}_j(x, y) \subseteq \mathcal{F}_j^L \cup \mathcal{F}_j(u_{\ell+1}, u_{r-1}) \cup \mathcal{F}_j^R$.

Since $\mathcal{F}$ is a ξ-family, the 2-curves in $\mathcal{F}_j^L$ that intersect u_ℓ have chromatic number at most ξ, and so do the 2-curves in $\mathcal{F}_j^L$ that intersect $u_{\ell+1}$. The remaining 2-curves $c \in \mathcal{F}_j^L$ (intersecting neither u_ℓ nor $u_{\ell+1}$) are pairwise disjoint, because their 1-curves $L(c)$ are contained in and $R(c)$ are disjoint from the part of $\operatorname{int} \gamma_i$ between u_ℓ and $u_{\ell+1}$. Thus $\chi(\mathcal{F}_j^L) \leqslant 2\xi + 1$. Similarly, $\chi(\mathcal{F}_j^R) \leqslant 2\xi + 1$. This yields $\ell + 1 \leqslant r - 1$ and $\chi(\mathcal{F}_j(u_{\ell+1}, u_{r-1})) \geqslant \chi(\mathcal{F}_j(x, y)) - 4\xi - 2 > h(2\xi)$.

Since $\mathcal{F}$ is a (ξ, h)-family, there is a subfamily $\mathcal{G} \subseteq \mathcal{F}_j(u_{\ell+1}, u_{r-1})$ with $\chi(\mathcal{G}) > 2\xi$ such that every 2-curve $c \in \mathcal{F}$ with a basepoint on $I(\mathcal{G})$ satisfies $u_{\ell+1} \prec c \prec u_{r-1}$.

Let $u_{\ell'}$ be the 1-curve in $\mathcal{U}_j$ with rightmost basepoint to the left of $I(\mathcal{G})$, and let $u_{r'}$ be the 1-curve in $\mathcal{U}_j$ with leftmost basepoint to the right of $I(\mathcal{G})$. Every 2-curve in $\mathcal{G}$ must intersect $u_{\ell'}$, some 1-curve in $\mathcal{U}_j(I(\mathcal{G}))$, or $u_{r'}$. Since $\mathcal{F}$ is a ξ-family, the 2-curves in $\mathcal{G}$ that intersect $u_{\ell'}$ have chromatic number at most ξ, and so do the 2-curves in $\mathcal{G}$ that intersect $u_{r'}$. Therefore, since $\chi(\mathcal{G}) > 2\xi$, some 2-curve in $\mathcal{G}$ must intersect a 1-curve in $\mathcal{U}_j(I(\mathcal{G}))$. In particular, the family $\mathcal{U}_j(I(\mathcal{G}))$ is non-empty.

Let $u^\star \in \mathcal{U}_j(I(\mathcal{G}))$. The 1-curve $u^\star$ is a subcurve of $L(c^\star)$ for some 2-curve $c^\star \in \mathcal{F}_{j-1}$. Since the basepoint of $L(c^\star)$ lies on $I(\mathcal{G})$, the property of $\mathcal{G}$ implies $u_{\ell+1} \prec c^\star \prec u_{r-1}$. Since $c^\star \in \mathcal{F}_{j-1} \subseteq \mathcal{F}_i$ and $\mathcal{F}_i$ is supported by $(\gamma_i, \mathcal{U}_i)$, the 1-curve $R(c^\star)$ intersects at least one of the 1-curves $u_{\ell+1}, \ldots, u_{r-1}$, say u_k. Let $a_{n+1} = c^\star$ and b_{n+1} be the 2-curve in $\mathcal{F}_{i-1}$ such that u_k is a subcurve of $L(b_{n+1})$. For $1 \leqslant t \leqslant n$, the 1-curves $R(a_t)$ and $L(b_t)$ intersect and

they are both contained in $\operatorname{int}\gamma_j$ (because $a_t, b_t \in \mathcal{H}$), the basepoint of $L(a_{n+1})$ is between the basepoints of $R(a_t)$ and $L(b_t)$, and $L(a_{n+1})$ intersects γ_j (as it contains $u^\star$). Therefore, $L(a_{n+1})$ intersects all $R(a_1), \dots, R(a_n)$. We conclude that $\big((a_1, b_1), \dots, (a_{n+1}, b_{n+1})\big)$ is a chain of length $n+1$. ◀

Proof of Lemma 15. Let $\zeta = f(2k+1)$, where f is the function claimed by Lemma 19 for ξ and h. Suppose $\chi(\mathcal{F}) > \zeta$. It follows that $\mathcal{F}$ contains a chain of length $2k+1$. This chain contains a subchain $\big((a_1, b_1), \dots, (a_{k+1}, b_{k+1})\big)$ of pairs of the same "type": $L(a_i)$ intersects $R(a_1), \dots, R(a_{i-1})$ for $2 \leqslant i \leqslant k+1$ and thus $\{a_1, \dots, a_{k+1}\}$ is a clique, or $R(b_i)$ intersects $L(b_1), \dots, L(b_{i-1})$ for $2 \leqslant i \leqslant k+1$ and thus $\{b_1, \dots, b_{k+1}\}$ is a clique. Thus $\omega(\mathcal{F}) > k$. ◀

3 Proof of Theorem 2

▶ **Lemma 20** (Fox, Pach, Suk [9, Lemma 3.2]). *For every $t \in \mathbb{N}$, there is a constant $\nu_t > 0$ such that every family of curves $\mathcal{F}$ any two of which intersect in at most t points has subfamilies $\mathcal{F}_1, \dots, \mathcal{F}_d \subseteq \mathcal{F}$ with the following properties:*

- *for $1 \leqslant i \leqslant d$, there is a curve $c_i \in \mathcal{F}_i$ intersecting all curves in $\mathcal{F}_i \smallsetminus \{c_i\}$,*
- *for $1 \leqslant i < j \leqslant d$, every curve in $\mathcal{F}_i$ is disjoint from every curve in $\mathcal{F}_j$,*
- *$|\mathcal{F}_1 \cup \dots \cup \mathcal{F}_d| \geqslant \nu_t |\mathcal{F}| / \log |\mathcal{F}|$.*

Proof of Theorem 2. Let $\mathcal{F}$ be a family of curves obtained from the edges of G by shortening them slightly so that they do not intersect at the endpoints but all other intersection points are preserved. If follows that $\omega(\mathcal{F}) \leqslant k-1$ (as G is k-quasi-planar) and any two curves in $\mathcal{F}$ intersect in at most t points. Let ν_t, $\mathcal{F}_1, \dots, \mathcal{F}_d$, and $c_1, \dots, c_d$ be as claimed by Lemma 20. For $1 \leqslant i \leqslant d$, since $\omega(\mathcal{F}_i \smallsetminus \{c_i\}) \leqslant \omega(\mathcal{F}) - 1 \leqslant k-2$, Theorem 1 yields $\chi(\mathcal{F}_i \smallsetminus \{c_i\}) \leqslant f_t(k-2)$. Thus $\chi(\mathcal{F}_1 \cup \dots \cup \mathcal{F}_d) \leqslant f_t(k-2)+1$. For every color class $\mathcal{C}$ in a proper coloring of $\mathcal{F}_1 \cup \dots \cup \mathcal{F}_d$ with $f_t(k-2)+1$ colors, the vertices of G and the curves in $\mathcal{C}$ form a planar topological graph, and thus $|\mathcal{C}| < 3n$. Thus $|\mathcal{F}_1 \cup \dots \cup \mathcal{F}_d| < 3(f_t(k-2)+1)n$. This, the third property in Lemma 20, and the fact that $|\mathcal{F}| < n^2$ yield $|\mathcal{F}| < 3\nu_t^{-1}(f_t(k-2)+1)n \log |\mathcal{F}| < 6\nu_t^{-1}(f_t(k-2)+1)n \log n$. ◀

References

1. Eyal Ackerman. On the maximum number of edges in topological graphs with no four pairwise crossing edges. *Discrete Comput. Geom.*, 41(3):365–375, 2009.
2. Pankaj K. Agarwal, Boris Aronov, János Pach, Richard Pollack, and Micha Sharir. Quasi-planar graphs have a linear number of edges. *Combinatorica*, 17(1):1–9, 1997.
3. Edgar Asplund and Branko Grünbaum. On a colouring problem. *Math. Scand.*, 8:181–188, 1960.
4. Peter Brass, William Moser, and János Pach. *Research Problems in Discrete Geometry.* Springer, New York, 2005.
5. James P. Burling. *On coloring problems of families of prototypes.* PhD thesis, University of Colorado, Boulder, 1965.
6. Maria Chudnovsky, Alex Scott, and Paul Seymour. Induced subgraphs of graphs with large chromatic number. V. Chandeliers and strings. arXiv:1609.00314.
7. Jacob Fox and János Pach. Coloring K_k-free intersection graphs of geometric objects in the plane. *European J. Combin.*, 33(5):853–866, 2012.
8. Jacob Fox and János Pach. Applications of a new separator theorem for string graphs. *Combin. Prob. Comput.*, 23(1):66–74, 2014.

9 Jacob Fox, János Pach, and Andrew Suk. The number of edges in k-quasi-planar graphs. *SIAM J. Discrete Math.*, 27(1):550–561, 2013.

10 András Gyárfás. On the chromatic number of multiple interval graphs and overlap graphs. *Discrete Math.*, 55(2):161–166, 1985. Corrigendum. *Discrete Math.*, 62(3):333, 1986.

11 Clemens Hendler. Schranken für Färbungs- und Cliquenüberdeckungszahl geometrisch repräsentierbarer Graphen. Master's thesis, Freie Universität Berlin, 1998.

12 Alexandr V. Kostochka. O verkhnikh otsenkakh khromaticheskogo chisla grafov (On upper bounds for the chromatic number of graphs). In Vladimir T. Dementyev, editor, *Modeli i metody optimizacii*, volume 10 of *Trudy Inst. Mat.*, pages 204–226. Akad. Nauk SSSR SO, Novosibirsk, 1988.

13 Alexandr V. Kostochka. Coloring intersection graphs of geometric figures with a given clique number. In János Pach, editor, *Towards a Theory of Geometric Graphs*, volume 342 of *Contemp. Math.*, pages 127–138. AMS, Providence, 2004.

14 Alexandr V. Kostochka and Jan Kratochvíl. Covering and coloring polygon-circle graphs. *Discrete Math.*, 163(1–3):299–305, 1997.

15 Tomasz Krawczyk, Arkadiusz Pawlik, and Bartosz Walczak. Coloring triangle-free rectangle overlap graphs with $O(\log \log n)$ colors. *Discrete Comput. Geom.*, 53(1):199–220, 2015.

16 Tomasz Krawczyk and Bartosz Walczak. On-line approach to off-line coloring problems on graphs with geometric representations. *Combinatorica.* in press.

17 Michał Lasoń, Piotr Micek, Arkadiusz Pawlik, and Bartosz Walczak. Coloring intersection graphs of arc-connected sets in the plane. *Discrete Comput. Geom.*, 52(2):399–415, 2014.

18 Jiří Matoušek. Near-optimal separators in string graphs. *Combin. Prob. Comput.*, 23(1):135–139, 2014.

19 Sean McGuinness. On bounding the chromatic number of L-graphs. *Discrete Math.*, 154(1–3):179–187, 1996.

20 Sean McGuinness. Colouring arcwise connected sets in the plane I. *Graphs Combin.*, 16(4):429–439, 2000.

21 Sean McGuinness. Colouring arcwise connected sets in the plane II. *Graphs Combin.*, 17(1):135–148, 2001.

22 János Pach, Radoš Radoičić, and Géza Tóth. Relaxing planarity for topological graphs. In Ervin Győri, Gyula O. H. Katona, and László Lovász, editors, *More Graphs, Sets and Numbers*, volume 15 of *Bolyai Soc. Math. Stud.*, pages 285–300. Springer, Berlin, 2006.

23 János Pach, Farhad Shahrokhi, and Mario Szegedy. Applications of the crossing number. *Algorithmica*, 16(1):111–117, 1996.

24 Arkadiusz Pawlik, Jakub Kozik, Tomasz Krawczyk, Michał Lasoń, Piotr Micek, William T. Trotter, and Bartosz Walczak. Triangle-free geometric intersection graphs with large chromatic number. *Discrete Comput. Geom.*, 50(3):714–726, 2013.

25 Arkadiusz Pawlik, Jakub Kozik, Tomasz Krawczyk, Michał Lasoń, Piotr Micek, William T. Trotter, and Bartosz Walczak. Triangle-free intersection graphs of line segments with large chromatic number. *J. Combin. Theory Ser. B*, 105:6–10, 2014.

26 Alexandre Rok and Bartosz Walczak. Outerstring graphs are χ-bounded. In Siu-Wing Cheng and Olivier Devillers, editors, *30th Annual Symposium on Computational Geometry (SoCG 2014)*, pages 136–143. ACM, New York, 2014.

27 Andrew Suk. Coloring intersection graphs of x-monotone curves in the plane. *Combinatorica*, 34(4):487–505, 2014.

28 Andrew Suk and Bartosz Walczak. New bounds on the maximum number of edges in k-quasi-planar graphs. *Comput. Geom.*, 50:24–33, 2015.

29 Pavel Valtr. Graph drawing with no k pairwise crossing edges. In Giuseppe Di Battista, editor, *5th International Symposium on Graph Drawing (GD 1997)*, volume 1353 of *Lecture Notes Comput. Sci.*, pages 205–218. Springer, Berlin, 1997.

Barcodes of Towers and a Streaming Algorithm for Persistent Homology*

Michael Kerber[1] and Hannah Schreiber[2]

1 **Graz University of Technology, Graz, Austria**
kerber@tugraz.at
2 **Graz University of Technology, Graz, Austria**
hschreiber@tugraz.at

Abstract

A tower is a sequence of simplicial complexes connected by simplicial maps. We show how to compute a filtration, a sequence of nested simplicial complexes, with the same persistent barcode as the tower. Our approach is based on the coning strategy by Dey et al. (SoCG 2014). We show that a variant of this approach yields a filtration that is asymptotically only marginally larger than the tower and can be efficiently computed by a streaming algorithm, both in theory and in practice. Furthermore, we show that our approach can be combined with a streaming algorithm to compute the barcode of the tower via matrix reduction. The space complexity of the algorithm does not depend on the length of the tower, but the maximal size of any subcomplex within the tower. Experimental evaluations show that our approach can efficiently handle towers with billions of complexes.

1998 ACM Subject Classification G.4 Algorithm Design and Analysis

Keywords and phrases Persistent Homology, Topological Data Analysis, Matrix reduction, Streaming algorithms, Simplicial Approximation

Digital Object Identifier 10.4230/LIPIcs.SoCG.2017.57

1 Introduction

Motivation and problem statement. Persistent homology [16, 6, 15] is a paradigm to analyze how topological properties of general data sets evolve across multiple scales. Thanks to the success of the theory in finding applications (see, e.g., [25, 18] for recent enumerations), there is a growing demand for efficient computations of the involved topological invariants.

In this paper, we consider a sequence of simplicial complexes $(\mathbb{K}_i)_{i=0,\dots,m}$ and simplicial maps $\phi_i : \mathbb{K}_i \to \mathbb{K}_{i+1}$ connecting them, calling this data a *(simplicial) tower* of length m. Applying the homology functor with an arbitrary field, we obtain a *persistence module*, a sequence of vector spaces connected by linear maps. Such a module decomposes into a *barcode*, a collection of intervals, each representing a homological feature in the tower that spans over the specified range of scales.

Our computational problem is to compute the barcode of a given tower efficiently. The most prominent case of a tower is when all maps f_i are inclusion maps. In this case one obtains a *filtration*, a sequence of nested simplicial complexes. A considerable amount of work went into the study of fast algorithms for the filtration case, which culminated in fast software libraries for this task. The more general case of towers recently received growing

* The authors are supported by the Austrian Science Fund (FWF) grant number P 29984-N35.

interest in the context of sparsification technique for the Vietoris-Rips and Čech complexes; see the related work section below for a detailed discussion.

Results. As our first result, we show that any tower can be *efficiently* converted into a *small* filtration with the same barcode. Dey et al. [13] give an explicit construction, called "coning", for the generalized case of *zigzag towers*. Using a simple variant of their strategy, we obtain a filtration whose size is only marginally larger than the length of the tower. Furthermore, we experimentally show that the size is even smaller on realistic instances.

To describe our improved coning strategy, we discuss the case that a simplicial map in the tower contracts two vertices u and v. The coning strategy by Dey et al. proposes to join u with the closed star of v, making all incident simplices of v incident to u without changing the homotopy type. The vertex u is then taken as the representative of the contracted pair. We refer to the number of simplices that the join operation adds to the complex as the *cost* of the contraction. Quite obviously, the method is symmetric in u and v, and we have two choices to pick the representative, leading to potentially quite different costs. We employ the self-evident strategy to pick the representative that leads to smaller costs. This idea leads to an asymptotically improved size bound on the filtration. We prove this by an abstraction to path decompositions on weighted forest. Altogether, the worst-case size of the filtration is $O(\Delta \cdot n \cdot \log(n_0))$, where Δ is the maximal dimension of any complex in the tower, and n/n_0 is the number of simplices/vertices added to the tower.

We also provide a conversion algorithm whose time complexity is roughly proportional to the total number of simplices in the resulting filtration. One immediate benefit is a generic solution to compute barcodes of towers: just convert the tower to a filtration and apply one of the efficient implementations for barcodes of filtrations. Indeed, we experimentally show that on not-too-large towers, our approach is competitive with, and sometimes outperforms Simpers, an alternative approach that computes the barcode of towers with *annotations*, a variant of the persistent cohomology algorithm.

Our second contribution is a space-efficient version of the just mentioned algorithmic pipeline that is applicable to very large towers. To motivate the result, let the *width* of a tower denote the maximal size of any simplicial complex among the $\mathbb{K}_i$. Consider a tower with a very large length (say, larger than the number of bytes in main memory) whose width remains relatively small. In this case, our conversion algorithm yields a filtration that is very large as well. Most implementations for barcode computation read the entire filtration on initialization and algorithms based on matrix reduction are required to keep previously reduced columns. This leads to a high memory consumption for the barcode computation.

We show that with minor modifications, the standard persistent algorithm can be turned into a streaming algorithm with smaller space complexity in the case of towers. The idea is that upon contractions, simplices become *inactive* and cannot get additional cofaces. Our approach makes use of this observation by modifying the boundary matrix such that columns associated to inactive simplices can be removed. Combined with our conversion algorithm, we can compute the barcode of a tower of width ω keeping only up to $O(\omega)$ columns of the boundary matrix in memory. This yields a space complexity of $O(\omega^2)$ and a time complexity of $O((\Delta \cdot n \cdot \log(n_0)) \cdot \omega^2)$ in the worst case. We implemented a practically improved variant that makes use of additional heuristics to speed up the barcode computation in practice and resembles the *chunk algorithm* presented in [1]. We tested our implementation on various challenging data sets. The source code of the implementation is available at `https://bitbucket.org/schreiberh/sophia/`.

Related work. Already the first works on persistent homology pointed out the existence of efficient algorithm to compute the barcode invariant (or equivalently, the persistent diagram) for filtrations [16, 27]. As a variant of Gaussian elimination, the worst-case complexity is cubic. Remarkable theoretical follow-up results are a persistence algorithm in matrix multiplication time [23], an output-sensitive algorithm to compute only high-persistent features with linear space complexity [9], and a conditional lower bound on the complexity relating the problem to rank computations of sparse matrices [17].

On realistic instances, the standard algorithm has shown a quasi-linear behavior in practice despite its pessimistic worst-case complexity. Nevertheless, many improvements of the standard algorithm have been presented in the last years which improve the runtime by several orders of magnitude. One line of research exploits the special structure of the boundary matrix to speed up the reduction process [8]. This idea has led to efficient parallel algorithms for persistence in shared [1] and distributed memory [2]. Moreover, of same importance as the reduction strategy is an appropriate choice of data structures in the reduction process as demonstrated by the PHAT library [3]. A parallel development was the development of dual algorithms using persistent cohomology, based on the observation that the resulting barcode is identical [12]. The *annotation algorithm* [13, 4] is an optimized variant of this idea realized in the GUDHI library [21]. It is commonly considered as an advantage of annotations that only a cohomology basis must be kept during the reduction process, making it more space efficient than reduction-based approaches. We refer to the comparative study [24] for further approaches and software for persistence on filtrations.

Moreover, generalizations of the persistence paradigm are an active field of study. *Zigzag persistence* is a variant of persistence where the maps in the filtration are allowed to map in either direction (that is, either $\phi_i : \mathbb{K}_i \hookrightarrow \mathbb{K}_{i+1}$ or $\phi_i : \mathbb{K}_i \hookleftarrow \mathbb{K}_{i+1}$) – see [25] for a comprehensive introduction. The initial algorithms to compute this barcode [7] has been improved recently [22]. Our case of towers of complexes and simplicial maps can be modeled as a zigzag filtration and therefore sits in-between the standard and the zigzag filtration case.

Dey et al. [13] described the first efficient algorithm to compute the barcode of towers. Instead of the aforementioned coning approach explained in their paper, their implementation handles contractions with an empirically smaller number of insertions, based on the link condition. Recently, the authors have released the SIMPERS library that implements their annotation algorithm from the paper.

The case of towers has received recent attention in the context of approximate Vietoris-Rips and Čech filtrations. The motivation for approximation is that the (exact) topological analysis of a set of n points in d-dimensions requires a filtration of size $O(n^{d+1})$ which is prohibitive for most interesting input sizes. The first such type of result by Sheehy [26] resulted in a approximate filtration; however, it has been observed that the concept of towers somewhat simplifies the approximation schemes conceptually. See [13, 5, 20, 10] for examples. Very recently, the SimBa library [14] brings these theoretical approximation techniques for Vietoris-Rips complexes into practice.

Outline. We introduce the necessary basic concepts in Section 2. We describe our conversion algorithm from general towers to barcodes in Section 3. The streaming algorithm for persistence is discussed in Section 4. Several proofs and constructions are only sketched due to space constraints; see the arxiv version [19] for full arguments.

2 Background

Simplicial Complexes. Given a finite *vertex set* V, a *simplex* is a non-empty subset of V; more precisely, a *k-simplex* is a subset consisting of $k+1$ vertices, and k is called the *dimension* of the simplex. For a k-simplex σ, a simplex τ is a *face* of σ if $\tau \subseteq \sigma$. If τ is of dimension ℓ, we call it a ℓ-face. If $\ell < k$, we call τ a *proper face* of σ, and if $\ell = k-1$, we call it a *facet*. For a simplex σ and a vertex $v \notin \sigma$, we define the *join* $v * \sigma$ as the simplex $\{v\} \cup \sigma$.

An *(abstract) simplicial complex* $\mathbb{K}$ over V is a set of simplices that is closed under taking faces. We call V the *vertex set* of $\mathbb{K}$ and write $\mathcal{V}(\mathbb{K}) := V$. The *dimension* of $\mathbb{K}$ is the maximal dimension of its simplices. For $\sigma, \tau \in \mathbb{K}$, we call σ a *coface* of τ in $\mathbb{K}$ if τ is a face of σ. In this case, σ is a *cofacet* of τ if their dimensions differ by exactly one. A simplicial complex $\mathbb{L}$ is a *subcomplex* of $\mathbb{K}$ if $\mathbb{L} \subseteq \mathbb{K}$. Given $\mathcal{W} \subseteq \mathcal{V}$, the *induced subcomplex* by $\mathcal{W}$ is the set of all simplices σ in $\mathbb{K}$ with $\sigma \subseteq \mathcal{W}$. For a subcomplex $\mathbb{L} \subseteq \mathbb{K}$ and a vertex $v \in \mathcal{V}(\mathbb{K}) \setminus \mathcal{V}(\mathbb{L})$, we define the join $v * \mathbb{L} := \{v * \sigma \mid \sigma \in \mathbb{L}\}$. For a vertex $v \in \mathbb{K}$, the *star* of v in $\mathbb{K}$, denoted by $\mathrm{St}(v, \mathbb{K})$, is the set of all cofaces of v in $\mathbb{K}$. In general, the star is not a subcomplex, but we can make it a subcomplex by adding all faces of star simplices, which is denoted by the *closed star* $\overline{\mathrm{St}}(v, \mathbb{K})$. Equivalently, the closed star is the smallest subcomplex of $\mathbb{K}$ containing the star of v. The *link* of v, $\mathrm{Lk}(v, \mathbb{K})$, is defined as $\overline{\mathrm{St}}(v, \mathbb{K}) \setminus \mathrm{St}(v, \mathbb{K})$. It can be checked that the link is a subcomplex of $\mathbb{K}$. When the complex is clear from context, we will sometimes omit the $\mathbb{K}$ in the notation of stars and links.

Simplicial maps. A map $\mathbb{K} \xrightarrow{\phi} \mathbb{L}$ between simplicial complexes is called *simplicial* if with $\sigma = \{v_0, \ldots, v_k\} \in \mathbb{K}$, $\phi(\sigma)$ is equal to $\{\phi(v_0), \ldots, \phi(v_k)\}$ and $\phi(\sigma)$ is a simplex in $\mathbb{L}$. By definition, a simplicial map maps vertices to vertices and is completely determined by its action on the vertices. Moreover, the composition of simplicial maps is again simplicial.

A simple example of a simplicial map is the inclusion map $\mathbb{L} \xhookrightarrow{\phi} \mathbb{K}$ where $\mathbb{L}$ is a subcomplex of $\mathbb{K}$. If $\mathbb{K} = \mathbb{L} \cup \{\sigma\}$ with $\sigma \notin \mathbb{L}$, we call ϕ an *elementary inclusion*. The simplest example of a non-inclusion simplicial map is $\mathbb{K} \xrightarrow{\phi} \mathbb{L}$ such that there exist two vertices $u, v \in \mathbb{K}$ with $\mathcal{V}(\mathbb{L}) = \mathcal{V}(\mathbb{K}) \setminus \{v\}$, $\phi(u) = \phi(v) = u$, and ϕ is the identity on all remaining vertices of $\mathbb{K}$. We call ϕ an *elementary contraction* and write $(u, v) \rightsquigarrow u$ as a shortcut. These notions were introduced by Dey, Fan and Wang in [13] and they also showed that any simplicial map $\mathbb{K} \xrightarrow{\phi} \mathbb{L}$ can be written as the composition of elementary contractions[1] and inclusions.

A *tower* of length m is a collection of simplicial complexes $\mathbb{K}_0, \ldots, \mathbb{K}_m$ and simplicial maps $\phi_i : \mathbb{K}_i \to \mathbb{K}_{i+1}$ for $i = 0, \ldots, m-1$. From this initial data, we obtain simplicial maps $\phi_{i,j} : \mathbb{K}_i \to \mathbb{K}_j$ for $i \leq j$ by composition, where $\phi_{i,i}$ is simply the identity map on $\mathbb{K}_i$. A tower is called a *filtration* if all ϕ_i are inclusion maps. The *dimension* of a tower is the maximal dimension among the $\mathbb{K}_i$, and the *width* of a tower is the maximal number of simplices in a $\mathbb{K}_i$. For filtrations, dimension and width are determined by the dimension and size of the last complex $\mathbb{K}_m$, but this is not necessarily true for general towers.

Homology and Collapses. For a fixed base field $\mathbb{F}$, let $H_p(\mathbb{K}) := H_p(\mathbb{K}, \mathbb{F})$ the *p-dimensional homology group* of $\mathbb{K}$. It is well-known that $H_p(\mathbb{K})$ is a $\mathbb{F}$-vector space. Moreover, a simplicial map $\mathbb{K} \xrightarrow{\phi} \mathbb{L}$ induces a linear map $H_p(\mathbb{K}) \xrightarrow{\phi^*} H_p(\mathbb{L})$. In categorical terms, the equivalent

[1] They talk about "collapses" instead of "contractions", but this notion clashes with the standard notion of simplicial collapses of free faces that we use later. Therefore, we decided to use "contraction", even though the edge between the contracted vertices might not be present in the complex.

statement is that homology is a functor from the category of simplicial complexes and simplicial maps to the category of vector spaces and linear maps.

We will make use of the following homology-preserving operation: a *free face* in $\mathbb{K}$, is a simplex with exactly one proper coface in $\mathbb{K}$. An *elementary collapse* in $\mathbb{K}$ is the operation of removing a free face and its unique coface from $\mathbb{K}$, yielding a subcomplex of $\mathbb{K}$. We say that $\mathbb{K}$ *collapses to* $\mathbb{L}$, if there is a sequence of elementary collapses transforming $\mathbb{K}$ into $\mathbb{L}$. It is then well-known that the inclusion map $\mathbb{L} \overset{\phi}{\hookrightarrow} \mathbb{K}$ induces an isomorphism ϕ_* between $H_p(\mathbb{L})$ and $H_p(\mathbb{K})$.

Barcodes. A *persistence module* is a sequence vector spaces $\mathbb{V}_0, \ldots, \mathbb{V}_m$ and linear maps $f_{i,j} : \mathbb{V}_i \to \mathbb{V}_j$ for $i < j$ such that $f_{i,i} = \mathrm{id}_{\mathbb{V}_i}$ and $f_{i,k} = f_{j,k} \circ f_{i,j}$ for $i \leq k \leq j$. Persistence modules admit a decomposition into indecomposable summands in the following sense. Writing $I_{b,d}$ with $b \leq d$ for the persistence module

$$\underbrace{0 \xrightarrow{0} \ldots \xrightarrow{0} 0}_{b-1 \text{ times}} \xrightarrow{0} \underbrace{\mathbb{F} \xrightarrow{\mathrm{id}} \ldots \xrightarrow{\mathrm{id}} \mathbb{F}}_{d-b+1 \text{ times}} \xrightarrow{0} \underbrace{0 \xrightarrow{0} \ldots \xrightarrow{0} 0}_{m-d \text{ times}},$$

we can write every persistence module as the direct sum $I_{b_1,d_1} \oplus \ldots \oplus I_{b_s,d_s}$, where the direct sum of persistence modules is defined component-wise for vector spaces and linear maps in the obvious way. Moreover, this decomposition is uniquely defined up to isomorphisms and re-ordering, thus the pairs $(b_1, d_1), \ldots, (b_s, d_s)$ are an invariant of the persistence module, called its *barcode*. When the persistence module was generated by a tower, we also talk about the barcode of the tower.

Matrix reduction. In this paragraph, we assume that $(\mathbb{K}_i)_{i=0,\ldots,m}$ is a filtration such that $\mathbb{K}_0 = \emptyset$ and $\mathbb{K}_{i+1}$ has exactly one more simplex than $\mathbb{K}_i$. We label the simplices of $\mathbb{K}_m$ accordingly as $\sigma_1, \ldots, \sigma_m$, with $\mathbb{K}_i \setminus \mathbb{K}_{i-1} = \{\sigma_i\}$. The filtration can be encoded as a *boundary matrix* ∂ of dimension $m \times m$, where the (ij)-entry is 1 if σ_i is a facet of σ_j, and 0 otherwise. In other words, the j-th column of ∂ encodes the facets of σ_j, and the i-th row of ∂ encodes the cofacets of σ_i. Moreover, ∂ is upper-triangular because every $\mathbb{K}_i$ is a simplicial complex. We will sometimes identify rows and columns in ∂ with the corresponding simplex in $\mathbb{K}_m$. Adding the k-simplex σ_i to $\mathbb{K}_{i-1}$ either introduces one new homology class (of dimension k), or turns a non-trivial homology class (of dimension $k-1$) trivial. We call σ_i and the i-th column of ∂ *positive* or *negative*, respectively (with respect to the given filtration).

For the computation of the barcode, we assume for simplicity homology over the base field $\mathbb{Z}_2$, and interpret the coefficients of ∂ accordingly. In an arbitrary matrix A, a *left-to-right column addition* is an operation of the form $A_k \leftarrow A_k + A_\ell$ with $\ell < k$, where A_k and A_ℓ are columns of the matrix. The *pivot* of a non-zero column is the largest non-zero index of the corresponding column. A non-zero entry is called a pivot if its row is the pivot of the column. A matrix R is called a *reduction* of A if R is obtained by a sequence of left-to-right column additions from A and no two columns in R have the same pivot. It is well-known that, although ∂ does not have a unique reduction, the pivots of all its reductions are the same. Moreover, the pivots $(b_1, d_1), \ldots, (b_s, d_s)$ of R are precisely the barcode of the filtration. A direct consequence is that a simplex σ_i is positive if and only if the i-th column in R is zero.

The standard persistence algorithm processes the columns from left to right. In the j-th iteration, as long as the j-th column is not empty and has a pivot that appears in a previous column, it performs a left-to-right column addition. In this work, we use a simple improvement of this algorithm that is called *compression*: before reducing the j-th column, it first scans through the non-zero entries of the column. If a row index i corresponds to

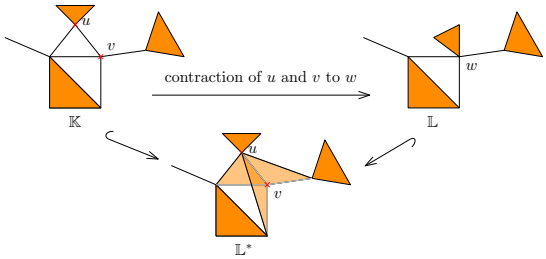


Figure 1 Construction example of $\mathbb{L}^*$, were u and v in $\mathbb{K}$ are contracted to w in $\mathbb{L}$.

a negative simplex (i.e., if the i-th column is not zero at this point in the algorithm), the row index can be deleted without changing the pivots of the matrix. After this initial scan, the column is reduced in the same way as in the standard algorithm. See [1, §. 3] for a discussion (we remark that this optimization was also used in [27]).

3 From towers to filtrations

We phrase now our first result which says that any tower can be converted into a filtration of only marginally larger size with a space-efficient streaming algorithm:

▶ **Theorem 1** (Conversion Theorem). *Let* $\mathcal{T}: \mathbb{K}_0 \xrightarrow{\phi_0} \mathbb{K}_1 \xrightarrow{\phi_1} \dots \xrightarrow{\phi_{m-1}} \mathbb{K}_m$ *be a tower where, w.l.o.g.,* $\mathbb{K}_0 = \emptyset$ *and each* ϕ_i *is either an elementary inclusion or an elementary contraction. Let* Δ *denote the dimension and* ω *the width of the tower, and let* $n \leq m$ *denote the total number of elementary inclusions, and* n_0 *the number of vertex inclusions. Then, there exists a filtration* $\mathcal{F}: \hat{\mathbb{K}}_0 \hookrightarrow \hat{\mathbb{K}}_1 \hookrightarrow \dots \hookrightarrow \hat{\mathbb{K}}_m$ *, where the inclusions are not necessarily elementary, such that* $\mathcal{T}$ *and* $\mathcal{F}$ *have the same barcode and the width of the filtration* $|\hat{\mathbb{K}}_m|$ *is at most* $O(\Delta \cdot n \log n_0)$*. Moreover,* $\mathcal{F}$ *can be computed from* $\mathcal{T}$ *with a streaming algorithm in* $O(\Delta \cdot |\hat{\mathbb{K}}_m| \cdot C_\omega)$ *time and space complexity* $O(\Delta \cdot \omega)$*, where* C_ω *is the cost of an operation in a dictionary with* ω *elements.*

The remainder of the section is organized as follows. We define $\mathcal{F}$ in Section 3.1 and prove that it yields the same barcode as $\mathcal{T}$ in Section 3.2. In Section 3.3, we prove the upper bound on the width of the filtration. In Section 3.4, we explain the algorithm to compute $\mathcal{F}$ and analyze its time and space complexity.

3.1 Active and small coning

Coning. We briefly revisit the *coning strategy* introduced by Dey, Fan and Wang [13]. Let $\phi : \mathbb{K} \to \mathbb{L}$ be an elementary contraction $(u, v) \rightsquigarrow u$ and define

$$\mathbb{L}^* = \mathbb{K} \cup \big(u * \overline{\mathrm{St}}(v, \mathbb{K})\big) \text{ (see Figure 1).}$$

Dey et al. show that $\mathbb{L} \subseteq \mathbb{L}^*$ and that the map induced by inclusion is an isomorphism between $H(\mathbb{L})$ and $H(\mathbb{L}^*)$. By applying this result at any elementary contraction, this implies that every zigzag tower can be transformed into a zigzag filtration with identical barcode.

Given a tower $\mathcal{T}$, we can also obtain an non-zigzag filtration using coning, if we continue the operation on $\mathbb{L}^*$ instead of going back to $\mathbb{L}$. More precisely, we set $\tilde{\mathbb{K}}_0 := \mathbb{K}_0$ and if ϕ_i is an inclusion of simplex σ, we set $\tilde{\mathbb{K}}_{i+1} := \tilde{\mathbb{K}}_i \cup \{\sigma\}$. If ϕ_i is a contraction $(u, v) \rightsquigarrow u$, we set $\tilde{\mathbb{K}}_{i+1} = \tilde{\mathbb{K}}_i \cup \big(u * \overline{\mathrm{St}}(v, \tilde{\mathbb{K}}_i)\big)$. Indeed, it can be proved that $(\tilde{\mathbb{K}}_i)_{i=0,\dots,m}$ has the same barcode as $\mathcal{T}$. However, the filtration will not be small, and we will define a smaller variant now.

Our new construction yields a sequence of complexes $\hat{\mathbb{K}}_0, \dots, \hat{\mathbb{K}}_m$ with $\hat{\mathbb{K}}_i \subseteq \hat{\mathbb{K}}_{i+1}$. During the construction, we maintain a flag for each vertex in $\hat{\mathbb{K}}_i$, which marks the vertex as *active* or *inactive*. A simplex is called *active* if all its vertices are active, and *inactive* otherwise. For a vertex u and a complex $\hat{\mathbb{K}}_i$, let $\mathrm{Act}\overline{\mathrm{St}}(u, \hat{\mathbb{K}}_i)$ denote its *active closed star*, which is the set of active simplices in $\hat{\mathbb{K}}_i$ in the closed star of u.

The construction is inductive, starting with $\hat{\mathbb{K}}_0 := \emptyset$. If $\mathbb{K}_i \xrightarrow{\phi_i} \mathbb{K}_{i+1}$ is an elementary inclusion with $\mathbb{K}_{i+1} = \mathbb{K}_i \cup \{\sigma\}$, set $\hat{\mathbb{K}}_{i+1} := \hat{\mathbb{K}}_i \cup \{\sigma\}$. If σ is a vertex, we mark it as active. It remains the case that $\mathbb{K}_i \xrightarrow{\phi_i} \mathbb{K}_{i+1}$ is an elementary contraction of the vertices u and v. If $|\mathrm{Act}\overline{\mathrm{St}}(u, \hat{\mathbb{K}}_i)| \leq |\mathrm{Act}\overline{\mathrm{St}}(v, \hat{\mathbb{K}}_i)|$, we set

$$\hat{\mathbb{K}}_{i+1} = \hat{\mathbb{K}}_i \cup \left(v * \mathrm{Act}\overline{\mathrm{St}}(u, \hat{\mathbb{K}}_i)\right)$$

and mark u as inactive. Otherwise, we do the same by inverting the role of u and v in the construction. This ends the description of the construction. We write $\mathcal{F}$ for the filtration $(\hat{\mathbb{K}}_i)_{i=0,\dots,m}$.

There are two major changes compared to the construction of $(\tilde{\mathbb{K}}_i)_{i=0,\dots,m}$. First, to counteract the potentially large growth of the involved cones, we restrict coning to active simplices. We will show below that the subcomplex of $\hat{\mathbb{K}}_i$ induced by the active vertices is isomorphic to $\mathbb{K}_i$. As a consequence, we add the same number of simplices by passing from $\hat{\mathbb{K}}_i$ to $\hat{\mathbb{K}}_{i+1}$ as in the approach by Dey et al. does when passing from $\mathbb{K}$ to $\mathbb{L}^*$.

A second difference is that our strategy exploits that an elementary contraction of two vertices u and v leaves us with a choice: we can either take u or v as the representative of the contracted vertex. In terms of simplicial maps, these two choices correspond to setting $\phi_i(u) = \phi_i(v) = u$ or $\phi_i(u) = \phi_i(v) = v$, if ϕ_i is the elementary contraction of u and v. It is obvious that both choices yield identical complexes $\mathbb{K}_{i+1}$ up to renaming of vertices. However, the choices make a difference in terms of the size of $\hat{\mathbb{K}}_{i+1}$, because the active closed star of u to v in $\hat{\mathbb{K}}_i$ might differ in size. Our construction simply choose the representative which causes the smaller $\hat{\mathbb{K}}_{i+1}$.

3.2 Topological equivalence

We assume w.l.o.g. that the vertices in $\mathbb{K}_i$ are named such that, whenever our construction encounters an elementary contraction ϕ_i of u and v and turns v inactive, we have $\phi_i(u) = \phi_i(v) = u$. With this convention, $\mathbb{K}_i$ is the subcomplex of $\hat{\mathbb{K}}_i$ induced by the active vertices.

▶ **Lemma 2.** *A simplex σ is in $\mathbb{K}_i$ if and only if σ is an active simplex in $\hat{\mathbb{K}}_i$.*

The proof works by induction on i, analyzing carefully the effect of a contraction on $\hat{\mathbb{K}}_i$ and $\mathbb{K}_i$. Moreover, when an elementary contraction of u and v turns v inactive, every simplex $\sigma = \{v, v_1, \dots, v_d\}$ that becomes inactive in $\hat{\mathbb{K}}_i$ has a corresponding simplex $\tau = \{u, v, v_1, \dots, v_d\}$ that also becomes inactive. The pairs (σ, τ) can be arranged in collapsible pairs, which implies with an inductive argument:

▶ **Lemma 3.** *For every $0 \leq i \leq m$, the complex $\hat{\mathbb{K}}_i$ collapses to $\mathbb{K}_i$.*

▶ **Proposition 4.** *$\mathcal{T}$ and $\mathcal{F}$ have the same barcode.*

Proof Sketch. Let $\hat{\phi}_i : \hat{\mathbb{K}}_i \to \hat{\mathbb{K}}_{i+1}$ and $\mathrm{inc}_i : \mathbb{K}_i \to \hat{\mathbb{K}}_i$ denote inclusion maps. Lemma 3 implies that the induced homology map $\mathrm{inc}_i^* : H(\mathbb{K}_i) \to H(\hat{\mathbb{K}}_i)$ is an isomorphism for all

$0 \leq i \leq m$. The following diagram connects the persistence modules induced by $\mathcal{T}$ and $\mathcal{F}$:

$$\begin{array}{ccccccc} H(\mathbb{K}_0) & \xrightarrow{\phi_0^*} & H(\mathbb{K}_1) & \xrightarrow{\phi_1^*} & \ldots & \xrightarrow{\phi_{m-1}^*} & H(\mathbb{K}_m) \\ \downarrow{\scriptstyle \mathrm{inc}_0^*} & & \downarrow{\scriptstyle \mathrm{inc}_1^*} & & & & \downarrow{\scriptstyle \mathrm{inc}_m^*} \\ H(\hat{\mathbb{K}}_0) & \xrightarrow{\hat{\phi}_0^*} & H(\hat{\mathbb{K}}_1) & \xrightarrow{\hat{\phi}_1^*} & \ldots & \xrightarrow{\hat{\phi}_{m-1}^*} & H(\hat{\mathbb{K}}_m) \end{array} \tag{1}$$

Our result follows from the *Persistence Equivalence Theorem* [15, p.159] which asserts that $(\mathbb{K}_j)_{j=0,\ldots,m}$ and $(\hat{\mathbb{K}}_j)_{j=0,\ldots,m}$ have the same barcode if (1) commutes, that is, if $\mathrm{inc}_{i+1}^* \circ \phi_i^* = \hat{\phi}_i^* \circ \mathrm{inc}_i^*$, for all $0 \leq i < m$. The latter statements follows from the fact that $\mathrm{inc}_{i+1} \circ \phi_i$ and $\hat{\phi}_i \circ \mathrm{inc}_i$ are contiguous maps. ◀

3.3 Size analysis

The contracting forest. We associate a rooted labeled forest $\mathcal{W}_j$ to a prefix $\emptyset = \mathbb{K}_0 \xrightarrow{\phi_0} \ldots \xrightarrow{\phi_{j-1}} \mathbb{K}_j$ of $\mathcal{T}$ inductively as follows: For $j = 0$, $\mathcal{W}_0$ is the empty forest. Let $\mathcal{W}_{j-1}$ be the forest of $\mathbb{K}_0 \to \ldots \to \mathbb{K}_{j-1}$. If ϕ_{j-1} is an elementary inclusion of a d-simplex, we have two cases: if $d > 0$, set $\mathcal{W}_j := \mathcal{W}_{j-1}$. If a vertex v is included, $\mathcal{W}_j := \mathcal{W}_{j-1} \cup \{x\}$, with x a single node tree labeled with v. If ϕ_{j-1} is an elementary contraction contracting two vertices u and v in $\mathbb{K}_{j-1}$, there are two trees in $\mathcal{W}_{j-1}$, whose roots are labeled u and v. In $\mathcal{W}_j$, these two trees are merged by making their roots children of a new root, which is labeled with the vertex that u and v are mapped to. So $\mathcal{W}_j$ is *full*, that is, every node has 0 or 2 children.

Let $\mathcal{W} := \mathcal{W}_m$ denote the forest of the tower $\mathcal{T}$. Let Σ denote the set of all simplices that are added at elementary inclusions in $\mathcal{T}$, and recall that $n = |\Sigma|$. For a node x in $\mathcal{W}$, we denote by $E(x) \subseteq \Sigma$ the subset of simplices with at least one vertex that appears as label in the subtree of $\mathcal{W}$ rooted at x. If y_1 and y_2 are the children of x, the following follows at once:

$$|E(x)| \quad \geq \quad |E(y_1)| + |E(y_2) \setminus E(y_1)|\,. \tag{2}$$

We say that the set N of nodes in $\mathcal{W}$ is *independent*, if there are no two nodes $x_1 \neq x_2$ in N, such that x_1 is an ancestor of x_2 in $\mathcal{W}$. A vertex in $\mathbb{K}_i$ appears as label in at most one $\mathcal{W}$-subtree rooted at a vertex in the independent set N. Thus, a d-simplex σ can only appear in up to $d+1$ E-sets of vertices in N. That implies:

▶ **Lemma 5.** *Let N be an independent set of vertices in $\mathcal{W}$. Then, $\sum_{x \in N} |E(x)| \leq (\Delta+1) \cdot n$.*

The cost of contracting. In order to bound the total size of $\hat{\mathbb{K}}_m$, we need to bound the number of simplices added in all these contractions. We define the *cost* of a contraction ϕ_i as $|\hat{\mathbb{K}}_{i+1} \setminus \hat{\mathbb{K}}_i|$. Since each contraction corresponds to a node x in $\mathcal{W}$, we can associate these costs to the internal nodes in the forest, denoted by $c(x)$. The leaves get cost 0.

▶ **Lemma 6.** *For an internal node x of $\mathcal{W}$ with children y_1, y_2, $c(x) \leq 2 \cdot |E(y_1) \setminus E(y_2)|$.*

Proof Sketch. Let $\phi_i : \mathbb{K}_i \to \mathbb{K}_{i+1}$ denote the contraction that is represented by the node x, and let w_1 and w_2 the labels of its children y_1 and y_2, respectively. By construction, w_1 and w_2 are vertices in $\mathbb{K}_i$ that are contracted by ϕ_i. Let $C_1 = \overline{\mathrm{St}}(w_1, \mathbb{K}_i) \setminus \overline{\mathrm{St}}(w_2, \mathbb{K}_i)$ and $C_2 = \overline{\mathrm{St}}(w_2, \mathbb{K}_i) \setminus \overline{\mathrm{St}}(w_1, \mathbb{K}_i)$. By Lemma 2, $\overline{\mathrm{St}}(w_1, \mathbb{K}_i) = \mathrm{Act}\overline{\mathrm{St}}(w_1, \hat{\mathbb{K}}_i)$, and the same for w_2. So, because the common simplices of the two active closed stars do not influence the cost of the contraction, we have $c(x) \leq \min\{|C_1|, |C_2|\}$, because the contraction is defined such that the resulting complex is as small as possible.

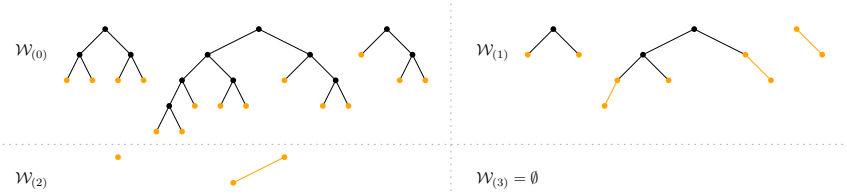


Figure 2 Iterations of the pruning procedure. The only-child-paths are marked in color.

In particular, $c(x) \leq |C_1|$. It is left to show that $|C_1| \leq 2 \cdot |E(y_1) \setminus E(y_2)|$. We do this by a simple charging scheme which attributes the existence of a simplex in C_1 to a simplex in $E(y_1) \setminus E(y_2)$ such that no simplex in the latter set is charged more than twice. ◀

An *ascending path* $(x_1, ..., x_L)$, with $L \geq 1$, is a path in a forest such that x_{i+1} is the parent of x_i, for $1 \leq i < L$. We call L the *length* of the path and x_L its *endpoint*. For ascending paths in $\mathcal{W}$, the *cost* of the path is the sum of the costs of the nodes. The set P of ascending paths is *independent*, if the endpoints in P are pairwise different and form an independent set of nodes. We define the *cost* of P as the sum of the costs of the paths in P.

▶ **Lemma 7.** *An ascending path with endpoint x has cost at most $2 \cdot |E(x)|$. An independent set of ascending paths in $\mathcal{W}$ has cost at most $2 \cdot (\Delta + 1) \cdot n$.*

Proof. For the first statement, let $p = (x_1, ..., x_L)$ be an ascending path with $v_L = v$. Without loss of generality, we can assume the the path starts with a leaf x_1, because otherwise, we can always extend the path to a longer path with at least the same cost. We let $p_i = (x_1, \ldots, x_i)$ denote the sub-path ending at x_i, for $i = 1, \ldots, L$, so that $p_L = p$. We let $c(p_i)$ denote the cost of the path p_i and show by induction that $c(p_i) \leq 2 \cdot |E(x_i)|$. For $i = 1$, this follows because $c(p_1) = 0$. For $i = 2, \ldots, L$, x_i is an internal node, and its two children are x_{i-1} and some other node x'_{i-1}. Using induction and Lemma 6, we have that

$$c(p_i) = c(p_{i-1}) + c(x_i) \leq 2 \cdot (|E(x_{i-1})| + |E(x'_{i-1}) \setminus E(x_{i-1})|) \leq 2 \cdot |E(x_i)|,$$

where the last inequality follows from (2). The second statement follows from Lemma 5 because the endpoints of the paths form an independent set in $\mathcal{W}$. ◀

Ascending path decomposition. An *only-child-path* in a binary tree is an ascending path starting in a leaf and ending at the first encountered node that has a sibling, or at the root of the tree. Consider the following pruning procedure for a full binary forest $\mathcal{W}$. Set $\mathcal{W}_{(0)} \leftarrow \mathcal{W}$. In iteration i, we obtain the forest $\mathcal{W}_{(i)}$ from $\mathcal{W}_{(i-1)}$ by deleting the only-child-paths of $\mathcal{W}_{(i-1)}$. We stop when $\mathcal{W}_{(i)}$ is empty. Figure 2 shows the pruning procedure on an example. We define the following integer valued function for nodes in $\mathcal{W}$:

$$r(x) = \begin{cases} 1, & \text{if } x \text{ is a leaf,} \\ r(y_1) + 1, & \text{if } x \text{ has children } y_1, y_2 \text{ and } r(y_1) = r(y_2), \\ \max\{r(y_1), r(y_2)\}, & \text{if } x \text{ has children } y_1, y_2 \text{ and } r(y_1) \neq r(y_2). \end{cases}$$

With two simple inductive arguments, we can show the next two lemmas:

▶ **Lemma 8.** *A node x of a full binary forest $\mathcal{W}$ is deleted in the pruning procedure during the $r(x)$-th iteration.*

▶ **Lemma 9.** *For a node x in a full binary forest, let $s(x)$ denote the number of nodes in the subtree rooted at x. Then $s(x) \geq 2^{r(x)} - 1$. In particular, $r(x) \leq \log_2(s(x) + 1)$.*

With that, we can bound the size of the constructed filtration.

▶ **Proposition 10.** *$|\hat{\mathbb{K}}_m| \leq n + 2 \cdot (\Delta + 1) \cdot n \cdot (1 + \log_2(n_0)) = O(n \cdot \Delta \cdot \log_2(n_0))$, where n_0 is the number of vertices included in $\mathcal{T}$.*

Proof Sketch. The first summand counts the number of elementary inclusions, the second one the total cost of the contractions. The costs of all nodes removed in one iteration of the pruning procedure are at most $2 \cdot (\Delta + 1) \cdot n$ by Lemma 7 because the considered only-child-paths form an independent set of ascending paths. By Lemma 9, all nodes have been considered after $1 + \log_2(n_0)$ iterations. ◀

3.4 Algorithm

A *dictionary* is a data structure that stores a set of *items* of the form `(k,v)`, where `k` is called the *key* and is unique and `v` is called the *value* of the item. The dictionary supports three operations: `insert(k,v)` adds a new item, `delete(k)` removes the item with key `k` and `search(k)` returns the item with key `k`, or returns that no such item exists. Common realizations are balanced binary search trees [11, §12] and hash tables [11, §11].

Simplicial complexes by dictionaries. The main data structure of our algorithm is a dictionary D that represents a simplicial complex. Every item stored in the dictionary represents a simplex, whose key is the list of its vertices. Every simplex σ itself stores an dictionary CoF_σ. Every item in CoF_σ is a pointer to another item in D, representing a cofacet τ of σ. The key of the item is a vertex identifier (e.g., an integer) for v such that $\tau = v * \sigma$. Assuming that the size of a dictionary is linear in the number n of stored elements, the size of D is in $O(n\Delta)$, if Δ is the dimension of the represented complex. With the right `search(k)` function and key encoding, each simplex insertion and deletion requires $O(\Delta)$ dictionary operations. In what follows, it is convenient to assume that dictionary operations have unit costs; we multiply the time complexity with the cost of a dictionary operation at the end to compensate for this simplification.

The conversion algorithm. We assume that the tower $\mathcal{T}$ is given to us as a stream where each element represents a simplicial map ϕ_i in the tower: an element starts with a token {`I`, `C`} that specifies the type of the map and ends with the identifiers of the involved vertices. The algorithm outputs a stream of simplices specifying the filtration $\mathcal{F}$: while handling the i-th input element, it outputs the simplices of $\hat{\mathbb{K}}_{i+1} \setminus \hat{\mathbb{K}}_i$ in increasing order of dimension. We use an initially empty dictionary D and maintain the invariant that after the i-th iteration, D represents the active subcomplex of $\hat{\mathbb{K}}_i$, which is equal to $\mathbb{K}_i$ by Lemma 2.

If the algorithm reads an inclusion of a simplex σ from the stream, it simply adds σ to D and writes σ to the output stream. If the algorithm reads a contraction of two vertices u and v, from $\mathbb{K}_i$ to $\mathbb{K}_{i+1}$, we let $c_i = |\hat{\mathbb{K}}_{i+1} \setminus \hat{\mathbb{K}}_i|$ denote the cost of the contraction. The first step is to determine which of the vertices has the smaller closed star in $\mathbb{K}_i$. The size of the closed star of a vertex v could be computed by a simple graph traversal in D, starting at a vertex v and following the cofacet pointers recursively. However, we want to identify the smaller star with only $O(c_i)$ operations. Therefore, we change the traversal in several ways: First of all, observe that $|\overline{\mathrm{St}}(u)| \leq |\overline{\mathrm{St}}(v)|$ if and only if $|\mathrm{St}(u)| \leq |\mathrm{St}(v)|$. Now define $\mathrm{St}(u, \neg v) := \mathrm{St}(u) \setminus \mathrm{St}(v)$. Then, $|\mathrm{St}(u)| \leq |\mathrm{St}(v)|$ if and only if $|\mathrm{St}(u, \neg v)| \leq |\mathrm{St}(v, \neg u)|$,

because we subtracted the intersection of the stars on both sides. Finally, note that $\min\{|\mathrm{St}(u,\neg v)|, |\mathrm{St}(v,\neg u)|\} \leq c_i$. Moreover, we can count the size of $\mathrm{St}(u,\neg v)$ by a cofacet traversal from u, ignoring cofacets that contain v in $O(|\mathrm{St}(u,\neg v)|)$ time. Finally, we count the sizes of $\mathrm{St}(u,\neg v)$ and $\mathrm{St}(v,\neg u)$ at the same time by a simultaneous graph traversal of both, terminating as soon as one of the traversal stops. The running time is then proportional to $2 \cdot \min\{|\mathrm{St}(u,\neg v)|, |\mathrm{St}(v,\neg u)|\} = O(c_i)$, as required. Assume w.l.o.g. that $|\overline{\mathrm{St}}(u)| \leq |\overline{\mathrm{St}}(v)|$. Also in time $O(c_i)$, we can obtain $\mathrm{St}(u,\neg v)$. We sort its elements by increasing dimension, which can be done in $O(c_i + \Delta)$ using integer sort. For each simplex $\sigma = \{u, v_1, \ldots, v_k\} \in \mathrm{St}(u,\neg v)$ in order, we check whether $\{v, v_1, \ldots, v_k\}$ is in D. If not, we add it to D and also write it to the output stream. Then, we output $\{u, v, v_1, \ldots, v_k\}$. At the end of the loop, we wrote exactly the simplices in $\mathbb{K}_{i+1} \setminus \mathbb{K}_i$ to the output stream. It remains to maintain the invariant on D. Assuming still that $|\overline{\mathrm{St}}(u)| \leq |\overline{\mathrm{St}}(v)|$, u turns inactive in $\hat{\mathbb{K}}_{i+1}$. We simply traverse over all cofaces of u and remove all encountered simplices from D.

Complexity analysis. By applying the operation costs on the above described algorithm, we obtain the following statement. Combined with Propositions 4 and 10, it completes the proof of Theorem 1.

▶ **Proposition 11.** *The algorithm requires $O(\Delta \cdot \omega)$ space and $O(\Delta \cdot |\hat{\mathbb{K}}_m| \cdot C_\omega)$ time, where $\omega = \max_{i=0,\ldots,m} |\mathbb{K}_i|$ and C_ω is the cost of an operation in a dictionary with ω elements.*

Proof Sketch. By the above description, the cost of a contraction can be bounded by $O(\Delta(c_i + d_i)C_\omega)$, where c_i is the number of simplices added, and d_i the number of simplices that become inactive in the i-th iteration. Because $\sum c_i$ and $\sum d_i$ are both in $O(|\mathbb{K}_m|)$, the result follows. ◀

Using balanced binary trees as dictionary, we get $C_\omega = O(\Delta \log \omega)$ because comparing two keys costs $O(\Delta)$. Using hash tables, the expected complexity is $C_\omega = O(\Delta)$.

Experimental results. The following tests where made on a 64-bit Linux (Ubuntu) HP machine with a 3.50 GHz Intel processor and 63 GB RAM. The programs were all implemented in C++ and compiled with optimization level –O2.

To test the performance of our algorithm, we compared it to the software Simpers (downloaded in May 2016)[2], which is the implementation of the Annotation Algorithm from Dey, Fan and Wang described in [13]. Simpers computes the persistence of the given filtration, so we add to our time the time the library PHAT (version 1.4.1) needs to compute the persistence from the generated filtration (with default parameters).

The results of the tests are in Table 1. The timings for File IO are not included in the process time of PHAT and Simpers. The memory peak was obtained via the '/usr/bin/time –v' Linux command. The first three towers in the table, data1-3, were generated incrementally on a set of n_0 vertices: In each iteration, with 90% probability, a new simplex is included, that is picked uniformly at random among the simplices whose facets are all present in the complex, and with 10% probability, two randomly chosen vertices of the complex are contracted. This is repeated until the complex on the remaining k vertices forms a $k-1$-simplex, in which case no further simplex can be added. The remaining data was generated from the SimBa (downloaded in June 2016) library with default parameters using the point clouds from [14]. To obtain the towers that SimBa computes internally, we included a print command at a suitable location in the SimBa code.

[2] `http://web.cse.ohio-state.edu/~tamaldey/SimPers/Simpers.html`

Table 1 Experimental results. The symbol ∞ means that the calculation time exceeded 12 hours.

						Alg1 + PHAT			Simpers	
	c	n	n_0	Δ	ω	filtration size	time (s)	mem. peak (kB) Alg1 / total	time (s)	mem. peak (kB)
data1	495	4 833	500	4	2 908	19 747	0.12	4 644 / 7 040	2.49	10188
data2	795	7 978	800	4	4 816	35 253	0.20	5 424 / 10 228	13.97	20308
data3	794	8 443	800	5	5 155	38 101	0.22	5 744 / 10 916	19.29	24 924
GPS	1 746	8 585	1 747	3	1 747	9 063	0.07	4 072 / 5 292	0.35	6 064
KB	22 499	95 019	22 500	3	22 500	133 433	0.50	10 520 / 18 712	2.83	24 460
MC	23 074	143 928	23 075	3	28 219	185 447	0.72	14 636 / 25 272	4.12	26 020
S3	252 995	1 473 580	252 996	4	252 996	1 824 461	10.09	94 020 / 221 636	49.86	239 404
PC25	14 999	10 246 125	15 000	3	2 191 701	12 283 003	135.02	1 029 680 / 2 223 544	∞	–

To verify that the space consumption of our algorithm does not dependent on the length of the tower, we constructed an additional example whose size exceeds our RAM capacity, but whose width is small: we obtained a tower of length about $3.5 \cdot 10^9$ which has a file size of about 73 GB, but only has a width of 367. Our algorithm took about 2 hours to convert this tower into a filtration of size roughly $4.6 \cdot 10^9$. During the conversion, the virtual memory used was constantly around 22 MB and the resident set size about 3.8 MB only, confirming the theoretical prediction that the space consumption is independent of the length.

4 Persistence by Streaming

If the original tower is small, we want to be able to compute its persistence even if the tower is extremely long. So we design here a streaming variation of the reduction algorithm that computes the barcode of filtrations with a more efficient memory use than the standard algorithm. More precisely, we will prove the following theorem:

▶ **Theorem 12.** *With the same notation as in Theorem 1, we can compute the barcode of a tower $\mathcal{T}$ in worst-case time $O(\omega^2 \cdot \Delta \cdot n \cdot \log n_0)$ and space complexity $O(\omega^2)$*

Algorithmic description. The input to the algorithm is a stream of elements, each starting with a token {`Add`, `I`} followed by an identifier which represents a simplex σ. In the `Add` case, this is followed by the identifiers of the facets of σ. The `I` means that σ has become inactive.

The algorithm uses a matrix data type M as its main data structure. We realize M as a dictionary of columns, indexed by a simplex identifier. Each column is a sorted linked list of identifiers corresponding to the non-zero row indices of the column. In particular, we can access the pivot of the column in constant time and we can add two columns in time proportional to the maximal size of the involved lists. There are two secondary data structures that we mention briefly. Firstly, a dictionary where the keys represents row indices r and their corresponding value is the column that has r as pivot. Secondly, a dictionary representing the set of active simplex identifiers, plus a positive/negative flag. It is straight-forward to maintain these structures during the algorithm, and we will omit the description of the required operations.

The algorithm uses two subroutines. The first one, called `reduce_column`, takes a column identifier j as input and iterate through the non-zero row indices of j: if an index i is the index of an inactive and negative column in M, remove the entry from the column j (cf. to "compression" at end of Section 2). Afterwards, while the column is non-empty, and its

pivot i is the pivot of another column $k < j$ in the matrix, add column k to column j. The second subroutine, `remove_row`, takes a index ℓ as input: let j be the column with ℓ as pivot. Traverse all non-zero columns of the matrix except column j. If a column $i \neq j$ has a non-zero entry at row ℓ, add column j to column i. After traversing all columns, remove column j from M.

The main algorithm can be described easily now: if we add a simplex, we add the column to M and call `reduce_column` on it. If at the end of that routine, the column is empty, it is removed from M. If the column is not empty and has pivot ℓ, we report (ℓ, j) as a persistence pair and check whether ℓ is active. If not, we call `remove_row(`ℓ`)`. If the input stream specifies that simplex ℓ becomes inactive, we check whether ℓ appears as pivot in the matrix and call `remove_row(`ℓ`)` in this case.

▶ **Proposition 13.** *The algorithm computes the correct barcode.*

Proof Sketch. First, note that removing a column from M within the procedure `remove_row` does not affect further reduction steps. Then, `remove_row` might also include right-to-left column additions. But we can easily show that a column reduced with right-to-left additions can also be expressed by a sequence of left-to-right column additions, and thus yields the same pivot as in the standard algorithm. ◀

Complexity analysis. We analyze how large the structure M can become during the algorithm. After every iteration, the matrix represents the reduced boundary matrix of some intermediate complex $\hat{\mathbb{L}}$ with $\hat{\mathbb{K}}_i \subseteq \hat{\mathbb{L}} \subseteq \hat{\mathbb{K}}_{i+1}$ for some $i = 0, \dots, m$. Moreover, the active simplices define a subcomplex $\mathbb{L} \subseteq \hat{\mathbb{L}}$ and there is a moment during the algorithm where $\hat{\mathbb{L}} = \hat{\mathbb{K}}_i$ and $\mathbb{L} = \mathbb{K}_i$, for every $i = 0, \dots, m$. We call this the *i-th checkpoint*.

▶ **Lemma 14.** *At every moment, the number of columns stored in M is at most 2ω.*

This come from the fact that, throughout the algorithm, a column is stored in M only if it has an active pivot. The number of rows is more difficult to bound because we cannot guarantee that each column in M corresponds to an active simplex. Still, the number of rows is asymptotically the same:

▶ **Lemma 15.** *At every moment, the number of rows stored in M is at most 4ω.*

Proof Sketch. Consider a row index ℓ and a time in the algorithm where M represents $\hat{\mathbb{L}}$. There are at most 2ω active row indices at any time. Moreover, following the algorithm, ℓ cannot represent an inactive negative simplex neither an active one that was paired with another index. Therefore, we restrict our attention to the remaining case, that ℓ is inactive, positive and has not been paired so far. It is well-known that in this case, ℓ is the generator of an homology class of $\hat{\mathbb{L}}$. Let $\beta(\hat{\mathbb{L}}) := \sum_{i=0}^{\Delta} \beta_i(\hat{\mathbb{L}})$ denote the sum of the Betti numbers of the complex. Then, it follows that the number of such row indices is at most $\beta(\hat{\mathbb{L}})$. We have that $\beta(\hat{\mathbb{K}}_i) = \beta(\mathbb{K}_i)$ by Lemma 3, and since $\mathbb{K}_i$ has at most ω simplices, $\beta(\mathbb{K}_i) \leq \omega$. Since we add at most ω simplices to get from $\hat{\mathbb{K}}_i$ to $\hat{\mathbb{L}}$, and each addition can increase β by at most one, we have that $\beta(\hat{\mathbb{L}}) \leq 2\omega$. ◀

▶ **Proposition 16.** *The algorithm runs in time $O(\omega^2 \cdot \Delta \cdot n \cdot \log n_0)$ with $O(\omega^2)$ space.*

Proof. The space complexity is immediately clear from the preceding two lemmas, as M is the dominant data structure in terms of space consumption. For the time complexity, we observe that both subroutines `reduce_column` and `remove_row` need $O(\omega)$ column additions and $O(\omega)$ dictionary operations in the worst case. A column addition costs $O(\omega)$, and a

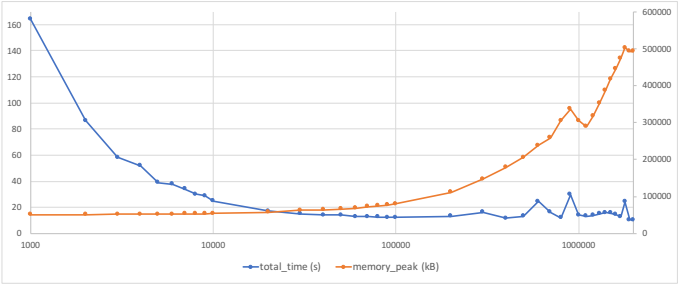


Figure 3 Evolution of processing time (left Y-axis in sec) and process memory peak (right Y-axis in kB) depending on the chunk size (logarithmic X-axis).

dictionary operation is not more expensive So, the complexity of both methods is $O(\omega^2)$. Since each routine is called at most once per input element, and there are $O(\Delta \cdot n \cdot \log n_0)$ elements by Theorem 1, the bound follows. ◀

Implementation. The algorithm just described is not efficient in practice, partially because `remove_row` scans the entire matrix M which should be avoided. We outline a variant that behaves better in practice. The idea is to perform a "batch" variant of the previous algorithm: We define a *chunk size* C and read in C elements from the stream; we insert added columns in the matrix, but not reducing the columns yet. After having read C elements, we start the reduction of the newly inserted columns using the *clearing optimization*: that is, we go in decreasing dimension and remove a column as soon as its index becomes the pivot of another column; see [8] for details. After the reduction ends, except for the last chunk, we go over the columns of the matrix and check for each pivot whether it is active. If it is, we traverse its row entries in decreasing order, but skipping the pivot. Let ℓ be the current entry. If ℓ is the inactive pivot of some column j, we add j to the current column. If ℓ is inactive and represents a negative column, we delete ℓ from the current column. After performing these steps for all remaining columns of the matrix, we go over all columns again, deleting every column with inactive pivot.

How to choose the parameter C? The chunk provides a trade-off between time and space efficiency. Roughly speaking, the matrix can have up to $O(\omega + C)$ columns during this reduction, but the larger the chunks are, the more benefit one can draw from clearing.

Experimental evaluation. The tests were made with the same setup as in Section 3.4. Figure 3 shows the effect of the chunk size parameter C on the runtime and memory consumption of the algorithm. The data used is `S3` (see Section 3.4); we also performed the tests on the other examples from Table 1, with similar outcome. The File IO operations are included in the measurements. Confirming the theory, as the chunk size decreases, our implementation needs less space but more computation time (while the running time seems to increase slightly again for larger chunk sizes).

For the $4.6 \cdot 10^9$-inclusions-tower from Section 3.4, with $C = 200\,000$, the algorithm took around 4.5 hours, the virtual memory used was constantly around 68 MB and the resident set size constantly around 49 MB, confirming the theoretical statement that the memory size does not depend on the length of the filtration.

References

1 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.

2 U. Bauer, M. Kerber, and J. Reininghaus. Distributed Computation of Persistent Homology. In *Workshop on Algorithm Engineering and Experiments (ALENEX)*, pages 31–38, 2014.

3 U. Bauer, M. Kerber, J. Reininghaus, and H. Wagner. Phat – Persistent Homology Algorithms Toolbox. *Journal of Symbolic Computation*, 78:76–90, 2017.

4 J.-D. Boissonnat, T. Dey, and C. Maria. The Compressed Annotation Matrix: An Efficient Data Structure for Computing Persistent Cohomology. In *European Symp. on Algorithms (ESA)*, pages 695–706, 2013.

5 M. Botnan and G. Spreemann. Approximating Persistent Homology in Euclidean space through collapses. *Applied Algebra in Engineering, Communication and Computing*, 26:73–101, 2015.

6 G. Carlsson. Topology and Data. *Bulletin of the AMS*, 46:255–308, 2009.

7 G. Carlsson, V. de Silva, and D. Morozov. Zigzag Persistent Homology and Real-valued Functions. In *ACM Symp. on Computational Geometry (SoCG)*, pages 247–256, 2009.

8 C. Chen and M. Kerber. Persistent Homology Computation With a Twist. In *European Workshop on Computational Geometry (EuroCG)*, pages 197–200, 2011.

9 C. Chen and M. Kerber. An output-sensitive algorithm for persistent homology. *Computational Geometry: Theory and Applications*, 46:435–447, 2013.

10 A. Choudhary, M. Kerber, and S. Raghvendra. Polynomial-Sized Topological Approximations Using The Permutahedron. In *32nd Int. Symp. on Computational Geometry (SoCG)*, pages 31:1–31:16, 2016.

11 T. Cormen, C. Leiserson, R. Rivest, and C. Stein. *Introduction to algorithms*. The MIT press, 3rd edition, 2009.

12 V. de Silva, D. Morozov, and M. Vejdemo-Johansson. Dualities in persistent (co)homology. *Inverse Problems*, 27:124003, 2011.

13 T. Dey, F. Fan, and Y. Wang. Computing Topological Persistence for Simplicial Maps. In *ACM Symp. on Computational Geometry (SoCG)*, pages 345–354, 2014.

14 T. 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.

15 H. Edelsbrunner and J. Harer. *Computational Topology: an introduction*. American Mathematical Society, 2010.

16 H. Edelsbrunner, D. Letscher, and A. Zomorodian. Topological Persistence and Simplification. *Discrete & Computational Geometry*, 28:511–533, 2002.

17 H. Edelsbrunner and S. Parsa. On the Computational Complexity of Betti Numbers: Reductions from Matrix Rank. In *ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 152–160, 2014.

18 M. Kerber. Persistent Homology: State of the art and challenges. *Internationale Mathematische Nachrichten*, 231:15–33, 2016.

19 M. Kerber and H. Schreiber. Barcodes of Towers and a Streaming Algorithm for Persistent Homology. *arXiv*, abs/1701.02208, 2017. URL: `http://arxiv.org/abs/1701.02208`.

20 M. Kerber and R. Sharathkumar. Approximate Čech Complex in Low and High Dimensions. In *Int. Symp. on Algortihms and Computation (ISAAC)*, pages 666–676, 2013.

21 C. Maria, J.-D. Boissonnat, M. Glisse, and M. Yvinec. The Gudhi Library: Simplicial Complexes and Persistent Homology. In *Int. Congress on Mathematical Software (ICMS)*, volume 8592 of *Lecture Notes in Computer Science*, pages 167–174, 2014.

22 C. Maria and S. Oudot. Zigzag Persistence via Reflections and Transpositions. In *ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 181–199, 2015.

23 N. Milosavljevic, D. Morozov, and P. Skraba. Zigzag persistent homology in matrix multiplication time. In *ACM Symp. on Computational Geometry (SoCG)*, pages 216–225, 2011.

24 N. Otter, M. Porter, U. Tillmann, P. Grindrod, and H. Harrington. A roadmap for the computation of persistent homology. *arXiv*, abs/1506.08903, 2015.

25 S. Oudot. *Persistence theory: From Quiver Representation to Data Analysis*, volume 209 of *Mathematical Surveys and Monographs*. American Mathematical Society, 2015.

26 D. Sheehy. Linear-size approximation to the Vietoris-Rips Filtration. *Discrete & Computational Geometry*, 49:778–796, 2013.

27 A. Zomorodian and G. Carlsson. Computing Persistent Homology. *Discrete & Computational Geometry*, 33:249–274, 2005.

Algorithmic Interpretations of Fractal Dimension*†

Anastasios Sidiropoulos[1] and Vijay Sridhar[2]

1 **Dept. of Mathematics and Dept. of Computer Science & Engineering, The Ohio State University, Columbus, OH, USA**
sidiropoulos.1@osu.edu

2 **Dept. of Computer Science & Engineering, The Ohio State University, Columbus, OH, USA**
sridhar.38@buckeyemail.osu.edu

Abstract

We study algorithmic problems on subsets of Euclidean space of low *fractal dimension*. These spaces are the subject of intensive study in various branches of mathematics, including geometry, topology, and measure theory. There are several well-studied notions of fractal dimension for sets and measures in Euclidean space. We consider a definition of fractal dimension for finite metric spaces which agrees with standard notions used to empirically estimate the fractal dimension of various sets. We define the fractal dimension of some metric space to be the infimum $\delta > 0$, such that for any $\varepsilon > 0$, for any ball B of radius $r \geq 2\varepsilon$, and for any ε-net N (that is, for any maximal ε-packing), we have $|B \cap N| = O((r/\varepsilon)^{\delta})$.

Using this definition we obtain faster algorithms for a plethora of classical problems on sets of low fractal dimension in Euclidean space. Our results apply to exact and fixed-parameter algorithms, approximation schemes, and spanner constructions. Interestingly, the dependence of the performance of these algorithms on the fractal dimension nearly matches the currently best-known dependence on the standard Euclidean dimension. Thus, when the fractal dimension is strictly smaller than the ambient dimension, our results yield improved solutions in all of these settings.

We remark that our definition of fractal definition is equivalent up to constant factors to the well-studied notion of doubling dimension. However, in the problems that we consider, the dimension appears in the exponent of the running time, and doubling dimension is not precise enough for capturing the best possible such exponent for subsets of Euclidean space. Thus our work is *orthogonal* to previous results on spaces of low doubling dimension; while algorithms on spaces of low doubling dimension seek to extend results from the case of low dimensional Euclidean spaces to more *general* metric spaces, our goal is to obtain faster algorithms for *special* pointsets in Euclidean space.

1998 ACM Subject Classification F.2.2 [Nonnumerical Algorithms and Problems] Geometrical Problems and Computations

Keywords and phrases fractal dimension, exact algorithms, fixed parameter tractability, approximation schemes, spanners

Digital Object Identifier 10.4230/LIPIcs.SoCG.2017.58

* A full version of the paper is available at https://arxiv.org/abs/1703.09324.

† This work was supported by the National Science Foundation (NSF) under grant CCF 1423230 and award CAREER 1453472.

1 Introduction

Sets of non-integral dimension are ubiquitous in nature and can be used to model a plethora of processes and phenomena in science and engineering [26]. Sets and measures in Euclidean space of certain fractal dimension are the subject of study in several branches of mathematics, including geometry, topology, and measure theory.

In many problems in computational geometry, the dimension of the input set often determines the complexity of the best-possible algorithms. In this work we study the computational complexity of geometric problems on sets of bounded fractal dimension in low-dimensional Euclidean space. We observe the following interesting phenomenon: For many problems, it is possible to obtain algorithms with dependence on the fractal dimension similar to the best-possible dependence to the standard Euclidean dimension. This implies asymptotically faster algorithms when the fractal dimension of the input is smaller than the ambient dimension.

1.1 Definition of fractal dimension

Intuitively, some $X \subseteq \mathbb{R}^d$ has fractal dimension $\delta \in [0, d]$ if when scaling X by a factor of $\alpha > 0$, the "volume" of X is multiplied by a factor α^δ. There are many different ways this intuition can be formalized, such as Hausdorff dimension, Minkowski dimension, and so on. Unfortunately, some of these definitions are not directly applicable in the context of discrete computational problems. For example, the Hausdorff dimension of any countable set is 0.

Despite this, there are some natural methods that are used to estimate the fractal dimension of a set in practice. Let $X \subseteq \mathbb{R}^d$. Let Γ_ε be a d-dimensional grid where each cell has width $\varepsilon > 0$, and let $I_\varepsilon(X)$ be the number of cells in Γ_ε that intersect X. The *fractal box-counting dimension* of X is defined to be $\lim_{\varepsilon \to 0} \log(I_\varepsilon(X))/\log(1/\varepsilon)$ [8]. This definition is often used experimentally as follows: Intersect X with a regular lattice $(\varepsilon\mathbb{Z})^d$, and estimate the rate by which the cardinality of the intersection grows when $\varepsilon \to 0$. In that context, X has fractal dimension δ when the size of the intersection grows as $(1/\varepsilon)^\delta$ [18].

We consider a definition that is closely related to box-counting dimension, but is more easily amenable to algorithmic analysis. Let $S \subseteq A$. We say that S is an *ε-covering* of A if for any $x \in A$ we have that $\mathsf{dist}(\{x\}, S) \leq \varepsilon$. For any $x \in A$ and $y \in S$ we say that x is covered by y if $\rho(x, y) \leq \varepsilon$. S is an *ε-packing* if for any $x, y \in S$ we have $\rho(x, y) \geq \varepsilon$. If S is both an ε-covering and an ε-packing of A then we say that S is an ε-net of A. We define the *fractal dimension* of some family of pointsets $P \subseteq \mathbb{R}^d$, denoted by $\dim_{\mathsf{f}}(P)$, to be the infimum δ, such that for any $\varepsilon > 0$ and $r \geq 2\varepsilon$, for any ε-net[1] N of P, and for any $x \in \mathbb{R}^d$, we have $|N \cap \mathsf{ball}(x, r)| = O((r/\varepsilon)^\delta)$. For the sake of notational simplicty, we will be referring to the fractal dimension of some familty of pointsets P, as the fractal dimension of the pointset P, with the understanding that in the asymptotic notation $|P|$ is unbounded.

Figure 1 depicts an example of an infinite family of discrete pointsets P with non-integral fractal dimension constructed as follows: We begin with the $3^k \times 3^k$ integer grid, for some $k \in \mathbb{N}$, we partition it into 9 subgrids of equal size, we delete all the points in the central subgrid, and we recurse on the remaining 8 subgrids. The recursion stops when we arrive at a subgrid containing a single point. This is a natural discrete variant of the Sierpiński carpet. It can be shown that $\dim_{\mathsf{f}}(P) = \log_3 8$, which is equal to the Hausdorff dimension of the standard Sierpiński carpet.

[1] We arrive at an equivalent definition if we require N to be a ε-packing instead of a ε-net.

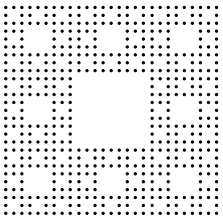


Figure 1 A discrete variant of the Sierpiński carpet for $k = 3$.

1.2 Why yet another notion of dimension?

We now briefly compare the above notion of fractal dimension to previous definitions and motivate its importance. The most closely related notion that has been previously studied in the context of algorithm design is *doubling dimension* [2, 15, 10]. We recall that the doubling dimension of some metric space M, denoted by $\dim_{\mathsf{d}}(M)$, is defined to be $\log \kappa$, where κ is the minimum integer such that for all $r > 0$, any ball in M of radius r can be covered by at most κ balls of radius $r/2$. It is easy to show that for any metric space M, we have[2] $\dim_{\mathsf{d}}(M) = \dim_{\mathsf{f}}(M) + O(1)$ and $\dim_{\mathsf{f}}(M) = O(\dim_{\mathsf{d}}(M))$. Thus our definition is equivalent to doubling dimension up to constant factors. However, in the problems we consider, the dimension appears in the *exponent* of the running time of the best-known algorithms; therefore, determining the best-possible constant is of importance. As we shall see, for several algorithmic problems, our definition yields nearly optimal bounds on this exponent, while doubling dimension is not precise enough for this task.

Let us illustrate this phenomenon on the problem of solving TSP exactly of a set of n points in the Euclidean plane. It is known that TSP admits an algorithm with running time $2^{O(\sqrt{n}\log n)}n^{O(1)}$ in this case [25]. Moreover, the exponent of $O(\sqrt{n}\log n)$ is known to be nearly optimal assuming the Exponential Time Hypothesis (ETH) [23] (see later in this Section for a more precise statement). We show that for sets of fractal dimension $\delta \in (1, 2]$, there exists an algorithm with running time $2^{O(n^{1-1/\delta}\log n)}$. Thus, for any fixed $\delta < 2$, we achieve an asymptotically faster algorithm than what is possible for general pointsets (assuming ETH). On the other hand, it is known that the unit disk cannot be covered with 6 disks of radius $1/2$ (see [29]). Thus $\dim_{\mathsf{d}}(\mathbb{R}^2) \geq \log_2 7 > 2.807$, while $\dim_{\mathsf{f}}(\mathbb{R}^2) = 2$. Therefore doubling dimension is not precise enough to capture the best-possible exponent in this setting.

In summary, while algorithms on spaces of low doubling dimension seek to extend results from the case of low dimensional Euclidean space to a more *general* setting, our goal is to obtain faster algorithms for *special* classes of pointsets in Euclidean space.

1.3 Our results

We obtain algorithms for various problems on sets of low fractal dimension in Euclidean space. We consider exact algorithms, fixed parameter algorithms, and approximation schemes. In each one of these settings, we pick classical representative problems. We believe that our techniques should be directly applicable to many other problems.

[2] Note that for a set X containing two distinct points we have $\dim_{\mathsf{f}}(X) = 0$ while $\dim_{\mathsf{d}}(X) = 1$ and thus it is not always the case that $\dim_{\mathsf{d}}(X) = O(\dim_{\mathsf{f}}(X))$.

Exact algorithms. We first consider exact algorithms in $\mathbb{R}^d$. It is known that for any fixed d, TSP on a set of n points in $\mathbb{R}^d$ can be solved in time $2^{O(n^{1-1/d}\log n)}$ [25]. By adapting ideas from the Euclidean setting, we show that TSP on a set of n points of fractal dimension $\delta > 1$ in constant-dimensional Euclidean space, can be solved in time $2^{O(n^{1-1/\delta}\log n)}$. When $\delta = 1$ and $\delta < 1$, our algorithm has running time $n^{O(\log^2 n)}$ and $n^{O(\log n)}$ respectively. We remark that it has been shown by Marx and Sidiropoulos [23] that assuming ETH, there is no algorithm for TSP in $\mathbb{R}^d$ with running time $2^{O(n^{1-1/d-\varepsilon})}$, for any $\varepsilon > 0$. Thus, our result bypasses this lower bound for sets of low fractal dimension. In particular, our result implies that, in a certain sense, the hardest instances for TSP in $\mathbb{R}^d$ must be close to *full-dimensional*; that is, they must have fractal dimension close to d. Our technique also extends to the Minimum Rectilinear Steiner Tree problem in $\mathbb{R}^2$. Due to lack of space, this extension is omitted.

Parameterized problems. We also consider algorithms for problems parameterized by the value of the optimum solution. A prototypical geometric problem in this setting is Independent Set of unit balls in $\mathbb{R}^d$. Formally, we show that given a set D of unit balls in $\mathbb{R}^d$, the k-Independent Set problem on D can be solved in time $n^{O(k^{1-1/\delta})}$, for any fixed d, where $\delta > 1$ is the fractal dimension of the set of centers of the disks in D. When $\delta \leq 1$, we get an algorithm with running time $n^{O(\log k)}$. Previously known algorithms for this problem in d-dimensional Euclidean space have running time $n^{O(k^{1-1/d})}$, for any $d \geq 2$ [1, 23]. Moreover, it has been shown that there is no algorithm with running time $f(k)n^{o(k^{1-1/d})}$, for any computable function f, assuming ETH [23] (see also [22]). Thus, our result implies that this lower bound can also be bypassed for sets of fractal dimension $\delta < d$.

Approximation schemes. We next consider approximation schemes. Let P be a set of n points of fractal dimension $\delta > 0$, in d-dimensional Euclidean space. We show that for any $R > 0$, for any $\ell > 0$, we can compute a $(1 + d/\ell)$-approximate R-cover of P in time $\ell^{d+\delta} n^{O((\ell\sqrt{d})^{\delta})}$. This matches the performance of the algorithm of Hochbaum and Maass [16] after replacing δ by d. We also obtain a similar algorithm for the R-packing problem.

Spanners and pathwidth. Recall that for any pointset in $\mathbb{R}^d$, and for any $c \geq 1$, a *c-spanner* for P is a graph G with $V(G) = P$, such that for all $x, y \in P$, we have $\|x - y\|_2 \leq d_G(x, y) \leq c \cdot \|x - y\|_2$, where d_G denotes the shortest path distance in G. The parameter c is called the *dilation* of G. It is known that for any $\varepsilon > 0$, any set of n points in $\mathbb{R}^d$ admits a $(1+\varepsilon)$-spanner of size $n(1/\varepsilon)^{O(d)}$ [24, 28]. We strengthen this result in the following way. We show that for any $\varepsilon > 0$, any set of n points of fractal dimension δ in constant-dimensional Euclidean space admits a $(1 + \varepsilon)$-spanner of size $n(1/\varepsilon)^{O(d)}$, and of pathwidth at most $O(n^{1-1/\delta}\log n)$ if $\delta > 1$, at most $O(\log^2 n)$ if $\delta = 1$, and at most $O(\log n)$ if $\delta < 1$. Our spanner is obtained via a modification of the construction due to Vaidya [28]. This provides a general polynomial-time reduction for geometric optimization problems on Euclidean instances of low fractal dimension to corresponding graph instances of low pathwidth. This result can be understood as justification for the fact that instances of low fractal dimension appear to be "easier" than arbitrary instances. We remark that our construction also implies, as a special case, that arbitrary n-pointsets in $\mathbb{R}^d$ admit $(1 + \varepsilon)$-spanners of size $n(1/\varepsilon)^{O(d)}$ and pathwidth $O(n^{1-1/d}\log n)$; this bound on the pathwidth appears to be new, even for the case $d = 2$.

1.4 Related work

There is a large body of work on various notions of dimensionality in computational geometry. Most notably, there has been a lot of effort on determining the effect of doubling dimension

on the complexity of many problems [14, 3, 7, 19, 9, 21, 4, 6, 11, 27]. Other notions that have been considered include low-dimensional negatively curved spaces [20], growth-restricted metrics [17], as well as generalizations of doubling dimension to metrics of bounded global growth [5].

A common goal in all of the above lines of research is to extend tools and ideas from the Euclidean setting to more general geometries. In contrast, as explained above, we study restricted classes of Euclidean instances, with the goal of obtaining faster algorithms than what is possible for arbitrary Euclidean pointsets.

1.5 Notation and definitions

Let (X, ρ) be some metric space. For any $x \in X$ and $r \geq 0$, we define $\mathsf{ball}(x, r) = \{y \in X : \rho(x, y) \leq r\}$ and $\mathsf{sphere}(x, r) = \{y \in X : \rho(x, y) = r\}$. For some $A, B \subseteq X$, we write $\mathsf{dist}(A, B) = \inf_{x \in A, y \in B}\{\rho(x, y)\}$. For some $r \geq 0$, we write $N(A, r) = \{x \in X : \mathsf{dist}(A, \{x\}) \leq r\}$. Let $S \subseteq A$. We say that S is an *ε-covering* of A if for any $x \in A$ we have that $\mathsf{dist}(\{x\}, S) \leq \varepsilon$. For any $x \in A$ and $y \in S$ we say that x is covered by y if $\rho(x, y) \leq \varepsilon$. S is an *ε-packing* if for any $x, y \in S$ we have $\rho(x, y) \geq \varepsilon$. If S is both an ε-covering and an ε-packing of A then we say that S is an ε-net of A.

We recall the following definition from [25]. Let D be a collection of subsets of $\mathbb{R}^d$. D is said to be *κ-thick* if no point is covered by more than κ elements of D. Let D' be any subset of D such that the ratio between the diameters of any pair of elements in D' is at most λ. Then D' is said to be *λ-related*. D is said to be *(λ, κ)-thick* if no point is covered by more than κ elements of any λ-related subset of D.

The pathwidth of some graph G, denoted by $\mathsf{pw}(G)$, is the minimum integer $k \geq 1$, such that there exists a sequence $C_1, \ldots, C_\ell$ of subsets of $V(G)$ of cardinality at most $k + 1$, such that for all $\{u, v\} \in E(G)$, there exists $i \in \{1, \ldots, \ell\}$ with $\{u, v\} \subseteq C_i$, and for all $w \in V(G)$, for all $i_1 < i_2 < i_3 \in \{1, \ldots, \ell\}$, if $w \in C_{i_1} \cap C_{i_3}$ then $w \in C_{i_2}$.

1.6 Organization

The rest of the paper is organized as follows. In Section 2 we derive a separator Theorem for a set of balls whose set of centers has bounded fractal dimension. In Section 3 we present our exact algorithm for TSP. In Section 4 we give a fixed-parameter algorithm for Independent Set of unit balls. In Section 5 we give approximation schemes for packing and covering unit balls. Finally, in Section 6 we present our spanner construction. Due to lack of space, some of the proofs have been deferred to the Appendix.

2 A separator theorem

In this section we prove a separator theorem for a set of d-balls intersecting a set of points with bounded fractal dimension. Subsequently, this result will form the basis for some of our algorithms. The proof uses an argument due to Har-Peled [13].

▶ **Theorem 1.** *Let $d \geq 2$ be some integer, and let $\delta \in (0, d]$ be some real number. Let $P \subset \mathbb{R}^d$ such that $\dim_{\mathsf{f}}(P) = \delta$. Let B be a (λ, κ)-thick set of d-balls in $\mathbb{R}^d$, with $|B| = n$, $\lambda \geq 2$ and such that for all $b \in B$ we have $b \cap P \neq \emptyset$. Then there exists a $(d-1)$-sphere C such that at most $(1 - 2^{-O(d)})n$ of the elements in B are entirely contained in the interior of C, at most*

$(1-2^{-O(d)})n$ *of the elements in* B *are entirely outside* C*, and*

$$|A| = \begin{cases} O\left(\kappa(5\lambda)^d 6^\delta \frac{\lambda}{1-\lambda^{(1-\delta)}} n^{1-1/\delta}\right) & \text{if } \delta > 1 \\ O\left(\kappa(5\lambda)^d 6^\delta \log n\right) & \text{if } \delta = 1 \\ O\left(\frac{\kappa(5\lambda)^d 6^\delta}{\lambda^{1-\delta}-1}\right) & \text{if } \delta < 1 \end{cases},$$

where $A = \{b \in B : \mathsf{diam}(b) \leq \mathsf{diam}(C)$ *and* $b \cap C \neq \emptyset\}$.

Proof. It is known that any ball in $\mathbb{R}^d$ of radius r can be covered by at most $k(d) = 2^{O(d)}$ balls of radius $\frac{r}{2}$. Let C' be the d-ball of minimum radius that contains at least $\frac{1}{k(d)+1}n$ of the elements in B, breaking ties by choosing the ball that contains the maximum number of elements in B. Let $\mathbf{o}$ denote the origin in $\mathbb{R}^d$. Without loss of generality we can scale and translate the elements of B and P until the radius of C' is 1 and it is centered at $\mathbf{o}$. Now, let B^* denote the set of d-balls in B of diameter less than or equal to 4 after scaling. We pick uniformly at random $r \in [1,2]$ and let $C = \mathsf{sphere}(\mathbf{o}, r)$. Now we are ready to obtain an upper bound on the number of elements of B^* that intersect $\mathsf{sphere}(\mathbf{o}, r)$ in expectation.

Consider any d-ball $b \in B^*$ of diameter x. The probability that $\mathsf{sphere}(\mathbf{o}, r)$ intersects b is at most x. Now let $M_1 = \{b \in B^* : \mathsf{diam}(b) \leq n^{\frac{-1}{\delta}}$ and $b \cap \mathsf{sphere}(\mathbf{o}, r) \neq \emptyset\}$ and $M_2 = \{b \in B^* : n^{\frac{-1}{\delta}} < \mathsf{diam}(b) \leq 4$ and $b \cap \mathsf{sphere}(\mathbf{o}, r) \neq \emptyset\}$. $|M_1|$ in expectation is at most $O(n^{1-\frac{1}{\delta}})$ as $|B^*| \leq n$. It remains to bound the expected value of $|M_2|$.

Let $B_i = \{b \in B^* : \lambda^i n^{\frac{-1}{\delta}} < \mathsf{diam}(b) \leq \min\{\lambda^{i+1} n^{\frac{-1}{\delta}}, 4\}$ and $b \cap \mathsf{sphere}(\mathbf{o}, r) \neq \emptyset\}$. Let n_i denote $|B_i|$. We will construct a $\lambda^i n^{\frac{-1}{\delta}}$-net of P as follows. Let $B_i' = B_i$. Let π be some arbitrary ordering of the elements of B_i'. In the sequence determined by π pick the next d-ball b from B_i'. Remove all d-balls from B_i' that are entirely within a ball of diameter $5 \cdot \lambda^{i+1} n^{\frac{-1}{\delta}}$ centered at the center of b. Repeat this procedure for the next element determined by π until all the remaining d-balls in B_i' have been visited. From the fact that B is (λ, κ)-thick we have that there can be at most $\kappa 5^d \lambda^d$ elements in B_i' that are contained within a ball of diameter $5 \cdot \lambda^{i+1} n^{\frac{-1}{\delta}}$. This implies that we retain at least a constant fraction of the elements of B_i in B_i'. Now from each $b \in B_i'$ pick a point p_b that also belongs to P and take the union of all such points to get a set of points N_i. From the choice of d-balls in the above argument $|N_i| \geq \frac{1}{\kappa 5^d \lambda^d} n_i$ and N_i is $\lambda^i n^{\frac{-1}{\delta}}$-packing. We can add more points from P to N_i to obtain a $\lambda^i n^{\frac{-1}{\delta}}$-net N_i'. We have that $|N_i| \leq |N_i' \cap \mathsf{ball}(\mathbf{o}, 6)| \leq O((\frac{6}{\lambda^i n^{\frac{-1}{\delta}}})^\delta)$ since $\dim_{\mathsf{f}}(P) = \delta$ and the points of N_i are contained within the ball of radius 6 centered at the origin. This implies that $|B_i| \leq O(\kappa(5\lambda)^d 6^\delta \lambda^{-i\delta} n)$. Since the d-balls in B_i are intersected by $\mathsf{sphere}(\mathbf{o}, r)$ with probability at most $\lambda^{i+1} n^{\frac{-1}{\delta}}$ we have that the expected number of elements of B_i that are intersected by $\mathsf{sphere}(\mathbf{o}, r)$ is $O(\kappa(5\lambda)^d 6^\delta \lambda^{i+1-i\delta} n^{1-\frac{1}{\delta}})$. We thus get $\mathbb{E}[|M_2|] \leq \sum_{i=0}^{\frac{\log n}{\delta}+2} |B_i| \lambda^{i+1} n^{\frac{-1}{\delta}} \leq \sum_{i=0}^{\frac{\log n}{\delta}+2} O(\kappa(5\lambda)^d 6^\delta \lambda^{i+1-i\delta} n^{1-\frac{1}{\delta}})$. When $\delta > 1$ this implies $\mathbb{E}[|M_2|] \leq O(\kappa(5\lambda)^d 6^\delta (\frac{\lambda}{1-\lambda^{(1-\delta)}}) n^{1-\frac{1}{\delta}})$. When $\delta = 1$ we have $\mathbb{E}[|M_2|] \leq O(\kappa(5\lambda)^d 6^\delta \lambda (\frac{\log n}{\delta} + 3) n^{1-\frac{1}{\delta}}) \leq O(\kappa(5\lambda)^d 6^\delta \log n)$. When $\delta < 1$ we have $\mathbb{E}[|M_2|] \leq O(\kappa(5\lambda)^d 6^\delta \lambda (\frac{\lambda^{(\frac{\log n}{\delta}+3)(1-\delta)}-1}{\lambda^{(1-\delta)}-1}) n^{1-\frac{1}{\delta}}) \leq O(\frac{\kappa(5\lambda)^d 6^\delta}{\lambda^{1-\delta}-1})$.

For any $r \in [1,2]$ we have that $A \subseteq B^*$. Thus $\mathbb{E}[|A|] = \mathbb{E}[|M_1|] + \mathbb{E}[|M_2|] \leq O(n^{1-\frac{1}{\delta}}) + \mathbb{E}[|M_2|]$, which implies that

$$\mathbb{E}[|A|] = \begin{cases} O(\kappa(5\lambda)^d 6^\delta (\frac{\lambda}{1-\lambda^{(1-\delta)}}) n^{1-\frac{1}{\delta}}) & \text{if } \delta > 1 \\ O(\kappa(5\lambda)^d 6^\delta \log n) & \text{if } \delta = 1 \\ O(\frac{\kappa(5\lambda)^d 6^\delta}{\lambda^{1-\delta}-1}) & \text{if } \delta < 1 \end{cases}$$

Finally we need to ensure that C separates a constant fraction of the elements of B. The choice of C' ensures that at least $\frac{1}{k(d)+1}n = \frac{1}{2^{O(d)}}n$ of the elements in B are entirely contained in the interior of C. This implies that at most $(1-2^{-O(d)})n$ of the elements of B are in the exterior of C. Since the $(d-1)$-ball of radius 2 is covered by the union of at most $k(d)$ $(d-1)$-balls of unit radius we have that there are at most $\frac{k(d)}{k(d)+1}n = (1-2^{-O(d)})n$ of the elements in B contained in the interior of C. We note that the upper bound on $\mathbb{E}[|A|]$ remains unaltered for any choice of C'. We further remark that using a more complicated argument similar to the one used by Smith and Wormald [25] a cube separator can be found that separates a constant fraction of d-balls where the constant is independent of d. ◀

3 Exact algorithms

In this Section we give an exact algorithm for TSP. We first use Theorem 1 with the following Lemmas due to Smith and Wormald [25] to obtain a separator for any optimal TSP solution.

▶ **Lemma 2** (Smith and Wormald [25]). *Let $d \geq 2$ be some integer, and let $P \subset \mathbb{R}^d$. Let W be the edge set of an optimal traveling salesman tour of the points of P. Let B be the set of circumballs of the edges of W. Then B is $(2,\kappa)$-thick where $\kappa = 2^{O(d)}$.*

▶ **Lemma 3** (Smith and Wormald [25]). *Let $d \geq 2$ be some integer, and let $P \subset \mathbb{R}^d$. Let W be the edge set of an optimal traveling salesman tour of the points of P. For any $x \in \mathbb{R}^d$ let $W_x = \{w \in W : \mathsf{diam}(w) \geq 1 \text{ and } w \cap \mathsf{ball}(x,1) \neq \emptyset\}$. Then $|W_x| \leq 2^{O(d)}$ for all $x \in \mathbb{R}^d$.*

▶ **Theorem 4.** *Let $d \geq 2$ be some integer, and let $\delta \in (0,d]$ be some real number. Let P be a set of n points in $\mathbb{R}^d$ with $\dim_{\mathsf{f}}(P) = \delta$. Let W be the set of edges of any optimal Euclidean TSP tour of P. Then there exists a $(d-1)$-sphere C such that at most $(1-2^{-O(d)})n$ points in P are contained in the interior of C, at most $(1-2^{-O(d)})n$ points in P are contained outside C, and*

$$|W_C| = \begin{cases} O(n^{1-1/\delta}) & \text{if } \delta > 1 \\ O(\log n) & \text{if } \delta = 1 \\ O\left(\frac{1}{2^{1-\delta}-1}\right) & \text{if } \delta < 1 \end{cases},$$

where $W_C = \{w \in W : w \cap C \neq \emptyset\}$.

Proof. Let B be the set of circumballs of the edges in W. From Lemma 2 we have that B is $(2, 2^{O(d)})$-thick. Every ball in B contains an edge in W and therefore also two points in P. Therefore we can use Theorem 1 on B to find a separator C. It remains to bound the number of edges in W that are intersected by C. Let $W_1 = \{w \in W : \mathsf{diam}(w) \leq \mathsf{diam}(C) \text{ and } w \cap C \neq \emptyset\}$ and $W_2 = \{w \in W : \mathsf{diam}(w) > \mathsf{diam}(C) \text{ and } w \cap C \neq \emptyset\}$. Therefore $W_C = W_1 \cup W_2$. Let B_1 denote the circumballs of the edges in W_1 and B_2 denote the circumballs of the edges in W_2. If an edge in W_1 is intersected by C then the corresponding circumball in B_1 is also intersected by C. From Theorem 1 we have that

$$|W_1| = \begin{cases} O(n^{1-1/\delta}) & \text{if } \delta > 1 \\ O(\log n) & \text{if } \delta = 1 \\ O\left(\frac{1}{2^{1-\delta}-1}\right) & \text{if } \delta < 1 \end{cases}$$

W.l.o.g. we can assume that C has unit radius and is centered at the origin by scaling and translation. Therefore any edge in W_2 also intersects the unit ball centered at the origin. Combining this with Lemma 3 we have that $|W_2| \leq O(1)$. Since $|W_C| \leq |W_1| + |W_2|$, this concludes the proof. ◀

We now use Theorem 4 to obtain an exact algorithm for TSP. We note that the O-notation hides a factor of $n^{O(1)^d}$.

▶ **Theorem 5.** *Let $d \geq 2$ be some fixed integer, and let $\delta \in (0, d]$ be some real number. Let P be a set of n points in $\mathbb{R}^d$ with $\dim_{\mathsf{f}}(P) = \delta$. Then for any fixed d an optimal Euclidean TSP tour for P can be found in time $T(n)$, where*

$$T(n) = \begin{cases} n^{O(n^{1-1/\delta})} & \text{if } \delta > 1 \\ n^{O(\log^2 n)} & \text{if } \delta = 1 \\ n^{O(\log n)} & \text{if } \delta < 1 \end{cases}$$

Proof. First we observe that the $(d-1)$-sphere separator C described in Theorem 4 can be assumed to intersect at least $d+1$ points in P. This is because we can always decrease the radius of C without changing W_C until at least one point in P lies on it. We exhaustively consider all separating $(d-1)$-spheres to find the separator from Theorem 4. Since every relevant $(d-1)$-sphere is uniquely defined by at most $d+1$ points of P intersecting it, there are at most $n^{O(d)}$ spheres to consider. Let $f(n, \delta)$ denote the number of edges intersected by the separator C. From Theorem 4 we have that

$$f(n, \delta) = \begin{cases} O(n^{1-1/\delta}) & \text{if } \delta > 1 \\ O(\log n) & \text{if } \delta = 1 \\ O(1) & \text{if } \delta < 1 \end{cases}$$

We guess a set E' of at most $f(n, \delta)$ edges in the optimal tour that intersect C. For each such guess E', we also guess the permutation of E' defined by the order in which the optimal tour traverses the edges in E'. For each such permutation we solve the two sub-problems in the exterior and interior of the separator respecting the boundary conditions. The resulting running time is $T(n) \leq n^{O(d)} n^{O(f(n,\delta))} 2T\left((1 - 2^{-O(d)})n\right)$ which implies that for any fixed d implies the assertion. ◀

4 Parameterized problems

In this section we present an algorithm for the parameterized version of the Independent Set problem on a set of unit d-balls in $\mathbb{R}^d$, where set of centers of the d-balls has bounded fractal dimension. We first prove a separator theorem which will be used in the algorithm.

▶ **Theorem 6.** *Let $d \geq 2$ be an integer. Let $\delta \in (0, d]$ be a real number. Let P be a set of n points in $\mathbb{R}^d$ with $\dim_{\mathsf{f}}(P) = \delta$. Let $D = \{\mathsf{ball}(x, 1) : x \in P\}$. Let $D' \subseteq D$ be a set of disjoint elements of D such that $|D'| = k$. Then there exists $c \in \mathbb{R}^d$ and $r > 0$ such that at most H d-balls in D' intersect $\mathsf{sphere}(c, r)$ and at most $(1 - 2^{-O(d)})k$ d-balls in D' are contained on either side (interior and exterior) of $\mathsf{sphere}(c, r)$ where*

$$H = \begin{cases} O(k^{1-\frac{1}{\delta}}) & \text{if } \delta > 1 \\ O(1) & \text{if } \delta \leq 1 \end{cases}$$

Proof. Let P' denote the set of centers of the d-balls in D'. We have $|P'| = |D'| = k$. Also since the d-balls in D' are disjoint we have that P' is a 2-packing of P. Consider any $c \in \mathbb{R}^d$ and any $r \geq 1$. Consider a random $(d-1)$-sphere $\mathsf{sphere}(c, r')$ with radius $r' \in [r, 2r]$ chosen uniformly at random. Now we can bound the number of d-balls in D' that intersect the $\mathsf{sphere}(c, r')$. First we note that the center of any d-ball in D' that potentially intersects $\mathsf{sphere}(c, r')$ lies within $\mathsf{ball}(c, 2r+1)$. Therefore the number of d-balls that potentially intersect

$\mathsf{sphere}(c, r')$ is at most $|P' \cap \mathsf{ball}(c, 2r+1)|$. Since P' is a 2-packing that can be augmented into a 2-net by only adding points, we have that $|P' \cap \mathsf{ball}(c, 2r+1)| \leq O((\frac{2r+1}{2})^\delta) = O(r^\delta)$. Therefore we have that the number of d-balls that potentially intersect $\mathsf{sphere}(c, r')$ is at most $\min\{k, O(r^\delta)\}$. Any d-ball in D' intersects $\mathsf{sphere}(c, r')$ with probability at most $\frac{2}{r}$ since r' is chosen uniformly at random from the interval $[r, 2r]$. So in expectation the number of d-balls in D' that intersect $\mathsf{sphere}(c, r')$ is at most $\min\{k \cdot \frac{2}{r}, O(r^\delta) \cdot \frac{2}{r}\}$. When $r \leq k^{\frac{1}{\delta}}$ and $\delta > 1$ this is at most $O(r^\delta) \cdot \frac{2}{r} = O(k^{1-\frac{1}{\delta}})$, when $r \leq k^{\frac{1}{\delta}}$ and $\delta \leq 1$ this is at most $O(r^\delta) \cdot \frac{2}{r} = O(r^{\delta-1}) = O(1)$, and when $r > k^{\frac{1}{\delta}}$ this is again at most $k \cdot \frac{2}{r} = O(k^{1-\frac{1}{\delta}})$. This implies that there exists some specific $r' \in [r, 2r]$ such that the number of d-balls in D' that intersect $\mathsf{sphere}(c, r')$ is at most $O(k^{1-\frac{1}{\delta}})$ when $\delta > 1$ and $O(1)$ when $\delta \leq 1$.

Now it remains to specify our choice of c and r so that $\mathsf{sphere}(c, r')$ induces a balanced separator. We will use the fact that for any $r > 0$ any d-ball of radius $2r$ can be covered by at most $g(d) = 2^{O(d)}$ d-balls of radius r. Let c and r be chosen such that $\mathsf{sphere}(c, r)$ is the $(d-1)$-sphere with minimum radius that also contains in its interior $\frac{1}{g(d)+1}k$ elements of D'. Since the d-balls have unit radius it follows that the $r \geq 1$. This ensures that there are at least $\frac{1}{2^{O(d)}}$ elements of D' in the interior of $\mathsf{sphere}(c, r)$ and therefore at most $(1 - 2^{-O(d)})k$ elements of D' in the exterior of $\mathsf{sphere}(c, r)$. We have that $\mathsf{ball}(c, r')$ is contained within $\mathsf{ball}(c, 2r)$. Since $\mathsf{ball}(c, 2r)$ can be covered by at most $g(d)$ d-balls of radius r, by our choice of c and r we have that $\mathsf{sphere}(c, r')$ encloses at most $\frac{g(d)}{g(d)+1}k = (1 - 2^{-O(d)})k$ d-balls in D' concluding the proof. ◀

▶ **Theorem 7.** *Let $d \geq 2$ be an integer. Let $\delta \in (0, d]$ be a real number. Let P be a set of n points in $\mathbb{R}^d$ with $\dim_{\mathsf{f}}(P) = \delta$. Let $D = \{\mathsf{ball}(x, 1) : x \in P\}$. Then there exists an algorithm that computes an independent set in D of size k, if one exists, in time $T(n, k)$, where for any fixed d we have*

$$T(n,k) = \begin{cases} n^{O(k^{1-1/\delta})} & \text{if } \delta > 1 \\ n^{O(\log k)} & \text{if } \delta \leq 1 \end{cases}$$

Proof. Let $D' \subseteq D$ denote the set of k disjoint d-balls in any fixed optimal solution. Let P' denote the set of centers of the d-balls in D'. We have $|P'| = |D'| = k$. We use a divide and conquer approach using the separator from Theorem 6. First we guess the center c and radius r of the smallest $(d-1)$-sphere enclosing $\frac{1}{g(d)+1}$ of the d-balls in D'. W.l.o.g. we can assume that there exist a set of d-balls in D' that are tangential to $\mathsf{sphere}(c, r)$ and are enclosed by $\mathsf{sphere}(c, r)$, of cardinality $d+1$. Moreover $\mathsf{sphere}(c, r)$ is uniquely defined by the d-balls that it is tangential to. This implies that $\mathsf{sphere}(c, r-1)$ intersects at least $d+1$ points in P and can be uniquely defined by at most $d+1$ points in P. We can exhaustively guess c by searching through all $(d-1)$-spheres uniquely defined by at most $d+1$ points in P in time $n^{O(d)}$. Next we can assume w.l.o.g. that $\mathsf{sphere}(c, r')$ from Theorem 6 is tangential to at least one d-ball in D' (otherwise r' can be increased or decreased until this condition is met without altering the set of d-balls in D' that are intersected by the separator). This means that given a fixed center c we need to search through at most $2n$ different radii to guess r'. We enumerate over all such separators. For each such separator we again enumerate over all ways to pick the d-balls in D' that are intersected. This can be done in time $n^{O(k^{1-1/\delta})}$ when $\delta > 1$ and $n^{O(1)}$ when $\delta \leq 1$. Therefore we have $T(n,k) = n^{O(d)} \cdot O(n) \cdot n^{O(k^{1-1/\delta})} \cdot 2 \cdot T\left(n, (1 - 2^{-O(d)})k\right)$ when $\delta > 1$ or $T(n,k) = n^{O(d)} \cdot O(n) \cdot n^{O(1)} \cdot 2 \cdot T\left(n, (1 - 2^{-O(d)})k\right)$ when $\delta \leq 1$, which solves to the desired bound. ◀

5 Approximation schemes

In this section we describe polynomial time approximation schemes for covering and packing problems. We use the approach of Hochbaum and Maass [16].

▶ **Theorem 8.** *Let $d \geq 2$ be some integer, and let $\delta \in (0, d]$ be some real number. Let P be a set of n points in $\mathbb{R}^d$ with $\dim_f(P) = \delta$. Then there exists a polynomial time approximation scheme which given a natural number $l > 0$ and any $\varepsilon > 0$, computes a $(1 + \frac{d}{l})$-approximation to the ε-cover of P, in time $l^{d+\delta} n^{O((l\sqrt{d})^\delta)}$.*

Proof. Let A be a d-rectangle that encloses the points in P. Consider a set of hyperplanes perpendicular to an axis of the ambient space that subdivide A into strips of width $2l\varepsilon$, which are left closed and right open. This gives a partition P_0 of A where each strip has width $2l\varepsilon$. Now for any integer i where $0 < i < l$ we shift the hyperplanes that define the partition P_0 by $2i\varepsilon$ to the right to get the partition P_i. Let $S = \{P_0, P_1, \ldots, P_{l-1}\}$. Let OPT be the optimal ε-cover of P. Let D be the set of d-balls of radius ε centered at the points in OPT. Any d-ball in D intersects the hyperplanes from at most one partition in S. Therefore there exists a partition P_i such that at most $\frac{|D|}{l}$ d-balls in D are intersected by the hyperplanes defining P_i. In other words at most $\frac{|D|}{l}$ d-balls in D intersect more than one strip in P_i. Now we consider partitioning A similarly along each axis to get a grid of hypercubes of side length $2l\varepsilon$, which we call *cells*. Using the argument described above it follows that there exits a partition P' such that at most $\frac{d|D|}{l}$ d-balls in D intersect more than one cell in P'.

Now consider a cell C of side length $2l\varepsilon$. Since $\dim_f(P) = \delta$ and C is contained in a ball of radius $\sqrt{d}l\varepsilon$ we have that there exists an ε-cover of the points in C of cardinality at most $O(\frac{\sqrt{d}l\varepsilon}{\varepsilon})^\delta = O(\sqrt{d}l)^\delta$.

We combine the above observations to obtain our algorithm as follows. The algorithm enumerates all l^d partitions of P into cells of side length $2l\varepsilon$. Next it enumerates exhaustively all ε-covers of cardinality at most $O((\sqrt{d}l)^\delta)$ for each cell. Since verifying whether a set of points is a valid cover takes time $O(n(\sqrt{d}l)^\delta) = O(nl^\delta)$ this step overall takes time at most $n^{O((\sqrt{d}l)^\delta)} \cdot l^\delta$. Finally the algorithm takes the union of the ε-covers of all the cells to get an ε-cover of P and returns the best solution over all partitions. Since there exists at least one partition where at most $\frac{d|D|}{l}$ d-balls in D intersect more than one cell in the partition, we have that the size of the solution returned is at most $(1 + \frac{d}{l})|D| = (1 + \frac{d}{l})|\mathsf{OPT}|$. The running time of the algorithm is $l^d \cdot n^{O((\sqrt{d}l)^\delta)} \cdot l^\delta = l^{d+\delta} n^{O((l\sqrt{d})^\delta)}$. ◀

▶ **Theorem 9.** *Let $d \geq 2$ be some integer, and let $\delta \in (0, d]$ be some real number. Let P be a set of n points in $\mathbb{R}^d$ with $\dim_f(P) = \delta$. There exists a polynomial time approximation scheme which given a natural number $l > 0$ and any $\varepsilon > 0$, computes a $(1 + \frac{d}{l-d})$-approximation to the ε-packing of P, in time $l^{d+\delta} n^{O((l\sqrt{d})^\delta)}$.*

Proof. We use the partitioning approach described in Theorem 8. We consider cells of side length $l\varepsilon$. Since any ε-packing can be augmented into an ε-net we have that any ε-packing of the points in a cell has cardinality at most $O((\frac{\sqrt{d}l}{2})^\delta)$. We consider ε-packings for each cell where the points in the packing are all at least distance $\frac{\varepsilon}{2}$ from the boundary of the cell; this ensures that the d-balls of radius $\frac{\varepsilon}{2}$ centered at these points do not intersect multiple cells. Then we take the union of these points over all cells and take the minimum cardinality set over all partitions. The running time is $l^{d+\delta} n^{O((l\sqrt{d})^\delta)}$ by the same reasoning used in Theorem 8. Let OPT be the optimal ε-packing of P. Since at most $\frac{d}{l}|\mathsf{OPT}|$ d-balls in the optimal packing intersect more than one cell we have that the solution returned by the algorithm has cardinality at least $(1 - \frac{d}{l})|\mathsf{OPT}|$, as required. ◀

6 Spanners and pathwidth

We remark that several other constructions of $(1+\varepsilon)$-spanners for finite subsets of d-dimensional Euclidean space are known. However, they do not yield graphs of small pathwidth. Here we use a construction that is a modified version of the spanner due to Vaidya [28]. Let P be a set of n points in $\mathbb{R}^d$. Let us first recall the construction from [28]. Let $\varepsilon > 0$. We will define a graph G with $V(G) = P$, that is a $(1+\varepsilon)$-spanner for P.

Let $I_1, \ldots, I_d \subset \mathbb{R}$ be intervals, all having the same length, and such that each I_i is either closed, open, or half-open. Then we say that $b = I_1 \times \ldots \times I_d$ is a *box*. We define $\mathsf{size}(b)$ to be the length of the interval I_1. For each $i \in \{1, \ldots, d\}$, let $\psi(b)_i$ be the center of I_i, and define the half-spaces $L_i(b) = \{(x_1, \ldots, x_d) \in \mathbb{R}^d : x_i < \psi_i(b)\}$ and $R_i(b) = \{(x_1, \ldots, x_d) \in \mathbb{R}^d : x_i \geq \psi_i(b)\}$ Let $S(b)$ be the set of boxes such that $S(b) = \left\{b' : b' = b \cap \left(\bigcap_{i=1}^d f_i\right), \text{ where for all } i \in \{1, \ldots, d\}, f_i = L_i(b) \text{ or } f_i = R_i(b)\right\}$. We also define $\mathsf{shrunk}(b)$ to be some box satisfying the following conditions:

1. If $|b \cap P| \leq 1$ then $\mathsf{shrunk}(b) = b \cap P$. Note that we allow $\mathsf{shrunk}(b)$ to be empty.
2. If $|b \cap P| \geq 2$ then $\mathsf{shrunk}(b)$ is some minimal box contained in b with $\mathsf{shrunk}(b) \cap P = b \cap P$. Note that if there are multiple choices for $\mathsf{shrunk}(b)$, then we choose one arbitrarily.

For some box b with $|b \cap P| \geq 2$, we define $S'(b)$ to be the set of boxes such that $S'(b) = \{b' : \text{ there exists } b'' \in S(b) \text{ s.t. } b'' \cap P \neq \emptyset \text{ and } b' = \mathsf{shrunk}(b'')\}$. If $|b \cap P| \leq 1$, then we define $S'(b) = \emptyset$.

The *box-tree* of P is defined to be a tree T where every node is some box. We set the root of T to be some minimal box b^* containing P. For each $b \in V(T)$, the set of children of b in T is $S'(b)$. Note that $|b \cap P| = 1$ if and only if b is a leaf of T. For each $b \in V(T) \setminus \{b^*\}$ we denote by $\mathsf{father}(b)$ the father of b in T.

For each $b \in V(T)$ let

$$\mathsf{Near}(b) = \left\{b' \in V(T) \setminus \{b^*\} : \mathsf{size}(b') < \mathsf{size}(b) \leq \mathsf{size}(\mathsf{father}(b')) \text{ and } \mathsf{dist}(b, b') \leq \frac{6\sqrt{d}}{\varepsilon}\mathsf{size}(b)\right\}.$$

It follows by the construction that for each $b \in V(T)$, we have $b \cap P \neq \emptyset$. For each $b \in V(T)$ pick some arbitrary point $\mathsf{rep}(b) \in b \cap P$. We say that $\mathsf{rep}(b)$ is the *representative* of b. We further impose the constraint that for each non-leaf $b \in V(T)$, if b' is the unique child of b with $\mathsf{rep}(b) \in b'$, then $\mathsf{rep}(b') = \mathsf{rep}(b)$. This implies that for every $b \in V(T)$, there exists a branch in T starting at b and terminating at some leaf, such that all the boxes in the branch have the same representative as b. We remark that this additional requirement is not necessary in the original construction of Vaidya [28].

We define $E(G) = E_1 \cup E_2$, where $E_1 = \{\{\mathsf{rep}(b), \mathsf{rep}(b')\} : b \in V(T), b' \in S'(b), \mathsf{rep}(b) \neq \mathsf{rep}(b')\}$ and $E_2 = \{\{\mathsf{rep}(b), \mathsf{rep}(b')\} : b \in V(T), b' \in \mathsf{Near}(\mathsf{father}(b))\}$. This completes the description of the spanner construction due to Vaidya [28]. His result is summarized in the following.

▶ **Theorem 10** (Vaidya [28]). *G is a $(1+\varepsilon)$-spanner for P. Moreover $|E(G)| = O(\varepsilon^{-d}n)$.*

For each $e = \{u, v\} \in E(G)$, let D_e be the circumscribed ball for the segment u-v. Let $\mathcal{D} = \bigcup_{e \in E(G)}\{D_e\}$. For each $i \in \{1, 2\}$ let $\mathcal{D}_i = \bigcup_{e \in E_i}\{D_e\}$.

▶ **Lemma 11.** *$\mathcal{D}_1$ is $(2, d^{O(d)})$-thick.*

Proof. Let $r > 0$ and define $E_{1,r} = \{\{x, y\} \in E_1 : r \leq \|x - y\|_2 < 2r\}$. Let $\mathcal{D}_{1,r} = \{D_e \in \mathcal{D}_1 : e \in E_{1,r}\}$. It suffices to show that $\mathcal{D}_{1,r}$ is $d^{O(d)}$-thick.

For each $e = \{x, y\} \in E_{1,r}$ we define some unordered pair of boxes $\gamma(e) = \{B(e), B'(e)\}$, as follows. By the definition of E_1, there exists some $b \in V(T)$, $b' \in S'(b)$, with $\mathsf{rep}(b) \neq \mathsf{rep}(b')$, such that $\{x, y\} = \{\mathsf{rep}(b), \mathsf{rep}(b')\}$. Assume w.l.o.g. that $x = \mathsf{rep}(b)$ and $y = \mathsf{rep}(b')$. By the choice of the representatives, there exist some branch $b_0, \ldots, b_t$ of T, for some $t \geq 1$ with $b_0 = b$, that terminates at some leaf b_t, such that $x = \mathsf{rep}(b) = \mathsf{rep}(b_0) = \ldots = \mathsf{rep}(b_t)$. Since $x, y \in b$, it follows that $r \leq \|x - y\|_2 \leq \sqrt{d} \cdot \mathsf{size}(b)$. Since b_t is a leaf, we have $\mathsf{size}(b_t) = 0$. Let $t^* \in \{1, \ldots, t\}$ be the maximum integer such that $\mathsf{size}(b_{t^*-1}) \geq r/\sqrt{d}$. Let $A \in S(b_{t^*-1})$ such that $b_{t^*} \subseteq A$. Note that $\mathsf{size}(A) \geq r/(2\sqrt{d})$, and $\mathsf{size}(b_{t^*}) < r/\sqrt{d}$. Pick some box $B(e)$, with $b_{t^*} \subseteq B(e) \subseteq A$, such that

$$\mathsf{size}(B(e)) \in \left[r/(2\sqrt{d}), r/\sqrt{d}\right] \tag{1}$$

in a consistent fashion (i.e. for a fixed choice of b_{t^*} and A we always pick the same box). Similarly, let $b'_0, \ldots, b'_s$ be a sequence of boxes such that $b'_0 \in S'(b)$, with $b' \subseteq b'_0$, and $b'_1, \ldots, b'_s$ is a branch of T starting at $b'_1 = b'$ and terminating at some leaf b'_s. Arguing as before, let $s^* \in \{1, \ldots, s\}$ be the maximum integer such that $\mathsf{size}(b'_{s^*-1}) \geq r/(2\sqrt{d})$. If $s^* = 1$ then let $A' \in S'(b'_{s^*-1})$, with $b'_{s^*} \subseteq A'$; pick some box $B'(e)$, with $b_{s^*} \subseteq B'(e) \subseteq A'$, such that

$$\mathsf{size}(B'(e)) \in \left[r/(4\sqrt{d}), r/(2\sqrt{d})\right] \tag{2}$$

in a consistent fashion.

We say that e is *charged* to $\gamma(e)$. By construction, there exists at most one edge in $E_{1,r}$ that is charged to each pair of boxes.

By (1) and (2) we have that for each $e \in E_{1,r}$, the pair $\gamma(e)$ consists of two boxes, each of size $\Theta(r/\sqrt{d})$. Moreover by construction and our choice of boxes we have that for any $e, f \in E_{1,r}$ $B(e)$ and $B(f)$ are disjoint or equal. Similarly $B'(e)$ and $B'(f)$ are also disjoint or equal. Thus, each point in $\mathbb{R}^d$ can be contained in at most $O(1)$ boxes in all the pairs $\gamma(e)$, for all $e \in E_{1,r}$. Moreover, $\mathsf{dist}(B(e), B'(e)) \leq \|x - y\| < 2r$. Thus, each box participates in at most $(\sqrt{d})^{O(d)} = d^{O(d)}$ pairs. For each $e \in E_{1,r}$, let $A(e) = N(B(e), r) \cup N(B'(e), r)$, where $N(X, r)$ denotes the r-neighborhood of X in $\mathbb{R}^d$. It follows that $\{A(e)\}_{e \in E_{1,r}}$ is $d^{O(d)}$-thick. Since for each $e \in E_{1,r}$, we have $D_e \in A(e)$, it follows that $\mathcal{D}_1$ is $d^{O(d)}$-thick, as required. ◀

▶ **Lemma 12.** *$\mathcal{D}_2$ is $(2, (d/\varepsilon)^{O(d)})$-thick.*

Proof. Let $r > 0$ and define $E_{2,r} = \{\{x, y\} \in E_2 : r \leq \|x - y\|_2 < 2r\}$. Let $\mathcal{D}_{2,r} = \{D_e \in \mathcal{D}_2 : e \in E_{2,r}\}$. It suffices to show that $\mathcal{D}_{2,r}$ is $d^{O(d)}$-thick.

As in the proof of Lemma 11, for each $e \in \mathcal{D}_{1,r}$ we define some unordered pair of boxes $\gamma(e) = \{B(e), B'(e)\}$. By the definition of E_2, there exists some $b \in V(T)$, $b' \in \mathsf{Near}(\mathsf{father}(b))$, such that $\{x, y\} = \{\mathsf{rep}(b), \mathsf{rep}(b')\}$. Assume w.l.o.g. that $x = \mathsf{rep}(b)$ and $y = \mathsf{rep}(b')$. Thus we have $\mathsf{size}(b') < \mathsf{size}(\mathsf{father}(b)) \leq \mathsf{size}(\mathsf{father}(b'))$ and $\mathsf{dist}(b', \mathsf{father}(b)) \leq \frac{6\sqrt{d}}{\varepsilon}\mathsf{size}(\mathsf{father}(b))$. Thus $r \leq \|x - y\|_2 \leq \sqrt{d} \cdot \mathsf{size}(b) + \sqrt{d} \cdot \mathsf{size}(b') + \mathsf{dist}(b, b') < \sqrt{d} \cdot \mathsf{size}(\mathsf{father}(b)) + \sqrt{d} \cdot \mathsf{size}(\mathsf{father}(b)) + \mathsf{dist}(b', \mathsf{father}(b)) + \sqrt{d} \cdot \mathsf{size}(\mathsf{father}(b)) \leq (3 + 6/\varepsilon)\sqrt{d} \cdot \mathsf{size}(\mathsf{father}(b))$. Thus $\mathsf{size}(\mathsf{father}(b)) > r\varepsilon/(9\sqrt{d})$. Let $b_0, \ldots, b_t$ be a branch in T with $b_0 = \mathsf{father}(b)$, and $b_t = \{x\}$. Arguing as in Lemma 11, let $t^* \in \{0, \ldots, t-1\}$ be the maximum integer such that $\mathsf{size}(b_{t^*}) \geq r\varepsilon/(9\sqrt{d})$. Let $A \in S'(b_{t^*})$, with $b_{t^*+1} \subseteq A$, and pick some box $B(e) \subset A$, with

$$\mathsf{size}(B(e)) \in \left[(r\varepsilon)/(18\sqrt{d}), (r\varepsilon)/(9\sqrt{d})\right] \tag{3}$$

Similarly, let $b'_0, \ldots, b'_s$ be a branch of T with $b'_0 = \mathsf{father}(b')$, and b'_s is a leaf with $b'_s = \{y\}$. Arguing as in Lemma 11, let $s^* \in \{0, \ldots, s-1\}$ be the maximum integer such that $\mathsf{size}(b'_{s^*-1}) \geq (r\varepsilon)/(9\sqrt{d})$. Let $A' \in S'(b_{s^*-1})$ with $b_{s^*} \subseteq A$, and pick some box $B'(e)$, with $b_{s^*} \subseteq A \subseteq B'(e)$, such that

$$\mathsf{size}(B'(e)) \in \left[(\varepsilon r)/(18\sqrt{d}), (\varepsilon r)/(9\sqrt{d})\right] \tag{4}$$

We say that e is *charged* to $\gamma(e)$. By construction, there exists at most one edge in $E_{1,r}$ that is charged to each pair of boxes.

By (3) and (4) we have that for each $e \in E_{2,r}$, the pair $\gamma(e)$ consists of two boxes, each of size $\Theta((\varepsilon r)/\sqrt{d})$. Thus, each point in $\mathbb{R}^d$ can be contained in at most $O(1)$ distinct boxes in all the pairs $\gamma(e)$, for all $e \in E_{2,r}$. Moreover, $\mathsf{dist}(B(e), B'(e)) \leq \|x - y\| < 2r$. Thus, each box participates in at most $(\sqrt{d}/\varepsilon)^{O(d)} = (d/\varepsilon)^{O(d)}$ pairs. For each $e \in E_{2,r}$, let $A(e) = N(B(e), r) \cup N(B'(e), r)$. It follows that $\{A(e)\}_{e \in E_{2,r}}$ is $(d/\varepsilon)^{O(d)}$-thick. Since for each $e \in E_{2,r}$, we have $D_e \in A(e)$, it follows that $\mathcal{D}_2$ is $(d/\varepsilon)^{O(d)}$-thick, as required. ◀

▶ **Lemma 13.** *$\mathcal{D}$ is $(2, (d/\varepsilon)^{O(d)})$-thick.*

Proof of Lemma 13. By Lemma 11 we have that $\mathcal{D}_1$ is $(2, d^{O(d)})$-thick, and by Lemma 12 we have that $\mathcal{D}_2$ is $(2, (d/\varepsilon)^{O(d)})$-thick. Since $\mathcal{D} = \mathcal{D}_1 \cup \mathcal{D}_2$, we get that $\mathcal{D}$ is $(2, \kappa)$-thick, where $\kappa = d^{O(d)} + (d/\varepsilon)^{O(d)} = (d/\varepsilon)^{O(d)}$, as required. ◀

Let $x, y, z, w \in \mathbb{R}^d$. We say that zw is a *shortcut* for xy if the following conditions holds:

1. $\|x - z\|_2 \leq \varepsilon \|z - w\|_2/20$.
2. The angle formed by the segments x-y and x-$(w - z + x)$ is at most $\varepsilon/20$.

We now proceed to modify G to obtain a graph G'. Initially, G' contains no edges. We consider all edges in G in increasing order of length. When considering an edge $e = \{x, y\}$, if there exists $\{z, w\} \in E(G')$ such that either zw is a shortcut for xy or zw is a shortcut for yx, then we do not add e to G'; otherwise we add e to G'. This completes the construction of G'. We next argue that G' is a spanner with low dilation for P. The proof of the following is standard (see e.g. [12]).

▶ **Lemma 14.** *G' is a $(1 + 2\varepsilon)$-spanner for P.*

Due to lack of space, the proof of Lemma 14 is deferred to the full version.

▶ **Lemma 15.** *Let $c \in \mathbb{R}^d$ and let $r > 0$. Let $E^* = \{\{x, y\} \in E(G') : \|x - y\| > 2r$ and x-$y \cap \mathsf{sphere}(c, r) \neq \emptyset\}$. Then $|E^*| \leq (d/\varepsilon)^{O(d)}$.*

Proof of Lemma 15. Let $E^*_0 = \{\{x, y\} \in E^* : \|x - y\| \leq 100r/\varepsilon\}$. We can partition E^*_0 into $O(\log(1/\varepsilon))$ buckets, where the i-th bucket contains the balls with radius in $[r2^i, r2^{i+1})$. Since by Lemma 13, $\mathcal{D}$ is $(2, (d/\varepsilon)^{O(d)})$-thick, and all the balls in E^* are contained in a ball of radius $O(r/\varepsilon)$, it follows that each bucket can contain at most $(1/\varepsilon)^{O(d)} \cdot (d/\varepsilon)^{O(d)}$ balls. Thus $|E^*_0| = O(\log(1/\varepsilon)) \cdot (1/\varepsilon)^{O(d)} \cdot (d/\varepsilon)^{O(d)} = (d/\varepsilon)^{O(d)}$.

Let $E^*_1 = E^* \setminus E^*_0$. Suppose that $|E^*_1| > (d/\varepsilon)^{Cd}$. Setting C to be a sufficiently large universal constant it follows that there exist distinct edges $\{x, y\}, \{z, w\} \in E^*_1$ that form an angle of less than $\varepsilon/20$. Assume w.l.o.g. that $\|x - y\|_2 \geq \|z - w\|_2$, $x \in \mathsf{ball}(c, r)$, and $z \in \mathsf{ball}(c, r)$. Then zw must be a shortcut for xy, which is a contradiction since $\{x, y\} \in E(G')$, concluding the proof. ◀

We now prove the main result of this section.

▶ **Theorem 16.** *Let $d \geq 2$ be some fixed integer, and let $\delta \in (0, d]$ be some real number. Let $P \subset \mathbb{R}^d$ be some finite point set with $|P| = n$, such that $\dim_{\mathsf{f}}(P) = \delta$. Then, for any fixed $\varepsilon \in (0, 1]$, there exists a $(1+\varepsilon)$-spanner, G', for P, with a linear number of edges, and with*

$$\mathsf{pw}(G) = \begin{cases} O(n^{1-1/\delta} \log n) & \text{if } \delta > 1 \\ O(\log^2 n) & \text{if } \delta = 1 \\ O(\log n) & \text{if } \delta < 1 \end{cases}$$

Moreover, given P, the graph G' can be computed in polynomial time.

Proof. Let G' be the spanner constructed above. The bound on the number of edges of G' follows by Theorem 10 since $G' \subseteq G$. We will bound the pathwidth of G'. By Lemma 13 we have that $\mathcal{D}$ is $(2, (d/\varepsilon)^{O(d)})$-thick. By Theorem 1 there exists some $(d-1)$-sphere C with radius r such that at most $(1 - 2^{-O(d)})n$ points of P are contained in either side of C, and $|A| \leq M$, where

$$M = \begin{cases} O(n^{1-1/\delta}) & \text{if } \delta > 1 \\ O(\log n) & \text{if } \delta = 1 \\ O(1) & \text{if } \delta < 1 \end{cases},$$

and $A = \{\{x, y\} \in E(G) : \|x - y\| \leq 2r \text{ and } x\text{-}y \cap C \neq \emptyset\}$.

Let $A' = \{\{x, y\} \in E(G') : \|x - y\| \leq 2r \text{ and } x\text{-}y \cap C \neq \emptyset\}$. We have $A' \subseteq A$, and thus $|A'| \leq |A|$. Let $A'' = \{\{x, y\} \in E(G') : \|x - y\| > 2r \text{ and } x\text{-}y \cap C \neq \emptyset\}$. By Lemma 15 we have $|A''| = O(1)$ (for fixed d and ε). Let S be the set of all endpoints of all the edges in $A' \cup A''$. We have $|S| \leq 2|A' \cup A''| = O(|A|)$. Let U (resp. U') be the set of points in P that are inside (resp. outside) C. Then S separates in G' every vertex in U from every vertex in U'. We may thus recurse on $G'[U \setminus (S \cup U')]$ and $G'[U' \setminus (S \cup U)]$ and obtain path decompositions $X_1, \ldots, X_t$ and $Y_1, \ldots, Y_s$ respectively. We now obtain the path decomposition $X_1 \cup S, \ldots, X_t \cup S, Y_1 \cup S, \ldots, Y_s \cup S$ for G'. The width of the resulting path decomposition is at most $O(M \log n)$, concluding the proof. ◀

Acknowledgements. The authors wish to thank Sariel Har-Peled, Dimitrios Thilikos, and Yusu Wang for fruitful discussions.

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 Patrice Assouad. Plongements lipschitziens dans $\mathbb{R}^n$. *Bulletin de la Société Mathématique de France*, 111:429–448, 1983.

3 Yair Bartal, Lee-Ad Gottlieb, and Robert Krauthgamer. The traveling salesman problem: low-dimensionality implies a polynomial time approximation scheme. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 663–672. ACM, 2012.

4 T. H. Hubert Chan and Anupam Gupta. Small hop-diameter sparse spanners for doubling metrics. *Discrete & Computational Geometry*, 41(1):28–44, 2009.

5 T. H. Hubert Chan and Anupam Gupta. Approximating tsp on metrics with bounded global growth. *SIAM Journal on Computing*, 41(3):587–617, 2012.

6 T. H. Hubert Chan, Anupam Gupta, Bruce M. Maggs, and Shuheng Zhou. On hierarchical routing in doubling metrics. In *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 762–771. Society for Industrial and Applied Mathematics, 2005.

7 Richard Cole and Lee-Ad Gottlieb. Searching dynamic point sets in spaces with bounded doubling dimension. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 574–583. ACM, 2006.

8 Kenneth Falconer. *Fractal geometry: mathematical foundations and applications*. John Wiley & Sons, 2004.

9 Lee-Ad Gottlieb and Liam Roditty. An optimal dynamic spanner for doubling metric spaces. In *European Symposium on Algorithms*, pages 478–489. Springer, 2008.

10 Anupam Gupta, Robert Krauthgamer, and James R. Lee. Bounded geometries, fractals, and low-distortion embeddings. In *Foundations of Computer Science, 2003. Proceedings. 44th Annual IEEE Symposium on*, pages 534–543. IEEE, 2003.

11 Anupam Gupta and Kevin Lewi. The online metric matching problem for doubling metrics. In *International Colloquium on Automata, Languages, and Programming*, pages 424–435. Springer, 2012.

12 Sariel Har-Peled. *Geometric approximation algorithms*, volume 173. American Mathematical Society, Providence, 2011.

13 Sariel Har-Peled. A simple proof of the existence of a planar separator. *arXiv preprint arXiv:1105.0103*, 2011.

14 Sariel Har-Peled and Manor Mendel. Fast construction of nets in low-dimensional metrics and their applications. *SIAM Journal on Computing*, 35(5):1148–1184, 2006.

15 Juha Heinonen. *Lectures on analysis on metric spaces*. Springer Science & Business Media, 2012.

16 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.

17 David R. Karger and Matthias Ruhl. Finding nearest neighbors in growth-restricted metrics. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing*, pages 741–750. ACM, 2002.

18 Marc Khoury and Rephael Wenger. On the fractal dimension of isosurfaces. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1198–1205, 2010.

19 Robert Krauthgamer and James R. Lee. The black-box complexity of nearest-neighbor search. *Theoretical Computer Science*, 348(2):262–276, 2005.

20 Robert Krauthgamer and James R. Lee. Algorithms on negatively curved spaces. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 119–132. IEEE, 2006.

21 Robert Krauthgamer, James R. Lee, Manor Mendel, and Assaf Naor. Measured descent: A new embedding method for finite metrics. *Geometric & Functional Analysis GAFA*, 15(4):839–858, 2005.

22 Dániel Marx. Efficient approximation schemes for geometric problems? In *European Symposium on Algorithms*, pages 448–459. Springer, 2005.

23 Dániel Marx and Anastasios Sidiropoulos. The limited blessing of low dimensionality: when 1-1/d is the best possible exponent for d-dimensional geometric problems. In *Proceedings of the thirtieth annual symposium on Computational geometry*, page 67. ACM, 2014.

24 Jeffrey S Salowe. Construction of multidimensional spanner graphs, with applications to minimum spanning trees. In *Proceedings of the seventh annual symposium on Computational geometry*, pages 256–261. ACM, 1991.

25 Warren D. Smith and Nicholas C. Wormald. Geometric separator theorems and applications. In *Foundations of Computer Science, 1998. Proceedings. 39th Annual Symposium on*, pages 232–243. IEEE, 1998.

26 Hideki Takayasu. *Fractals in the physical sciences*. Manchester University Press, 1990.

27 Kunal Talwar. Bypassing the embedding: algorithms for low dimensional metrics. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 281–290. ACM, 2004.

28 Pravin M. Vaidya. A sparse graph almost as good as the complete graph on points ink dimensions. *Discrete & Computational Geometry*, 6(3):369–381, 1991.

29 C. T. Zahn. Black box maximization of circular coverage. *Journal of Research of the National Bureau of Standards B*, 66:181–216, 1962.

Disjointness Graphs of Segments*

János Pach[1], Gábor Tardos[2], and Géza Tóth[3]

1 **Ecole Polytechnique Fédérale de Lausanne, Lausanne, Switzerland; and Rényi Institute, Hungarian Academy of Sciences, Budapest, Hungary**
pach@renyi.hu

2 **Rényi Institute, Hungarian Academy of Sciences, Budapest, Hungary**
tardos@renyi.hu

3 **Rényi Institute, Hungarian Academy of Sciences, Budapest, Hungary**
geza@renyi.hu

Abstract

The *disjointness graph* $G = G(\mathcal{S})$ of a set of segments $\mathcal{S}$ in $\mathbb{R}^d$, $d \geq 2$, is a graph whose vertex set is $\mathcal{S}$ and two vertices are connected by an edge if and only if the corresponding segments are disjoint. We prove that the chromatic number of G satisfies $\chi(G) \leq (\omega(G))^4 + (\omega(G))^3$, where $\omega(G)$ denotes the clique number of G. It follows, that $\mathcal{S}$ has $\Omega(n^{1/5})$ pairwise intersecting or pairwise disjoint elements. Stronger bounds are established for lines in space, instead of segments.

We show that computing $\omega(G)$ and $\chi(G)$ for disjointness graphs of lines in space are NP-hard tasks. However, we can design efficient algorithms to compute proper colorings of G in which the number of colors satisfies the above upper bounds. One cannot expect similar results for sets of continuous arcs, instead of segments, even in the plane. We construct families of arcs whose disjointness graphs are triangle-free ($\omega(G) = 2$), but whose chromatic numbers are arbitrarily large.

1998 ACM Subject Classification F.2.2 Geometrical Problems and Computations

Keywords and phrases disjointness graph, chromatic number, clique number, χ-bounded

Digital Object Identifier 10.4230/LIPIcs.SoCG.2017.59

1 Introduction

Given a set of (geometric) objects, their *intersection graph* is a graph whose vertices correspond to the objects, two vertices being connected by an edge if and only if their intersection is nonempty. Intersection graphs of intervals on a line [19], more generally, chordal graphs [3, 8] and comparability graphs [7], turned out to be *perfect graphs*, that is, for them and for each of their induced subgraph H, we have $\chi(H) = \omega(H)$, where $\chi(H)$ and $\omega(H)$ denote the chromatic number and the clique number of H, respectively. It was shown [18] that the complements of these graphs are also perfect, and based on these results, Berge [3] conjectured and Lovász [29] proved that the complement of every perfect graph is perfect.

Most geometrically defined intersection graphs are not perfect. However, in many cases they still have nice coloring properties. For example, Asplund and Grünbaum [2]

* János Pach was supported by Swiss National Science Foundation Grants 200021-165977 and 200020-162884, Gábor Tardos was supported by the Cryptography "Lendület" project of the Hungarian Academy of Sciences and by the National Research, Development and Innovation Office, NKFIH, projects K-116769 and SNN-117879, Géza Tóth was supported by National Research, Development and Innovation Office, NKFIH, K-111827.

proved that every intersection graph G of axis-parallel rectangles in the plane satisfies $\chi(G) = O((\omega(G))^2)$. It is not known if the stronger bound $\chi(G) = O(\omega(G))$ also holds for these graphs. For intersection graphs of chords of a circle, Gyárfás [13, 14] established the bound $\chi(G) = O((\omega(G))^2 4^{\omega(G)})$, which was improved to $O(2^{\omega(G)})$ in [24]. Here we have examples of $\chi(G)$ slightly superlinear in $\omega(G)$ [25]. In some cases, there is no functional dependence between χ and ω. The first such example was found by Burling [5]: there are sets of axis-parallel boxes in $\mathbb{R}^3$, whose intersection graphs are *triangle-free* ($\omega = 2$), but their chromatic numbers are arbitrarily large. Following Gyárfás and Lehel [16], we call a family $\mathcal{G}$ of graphs *χ-bounded* if there exists a function f such that all elements $G \in \mathcal{G}$ satisfy the inequality $\chi(G) \leq f(\omega(G))$. The function f is called a *bounding function* for $\mathcal{G}$. Heuristically, if a family of graphs is χ-bounded, then its members can be regarded "nearly perfect". Consult [17, 15, 23] for surveys.

At first glance, one might believe that, in analogy to perfect graphs, a family of intersection graphs is χ-bounded if and only if the family of their complements is. Burling's above mentioned constructions show that this is not the case: the family of complements of intersection graphs of axis-parallel boxes in $\mathbb{R}^d$ is χ-bounded with bounding function $f(x) = O(x \log^{d-1} x)$, see [21]. More recently, Pawlik, Kozik, Krawczyk, Lasoń, Micek, Trotter, and Walczak [33] have proved that Burling's triangle-free graphs can be realized as intersection graphs of segments in the plane. Consequently, the family of these graphs is *not* χ-bounded either. On the other hand, the family of their complements is, see Theorem 0.

To simplify the exposition, we call the complement of the intersection graph of a set of objects their *disjointness graph*. That is, in the disjointness graph two vertices are connected by an edge if and only if the corresponding objects are disjoint. Using this terminology, the following is a direct consequence of a result of Larman, Matoušek, Pach, and Törőcsik.

▶ **Theorem 0** ([28]). *The family of disjointness graphs of segments in the plane is χ-bounded. More precisely, every such graph G satisfies the inequality $\chi(G) \leq (\omega(G))^4$.*

For the proof of Theorem 0, one has to introduce four partial orders on the family of segments, and apply Dilworth's theorem [7] four times. Although this method does not seem to generalize to higher dimensions, the statement does. We establish the following.

▶ **Theorem 1.** *The disjointness graph G of any system of segments in $\mathbb{R}^d, d \geq 2$ satisfies the inequality $\chi(G) \leq (\omega(G))^4 + (\omega(G))^3$.*

Moreover, there is a polynomial time algorithm that, given the segments corresponding to the vertices of G, finds a complete subgraph $K \subseteq G$ and a proper coloring of G with at most $|V(K)|^4 + |V(K)|^3$ colors.

If we consider full lines in place of segments, we obtain stronger bounds.

▶ **Theorem 2.**

(i) *Let G be the disjointness graph of a set of lines in $\mathbb{R}^d$, $d \geq 3$. Then we have $\chi(G) \leq (\omega(G))^3$.*

(ii) *Let G be the disjointness graph of a set of lines in the projective space $\mathbb{P}^d$, $d \geq 3$. Then we have $\chi(G) \leq (\omega(G))^2$.*

In both cases, there are polynomial time algorithms that, given the lines corresponding to the vertices of G, find complete subgraphs $K \subseteq G$ and proper colorings of G with at most $|V(K)|^3$ and $|V(K)|^2$ colors, respectively.

Note that the difference between the two scenarios comes from the fact that parallel lines in the Euclidean space are disjoint, but the corresponding lines in the projective space intersect.

Most computational problems for geometric intersection and disjointness graphs are hard. It was shown by Kratochvíl and Nešetřil [26] and by Cabello, Cardinal, and Langerman [6] that finding the clique number $\omega(G)$ resp. the independence number $\alpha(G)$ of disjointness graphs of segments in the plane are NP-hard. It is also known that computing the chromatic number $\chi(G)$ of disjointness and intersection graphs of segments in the plane is NP-hard [9]. Our next theorem shows that some of the analogous problems are also NP-hard for disjointness graphs of lines in space, while others are tractable in this case. In particular, according to Theorem 3 (i), in a disjointness graph G of lines, it is NP-hard to determine $\omega(G)$ and $\chi(G)$. In view of this, it is interesting that one can design polynomial time algorithms to find proper colorings and complete subgraphs in G, where the number of colors is bounded in terms of the size of the complete subgraphs, in the way specified in the closing statements of Theorems 1 and 2.

▶ **Theorem 3.**

(i) *Computing the clique number $\omega(G)$ and the chromatic number $\chi(G)$ of disjointness graphs of lines in $\mathbb{R}^3$ or in $\mathbb{P}^3$ are NP-hard problems.*

(ii) *Computing the independence number $\alpha(G)$ of disjointness graphs of lines in $\mathbb{R}^3$ or in $\mathbb{P}^3$, and deciding for a fixed k whether $\chi(G) \leq k$, can be done in polynomial time.*

The bounding functions in Theorems 0 1, and 2 are not likely to be optimal. As for Theorem 2 (i), we will prove that there are disjointness graphs G of lines in $\mathbb{R}^3$ for which $\frac{\chi(G)}{\omega(G)}$ are arbitrarily large. Our best constructions for disjointness graphs G' of lines in the projective space satisfy $\chi(G') \geq 2\omega(G') - 1$; see Theorem 9.

The proof of Theorem 1 is based on Theorem 0. Any strengthening of Theorem 0 leads to improvements of our results. For example, if $\chi(G) = O((\omega(G))^\gamma)$ holds with any $3 \leq \gamma \leq 4$ for the disjointness graph of every set of segments in the plane, then the proof of Theorem 1 implies the same bound for disjointness graphs of segments in higher dimensions. In fact, it is sufficient to verify this statement in 3 dimensions. For $d \geq 4$, we can find a projection in a generic direction to the 3-dimensional space that does not create additional intersections and then we can apply the 3-dimensional bound. We focus on the case $d = 3$.

It follows immediately from Theorem 0 that the disjointness (and, hence, the intersection) graph of any system of n segments in the plane has a clique or an independent set of size at least $n^{1/5}$. Indeed, denoting by $\alpha(G)$ the maximum number of independent vertices in G, we have

$$\alpha(G) \geq \frac{n}{\chi(G)} \geq \frac{n}{(\omega(G))^4},$$

so that $\alpha(G)(\omega(G))^4 \geq n$. Analogously, Theorem 1 implies that $\max(\alpha(G), \omega(G)) \geq (1 - o(1))n^{1/5}$ holds for disjointness (and intersection) graphs of segments in any dimension $d \geq 2$. For disjointness graphs of n lines in $\mathbb{R}^d$ (respectively, in $\mathbb{P}^d$), we obtain that $\max(\alpha(G), \omega(G))$ is $\Omega(n^{1/4})$ (resp., $\Omega(n^{1/3})$). Using more advanced algebraic techniques, Cardinal, Payne, and Solomon [35] proved the stronger bounds $\Omega(n^{1/3})$ (resp., $\Omega(n^{1/2})$).

If the order of magnitude of the bounding functions in Theorems 0 and 1 are improved, then the improvement carries over to the lower bound on $\max(\alpha(G), \omega(G))$. Despite many efforts [28, 22, 27] to construct intersection graphs of planar segments with small clique and independence numbers, the best known construction, due to Kynčl [27], gives only

$$\max(\alpha(G), \omega(G)) \leq n^{\log 8/\log 169} \approx n^{0.405},$$

where n is the number of vertices. This bound is roughly the square of the best known lower bound.

Our next theorem shows that any improvement on the lower bound on $\max(\alpha(G), \omega(G))$ in the plane, even if it was not achieved by an improvement of the bounding function in Theorem 0, would also carry over to higher dimensions.

▶ **Theorem 4.** *If the disjointness graph of any set of n segments in the plane has a clique or an independent set of size $\Omega(n^\beta)$ for some fixed $\beta \leq 1/4$, then the same is true for disjointness graphs of segments in $\mathbb{R}^d$ for any $d > 2$.*

A continuous arc in the plane is called a *string*. One may wonder whether Theorem 0 can be extended to disjointness graphs of strings in place of segments. The answer is no, in a very strong sense.

▶ **Theorem 5.** *There exist triangle-free disjointness graphs of n strings in the plane with arbitrarily large chromatic numbers. Moreover, we can assume that these strings are simple polygonal paths consisting of at most 4 segments.*

Very recently, Mütze, Walczak, and Wiechert [32] improved this result. They proved that the statement holds even if the strings are simple polygonal paths of at most 3 segments, moreover, any two intersect at most once.

The following problems remain open.

▶ **Problem 6.**

(i) *Is the family of disjointness graphs of polygonal paths, each consisting of at most two segments, χ-bounded?*

(ii) *Is the previous statement true under the additional assumption that any two of the polygonal paths intersect in at most one point?*

▶ **Problem 7.** *Is the family of intersection graphs of lines in $\mathbb{R}^3$ χ-bounded?*

By Theorem 2, the family of *complements* of intersection graphs of lines in $\mathbb{R}^3$ is χ-bounded.

This paper is organized as follows. In the next section, we prove Theorem 2, which is needed for the proof of Theorem 1. Theorem 1 is established in Section 3. In Section 4, we construct several examples of disjointness graphs whose chromatic numbers are much larger than their clique numbers. In particular, we prove Theorem 5 and some similar statements. The last section contains the proof of Theorem 3 and remarks on the computational complexity of related problems. The proof of Theorem 4 is omitted in this conference version.

2 Disjointness graphs of lines – Proof of Theorem 2

▶ **Claim 8.** *Let G be the disjointness graph of a set of n lines in $\mathbb{P}^d$. If G has an isolated vertex, then G is perfect.*

Proof. Let $L_0 \in V(G)$ be a line representing an isolated vertex of G. Consider the bipartite multigraph H with vertex set $V(H) = A \cup B$, where A consists of all points of L_0 that belong to at least one other line $L \in V(G)$, and B is the set of all (2-dimensional) planes passing through L_0 that contain at least one other line $L \in V(G)$ different from L_0. We associate with any line $L \in V(G)$ different from L_0 an edge e_L of H, connecting the point $p = L \cap L_0 \in A$ to the plane $\pi \in B$ that contains L. Note that there may be several parallel edges in H. See Figure 1.

Observe that two lines $L, L' \in V(G) \setminus \{L_0\}$ intersect if and only if e_L and $e_{L'}$ share an endpoint. This means that G minus the isolated vertex L_0 is isomorphic to the complement

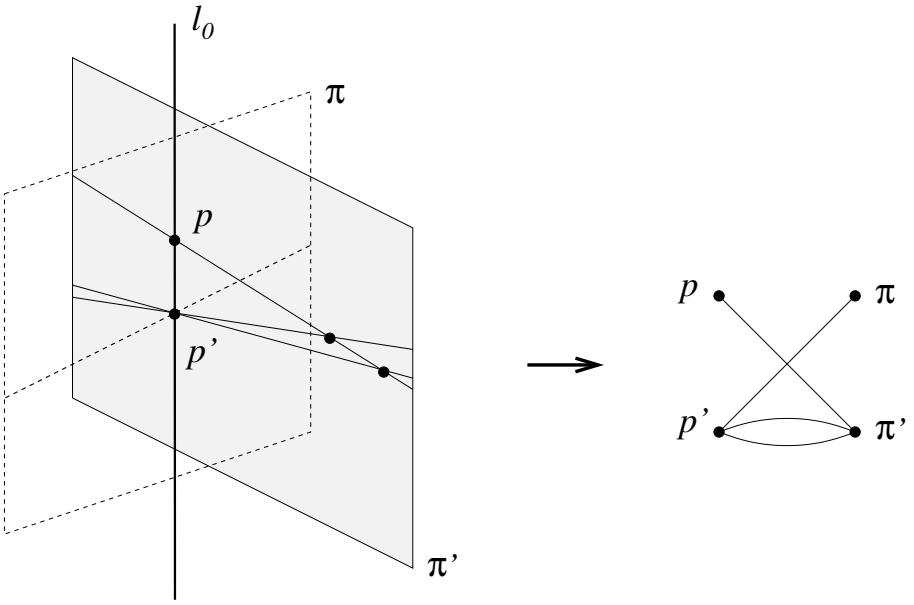


Figure 1 Construction of graph H in the proof of Claim 8.

of the line graph of H. The line graphs of bipartite multigraphs and their complements are known to be perfect. (For the complements of line graphs, this is the König-Hall theorem; see, e. g., [31].) The graph G can be obtained by adding the isolated vertex L_0 to a perfect graph, and is, therefore, also perfect. ◀

Proof of Theorem 2. We start with the proof of part (ii). Let G be a disjointness graph of lines in $\mathbb{P}^d$. Let $C \subseteq G$ be a maximal clique in G. Clearly, $|C| \leq \omega(G)$. By the maximality of C, for every $L \in V(G) \setminus C$, there exists $c \in C$ that is not adjacent to L in G. Hence, there is a partition of $V(G)$ into disjoint sets $V_c, c \in C$, such that $c \in V_c$ and c is an isolated vertex in the induced subgraph $G[V_c]$ of G. Applying Claim 8 separately to each subgraph $G[V_c]$, we obtain

$$\chi(G) \leq \sum_{c \in C} \chi(G[V_c]) = \sum_{c \in C} \omega(G[V_c]) \leq |C|\omega(G) \leq (\omega(G))^2.$$

Now we turn to the proof of part (i) of Theorem 2. Let G be a disjointness graph of lines in $\mathbb{R}^d$. Consider the lines in $V(G)$ as lines in the projective space $\mathbb{P}^d$, and consider the disjointness graph G' of these projective lines. Clearly, G' is a subgraph of G with the lines $L, L' \in V(G)$ adjacent in G but not adjacent in G' if and only if L and L' are parallel. Thus, an independent set in G' induces a disjoint union of complete subgraphs in G, where the vertices of each complete subgraph correspond to pairwise parallel lines. If k is the maximal number of pairwise parallel lines in $V(G)$, then $k \leq \omega(G)$ and each independent set in G' can be partitioned into at most k independent sets in G. Applying part (ii), we obtain

$$\chi(G) \leq k\chi(G') \leq \omega(G)(\omega(G'))^2 \leq (\omega(G))^3.$$

Finally, we prove the last claim concerning polynomial time algorithms. In the proof of part (ii), we first took a maximal clique C in G. Such a clique can be efficiently found by a greedy algorithm. The partition of $V(G)$ into subsets $V_c, c \in C$, such that $c \in V_c$ is an isolated vertex in the subgraph $G[V_c]$, can also be done efficiently. It remains to find a clique

of maximum size and a proper coloring of each perfect graph $G[V_c]$ with the smallest number of colors. It is well known that for perfect graphs, both of these tasks can be completed in polynomial time. See e.g. Corollary 9.4.8 on page 298 of [12]. Alternatively, notice that in the proof of Claim 8 we showed that $G[V_c]$ is, in fact, the complement of the line graph of a bipartite multigraph (plus an isolated vertex). Therefore, finding a maximum size complete subgraph corresponds to finding a maximum size matching in a bipartite graph, while finding an optimal proper coloring of $G[V_c]$ corresponds to finding a minimal size vertex cover in a bipartite graph. This can be accomplished by much simpler and faster algorithms than the general purpose algorithms developed for perfect graphs.

To finish the proof of the algorithmic claim for part (ii), we can simply output as K the set C or one of the largest maximum cliques in $G[V_c]$ over all $c \in C$, whichever is larger. We color each V_c optimally, with pairwise disjoint sets of colors.

For the algorithmic claim about part (i), first color the corresponding arrangement of projective lines, and then refine the coloring by partitioning each color class into at most k smaller classes, where k is the maximum number of parallel lines in the arrangement. It is easy to find the value of k, just partition the lines into groups of parallel lines. Output as K the set we found for the projective lines, or a set of k parallel lines, whichever is larger. ◀

▶ **Theorem 9.**

(i) *There exist disjointness graphs G of families of lines in $\mathbb{R}^3$ for which the ratio $\chi(G)/\omega(G)$ is arbitrarily large.*

(ii) *For any k one can find a system of lines in $\mathbb{P}^3$ whose disjointness graph G satisfies $\omega(G) = k$ and $\chi(G) = 2k - 1$.*

Proof. First, we prove (i). For some m and d to be determined later, consider the set W_m^d of integer points in the d-dimensional hypercube $[1, m]^d$. That is, $W_m^d = \{1, 2, \ldots, m\}^d$. A *combinatorial line* is a sequence of m distinct points of $x^1, \ldots x^m \in W_m^d$ such that for every $1 \leq i \leq d$, their ith coordinates $(x^j)_i$ are either the same for all $1 \leq j \leq m$ or we have $(x^j)_i = j$ for all $1 \leq j \leq m$. Note that the points of any combinatorial line lie on a geometric straight line. Let $\mathcal{L}$ denote the set of these geometric lines.

Let G denote the disjointness graph of $\mathcal{L}$. Since each line in $\mathcal{L}$ passes through m points of W_m^d, and $|W_m^d| = m^d$, we have $\omega(G) \leq m^{d-1}$. (It is easy to see that equality holds here, but we do not need this fact for the proof.)

Consider any proper coloring of G. The color classes are families of pairwise crossing lines in $\mathcal{L}$. Observe that any such family has a common point in W_m^d, except some families consisting of 3 lines. Take an optimal proper coloring of G with $\chi(G)$ colors, and split each 3-element color class into two smaller classes. In the resulting coloring, there are at most $2\chi(G)$ color classes, each of which has a point of W_m^d in common. This means that the set of at most $2\chi(G)$ points of W_m^d (the "centers" of the color classes) "hits" every combinatorial line. By the density version of the Hales-Jewett theorem, due to Furstenberg and Katznelson [4, 11], if d is large enough relative to m, then any set containing fewer than half of the points of W_m^d will miss an entire combinatorial line. Choosing any m and a sufficiently large d depending on m, we conclude that $2\chi(G) \geq m^d/2$ and $\chi(G)/\omega(G) \geq m/4$.

Note that the family $\mathcal{L}$ consists of lines in $\mathbb{R}^d$. To find a similar family in 3-space, simply take the image of $\mathcal{L}$ under a projection to $\mathbb{R}^3$. One can pick a generic projection that does not change the disjointness graph G. This completes the proof of part (i). Note that the same construction does not work for projective lines, as the combinatorial lines in W_m^d fall into $2^d - 1$ parallel classes, so the chromatic number of the corresponding projective disjointness graph is smaller than 2^d.

To establish part (ii), fix a positive integer k, and consider a set S of $2k+1$ points in general position (no four in a plane) in $\mathbb{R}^3 \subseteq \mathbb{P}^3$. Let $\mathcal{L}$ denote the set of $\binom{2k+1}{2}$ lines determined by them. Note that by the general position assumption, two lines in $\mathcal{L}$ intersect if and only if they have a point of S in common. This means that the disjointness graph G of $\mathcal{L}$ is isomorphic to the *Kneser graph* $G^*(2k+1,2)$ formed by all 2-element subsets of a $(2k+1)$-element set. Obviously, $\omega(G^*(n,m)) = \lfloor n/m \rfloor$, so $\omega(G) = k$. By a celebrated result of Lovász [30], $\chi G^*(n,m) = n - 2m + 2$ for all $n \geq 2m-1$. Thus, we have $\chi(G) = 2k-1$, as claimed. ◀

3 Disjointness graphs of segments – Proof of Theorem 1

If all segments lie in the same plane, then by Theorem 0 we have $\chi(G) \leq (\omega(G))^4$. Our next theorem generalizes this result to the case where the segments lie in a bounded number of distinct planes.

▶ **Theorem 10.** *Let G be the disjointness graph of a set of segments in $\mathbb{R}^d, d > 2$, that lie in the union of k two-dimensional planes. We have*

$$\chi(G) \leq (k-1)\omega(G) + (\omega(G))^4.$$

Given the segments representing the vertices of G and k planes containing them, there is a polynomial time algorithm to find a complete subgraph $K \subseteq G$ and a proper coloring of G with at most $(k-1)|V(K)| + |V(K)|^4$ colors.

Proof. Let $\pi_1, \pi_2, \ldots, \pi_k$ be the planes containing the segments. Partition the vertex set of G into the classes $V_1, V_2, \ldots, V_k$ by putting a segment s into the class V_i, where i is the *largest* index for which π_i contains s.

For $i = 1, 2, \ldots, k$, we define subsets $W_i, Z_i \subseteq V_i$ with $Z_i \subseteq W_i \subseteq V_i$ by a recursive procedure, as follows. Let $W_1 = V_1$ and let $Z_1 \subseteq W_1$ be a *maximal size* clique in $G[W_1]$.

Assume that the sets $W_1, \ldots, W_i$ and $Z_1, \ldots, Z_i$ have already been defined for some $i < k$. Let W_{i+1} denote the set of all vertices in V_{i+1} that are adjacent to every vertex in $Z_1 \cup Z_2 \cup \ldots \cup Z_i$, and let Z_{i+1} be a maximal size clique in $G[W_{i+1}]$. By definition, $\bigcup_{i=1}^k Z_i$ induces a complete subgraph in G, and we have

$$\sum_{i=1}^{k} |Z_i| \leq \omega(G).$$

Let s be a segment belonging to Z_i, for some $1 \leq i < k$. A point p of s is called a *piercing point* if $p \in \pi_j$ for some $j > i$. Notice that in this case, s "pierces" the plane π_j in a single point, otherwise we would have $s \subset \pi_j$, contradicting our assumption that $s \in V_i$. Letting P denote the set of piercing points of all segments in $\bigcup_{i=1}^k Z_i$, we have

$$|P| \leq \sum_{i=1}^{k} (k-i)|Z_i| \leq (k-1)\sum_{i=1}^{k} |Z_i| \leq (k-1)\omega(G).$$

Let $V_0 = V(G) \setminus \bigcup_{i=1}^k W_i$. We claim that every segment in V_0 contains at least one piercing point. Indeed, if $s \in V_i \setminus W_i$ for some $i \leq k$, then s is not adjacent in G to at least one segment $t \in Z_1 \cup \ldots \cup Z_{i-1}$. Thus, s and t are not disjoint, and their intersection point is a piercing point, at which t pierces the plane π_i.

Assign a color to each piercing point $p \in P$. Coloring every segment in V_0 by the color of one of its piercing points, we get a proper coloring of $G[V_0]$ with $|P|$ colors, so that $\chi(G[V_0]) \leq |P|$.

For every $i \leq k$, all segments of W_i lie in the plane π_i. Therefore, we can apply Theorem 0 to their disjointness graph $G[W_i]$, to conclude that $\chi(G[W_i]) \leq (\omega(G[W_i]))^4$. By definition, Z_i induces a maximum complete subgraph in $G[W_i]$, hence $|Z_i| = \omega(G[W_i])$ and $\chi(G[W_i]) \leq |Z_i|^4$.

Putting together the above estimates, and taking into account that $\bigcup_{i=1}^{k} Z_i$ induces a complete subgraph in G, we obtain

$$\chi(G) \leq \chi(G[V_0]) + \sum_{i=1}^{k} \chi(G[W_i]) \leq |P| + \sum_{i=1}^{k} |Z_i|^4$$

$$\leq (k-1)\omega(G) + (\sum_{i=1}^{k} |Z_i|)^4 \leq (k-1)\omega(G) + (\omega(G))^4,$$

as required.

We can turn this estimate into a polynomial time algorithm as required, using the fact that the proof of Theorem 0 is constructive. In particular, we use that, given a family of segments in the plane, one can efficiently find a subfamily K of pairwise disjoint segments and a proper coloring of the disjointness graph with at most $|K|^4$ colors. This readily follows from the proof of Theorem 0, based on the four easily computable (semi-algebraic) partial orders on the family of segments, introduced in [28].

Our algorithm finds the sets V_i, as in the proof. However, finding W_i and a maximum size clique $Z_i \subseteq W_i$ is a challenge. Instead, we use the constructive version of Theorem 0 to find $Z_i \subseteq W_i$ and a proper coloring of $G[W_i]$. The definition of W_i remains unchanged. Next, the algorithm identifies the piercing points.

The algorithm outputs the clique $K = \bigcup Z_i$ and the coloring of G. The latter one is obtained by combining the previously constructed colorings of the subgraphs $G[W_i]$ (using disjoint sets of colors for different subgraphs), and coloring each remaining vertex by a previously unused color, associated with one of the piercing points the corresponding segment passes through. ◀

Proof of Theorem 1. Consider the set of all lines in the *projective* space $\mathbb{P}^d$ that contain at least one segment belonging to $V(G)$. Let $\bar{G}'$ denote the disjointness graph of these lines. Obviously, we have $\omega(\bar{G}') \leq \omega(G)$. Thus, Theorem 2(ii) implies that

$$\chi(\bar{G}') \leq (\omega(\bar{G}'))^2 \leq (\omega(G))^2.$$

Let C be the set of lines corresponding to the vertices of a maximum complete subgraph in $\bar{G}'$. Fix an optimal proper coloring of $\bar{G}'$. Suppose that we used k "planar" colors (each such color is given to a set of lines that lie in the same plane) and $\chi(\bar{G}') - k$ "pointed" colors (each given to the vertices corresponding to a set of lines passing through a common point).

Consider now G, the disjointness graph of the segments. Let G_0 denote the subgraph of G induced by the set of segments whose supporting lines received one of the k planar colors in the above coloring of $\bar{G}'$. These segments lie in at most k planes. Therefore, applying Theorem 10 to G_0, we obtain

$$\chi(G_0) \leq (k-1)\omega(G_0) + (\omega(G_0))^4 \leq (k-1)\omega(G) + (\omega(G))^4.$$

For $i, 1 \leq i \leq \chi(\bar{G}') - k$, let G_i denote the subgraph of G induced by the set of segments whose supporting lines are colored by the ith pointed color. It is easy to see that G_i is the complement of a chordal graph. That is, the complement of G_i contains no induced cycle of length larger than 3. According to a theorem of Hajnal and Surányi [18], any graph with this property is perfect, so that

$$\chi(G_i) = \omega(G_i) \leq \omega(G).$$

Putting these bounds together, we obtain that

$$\chi(G) \leq \chi(G_0) + \sum_{i=1}^{\chi(\bar{G}')-k} \chi(G_i) \leq (k-1)\omega(G) + (\omega(G))^4 + \sum_{i=1}^{\chi(\bar{G}')-k} \omega(G)$$

$$\leq ((\omega(\bar{G}'))^2 - 1)\omega(G) + (\omega(G))^4 < (\omega(G))^3 + (\omega(G))^4.$$

To prove the algorithmic claim in Theorem 1, we first apply the algorithm of Theorem 2 to the disjointness graph $\bar{G}'$. We distinguish between planar and pointed color classes and find the subgraphs G_i. We output a coloring of G, where for each $G_i, i > 0$ we use the smallest possible number of colors (G_i is perfect, so its optimal coloring can be found in polynomial time), and we color G_0 by the algorithm described in Theorem 10. The subgraphs G_i are colored using pairwise disjoint sets of colors. We output the largest clique K that we can find. This may belong to a subgraph G_i with $i > 0$, or may be found in G_0 or in $\bar{G}'$ by the algorithms given by Theorem 10 or Theorem 2, respectively. (In the last case, we need to turn a clique in $\bar{G}'$ into a clique of the same size in G, by picking an arbitrary segment from each of the pairwise disjoint lines.) ◀

4 Constructions – Proof of Theorem 5

The aim of this section is to describe various arrangements of geometric objects in 2, 3, and 4 dimensions with triangle-free disjointness graphs, whose chromatic numbers grow logarithmically with the number of objects. (This is much faster than the rate of growth in Theorem 9.) Our constructions can be regarded as geometric realizations of a sequence of graphs discovered by Erdős and Hajnal.

▶ **Definition 11** ([10]). Given $m > 1$, let H_m, the *m-th shift graph*, be a graph whose vertex set consists of all ordered pairs (i, j) with $1 \leq i < j \leq m$, and two pairs (i, j) and (k, l) are connected by an edge if and only if $j = k$ or $l = i$.

Obviously, H_m is triangle-free for every $m > 1$. It is not hard to show (see, e.g., [31], Problem 9.26) that $\chi(H_m) = \lceil \log_2 m \rceil$. Therefore, Theorem 5 follows directly from part (vii) of the next theorem.

▶ **Theorem 12.** *For every m, the shift graph H_m can be obtained as a disjointness graph, where each vertex is represented by*

- **(i)** *a line minus a point in $\mathbb{R}^2$;*
- **(ii)** *a two-dimensional plane in $\mathbb{R}^4$;*
- **(iii)** *the intersection of two general position half-spaces in $\mathbb{R}^3$;*
- **(iv)** *the union of two segments in $\mathbb{R}^2$;*
- **(v)** *a triangle in $\mathbb{R}^4$;*
- **(vi)** *a simplex in $\mathbb{R}^3$;*
- **(vii)** *a polygonal curve in $\mathbb{R}^2$, consisting of four line segments.*

Proof.

(i) Let $L_1, \ldots, L_m$ be lines in general position in the plane. For any $1 \leq i < j \leq m$, let us represent the pair (i, j) by the "pointed line" $p_{ij} = L_i \setminus L_j$.
Fix $1 \leq i < j \leq m$, $1 \leq k < l \leq m$, and set $X = p_{ij} \cap p_{kl} = (L_i \cap L_k) \setminus (L_j \cup L_l)$. If $i = k$, then X is an infinite set.
Otherwise, $L_i \cap L_k$ consists of a single point. In this case, X is empty if and only if this point belongs to $L_j \cup L_l$. By the general position assumption, this happens if and only if $j = k$ or $l = i$. Thus, the disjointness graph of the sets p_{ij}, $1 \leq i < j \leq m$, is isomorphic to the shift graph H_m.

(ii) Let $h_1, \ldots h_m$ be hyperplanes in general position in $\mathbb{R}^4$. For every i, fix another hyperplane h'_i, parallel (but not identical) to h_i. For any $1 \leq i < j \leq m$, represent the pair (i, j) by the two dimensional plane $p_{ij} = h_i \cap h'_j$.
Given $1 \leq i < j \leq m$, $1 \leq k < l \leq m$, the set $X = p_{ij} \cap p_{k,l} = h_i \cap h'_j \cap h_k \cap h'_l$ is the intersection of four hyperplanes. If the four hyperplanes are in general position, then X consists of a single point.
If the hyperplanes are not in general position, then some of the four indices must coincide. If $i = k$ or $j = l$, then two of the hyperplanes coincide and X is a line. In the remaining cases, when $j = k$ or $l = i$, among the four hyperplanes two are parallel, so their intersection X is empty.

(iii) For $i = 1, \ldots, m$, define the half-space h_i as

$$h_i = \{(x, y, z) \in \mathbb{R}^3 \mid ix + i^2 y + i^3 z < 1\}.$$

Note that the bounding planes of these half-spaces are in general position. For any $1 \leq i < j \leq m$, represent the pair (i, j) by $p_{ij} = h_j \setminus h_i$.
Now let $1 \leq i < j \leq m$, $1 \leq k < l \leq m$. If $j = k$ or $l = i$, the sets p_{ij} and p_{kl} are obviously disjoint. If $i = k$ or $j = l$, then $p_{ij} \cap p_{kl}$ is the intersection of at most 3 half-spaces in general position, so it is unbounded and not empty.
It remains to analyze the case when all four indices are distinct. This requires some calculation. We assume without loss of generality that $j < l$. Consider the point $P = (x, y, z) \in \mathbb{R}^3$ with $x = \frac{1}{i} + \frac{1}{j} + \frac{1}{k}$, $y = -\frac{1}{ij} - \frac{1}{jk} - \frac{1}{ki}$ and $z = \frac{1}{ijk}$. This is the intersection point of the bounding planes of h_i, h_j and h_k. Therefore, the polynomial $zu^3 + yu^2 + xu - 1$ vanishes at $u = i, j, k$, and it must be positive at $u = l$, as $l > i, j, k$ and the leading coefficient is positive. This means that P lies in the open half-space h_l. As the bounding planes of h_i, h_j and h_k are in general position, one can find a point P' arbitrarily close to P (the intersection point of these half-planes) with $P' \in h_j \setminus (h_i \cup h_k)$. If we choose P' close enough to P, it will also belong to h_l. Thus, $P' \in p_{ij} \cap p_{kl}$, and so p_{ij} and p_{kl} are not disjoint.

(iv), **(v)**, and **(vi)** directly follow from (i), (ii) and (iii), respectively, by replacing the unbounded geometric objects representing the vertices with their sufficiently large bounded subsets.

(vii) Let C be an almost vertical, very short curve (arc) in the plane, convex from the right (that is, the set of points to the right of C is convex) lying in a small neighborhood of $(0, 1)$. Let $p_1, p_2, \ldots, p_m$ be a sequence of m points on C such that p_j is above p_i if and only if $j > i$. For every $1 \leq i \leq m$, let T_i be an equilateral triangle whose base is horizontal, whose upper vertex is p_i, and whose center is on the x-axis. Let q_i and r_i be the lower right and lower left vertices of T_i, respectively. It is easy to see that T_j contains T_i in its interior if $j > i$. Let s_i be a point on $r_i p_i$, very close to p_i.
Let us represent the vertex (i, j) of the shift graph H_m by the polygonal curve $p_{ij} =$

$t_{ij}p_jq_jr_js_j$, where the point t_{ij} is on the x-axis slightly to the left of the line p_ip_j. Note that if C is short enough and close enough to vertical, then t_{ij} can be chosen so that it belongs to the interior of all triangles T_k for $1 \le k \le m$. In particular, the entire polygonal path p_{ij} belongs to T_j.
It depends on our earlier choices of the vertices $p_{i'}$, how close we have to choose s_i to p_i. Analogously, it depends on our earlier choices of $p_{i'}$ and $s_{i'}$, how close we have to choose t_{ij} to the line. Instead of describing an explicit construction, we simply claim that with proper choices of these points, we obtain a disjointness representation of the shift graph.

To see this, let $1 \le i < j \le m$, $1 \le k < l \le m$. If $j = l$, then three of the four line segments in p_{ij} and p_{kl} are the same, so they intersect. Otherwise, assume without loss of generality that $j < l$. As noted above, p_{ij} belongs to the triangle T_j, which, in turn, lies in the interior of T_l. Three segments of p_{kl} lie on the edges of T_l, so if p_{ij} and p_{kl} meet, the fourth segment, $t_{kl}p_l$, must meet p_{ij}. This segment enters the triangle T_j, so it meets one of its edges. Namely, for $j > k$ it follows from the convexity of the curve C that the segment $t_{kl}p_l$ intersects the edge p_jq_j and, hence, also p_{ij}. Analogously, if $j < k$, then $t_{kl}p_l$ intersects the interior of the edge r_jp_j. This is true even if t_{kl} were chosen *on* the line p_kp_l, so choosing s_j close enough to p_j, one can make sure that $t_{kl}p_l$ intersects r_js_j and, hence, also p_{ij}. On the other hand, if $j = k$, we choose t_{kl} so that $t_{kl}p_l$ is just slightly to the left of $p_j = p_k$, so it enters T_j through the interior of the segment s_jp_j that is *not* contained in p_{ij}. To see that in this case p_{ij} and p_{kl} are disjoint, it is enough to check that $t_{kl}p_l$ and $t_{ij}p_j$ are disjoint. This is true, because p_j is on the right of $t_{kl}p_l$ and (from the convexity of C) the slope of the segments is such that p_j is the closest point of the segment $t_{ij}p_j$ to $t_{kl}p_l$. ◀

5 Complexity issues – Proof of Theorem 3

The aim of this section is to outline the proof of Theorem 3 and to establish some related complexity results. For simplicity, we only consider systems of lines in the *projective* space $\mathbb{P}^3$. It is easy to see that by removing a generic hyperplane (not containing any of the intersection points), we can turn a system of projective lines into a system of lines into $\mathbb{R}^3$ without changing the corresponding disjointness graph.

It is more convenient to speak about intersection graphs rather than their complement in formulating the next theorem.

▶ **Theorem 13.**

(i) *If G is a graph with maximum degree at most 3, then G is an intersection graph of lines in $\mathbb{P}^3$.*

(ii) *For an arbitrary graph G the* line graph *of G is an intersection graph of lines in $\mathbb{P}^3$.*

Proof.

(i) Suppose first that G is triangle-free. Let $V(G) = \{v_1, \dots, v_k\}$. Let vertex v_1 be represented by an arbitrary line L_1. Suppose, recursively, that the line L_j representing vertex j has already been defined for every $j < i$. We will maintain the "general position" property that no doubly ruled surface contains more than 3 pairwise disjoint lines. We must choose L_i representing v_i such that
(a) it intersects the lines representing the neighbors v_j of v_i with $j < i$,
(b) it does not intersect the lines representing the non-neighbors v_j with $j < i$, and
(c) we maintain our general position conditions.
These are simple algebraic conditions. The vertex v_i has at most 3 neighbors among v_j for $j < i$, and they must be represented by pairwise disjoint lines. Thus, the Zariski-

closed conditions from (a) determine an irreducible variety of lines, so unless they force the violation of a specific other (Zariski-open) condition from (b) or (c), all of those conditions can be satisfied with a generic line through the lines representing the neighbors. In case v_i has three neighbors v_j with $j < i$, the corresponding condition forces L_i to be in one of the two families of lines on a doubly ruled surface Σ. This further forces L_i to intersect *all* lines of the other family on Σ, but due to the general position condition, none of the vertices of G is represented by lines there, except the three neighbors of v_i. We would violate the general position condition with the new line L_i if the family we choose it from already had three members representing vertices. However, this would mean that the degrees of the neighbors of v_i would be at least 4, a contradiction. In case v_i has fewer than 3 neighbors, the requirement of L_i intersecting the corresponding lines does not force L_i to intersect any further lines or to lie on any doubly ruled surface.

We prove the general case by induction on $|V(G)|$. Suppose that $a, b, c \in V(G)$ form a triangle in G and that the subgraph of G induced by $V(G) \setminus \{a, b, c\}$ can be represented as the intersection graph of distinct lines in $\mathbb{P}^3$. Note that each of a, b and c has at most a single neighbor in the rest of the graph. We extend the representation of the subgraph by adding three lines L_a, L_b and L_c, representing the vertices of the triangle. We choose these lines in a generic way so that they pass through a common point p, and L_a intersects the line representing the neighbor of a (in case it exists), and similarly for L_b and L_c. It is clear that we have enough degrees of freedom (at least six) to avoid creating any further intersection. For instance, it suffices to choose p outside all lines in the construction and all planes determined by intersecting pairs of lines.

(ii) Assign distinct points of $\mathbb{P}^3$ to the vertices of G so that no four points lie in a plane. Represent each edge $xx' \in E(G)$ by the line connecting the points assigned to x and x'. As no four points are coplanar, two lines representing a pair of edges will cross if and only if the edges share an endpoint. Therefore, the intersection graph of these lines is isomorphic to the edge graph of G. ◀

The following theorem implies Theorem 3, as the disjointness graph $H = \bar{G}$ is the complement of the intersection graph G, and we have $\omega(G) = \alpha(H)$, $\alpha(G) = \omega(H)$, $\chi(G) = \theta(H)$, and $\theta(G) = \chi(H)$. Here $\theta(H)$ denotes the *clique covering number* of H, that is, the smallest number of complete subgraphs of H whose vertex sets cover $V(H)$.

▶ **Theorem 14.** *Let H be an intersection graph of n lines in the Euclidean space $\mathbb{R}^3$ or in the projective space $\mathbb{P}^3$.*

(i) *Computing $\alpha(H)$, the independence number of H, is NP-hard.*
(ii) *Computing $\theta(H)$, the clique covering number of H, is NP-hard.*
(iii) *Deciding whether $\chi(H) \leq 3$, that is, whether H is 3-colorable, is NP-complete.*
(iv) *Computing $\omega(H)$, the clique number of H, is in P.*
(v) *Deciding whether $\theta(H) \leq k$ for a fixed k is in P.*
(vi) *All the above statements remain true if H is not given as an abstract graph, but with its intersection representation with lines.*

Proof. We only deal with the case where the lines are in $\mathbb{P}^3$. The reduction of the Euclidean case to this case is easy.

(i) The problem of determining the independence number of 3-regular graphs is NP-hard; see [1]. By Theorem 13 (i), all 3-regular graphs are intersection graphs of lines in $\mathbb{P}^3$.
(ii) The *vertex cover number* of a graph H is the smallest number of vertices with the property that every edge of H is incident to at least one of them. Note that the

vertex cover number of H is $|V(H)| - \alpha(H)$. In [34], it was shown that the problem of determining the *vertex cover number* is NP-hard even for triangle-free graphs. We can reduce this problem to the problem of determining the clique covering number of an intersection graph of lines. For this, note that each complete subgraph of the line graph H' of H corresponds to a star of H and thus $\theta(H')$ is the vertex cover number of H. The reduction is complete, as H' is the intersection graph of lines in $\mathbb{P}^3$, by Theorem 13 (ii).

(iii) Deciding whether the *chromatic index* (chromatic number of the line graph) of a 3-regular graph is 3 is NP-complete, see [20]. Using that the line graph of any graph is an intersection graph of lines in $\mathbb{P}^3$ (Theorem 13 (ii)), the statement follows.

(iv) A maximal complete subgraph corresponds to a set of lines passing through the same point p or lying in the same plane Π. Any such point p or plane Π is determined by two lines, and in both cases we can verify for each remaining line whether it belongs to the corresponding complete subgraph (whether it passes through p or belongs to Π, respectively). This gives an $O(n^3)$-time algorithm, but we suspect that the running time can be much improved.

(v) As we have seen in part (iv), there are polynomially many maximal complete subgraphs in H. We can check all k-tuples of them, and decide whether they cover all vertices in H.

(vi) For this, we need to consider the constructions of lines in the representations described in the proof of Theorem 13, and show that they can be built in polynomial time. This is obvious in part (ii) of the theorem. For part (i), the situation is somewhat more complex. To find many possible representations of the next vertex intersecting the lines it should, is an algebraically simple task. In polynomial time, we can find one of them that is generic in the sense needed for the construction. However, if the coordinates of each line would be twice as long as those of the preceding line (a condition that is hard to rule out *a priori*), then the whole construction takes more than polynomial time.
A simple way to avoid this problem is the following. First, color the vertices of the triangle-free graph G of maximal degree at most 3 by at most 4 colors, by a simple greedy algorithm. Find the lines representing the vertices in the following order: first for the first color class, next for second color class, etc. The coordinates of each line will be just slightly more complex than the coordinates of the lines representing vertices in *earlier color classes*. Therefore, the construction can be performed in polynomial time. A similar argument works also for graphs G with triangles: First we find a maximal subset of pairwise vertex-disjoint triangles in G. Let G_0 be the graph obtained from G by removing these triangles. Then we construct an auxiliary graph G' with these triangles as vertices by connecting two of them with an edge if there is an edge in G between the triangles. The graph G' has maximum degree at most 3, so it can be greedily 4-colored. If we construct G by adding back the triangles to G_0, in the order determined by their colors, then the procedure will end in polynomial time. ◀

References

1. Paola Alimonti and Viggo Kann. Some APX-completeness results for cubic graphs. *Theoretical Computer Science*, 237(1-2):123–134, 2000.
2. Edgar Asplund and Branko Grünbaum. On a coloring problem. *Mathematica Scandinavica*, 8(1):181–188, 1960.
3. Claude Berge. Färbung von Graphen, deren sämtliche bzw. ungerade Kreise starr sind (Zusammenfassung). *Wiss. Z. Martin-Luther-Univ. Halle Wittenberg Math. Natur. Reihe*, 114, 1961.

4 Béla Bollobás. *Modern Graph Theory, Graduate Texts in Mathematics vol. 184*. Springer-Verlag, New York, 1998.

5 James P. Burling. On coloring problems of families of prototypes. (PhD thesis), University of Colorado, Boulder, 1965.

6 Sergio Cabello, Jean Cardinal, and Stefan Langerman. The clique problem in ray intersection graphs. *Discrete & computational geometry*, 50(3):771–783, 2013.

7 Robert P. Dilworth. A decomposition theorem for partially ordered sets. *Annals of Mathematics*, pages 161–166, 1950.

8 Gabriel Andrew Dirac. On rigid circuit graphs. In *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, volume 25, pages 71–76. Springer, 1961.

9 Gideon Ehrlich, Shimon Even, and Robert Endre Tarjan. Intersection graphs of curves in the plane. *Journal of Combinatorial Theory, Series B*, 21(1):8–20, 1976.

10 Paul Erdős and András Hajnal. Some remarks on set theory. ix: Combinatorial problems in measure theory and set theory. *Michigan Math. J*, 11(2):107–127, 1964.

11 Hillel Furstenberg and Yitzhak Katznelson. A density version of the hales-jewett theorem. *Journal d'Analyse Mathématique*, 57(1):64–119, 1991.

12 Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric algorithms and combinatorial optimization*. Springer-Verlag, Berlin, 1988.

13 András Gyárfás. On the chromatic number of multiple interval graphs and overlap graphs. *Discrete mathematics*, 55(2):161–166, 1985.

14 András Gyárfás. Corrigendum. *Discrete mathematics*, 62(3):333, 1986.

15 András Gyárfás. Problems from the world surrounding perfect graphs. *Applicationes Mathematicae*, 19(3-4):413–441, 1987.

16 András Gyárfás and Jenő Lehel. Hypergraph families with bounded edge cover or transversal number. *Combinatorica*, 3(3-4):351–358, 1983.

17 András Gyárfás and Jenő Lehel. Covering and coloring problems for relatives of intervals. *Discrete Mathematics*, 55(2):167–180, 1985.

18 András Hajnal and János Surányi. Über die auflösung von graphen in vollständige teilgraphen. *Ann. Univ. Sci. Budapest, Eötvös Sect. Math*, 1:113–121, 1958.

19 György Hajós. Über eine Art von Graphen. *Internationale Mathematische Nachrichten*, 11:65, 1957.

20 Ian Holyer. The NP-completeness of edge-coloring. *SIAM Journal on Computing*, 10(4):718–720, 1981.

21 Gyula Károlyi. On point covers of parallel rectangles. *Periodica Mathematica Hungarica*, 23:105–107, 1991.

22 Gyula Károlyi, János Pach, and Géza Tóth. Ramsey-type results for geometric graphs, i. *Discrete & Computational Geometry*, 18(3):247–255, 1997.

23 Alexandr Kostochka. Coloring intersection graphs of geometric figures with a given clique number. *Contemporary Mathematics*, 342:127–138, 2004.

24 Alexandr Kostochka and Jan Kratochvíl. Covering and coloring polygon-circle graphs. *Discrete Mathematics*, 163(1):299–305, 1997.

25 Alexandr V. Kostochka. Upper bounds for the chromatic numbers of graphs. *Modeli i Metody Optim. (Russian)*, 10:204–226, 1988.

26 Jan Kratochvíl and Jaroslav Nešetřil. Independent set and clique problems in intersection-defined classes of graphs. *Commentationes Mathematicae Universitatis Carolinae*, 31(1):85–93, 1990.

27 Jan Kynčl. Ramsey-type constructions for arrangements of segments. *European Journal of Combinatorics*, 33(3):336–339, 2012.

28 David Larman, Jiří Matoušek, János Pach, and Jenő Törőcsik. A Ramsey-type result for convex sets. *Bulletin of the London Mathematical Society*, 26(2):132–136, 1994.

29 László Lovász. Normal hypergraphs and the perfect graph conjecture. *Discrete Mathematics*, 2(3):253–267, 1972.

30 László Lovász. Kneser's conjecture, chromatic number, and homotopy. *Journal of Combinatorial Theory, Series A*, 25(3):319–324, 1978.

31 László Lovász. *Combinatorial problems and exercises*. American Mathematical Soc., 1993.

32 Torsten Mütze, Bartosz Walczak, and Veit Wiechert. Realization of shift graphs as disjointness graphs of 1-intersecting curves in the plane. Manuscript, 2017.

33 Arkadiusz Pawlik, Jakub Kozik, Tomasz Krawczyk, Michał Lasoń, Piotr Micek, William T. Trotter, and Bartosz Walczak. Triangle-free intersection graphs of line segments with large chromatic number. *Journal of Combinatorial Theory, Series B*, 105:6–10, 2014.

34 Svatopluk Poljak. A note on stable sets and colorings of graphs. *Commentationes Mathematicae Universitatis Carolinae*, 15(2):307–309, 1974.

35 Noam Solomon, Michael S. Payne, and Jean Cardinal. Ramsey-type theorems for lines in 3-space. *Discrete Mathematics & Theoretical Computer Science*, 18, 2016.

Bicriteria Rectilinear Shortest Paths among Rectilinear Obstacles in the Plane*†

Haitao Wang

Department of Computer Science, Utah State University, Logan, UT, USA
haitao.wang@usu.edu

Abstract

Given a rectilinear domain $\mathcal{P}$ of h pairwise-disjoint rectilinear obstacles with a total of n vertices in the plane, we study the problem of computing bicriteria rectilinear shortest paths between two points s and t in $\mathcal{P}$. Three types of bicriteria rectilinear paths are considered: minimum-link shortest paths, shortest minimum-link paths, and minimum-cost paths where the cost of a path is a non-decreasing function of both the number of edges and the length of the path. The one-point and two-point path queries are also considered. Algorithms for these problems have been given previously. Our contributions are threefold. First, we find a critical error in all previous algorithms. Second, we correct the error in a not-so-trivial way. Third, we further improve the algorithms so that they are even faster than the previous (incorrect) algorithms when h is relatively small. For example, for computing a minimum-link shortest s-t path, the previous algorithm runs in $O(n \log^{3/2} n)$ time while the time of our new algorithm is $O(n + h \log^{3/2} h)$.

1998 ACM Subject Classification I.3.5 Computational Geometry and Object Modeling, F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases rectilinear paths, shortest paths, minimum-link paths, bicriteria paths, rectilinear polygons

Digital Object Identifier 10.4230/LIPIcs.SoCG.2017.60

1 Introduction

Let $\mathcal{P}$ be a *rectilinear domain* with a total of h holes and n vertices in the plane, i.e., $\mathcal{P}$ is a multiply-connected region whose boundary is a union of n axis-parallel line segments, forming $h + 1$ closed polygonal cycles (i.e., h holes plus an outer boundary). A simple rectilinear polygon is a special case of a rectilinear domain with $h = 0$. A *rectilinear* path is a path consisting of only horizontal and vertical line segments.

For a rectilinear path π, we define its *length* as the total sum of the lengths of the segments of π, and we define its *link distance* as the number of edges of π (each edge is also called a *link*). We use the *measure* of π to refer to both its length and its link distance. For any two points s and t in $\mathcal{P}$, a *shortest rectilinear path* from s to t is a rectilinear path connecting s to t in $\mathcal{P}$ with the minimum length, and a *minimum-link rectilinear path* is a rectilinear s-t path with the minimum link distance. Among all shortest rectilinear s-t paths, one with the minimum link distance is called a *minimum-link shortest s-t path*; among all minimum-link s-t paths, one with the minimum length is called a *shortest minimum-link s-t path*. We define the *cost* of π as a non-decreasing function f of both the length and the link distance of π. We assume that given the number of links of π and the length of π, its cost can be computed

* A full version of the paper is available at https://arxiv.org/abs/1703.04466.
† This research was supported in part by NSF under Grant CCF-1317143.

in constant time. Depending on the context, the measure of π may also refer to its cost. A *minimum-cost* path from s to t is a rectilinear s-t path in $\mathcal{P}$ with the minimum cost.

All the three types of paths discussed above (i.e., minimum-link shortest paths, shortest minimum-link paths, and minimum-cost paths) are called *bicriteria shortest paths*. In order to differentiate between "bicriteria shortest paths" and "shortest paths", we will use *optimal paths* to refer to these bicriteria shortest paths. Since some observations and algorithmic schemes may be applicable to all three types of optimal paths, unless otherwise stated, a statement made to "optimal paths" should be applicable to all three types of optimal paths.

In this paper, we study the problem of computing all three types of optimal paths between two points s and t in $\mathcal{P}$. Their one-point and two-point queries are also considered.

Previous Work. The following results are applicable to all three types of optimal paths.

Yang et al. [24] first presented an $O(nr + n \log n)$ time algorithm, where r is the number of extreme edges of $\mathcal{P}$ (an edge e is *extreme* if its two adjacent edges lie on the same side of the line containing e; $r = \Omega(n)$ in the worst case). Later, Yang et al. [25] proposed an algorithm of $O(n \log^2 n)$ time and $O(n \log n)$ space and another algorithm of $O(n \log^{3/2} n)$ time and space; Chen et. al. [2] improved it to $O(n \log^{3/2} n)$ time and $O(n \log n)$ space.

The *one-point optimal path query problem*, where s is the source and t is a query point, was also studied. Based on the algorithm of Yang et al. [25], Chen et. al. [2] built a data structure of $O(n \log n)$ size in $O(n \log^{3/2} n)$ time such that for each query point t, the measure of the optimal s-t path can be computed in $O(\log n)$ time and an actual path can be output in additional time linear in the number of edges of the path. For simplicity, in the following, when we say that the query time of a data structure for finding a path is $O(g(n))$, we mean that the measure of the path can be computed in $O(g(n))$ time and an actual path can be output in additional time linear in the number of edges of the path.

The *two-point optimal path query problem*, i.e., both s and t are query points, was also studied by Chen et. al. [2], where a data structure of $O(n^2 \log^2 n)$ size was built in $O(n^2 \log^2 n)$ time such that each two-point query can be answered in $O(\log^2 n)$ time.

Our Results. We provide a comprehensive study on these problems and our contributions are threefold.

First, we show that all above algorithms in the previous work are incorrect. More specifically, we find a critical error in the algorithm of Yang et al. [25]. Since the results of Chen et. al. [2] are all based on the method of Yang et al. [25], they are not correct either. A similar error also appears in [24]. Note that the technique of Chen et. al. [2], which follows the similar idea in [4] for computing L_1 shortest paths in arbitrary polygonal domains, would work if it was based on a correct algorithm (for example, it still works in our new algorithm).

Second, we fix the error of Yang et al. [25] in a not-so-trivial way. However, the complexities are not the same as before for all three types of optimal paths. Specifically, for computing a minimum-link shortest path, our corrected algorithm runs in $O(n \log^{3/2} n)$ time and $O(n \log n)$ space (with the help of the technique of Chen et. al. [2] to reduce a factor of $\log^{1/2} n$). For the other two types of optimal paths, however, the complexities have one more $O(n)$ factor, i.e., $O(n^2 \log^{3/2} n)$ time and $O(n^2 \log n)$ space.

Third, we further improve the algorithms in the way that the complexities only depend on h, in addition to $O(n)$. For computing a minimum-link shortest path, our algorithm runs in $O(n + h \log^{3/2} h)$ time and $O(n + h \log h)$ space. For computing other two types of optimal paths, our algorithm runs in $O(n + h^2 \log^2 h)$ time and $O(n + h^2 \log h)$ space. We also obtain data structures for one-point and two-point queries, and the results are summarized

Table 1 Summary of our data structures on one-point and two-point optimal path queries. Note that $\log^2 h \cdot 4^{\sqrt{\log h}} = O(h^\epsilon)$ for any $\epsilon > 0$. The "Prep. Time" stands for "Preprocessing Time".

Types of Paths		One-Point Queries	Two-Point Queries	
Min-Link Shortest	Prep. Time	$O(n + h\log^{3/2} h)$	$O(n + h^2\log^2 h)$	$O(n + h^2\log^2 h 4^{\sqrt{\log h}})$
	Space	$O(n + h\log h)$	$O(n + h^2\log^2 h)$	$O(n + h^2\log h 4^{\sqrt{\log h}})$
	Query Time	$O(\log n)$	$O(\log n + \log^2 h)$	$O(\log n)$
Shortest Min-Link	Prep. Time	$O(n + h^2\log^{3/2} h)$	$O(n + h^3\log^2 h)$	$O(n + h^3\log^2 h 4^{\sqrt{\log h}})$
	Space	$O(n + h^2\log h)$	$O(n + h^3\log^2 h)$	$O(n + h^3\log h 4^{\sqrt{\log h}})$
	Query Time	$O(\log n + \log^2 h)$	$O(\log n + \log^3 h)$	$O(\log n + \log^2 h)$
Minimum-Cost	Prep. Time	$O(n + h^2\log^{3/2} h)$	$O(n + h^3\log^2 h)$	$O(n + h^3\log^2 h 4^{\sqrt{\log h}})$
	Space	$O(n + h^2\log h)$	$O(n + h^3\log^2 h)$	$O(n + h^3\log h 4^{\sqrt{\log h}})$
	Query Time	$O(\log n + h\log h)$	$O(\log n + h\log^2 h)$	$O(\log n + h\log h)$
Minimum-Link	Prep. Time		$O(n + h^2\log^2 h)$	$O(n + h^2\log^2 h 4^{\sqrt{\log h}})$
	Space		$O(n + h^2\log^2 h)$	$O(n + h^2\log h 4^{\sqrt{\log h}})$
	Query Time		$O(\log n + \log^2 h)$	$O(\log n)$

in Table 1. Note that for two-point queries, we give two data structures for each problem with tradeoff between the preprocessing and the query time. We also consider the two-point query problem for minimum-link paths (without considering the lengths) since the problem was not studied before (but its one-point query problem has been solved, as discussed below).

Our results are particularly interesting when h is relatively small. For example if $h = O(n^{1/2-\epsilon})$ for any $\epsilon > 0$, then for finding a single optimal path of any type, our algorithm runs in $O(n)$ time, and our data structures for the minimum-link shortest path and minimum-link path queries are also optimal.

It is easy to see that the minimum-link shortest paths and the shortest minimum-link paths are special cases of minimum-cost paths, and we discuss them separately mainly because our results for the two special cases are generally better that those for the minimum-cost paths. In fact, as the cost function f is quite general, our algorithm for computing minimum-cost paths may find many applications. We give two examples below.

Polishchuk and Mitchell [19] gave an $O(kn\log^2 n)$ time algorithm for computing a shortest s-t path with at most k links for a given integer k, which improves the $O(kn^2)$ time algorithm in [24]. As indicated in [19], the problem can be solved using any algorithm that can find a minimum-cost path with the cost function defined as $f(a, b) = a$ if $b \leq k$ and $f(a, b) = \infty$ otherwise, where a and b are the length and the link distance of the path, respectively. Partially due to this reason, Polishchuk and Mitchell [19] already suspected that there is a misunderstanding on the algorithms of [2, 25] for computing minimum-cost paths. We thus confirm their suspicion. On the other hand, applying our new (and correct) algorithm for minimum-cost paths can solve the problem in $O(n + h^2\log^{3/2} h)$ time, which is faster than the algorithm in [19] when h is sufficiently small or when k is relatively large.

As a dual problem, finding a minimum-link s-t path with length at most a given value l was also studied in [24], where a worst-case $O(n^2(r + \log n))$ time algorithm was given with r as the number of extreme edges of $\mathcal{P}$. Note that $r \geq h$. The problem can also be solved using any minimum-cost path algorithm by defining the cost function as $f(a, b) = b$ if $a \leq l$ and $f(a, b) = \infty$ otherwise. Hence, applying our algorithm for minimum-cost paths can solve the problem in $O(n + h^2\log^{3/2} h)$, which clearly improves the algorithm of [24].

Other Related Work. If $\mathcal{P}$ is a simple rectilinear polygon (i.e., $h = 0$), then there always exists a rectilinear s-t path that has both the minimum length and the minimum link distance for any s and t in $\mathcal{P}$ [10, 11]. de Berg [10] built a data structure of $O(n \log n)$ size in $O(n \log n)$ time that can find such a path in $O(\log n)$ time for any two-point query. The preprocessing time and space were both reduced to $O(n)$ by Schuierer [21] (with $O(\log n)$ query time).

If $\mathcal{P}$ is a general rectilinear domain with $h \neq 0$, then there may not exist a rectilinear path that is both a minimum-link path and a shortest path [24]. The problems of finding only minimum-link paths or only shortest paths have been studied extensively. Imai and Asano [12] presented an $O(n \log n)$ time and space algorithm for finding a minimum-link s-t path in $\mathcal{P}$, and the space was reduced to $O(n)$ [9, 16, 20]. Recently, Mitchell et al. [17] proposed an $O(n + h \log h)$ time and $O(n)$ space algorithm for the problem, after $\mathcal{P}$ is triangulated (which can be done in $O(n \log n)$ time or $O(n + h \log^{1+\epsilon} h)$ time for any $\epsilon > 0$ [1]). The algorithms in [9, 16, 17] also construct an $O(n)$ size data structure that can answer each one-point minimum-link path query in $O(\log n)$ time.

For computing shortest s-t paths in $\mathcal{P}$, Clarkson et al. [7] gave an algorithm of $O(n \log^2 n)$ time and $O(n \log n)$ space, and as a tradeoff between time and space, they modified their algorithm so that it runs in $O(n \log^{3/2} n)$ time and space [8]. Wu et al. [23] proposed an $O(n \log r + r^2 \log r)$ time algorithm, where r is the number of extreme edges of $\mathcal{P}$, and the algorithm was later improved to $O(n \log r + r \log^{3/2} r)$ time [25]. Mitchell [14, 15] solved the problem in $O(n \log n)$ time and $O(n)$ space, and Chen and Wang [5, 6] reduced the time to $O(n + h \log h)$ after $\mathcal{P}$ is triangulated.

If $\mathcal{P}$ is an arbitrary polygonal domain (i.e., not rectilinear), then the results from [5, 6, 7, 8, 14, 15] are also applicable to finding arbitrary shortest paths under L_1 metric. In addition, the algorithms in [5, 6, 14, 15] can be used to compute an $O(n)$ size data structure so that each one-point L_1 shortest path query can be answered in $O(\log n)$ time. For two-point L_1 shortest path queries, Chen et al. [4] constructed a data structure of size $O(n^2 \log n)$ in $O(n^2 \log^2 n)$ time that can answer each query in $O(\log^2 n)$ time. Recently, Chen et al. [3] reduced the query time to $O(\log n)$ by building a data structure of size $O(n + h^2 \cdot \log h \cdot 4^{\sqrt{\log h}})$ in $O(n + h^2 \cdot \log^2 h \cdot 4^{\sqrt{\log h}})$ time.

To find a minimum-link s-t path between two points s and t in an arbitrary polygonal domain $\mathcal{P}$, Mitchell [18] gave an $O(E\alpha(n) \log^2 n)$ time algorithm, where $\alpha(n)$ is the inverse of Ackermann's function and E is the size of the visibility graph of $\mathcal{P}$ and $E = \Theta(n^2)$ in the worst case. The one-point query problem was also studied in [18].

In the following, unless otherwise stated, a path always refers to a rectilinear path.

Our Techniques. Given two points s and t in the rectilinear domain $\mathcal{P}$, to find an optimal s-t path, the algorithm of Yang et al. [25] first built a "path-preserving" graph G of size $O(n \log n)$ by using the idea of Clarkson et al. [7]. Then, it is shown that G contains an s-t path $\pi_G(s,t)$ that is homotopic to an optimal s-t path $\pi(s,t)$ in $\mathcal{P}$ with the same length, and further, $\pi(s,t)$ can be obtained from $\pi_G(s,t)$ by performing certain "dragging" operations. Motivated by this observation, Yang et al. [25] computed an optimal s-t path by applying Dijkstra's algorithm on G and simultaneously performing the dragging operations. We find a critical error in their way of applying Dijkstra's algorithm. We fix the error by using a "path-based" Dijkstra's algorithm and maintaining some additional information, and we prove that our algorithm is correct. Due to that we need to maintain more information on computing shortest minimum-link paths and minimum-cost paths, our algorithm for them runs slower than that for computing minimum-link shortest paths.

To further reduce the running time (for small h), our main idea is to use a reduced graph G_r of size $O(h \log h)$ instead of G. We show that G_r contains an s-t path $\pi_{G_r}(s,t)$ that

is homotopic to an optimal s-t path $\pi(s,t)$ in $\mathcal{P}$ with the same length, and further, $\pi(s,t)$ can be obtained from $\pi_{G_r}(s,t)$ by performing the dragging operations as in [25] and a new kind of operations, called *through-corridor-path generating operations*. The graph G_r is built based on a corridor structure of $\mathcal{P}$, which was used to find minimum-link paths in [17]. More specifically, we decompose $\mathcal{P}$ into $O(h)$ *junction rectangles* and $O(h)$ *corridors*. Each corridor is a simple rectilinear polygon. Although each corridor may have $\Theta(n)$ vertices, we show that we only need to consider at most four points of each corridor to build the graph G_r. To this end, we make use of the histogram partitions of rectilinear simple polygons [21].

For the one-point queries, Chen et al. [2] "insert" the query point t to the graph G to obtain a set $V_g(t)$ of $O(\log n)$ vertices (called "gateways") of G such that an optimal path can be obtained by performing the dragging operations from the gateways. We follow the similar scheme but on our reduced graph G_r, where only $O(\log h)$ gateways are necessary. Further, we also need to utilize the techniques of Schuierer [21] for simple rectilinear polygons.

For the two-point queries, the approach of [2] inserts both query points s and t to the graph G to obtain a set $V_g(s)$ of $O(\log n)$ gateways for s and a set $V_g(t)$ of $O(\log n)$ gateways for t, so that an optimal s-t path can be obtained by performing dragging operations from these gateways. The query time becomes $O(\log^2 n)$ because every pair of points (p,q) with $p \in V_g(s)$ and $q \in V_g(t)$ needs to be considered. We again use the same scheme but on the graph G_r with only $O(\log h)$ gateways for both s and t, and the query time is $O(\log n + \log^2 h)$. To reduce the query time to $O(\log n)$, we follow the scheme in [3] for two-point L_1 shortest path queries in arbitrary polygonal domains. The main idea is to build a larger graph by adding more vertices to G_r so that $O(\sqrt{\log h})$ gateways are sufficient for each query point.

The rest of the paper is organized as follows. We define some notation in Section 2. In Section 3, we review the algorithm given by Yang, Lee, and Wong [25] (we refer to it as the YLW algorithm), point out the error, and correct it. In Section 4, we further improve the algorithm for finding a single optimal s-t path. Due to the space limit, some details are omitted but can be found in the full paper [22]. Our data structures for the one-point and two-point path queries are also omitted and in the full paper.

2 Preliminaries

In this section, we define some concepts and notation. For any two points p and q of $\mathcal{P}$, if the line segment $\overline{pq}$ is in $\mathcal{P}$, then we say that p is *visible* to q. Consider a vertical line l and a point $p \in \mathcal{P}$. Let p' be the point on l whose y-coordinate is the same as that of p. We call p' the *horizontal projection* of p on l. If p is visible to p', then p is *horizontally visible* to l.

For any two points p and q, we use R_{pq} to denote the rectangle with $\overline{pq}$ as a diagonal. A path in $\mathcal{P}$ is *L-shaped* if it consists of a horizontal segment and a vertical segment (each of them may be empty). A path is *U-shaped* if it consists of three segments s_1, s_2, and s_3 such that s_1 and s_3 are on the same side of the line containing s_2. A path is called a *staircase path* if it does not contain a U-shaped subpath. Note that a staircase path is a shortest path.

Let $\mathcal{V}$ denote the set of all vertices of $\mathcal{P}$. We let $\mathcal{V}$ also include the two points s and t. We review a "path-preserving" graph $G(\mathcal{V})$ on $\mathcal{V}$ [7]. The vertex set of $G(\mathcal{V})$ consists of the points of $\mathcal{V}$ and *Steiner points* on some vertical lines, called *cut-lines*. The cut-lines and the Steiner points are defined as follows. Let v_m be the point of $\mathcal{V}$ with the median x-coordinate. The vertical line l_m through v_m is a *cut-line*. For each point $v \in \mathcal{V}$, if v is horizontally visible to l_m, then the horizontal projection of v on l_m is a *Steiner point*. Let $\mathcal{V}_l$ (resp., $\mathcal{V}_r$) be the points of $\mathcal{V}$ on the left (resp., right) side of l_m. The cut-lines and Steiner points on the left and right sides of l_m are defined on $\mathcal{V}_l$ and $\mathcal{V}_r$, recursively. We use a

binary tree $T(\mathcal{V})$ to represent the above recursive procedure, called *cut-line tree*. Each node $u \in T(\mathcal{V})$ corresponds to a cut-line $l(u)$ and a subset $V(u) \subseteq \mathcal{V}$. If u is the root, then $l(u)$ is l_m and $V(u) = \mathcal{V}$. The left and right subtrees of the root are defined recursively on $\mathcal{V}_l$ and $\mathcal{V}_r$. Hence, $T(\mathcal{V})$ has $O(n)$ nodes and each point of $\mathcal{V}$ can define a Steiner point on at most $O(\log n)$ cut-lines. Therefore, there are $O(n \log n)$ Steiner points in total.

The vertex set of $G(\mathcal{V})$ consists of all points of $\mathcal{V}$ and all Steiner points defined above. The edges of the graph are defined as follows. First, if a point $v \in \mathcal{V}$ defines a Steiner point v' on a cut-line, then $G(\mathcal{V})$ has an edge $\overline{vv'}$. Second, for any two adjacent Steiner points p_1 and p_2 on each cut-line, if they are visible to each other, then $G(\mathcal{V})$ has an edge $\overline{p_1p_2}$.

Clearly, $G(\mathcal{V})$ has $O(n \log n)$ nodes and edges. Each edge of $G(\mathcal{V})$ is either horizontal or vertical, whose weight is the length of the corresponding line segment. The graph $G(\mathcal{V})$ can be built in $O(n \log^2 n)$ time [7, 13, 25]. The following lemma was proved before [7, 13, 25].

▶ **Lemma 1** ([7, 13, 25]). *For any two points p and q in $\mathcal{V}$, if R_{pq} is empty (i.e., R_{pq} is in $\mathcal{P}$), then $G(\mathcal{V})$ contains a staircase path from p to q.*

For any path π in $\mathcal{P}$, let $L_1(\pi)$ denote its length and let $L_d(\pi)$ denote its link distance. For any two points a and b on π, if the context is clear, we often use $\pi(a, b)$ to denote the subpath of π between a and b. For any two points p and q in the plane, we say that q is to the *northeast* of p if q is in the first quadrant (including its boundary) with respect to p.

3 The YLW Algorithm and Our Correction

In this section, we first review the YLW algorithm [25] and then point out the error. Finally, we will fix the error. The YLW algorithm is essentially based on the following observation.

▶ **Lemma 2** (Yang et al. [25]). *For any optimal path π from s to t in $\mathcal{P}$, there is path π_G in $G(\mathcal{V})$ such that $L_1(\pi_G) = L_1(\pi)$ and π_G is homotopic to π (i.e., π_G can be continuously dragged to π without going outside of $\mathcal{P}$).*

We briefly review the proof of Lemma 2 because it will help to understand the algorithm.

Let π be any optimal path from s to t. It is shown (Lemma 2.1 [25]) that π can be divided into a sequence of staircase subpaths, and the two endpoints of each such subpath are in $\mathcal{V}$. Hence, it is sufficient to prove the lemma for any staircase subpath of π. Consider a staircase subpath $\pi(p, q)$ of π with p and q as the two endpoints. We further obtain a *pushed staircase path* as follows. Without loss of generality, we assume q is to the northeast of p and the segment of $\pi(p, q)$ incident to p is horizontal. We push the first vertical segment of $\pi(p, q)$ rightwards until either it hits a vertex of $\mathcal{V}$ or it becomes collinear with the second vertical segment of $\pi(p, q)$. In the latter case, we merge the two vertical segments and keep pushing the merged segment rightwards. In the former case, we push the next horizontal segment upwards in a similar way. The procedure stops until we arrive at the segment incident to q. Let π' denote the resulting path. Observe that $L_1(\pi') = L_1(\pi(p, q))$, π' is homotopic to $\pi(p, q)$, and π' is also a staircase path. π' is called a *pushed staircase path* [25]. Also note that each segment of π' contains at least one vertex of $\mathcal{V}$. There are eight types of pushed staircase paths from p to q depending on which quadrant of p the point q lies in and also depending on whether the first segment of the path incident to p is horizontal or vertical.

The vertices of $\mathcal{V}$ partition π' into subpaths. To prove the lemma, it is sufficient to show the following *claim*: for any subpath $\pi'(p', q')$ of π' between any two adjacent vertices p' and q' of $\mathcal{V}$ on π', there is a path $\pi_G(p', q')$ connecting p' and q' in $G(\mathcal{V})$ with the same length and the two paths are homotopic. Because every segment of π' contains at least one vertex

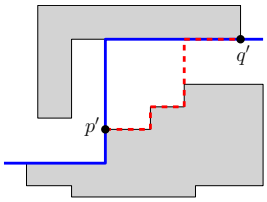


Figure 1 Converting $\pi_G(p', q')$ (the dashed red path) to $\pi'(p', q')$ (the solid blue path between p' and q').

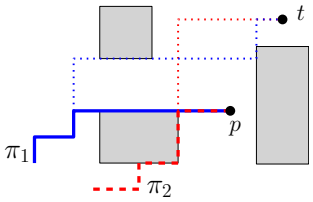


Figure 2 Illustrating a counter example for the YLW algorithm.

of $\mathcal{V}$, $\pi'(p', q')$ must be an L-shaped path. Without loss of generality, we assume q' is to the northwest of p'. If the rectangle $R_{p'q'}$ is empty, then by Lemma 1, the above claim is true. Otherwise, as shown in [25] (Lemma 4.5), there are some points of $\mathcal{V}$ in $R_{p'q'}$ that can be ordered as $p' = v_0, v_1, \ldots, v_t = q'$ with $R_{v_{i-1}v_i}$ being empty and v_i to the northwest of v_{i-1} for each $1 \leq i \leq t$, and further, $\pi'(p', q')$ is homotopic to the concatenation of $\overline{v_{i-1}v_i}$ for all $1 \leq i \leq t$. By Lemma 1, for each $1 \leq i \leq t$, $G(\mathcal{V})$ contains a staircase path connecting v_{i-1} and v_i and the path is in $R_{v_{i-1}v_i}$ (and thus is homotopic to $\overline{v_{i-1}v_i}$). Hence, by concatenating the staircase paths from v_{i-1} to v_i for all $i = 1, 2, \ldots, t$, we obtain a staircase path from p' to q' and the path is homotopic to $\pi'(p', q')$. Note that the staircase path has the same length as $\pi'(p', q')$ since $\pi'(p', q')$ is an L-shaped path. The above claim thus follows.

This proves Lemma 2. The proof actually constructs the path π_G in $G(\mathcal{V})$ corresponding to the optimal path π, and π_G is called a *target path*. Yang et al. [25] also showed that π can be obtained from π_G by applying certain *dragging* operations during searching the graph $G(\mathcal{V})$. Instead of describing the details of the operation (refer to our full paper or [25]), we give some intuition on how π can be obtained from π_G by using the dragging operations. Based on the above constructive proof for Lemma 2, we only need to show that for each L-shaped path $\pi'(p', q')$, it can be obtained from the corresponding staircase path $\pi_G(p', q')$ in $G(\mathcal{V})$. Without loss of generality, we assume that q' is to the northeast of p' and the segment incident to p' in $\pi'(p', q')$ is vertical. Because $\pi_G(p', q')$ is homotopic to $\pi'(p', q')$, we can convert $\pi_G(p', q')$ to $\pi'(p', q')$ as follows (e.g., see Fig. 1). Starting from p', for each horizontal segment of $\pi_G(p', q')$, drag it upwards until either it hits the horizontal segment of $\pi'(p', q')$ or it becomes collinear with the next horizontal segment of $\pi_G(p', q')$. In the former case, we have obtained $\pi'(p', q')$. In the latter case, we continue to drag the new horizontal segment upwards in the same way as before.

The YLW algorithm applies Dijkstra's algorithm using the measure vector $(L_1(\pi), L_d(\pi))$ for a path π. Initially, all vertices of $G(\mathcal{V})$ are in a priority queue Q with measure vectors (∞, ∞) except that the measure vector for s is $(0, 0)$. While Q is not empty, the algorithm removes from Q the vertex p with the smallest measure vector (lexicographically) and advance the paths stored at p to each of p's neighbor q by the dragging operations. Let $\pi(s, q)$ be a path obtained for q. There may be other paths already stored at q and the types of the last staircase subpaths of these paths are also stored (recall that there are eight types of pushed staircase subpaths). The YLW algorithm relies on the following two rules to determine whether the new path $\pi(s, q)$ should be stored at q, and if yes, whether some paths stored at q should be removed. Let $\pi'(s, q)$ be any path that has already been stored at q.

Rule(a) If the measure vectors of $\pi(s, q)$ and $\pi'(s, q)$ are not the same, then discard the one whose measure vector is strictly larger.

Rule(b) If $\pi(s, q)$ and $\pi'(s, q)$ have the same measure vector and of the same type, compare their last segments. If they overlap, discard the path whose last segment is longer.

It is claimed in [25] that once the point t is processed, among all paths stored at t, the one with the smallest measure vector is an optimal s-t path.

The Error. We find that the algorithm is not correct, mainly due to Rule(a). Figure 2 illustrates a counterexample. Assume that both π_1 and π_2 are paths from s to p with $L_1(\pi_1) = L_1(\pi_2)$ and $L_d(\pi_1) + 1 = L_d(\pi_2)$. Thus, the measure vector of π_1 is strictly smaller than that of π_2. According to Rule(a), we should discard π_2. Observe that we can obtain an s-t path from s to t using π_2 without any extra link. However, to obtain an s-t path using π_1, we need at least two more links. Therefore, π_2 can lead to a better s-t path than π_1, and thus, we should not discard π_2. Notice that the reason this happens is that although the measure vector of π_1 is strictly smaller than that of π_2, the last segment of π_2 is shorter than that of π_1 (and thus it may be "freely" dragged upwards higher than that of π_1).

In fact, the most essential reason for this error to happen might be the following. If π is a shortest s-t path, then for any two points p and q in π, the subpath $\pi(p, q)$ of π between p and q is also a shortest path from p to q. However, this may not be the case for minimum-link paths. Namely, if π is a minimum-link s-t path, then it is possible that for two points p and q in π, $\pi(p, q)$ is not a minimum-link path from p to q. Due to this reason, one can verify that the $O(nt + n \log n)$ time algorithm given by Yang et al. [24] for computing optimal paths is not correct either. Indeed, the approach in [24] also applies Dijkstra's algorithm on a graph to search the optimal paths using the measure vectors like $(L_1(\pi), L_d(\pi))$.

3.1 Our New Algorithm

To fix the error, we need to fix Rule(a). We first consider the minimum-link shortest paths. We replace Rule(a) by the following Rule(a_1), but still keep Rule(b). (Recall that $\pi'(s, q)$ denotes any path that has already been stored at q.)

Rule(a_1) Let π_1 be one of $\pi'(s, q)$ and $\pi(s, q)$, and π_2 the other. If $L_1(\pi_1) < L_1(\pi_2)$, or $L_1(\pi_1) = L_1(\pi_2)$ but $L_d(\pi_1) \leq L_d(\pi_2) - 2$, then we discard π_2.

By Rule(a_1), we may need to store two paths π_1 and π_2 at q even if the measure vector of one path is strictly smaller than that of the other, in which case $L_1(\pi_1) = L_1(\pi_2)$ and $L_d(\pi_1) = L_d(\pi_2) \pm 1$. Hence, unlike the YLW algorithm, each vertex q of $G(\mathcal{V})$ may store paths with different measure vectors. Therefore, we cannot apply the same "vertex-based" Dijkstra's algorithm as before. Instead, we propose a "path-based" Dijkstra's algorithm. Roughly speaking, we will process individual paths instead of vertices. Specifically, in the beginning there is only one path from s to s itself in the priority queue Q. In general, as long as Q is not empty, we remove from Q the path π with the smallest measure vector. Assume that the endpoint of π is p. Then, we advance π from p to each of p's neighbors q. If $\pi(s, q)$ is stored at q by our rules (i.e., both Rule(a_1) and Rule(b)), then we (implicitly) insert $\pi(s, q)$ to Q. The algorithm stops once Q is empty. Since we process paths following the increasing measure order, the algorithm will eventually stop. Finally, among all paths stored at t, we return the one with the smallest measure as the optimal solution. The correctness of the algorithm is proved in the full paper.

In terms of the running time, the YLW algorithm maintains at most eight paths at each vertex p of $G(\mathcal{V})$. To see this, due to Rule(b), for each type of staircase paths, p maintains at most one path. In our new algorithm, the paths maintained at p always have the same length but their link distances differ by at most one. Hence, again due to Rule(b), there are at most sixteen paths maintained at p. Clearly, this does not affect both the time and

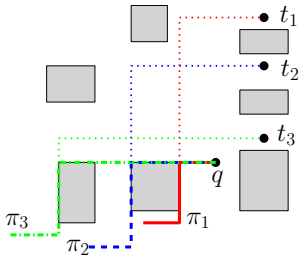


Figure 3 Illustrating an example on why we need Rule(a_2).

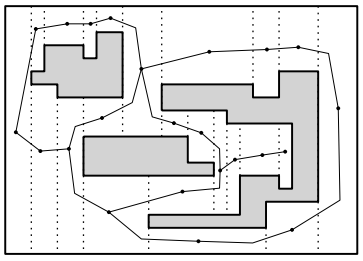


Figure 4 Illustrating the vertical visibility decomposition $VD(\mathcal{P})$ and its dual graph G_{vd}.

the space complexities of the algorithm asymptotically. Thus, the algorithm still runs in $O(n \log^2 n)$ time and $O(n \log n)$ space, as the YLW algorithm.

In addition, using another path-preserving graph $G^*(\mathcal{V})$ of $O(n \log^{1/2} n)$ vertices and $O(n \log^{3/2} n)$ edges [8], Yang et al. [25] proposed another $O(n \log^{3/2} n)$ time and space algorithm (see Section 4.2 of [25]). Further, Chen et al. [2] reduced the space of the algorithm to $O(n \log n)$ with the same $O(n \log^{3/2} n)$ time (similar technique was also used in [4]). By applying the techniques of both [25] and [2] to our new method, we can also obtain an algorithm of $O(n \log^{3/2} n)$ time and $O(n \log n)$ space. We omit the details.

We proceed on the problem of finding a minimum-cost s-t path. Recall that we have a cost function f. For any path π, we use $f(\pi)$ to denote the cost of the path. Our algorithm is the same as above with the following changes. First, the paths π in the priority Q are prioritized by $f(\pi)$. Second, we replace both Rule(a_1) and Rule(b) by the following rule.

Rule(a_2) Let π_1 be one of $\pi'(s,q)$ and $\pi(s,q)$, and π_2 the other. If the last segments of π_1 and π_2 are exactly the same and $f(\pi_1) \leq f(\pi_2)$, then we discard π_2.

We give some intuition on why we use the above rule. Refer to Fig. 3, where there are three paths π_1, π_2, and π_3 from s to q. Let s_i be the last segment of π_i for each $1 \leq i \leq 3$, and we assume that they overlap with $|s_1| < |s_2| < |s_3|$, where $|s_i|$ is the length of each s_i. We also assume that $L_d(\pi_1) = L_d(\pi_2) = L_d(\pi_3)$ and $L_1(\pi_1) > L_1(\pi_2) > L_1(\pi_3)$. In this case, we have to keep all three paths because any of them may lead to the best path from s to t. For example, for each $1 \leq i \leq 3$, the path π_i may lead to the best path from s to t_i. One can generalize the example so that a total of $\Omega(n)$ paths may need to be stored at p. However, $O(n)$ is the upper bound since the last segment of each such path starts from a different vertex of $G(\mathcal{V})$ in the horizontal line through q and there are $O(n)$ such vertices. For this reason, their are $O(n^2 \log n)$ paths stored in all $O(n \log n)$ vertices of $G(\mathcal{V})$. Hence, the running time of the algorithm becomes $O(n^2 \log^2 n)$ and the space becomes $O(n^2 \log n)$. As for the minimum-link shortest paths, by using the graph $G^*(\mathcal{V})$ and the techniques in [2, 25], we can reduce the running time by a factor of $\sqrt{\log n}$. We omit the details.

For computing a shortest minimum-link s-t path, we use a similar algorithm as above but with the following changes. First, we use the measure vector $(L_d(\pi), L_1(\pi))$ instead. Second, we use the following rule, which is similar to Rule(a_2). The complexities are the same as the above for minimum-cost paths.

Rule(a_3) Let π_1 be one of $\pi'(s,q)$ and $\pi(s,q)$, and π_2 the other. If the last segments of π_1 and π_2 are exactly the same and the measure vector of π_1 is no larger than that of π_2, then we discard π_2.

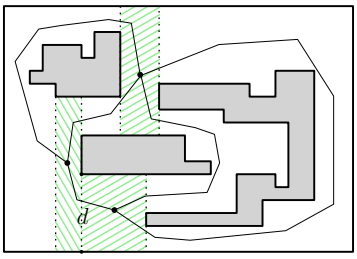


Figure 5 Illustrating the corridor structure and the corridor graph G_{cor} of three vertices. There are three junction rectangles, which are highlighted. Each connected white region is a corridor, which corresponds to an edge of G_{cor}. The diagonal d forms a degenerated corridor.

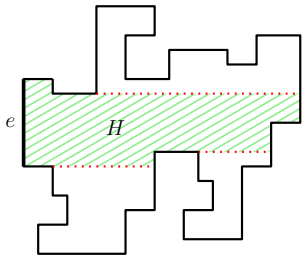


Figure 6 Illustrating the maximal histogram H, which has three windows shown with (red) dotted segments.

4 The Improved Algorithm

We further improve our algorithm, so that in addition to $O(n)$, the complexities of our improved algorithm only depend on h, i.e., the number of holes of $\mathcal{P}$. We first review the corridor structure of $\mathcal{P}$ [17] and the histogram partitions of rectilinear simple polygons [21].

The Corridor Structure of $\mathcal{P}$. For ease of exposition, we make a general position assumption that no two edges of $\mathcal{P}$ are collinear. The *vertical visibility decomposition* of $\mathcal{P}$, denoted by $VD(\mathcal{P})$, is obtained by extending each vertical edge of $\mathcal{P}$ until it hits the boundary of $\mathcal{P}$. Each cell of $VD(\mathcal{P})$ is a rectangle. Each extension segment is called a *diagonal* of $VD(\mathcal{P})$.

Let G_{vd} be the dual graph of $VD(\mathcal{P})$ (see Fig. 4), i.e., each node of G_{vd} corresponds to a cell of $VD(\mathcal{P})$ and two nodes have an edge if the corresponding cells share an edge. Based on G_{vd}, we obtain a *corridor graph* G_{cor} as follows. First, we keep removing every degree-one node from G_{vd} along with its incident edge until no such nodes remain. Second, we keep contracting every degree-two node from G_{vd} (i.e., remove the node and replace its two incident edges by a single edge) until no such nodes remain. The graph thus obtained is G_{cor}, which has $O(h)$ nodes and $O(h)$ edges [17]. See Fig. 5. The cells of $VD(\mathcal{P})$ corresponding to the nodes of G_{cor} are called *junction rectangles*. If we remove all junction rectangles from $\mathcal{P}$, each connected region is a simple rectilinear polygon, which is called a *corridor*. Each corridor has two diagonals each of which is on a vertical side of a junction rectangle, and we call them the *doors* of the corridor. For convenience, if a diagonal d bounds two junction rectangles (see Fig. 5), then we consider d itself as a "degenerate" corridor whose two doors are both d. With the degenerated corridors, each vertex of $\mathcal{P}$ lies in a unique corridor.

The decomposition $VD(\mathcal{P})$ can be computed in $O(n + h\log^{1+\epsilon} h)$ time for any $\epsilon > 0$ [1]. After $VD(\mathcal{P})$ is known, the corridor structure of $\mathcal{P}$ can be obtained in $O(n)$ time.

The Histogram Partitions. The histogram partition is a decomposition of a simple rectilinear polygon [21]. We will need to build the histogram partitions on the corridors of $\mathcal{P}$.

A simple rectilinear polygon H is called a *histogram* if its boundary can be divided into an x- or y-monotone chain and a single line segment, which is called the base of H.

Consider a simple rectilinear polygon Q (e.g., a corridor $\mathcal{C}$ of the corridor structure of $\mathcal{P}$) and let e be an edge of Q (e.g., a door of $\mathcal{C}$). A histogram partition of Q with respect to e, denoted by $\mathcal{H}(Q, e)$, is defined as follows. Let H be the *maximal histogram* with base

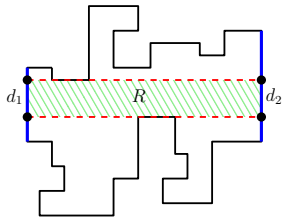


Figure 7 Illustrating an open corridor: the canal R and the two bridges are highlighted. The four points on the two doors are backbone points.

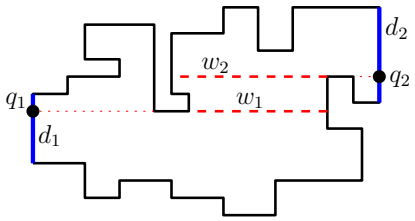


Figure 8 Illustrating a closed corridor. The points q_1 and q_2 are backbone points on d_1 and d_2, respectively.

e in Q, i.e., there is no other histogram in Q with base e that can properly contain it (see Fig. 6). A *window* of H is a maximal segment on the boundary of H that is contained in the interior of Q except its two endpoints. For each window w of H, it divides H into two subpolygons, and we let $Q(w)$ be the one that does not contain e. If H does not have a window, we are done with the histogram partition of Q. Otherwise, for each window w, we perform the above partition on $Q(w)$ recursively with respect to w.

For any points p and q in Q, it is known that there exists a path from p to q in Q that is both a shortest path and a minimum-link path [10, 11, 21], and we call it a *smallest path*.

4.1 A Reduced Path Preserving Graph

Recall that our algorithm in Section 3 use a graph $G(\mathcal{V})$, which is built on the vertices of $\mathcal{V}$ and has $O(n \log n)$ nodes and edges. In this section, as a major tool for reducing the complexities of our algorithm, we propose a *reduced graph* of $O(h \log h)$ nodes and edges. We first introduce a set $\mathcal{B}$ of $O(h)$ *backbone* points on the doors of the corridors of $\mathcal{P}$.

The Backbone Points. Consider a corridor $\mathcal{C}$ of the corridor structure of $\mathcal{P}$. Let d_1 and d_2 be the two doors of $\mathcal{C}$, which are both vertical. The region of $\mathcal{C}$ excluding the two doors is called the *interior* of $\mathcal{C}$. If there exist a point $p_1 \in d_1$ and a point $p_2 \in d_2$ such that $\overline{p_1p_2}$ is horizontal and $\overline{p_1p_2}$ in $\mathcal{C}$ then we say that $\mathcal{C}$ is an *open corridor*; otherwise, it is *closed*.

Consider an open corridor $\mathcal{C}$ (see Fig. 7). Let p_1 and p_2 be the points defined above. Imagine that we drag $\overline{p_1p_2}$ vertically upwards (resp., downwards) until we hit a vertex of $\mathcal{C}$, then the current locations of p_1 and p_2 are two *backbone points*. In this way, each door of $\mathcal{C}$ has two backbone points. Clearly, the rectangle R with the four backbone points as the vertices is in $\mathcal{C}$ and we call R the *canal* of $\mathcal{C}$. The two horizontal edges of R are called *bridges* of $\mathcal{C}$. Further, the top edge of R is the *upper* bridge and the bottom edge is the *lower* bridge.

If $\mathcal{C}$ is a degenerate corridor, which is a single diagonal d, then $\mathcal{C}$ is also an open corridor and the upper (resp., lower) bridge is degenerated to the upper (resp., lower) endpoint of d.

Next, we consider the case where $\mathcal{C}$ is closed (see Fig. 8). Let H_1 be the maximal histogram in $\mathcal{C}$ with base d_1. As $\mathcal{C}$ is closed, H_1 has a window w_1 that *separates* d_1 from d_2, that is, w_1 divides $\mathcal{C}$ into two sub-polygons that contain d_1 and d_2, respectively. By the definition of windows, if we extend w_1 to d_1, the extension will hit d_1 at a point, denoted by q_1, before it goes out of $\mathcal{C}$. Similarly, we define H_2, w_2, and q_2, with respect to the other door d_2. The two points q_1 and q_2 are *backbone points* of $\mathcal{C}$.

The above defines two backbone points on each door of every open corridor and one backbone point on each door of every closed corridor. Let $\mathcal{B}$ denote the set of all such backbone points. Since there are $O(h)$ corridors, the size of $\mathcal{B}$ is $O(h)$.

The Reduced Graph $G(\mathcal{B})$. In the sequel, we introduce the reduced graph, denoted by $G(\mathcal{B})$. We first consider the case where both s and t are in junction rectangles. With a little abuse of notation, we let $\mathcal{B}$ also contain both s and t.

We build the graph $G(\mathcal{B})$ with respect to the points of $\mathcal{B}$ in the same way as $G(\mathcal{V})$ with respect to $\mathcal{V}$ in Section 3. Hence, $G(\mathcal{B})$ has $O(h \log h)$ vertices and $O(h \log h)$ edges. In addition, we add the following $O(h)$ edges to $G(\mathcal{B})$. Consider a closed corridor $\mathcal{C}$ with the two backbone points q_1 and q_2 on its two doors. Note that q_1 and q_2 are also two vertices in $G(\mathcal{B})$. We add to $G(\mathcal{B})$ an edge $e(q_1, q_2)$ to connect q_1 and q_2 with length equal to $L_1(\pi(\mathcal{C}, q_1, q_2))$, where $\pi(\mathcal{C}, q_1, q_2)$ is a shortest path from q_1 to q_2 in $\mathcal{C}$. We call $e(q_1, q_2)$ a *corridor edge* of $G(\mathcal{B})$, and call $\pi(\mathcal{C}, q_1, q_2)$ a *corridor path* of $\mathcal{C}$. We do this for all closed corridors. This completes the construction of $G(\mathcal{B})$. Since there are $O(h)$ corridors, $G(\mathcal{B})$ has $O(h)$ corridor edges. For differentiation, other edges of $G(\mathcal{B})$ that are not corridor edges are called *ordinary edges*. Hence, $G(\mathcal{B})$ has $O(h \log h)$ edges in total. Note that every path $\pi_{G(\mathcal{B})}$ in $G(\mathcal{B})$ corresponds to a path π in $\mathcal{P}$ with the same length in the sense that if the path $\pi_{G(\mathcal{B})}$ contains a corridor edge, then π contains the corresponding corridor path.

The following lemma is analogous to Lemma 2, but on the reduced graph $G(\mathcal{B})$. It explains why the graph $G(\mathcal{B})$ can help to find optimal paths.

▶ **Lemma 3.** *There exists a path $\pi_{G(\mathcal{B})}$ in $G(\mathcal{B})$ from s to t that is homotopic to an optimal s-t path and the two paths have the same length; we call $\pi_{G(\mathcal{B})}$ a* target path*.*

Lemma 3 implies that a shortest s-t path in $G(\mathcal{B})$ is a shortest s-t path in $\mathcal{P}$. Hence, $G(\mathcal{B})$ is indeed a "path-preserving" graph. We can compute $G(\mathcal{B})$ in $O(n + h \log^2 h)$ time using the previous algorithm [7, 13, 25] as well as a so-called *reduced domain* $\mathcal{P}_r$, which consists of all junction rectangles and the canals of all open corridors of $\mathcal{P}$. The details are omitted.

4.2 Computing an Optimal Path

In this section, we compute an optimal s-t path using $G(\mathcal{B})$. We will show that an optimal s-t path can be computed by applying the dragging operations as in [25] on the ordinary edges of $\pi_{G(\mathcal{B})}$ and applying a new kind of operations, called *through-corridor-path generating operations*, on corridor edges of $\pi_{G(\mathcal{B})}$, where $\pi_{G(\mathcal{B})}$ is a target path defined in Lemma 3.

The algorithmic scheme is similar to that in Section 3.1. When we advance the searching process through an ordinary edge, we perform a dragging operation as in [25]. If we are advancing along a corridor edge, then we apply a through-corridor-path generating operation, which is introduced in the following. We first review some results from Schuierer [21].

Consider a closed corridor $\mathcal{C}$. Let d be a door of $\mathcal{C}$ and let q be the backbone point on d. Recall that q is an extension of a window w of the maximal histogram H in $\mathcal{C}$ with base d.

Let p be a point in $\mathcal{C}$. Following the terminology in [21], a rectilinear path from p to a point on d is called an *admissible path* if the last link is orthogonal to d. A *minimum-link admissible path* from p to d is an admissible path from p to any point of d with the smallest number of links, and we use $L_d(p, d)$ to denote the number of links in the path. Let $I_1(p, d)$ (resp., $I_2(p, d)$) denote the set of points on d that can be reached by p with an admissible path of at most $L_d(p, d)$ (resp., $L_d(p, d) + 1$) links (e.g., see Fig. 9). It is known that each of $I_1(p, d)$ and $I_2(p, d)$ is an interval of d, and $I_1(p, d) \subseteq I_2(p, d)$ [21]. Further, if p is not horizontally visible to d, then both intervals have q as one of their endpoints. By using the histogram partition $\mathcal{H}(\mathcal{C}, d)$, Schuierer [21] built a data structure in $O(|\mathcal{C}|)$ time such that given any point $p \in \mathcal{C}$, the two intervals $I_1(p, d)$ and $I_2(p, d)$ can be determined in $O(\log |\mathcal{C}|)$ time. With a little abuse of notation, we also use $\mathcal{H}(\mathcal{C}, d)$ to refer to the above data structure.

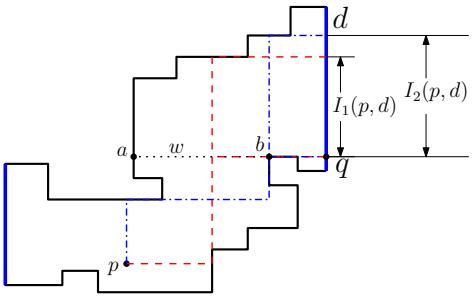


Figure 9 Illustrating the two intervals $I_1(p,d)$ and $I_2(p,d)$, where $L_d(p,d) = 3$ and $\overline{ab}$ is the window w. The two blue segments are doors of the corridor.

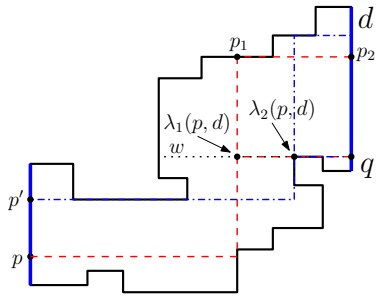


Figure 10 Illustrating the two points $\lambda_1(p,d)$ and $\lambda_2(p,d)$ on the window w. p' is also a backbone point.

Suppose p is a point on the other door of $\mathcal{C}$ than d (so p is not horizontally visible to d). Then, $I_1(p,d)$ is uniquely determined by a point, denoted by $\lambda_1(p,d)$, on the window w in the following way [21] (e.g., see Fig. 10). Recall that d is vertical and thus w is horizontal. Without loss of generality, assume that the histogram H is locally above w and locally on the left of d. We shoot a ray from $\lambda_1(p,d)$ upwards until a point p_1 on the boundary of $\mathcal{C}$ and then we project p_1 perpendicular to d and let p_2 be the projection point. The point p_2 is the other endpoint of the interval $I_1(p,d)$, i.e., $I_1(p,d) = \overline{qp_2}$. Note that p_2 is above q. Let $I_1'(p,d)$ denote the segment $\overline{\lambda_1(p,d)q}$, which is on the extension of the window w. We can also understand the two intervals $I_1(p,d)$ and $I_1'(p,d)$ in the following way. There exists an admissible path of $L_d(p,d)$ links from p to q, denoted by $\pi_1(\mathcal{C},p,q)$, which is actually a *smallest* path from p to q, and its last link is $I_1'(p,d)$; for any point $q' \in I_1(p,d)$, by dragging the last segment of $\pi_1(\mathcal{C},p,q)$ upwards until q', we can obtain an admissible path of $L_d(p,d)$ links from p to q'. The data structure $\mathcal{H}(\mathcal{C},d)$ can also report $\lambda_1(p,d)$ in $O(\log n)$ time and the path $\pi_1(\mathcal{C},p,q)$ can be output in additional time linear in the link distance of the path.

The interval $I_2(p,d)$ is uniquely determined by a point $\lambda_2(p,d)$ on the window w in the similar way as above. Similarly, we define $I_2'(p,d)$ and the corresponding admissible path of $L_d(p,d)+1$ links from p to q whose last link is $I_2'(p,d)$, denoted by $\pi_2(\mathcal{C},p,q)$, which is a *shortest* path (but not necessarily a smallest path) from p to q in $\mathcal{C}$ [21]. Similarly, the data structure $\mathcal{H}(\mathcal{C},d)$ can also report $\lambda_2(p,d)$ in $O(\log n)$ time and the path $\pi_2(\mathcal{C},p,q)$ can be output in additional time linear in the link distance of the path.

In the following, we introduce our through-corridor-path generating operations for advancing paths along corridor edges in our algorithm for searching the graph $G(\mathcal{B})$.

Consider a corridor edge $e(q_1,q_2)$ connecting two vertices q_1 and q_2 of $G(\mathcal{B})$. Note that q_1 and q_2 are two backbone points that are on the two doors d_1 and d_2 of a closed corridor $\mathcal{C}$, respectively. Consider a path $\pi(s,q_1)$ from s to q_1 maintained by our algorithm. Suppose we want to advance $\pi(s,q_1)$ from q_1 to q_2 along the corridor edge $e(q_1,q_2)$. We perform the following through-corridor-path generating operation that will extend $\pi(s,q_1)$ from q_1 to q_2 to obtain a path $\pi(s,q_2)$ from s to q_2.

Recall that q_1 is an extension of a window w_1 of the maximal histogram H_1 in $\mathcal{C}$ with base d_1. Hence, w_1 divides $\mathcal{C}$ into two sub-polygons that contain d_1 and d_2, respectively. Without loss of generality, we assume that the sub-polygon containing d_2 is locally above w_1. We also assume that $\mathcal{C}$ is locally on the right of d_1 (e.g., see Fig. 11).

Let α be the last segment of $\pi(s,q_1)$ (i.e., the one incident to q_1) and let p be the other endpoint of α than q_1. Suppose we have already built the data structure $\mathcal{H}(\mathcal{C},d_2)$ for $\mathcal{C}$ with respect to the door d_2. Depending on whether α is horizontal or vertical, there are two cases.

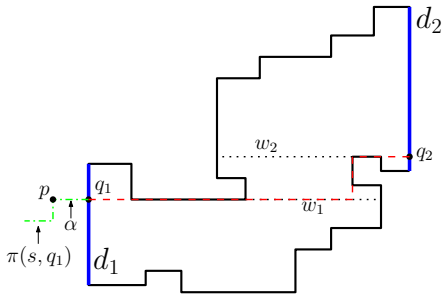


Figure 11 Illustrating the through-corridor-path generating operation for the case where α is horizontal. The path $\pi_1(\mathcal{C}, q_1, q_2)$ are shown with red dashed segments.

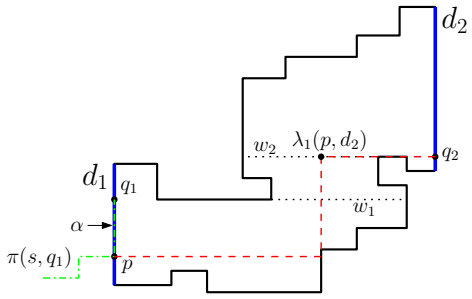


Figure 12 Illustrating the through-corridor-path generating operation for the case where α is vertical and p is on d_1 below q_1. The smallest path $\pi_{opt}(\mathcal{C}, q_1, q_2)$ are shown with red dashed segments. Note that $I_1'(p, d_2) = \overline{\lambda_1(p, d_2)q_2}$.

1. If α is horizontal (e.g., see Fig. 11), then p must be to the left of q_1 since $\mathcal{C}$ is locally on the right side of d_1. In this case, we use $\mathcal{H}(\mathcal{C}, d_2)$ to determine the path $\pi_1(\mathcal{C}, q_1, q_2)$ (whose last link is $I_1'(q_1, d_2)$) and concatenate it with $\pi(s, q_1)$ to obtain $\pi(s, q_2)$. We also compute the number of links of $\pi(s, q_2)$ and its length, and store them at q_2. Note that $L_1(\pi(s, q_1))$ and $L_d(\pi(s, q_1))$ are already stored at q_1.
2. If α is vertical, then depending on whether p is above q_1, there are two subcases.
   a. If p is above q_1, then we use the same approach as above to obtain $\pi(s, q_2)$. Note that in this case the path makes a turn at q_1 while there is no turn at q_1 in the above case.
   b. If p is below q_1, then depending on whether p is on d_1, there are further two subcases.
      i. If p is not on d_1, then we use the same approach as above to obtain $\pi(s, q_2)$.
      ii. If p is on d_1, this is the trickiest case. Let $\pi_{opt}(\mathcal{C}, p, q_2)$ denote the smallest path from p to q_2 in $\mathcal{C}$ whose last link is $I_1'(p, d_2)$ (e.g., see Fig. 12). $\pi_{opt}(\mathcal{C}, p, q_2)$ can also be determined in $O(\log n)$ time by the data structure $\mathcal{H}(\mathcal{C}, d_2)$. We obtain $\pi(s, q_2)$ by concatenating $\pi_{opt}(\mathcal{C}, p, q_2)$ with the subpath of $\pi(s, q_1)$ between s and p.

As a summary, to obtain $\pi(s, q_2)$, if Case 2(b)ii happens, then we connect the subpath of $\pi(s, q_1)$ between s and p with $\pi_{opt}(\mathcal{C}, p, q_2)$; otherwise, we connect $\pi(s, q_1)$ with $\pi_1(\mathcal{C}, q_1, q_2)$. Note that in either case the last link of $\pi(s, q_2)$ is $I_1'(q_1, d_2)$. In either case, let π' be the subpath of $\pi(s, q_2)$ contained in $\mathcal{C}$. With the histogram partition $\mathcal{H}(\mathcal{C}, d_2)$, we can obtain $L_1(\pi')$ and $L_d(\pi')$ as well as the first and last links of π' in $O(\log n)$ time (the path π' can be output in additional $O(L_d(\pi'))$ time). Hence, we can compute $L_1(\pi(s, p_2))$ and $L_d(\pi(s, p_2))$ as well as its last link in $O(\log n)$ time, without explicitly computing the path π'. Therefore, the through-corridor-path generating operation can be performed in $O(\log n)$ time.

As discussed before, our algorithm works in the same way as the one in Section 3 except that we apply through-corridor-path generating operations on corridor edges of $G(\mathcal{B})$ instead of the dragging operations. We can compute the histogram partitions for all closed corridors as the preprocessing for performing the through-corridor-path generating operations, and the total preprocessing time is $O(n)$ since the size of all corridors is $O(n)$. After the algorithm finishes, the path stored at t with the smallest measure is an optimal s-t path.

The above only discussed the case where both s and t are in junction rectangles. We can generalizes the approach to other cases if at least one of them is in a corridor. This is done by introducing *corridor-connection points* and adding a *beginning procedure* and a *concatenation procedure* to the algorithm. The details can be found in the full paper [22].

▶ **Theorem 4.** *We can compute a minimum-link shortest s-t path in $O(n + h\log^{3/2} h)$ time and $O(n + h\log h)$ space, and compute a shortest minimum-link s-t path or a minimum-cost s-t path in $O(n + h^2\log^{3/2} h)$ time and $O(n + h^2\log h)$ space.*

References

1. R. Bar-Yehuda and B. Chazelle. Triangulating disjoint Jordan chains. *International Journal of Computational Geometry and Applications*, 4(4):475–481, 1994.
2. D. Z. Chen, O. Daescu, and K. S. Klenk. On geometric path query problems. *International Journal of Computational Geometry and Applications*, 11(6):617–645, 2001.
3. D. Z. Chen, R. Inkulu, and H. Wang. Two-point L_1 shortest path queries in the plane. In *Proc. of the 30th Annual Symposium on Computational Geometry*, pages 406–415, 2014.
4. D. Z. Chen, K. S. Klenk, and H.-Y. T. Tu. Shortest path queries among weighted obstacles in the rectilinear plane. *SIAM Journal on Computing*, 29(4):1223–1246, 2000.
5. D. Z. Chen and H. Wang. A nearly optimal algorithm for finding L_1 shortest paths among polygonal obstacles in the plane. In *Proc. of the 19th European Symposium on Algorithms*, pages 481–492, 2011.
6. D. Z. Chen and H. Wang. L_1 shortest path queries among polygonal obstacles in the plane. In *Proc. of 30th Symp. on Theoretical Aspects of Computer Science*, pages 293–304, 2013.
7. K. Clarkson, S. Kapoor, and P. Vaidya. Rectilinear shortest paths through polygonal obstacles in $O(n\log^2 n)$ time. In *Proc. of the 3rd Annual Symposium on Computational Geometry*, pages 251–257, 1987.
8. K. Clarkson, S. Kapoor, and P. Vaidya. Rectilinear shortest paths through polygonal obstacles in $O(n\log^{2/3} n)$ time. Manuscript, 1988.
9. G. Das and G. Narasimhan. Geometric searching and link distance. In *Proc. of the 2nd Workshop of Algorithms and Data Structures*, pages 261–272, 1991.
10. M. de Berg. On rectilinear link distance. *Computational Geometry: Theory and Applications*, 1:13–34, 1991.
11. J. Hershberger and J. Snoeyink. Computing minimum length paths of a given homotopy class. *Computational Geometry: Theory and Applications*, 4(2):63–97, 1994.
12. H. Imai and T. Asano. Efficient algorithms for geometric graph search problems. *SIAM Journal on Computing*, 15(2):478–494, 1986.
13. D. T. Lee, C. D. Yang, and T. H. Chen. Shortest rectilinear paths among weighted obstacles. *International Journal of Computational Geometry and Applications*, 1(2):109–124, 1991.
14. J. S. B. Mitchell. An optimal algorithm for shortest rectilinear paths among obstacles. Abstracts of the *1st Canadian Conference on Computational Geometry*, 1989.
15. J. S. B. Mitchell. L_1 shortest paths among polygonal obstacles in the plane. *Algorithmica*, 8(1):55–88, 1992.
16. J. S. B. Mitchell, V. Polishchuk, and M. Sysikaski. Minimum-link paths revisited. *CGTA*, 47:651–667, 2014.
17. J. S. B. Mitchell, V. Polishchuk, M. Sysikaski, and H. Wang. An optimal algorithm for minimum-link rectilinear paths in triangulated rectilinear domains. In *Proc. of the 42nd International Colloquium on Automata, Languages and Programming*, pages 947–959, 2015.
18. J. S. B. Mitchell, G. Rote, and G. Woeginger. Minimum-link paths among obstacles in the plane. *Algorithmica*, 8:431–459, 1992.
19. V. Polishchuk and J. S. B. Mitchell. k-Link rectilinear shortest paths among rectilinear obstacles in the plane. In *Proc. of the 17th Canadian Conference on Computational Geometry (CCCG)*, pages 101–104, 2005.
20. M. Sato, J. Sakanaka, and T. Ohtsuki. A fast line-search method based on a tile plane. In *Proc. of the IEEE International Symposium on Circuits and Systems*, pages 588–597, 1987.

21 S. Schuierer. An optimal data structure for shortest rectilinear path queries in a simple rectilinear polygon. *International Journal of Computational Geometry and Applications*, 6:205–226, 1996.

22 H. Wang. Bicriteria rectilinear shortest paths among rectilinear obstacles in the plane. arXiv:1703.04466, 2017.

23 Y.-F. Wu, P. Widmayer, M. D. F. Schlag, and C. K. Wong. Rectilinear shortest paths and minimum spanning trees in the presence of rectilinear obstacles. *IEEE Transactions on Computers*, 36:321–331, 1987.

24 C. D. Yang, D. T. Lee, and C. K. Wong. On bends and lengths of rectilinear paths: A graph-theoretic approach. *Int. J. Comput. Geom. Appl.*, 02:61–74, 1992.

25 C. D. Yang, D. T. Lee, and C. K. Wong. Rectilinear path problems among rectilinear obstacles revisited. *SIAM Journal on Computing*, 24:457–472, 1995.

Quickest Visibility Queries in Polygonal Domains*†

Haitao Wang

Department of Computer Science, Utah State University, Logan, UT, USA
haitao.wang@usu.edu

Abstract

Let s be a point in a polygonal domain $\mathcal{P}$ of $h-1$ holes and n vertices. We consider the following *quickest visibility query* problem. Given a query point q in $\mathcal{P}$, the goal is to find a shortest path in $\mathcal{P}$ to move from s to *see* q as quickly as possible. Previously, Arkin et al. (SoCG 2015) built a data structure of size $O(n^2 2^{\alpha(n)} \log n)$ that can answer each query in $O(K \log^2 n)$ time, where $\alpha(n)$ is the inverse Ackermann function and K is the size of the visibility polygon of q in $\mathcal{P}$ (and K can be $\Theta(n)$ in the worst case). In this paper, we present a new data structure of size $O(n \log h + h^2)$ that can answer each query in $O(h \log h \log n)$ time. Our result improves the previous work when h is relatively small. In particular, if h is a constant, then our result even matches the best result for the simple polygon case (i.e., $h = 1$), which is optimal. As a by-product, we also have a new algorithm for the following *shortest-path-to-segment query* problem. Given a query line segment τ in $\mathcal{P}$, the query seeks a shortest path from s to all points of τ. Previously, Arkin et al. gave a data structure of size $O(n^2 2^{\alpha(n)} \log n)$ that can answer each query in $O(\log^2 n)$ time, and another data structure of size $O(n^3 \log n)$ with $O(\log n)$ query time. We present a data structure of size $O(n)$ with query time $O(h \log \frac{n}{h})$, which favors small values of h and is optimal when $h = O(1)$.

1998 ACM Subject Classification I.3.5 Computational Geometry and Object Modeling, F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases shortest paths, visibility, quickest visibility queries, shortest path to segments, polygons with holes

Digital Object Identifier 10.4230/LIPIcs.SoCG.2017.61

1 Introduction

Let $\mathcal{P}$ be a polygonal domain with $h-1$ holes and a total of n vertices, i.e., there is an outer simple polygon containing $h-1$ pairwise disjoint holes and each hole itself is a simple polygon. If $h = 1$, then $\mathcal{P}$ becomes a simple polygon. For any two points s and t in $\mathcal{P}$, a *shortest path* from s to t is a path in $\mathcal{P}$ connecting s and t with the minimum Euclidean length. Two points p and q are *visible* to each other if the line segment $\overline{pq}$ is in $\mathcal{P}$. For any point q in $\mathcal{P}$, its *visibility polygon* consists of all points of $\mathcal{P}$ visible to q, denoted by $Vis(q)$.

We consider the following *quickest visibility query* problem. Let s be a source point in $\mathcal{P}$. Given any point q in $\mathcal{P}$, the query asks for a path to move from s to *see* q as quickly as possible. Such a "quickest path" is actually a shortest path from s to all points of $Vis(q)$. The problem has been recently studied by Arkin et al. [1], who built a data structure of size $O(n^2 2^{\alpha(n)} \log n)$ that can answer each query in $O(K \log^2 n)$ time, where K is the size of $Vis(q)$. In this paper, we present a new data structure of $O(n \log h + h^2)$ size with $O(h \log h \log n)$ query time. Our result improves the previous work when h is relatively small. Interesting is that the query time is independent of K, which can be $\Theta(n)$ in the worst case. Our result is

* A full version of the paper is available at https://arxiv.org/abs/1703.03048.
† This research was supported in part by NSF under Grant CCF-1317143.

also interesting in that when $h = O(1)$, the data structure has $O(n)$ size and $O(\log n)$ query time, which matches the result for the simple polygon case [1] and is optimal.

As in [1], in order to solve the quickest visibility queries, we also solve a *shortest-path-to-segment query* problem (or *segment query* for short), which may have independent interest. Given any line segment τ in $\mathcal{P}$, the segment query asks for a shortest path from s to all points of τ. Arkin et al. [1] gave a data structure of size $O(n^2 2^{\alpha(n)} \log n)$ that can answer each query in $O(\log^2 n)$ time, and another data structure of size $O(n^3 \log n)$ with $O(\log n)$ query time. We present a new data structure of $O(n)$ size with $O(h \log \frac{n}{h})$ query time. Our result again favors small values of h and attains optimality when $h = O(1)$, which also matches the best result for the simple polygon case [1, 7].

Given the shortest path map of s, our quickest visibility query data structure can be built in $O(n \log h + h^2 \log h)$ time and our segment query data structure can be built in $O(n)$ time. Arkin et al.'s quickest visibility query data structure and their first segment query data structure can both be built in $O(n^2 2^{\alpha(n)} \log n)$ time, and their second segment query data structure can be built in $O(n^3 \log n)$ time [1].

Throughout the paper, whenever we talk about a query related to paths in $\mathcal{P}$, the query time always refers to the time for computing the path length, and to output the actual path, it needs additional time linear in the number of edges of the path by standard techniques.

1.1 Related Work

The traditional shortest path problem is to compute a shortest path to move from s to "reach" a query point. Each shortest path query can be answered in $O(\log n)$ time by using the *shortest path map* of s, denoted by $SPM(s)$, which is of $O(n)$ size. To build $SPM(s)$, Mitchell [14] gave an algorithm of $O(n^{3/2+\epsilon})$ time for any $\epsilon > 0$ and $O(n)$ space, and later Hershberger and Suri [10] presented an algorithm of $O(n \log n)$ time and space. If $\mathcal{P}$ is a simple polygon (i.e., $h = 1$), $SPM(s)$ can be built in $O(n)$ time, e.g., see [8].

For the quickest visibility queries, Arkin et al. [1] also built a "quickest visibility map" of $O(n^7)$ size in $O(n^8 \log n)$ time, which can answer each query in $O(\log n)$ time. In addition, Arkin et al. [1] gave a conditional lower bound on the problem by showing that the 3SUM problem on n numbers can be solved in $O(\tau_1 + n \cdot \tau_2)$ time, where τ_1 is the preprocessing time and τ_2 is the query time. Therefore, a data structure of $o(n^2)$ preprocessing time and $o(n)$ query time would lead to an $o(n^2)$ time algorithm for 3SUM.

In the simple polygon case (i.e., $h = 1$), better results are known. For the quickest visibility queries, Khosravi and Ghodsi [11] first proposed a data structure of $O(n^2)$ size that can answer each query in $O(\log n)$ time. Arkin et al. [1] gave an improved result and they built a data structure of $O(n)$ size in $O(n)$ time, with $O(\log n)$ query time. For the segment queries, Arkin et al. [1] built a data structure of $O(n)$ size in $O(n)$ time, with $O(\log n)$ query time. Chiang and Tamassia [7] achieved the same result for the segment queries and they also gave some more general results (e.g., when the query is a convex polygon).

Similar in spirit to the "point-to-segment" shortest path problem, Cheung and Daescu [6] considered a "point-to-face" shortest path problem in 3D and approximation algorithms were given for the problem.

1.2 Our Techniques

We first propose a decomposition $\mathcal{D}$ of $\mathcal{P}$ by $O(h)$ shortest paths from s to certain vertices of $SPM(s)$. The decomposition $\mathcal{D}$, whose size is $O(n)$, has $O(n)$ cells with the following three key properties. First, any segment τ in $\mathcal{P}$ can intersect at most $O(h)$ cells of $\mathcal{D}$. Second, for

each cell Δ of $\mathcal{D}$, $\tau \cap \Delta$ consists of at most two sub-segments of τ. Third, after $O(n)$ time preprocessing, for each sub-segment τ' of τ in any cell of $\mathcal{D}$, the shortest path from s to τ' can be computed in $O(\log n)$ time. With $\mathcal{D}$, we can easily answer each segment query in $O(h \log \frac{n}{h})$ time by a "pedestrian" algorithm.

To solve the quickest visibility queries, an observation is that the shortest path from s to see q is a shortest path from s to a *window* of $\mathit{Vis}(q)$, i.e., an extension of the segment $\overline{qu}$ for some reflex vertex u of $\mathcal{P}$. Hence, the query can be answered by calling segment queries on all $O(K)$ windows of $\mathit{Vis}(s)$. This leads to the $O(K \log^2 n)$ time query algorithm in [1].

If we follow the same algorithmic scheme and using our new segment query algorithm, then we would obtain an algorithm of $O(K \cdot h \cdot \log \frac{n}{h})$ time for the quickest visibility queries. We instead present a "smarter" algorithm that prunes some "unnecessary" portions of the windows such that it suffices to consider the remaining parts of the windows. Further, with the help of the decomposition $\mathcal{D}$, we show that a shortest path from s to the remaining windows can be found in $O((K + h) \log h \log n)$ time. We refer to it as *the preliminary result*. To achieve this result, we solve many other problems, which may be of independent interest. For example, we build a data structure of $O(n \log h)$ size such that given any query point t and line segment τ in $\mathcal{P}$, we can compute in $O(\log h \log n)$ time the intersection between τ and the shortest path from s to t in $\mathcal{P}$.

To further reduce the query time to $O(h \log h \log n)$, by using the extended corridor structure of $\mathcal{P}$ [3, 5], we show that there exists a set $\mathcal{S}(q)$ of $O(h)$ *candidate windows* such that a shortest path from s to see the query point q must be a shortest path from s to a window in $\mathcal{S}(q)$. This is actually quite consistent with the result in the simple polygon case, where only one window is needed for answering each quickest visibility query [1]. Once the set $\mathcal{S}(q)$ is computed, we can apply our pruning algorithm discussed above on $\mathcal{S}(q)$ to answer the quickest visibility query in additional $O(h \log h \log n)$ time. To compute $\mathcal{S}(q)$, we give an algorithm of $O(h \log n)$ time, without having to explicitly compute $\mathit{Vis}(s)$. The algorithm is based on a modification of the algorithm given in [4] that can compute $\mathit{Vis}(q)$ in $O(K \log n)$ time for any point q, after $O(n + h^2)$ space and $O(n + h^2 \log h)$ time preprocessing.

The rest of the paper is organized as follows. In Section 2, we define notation and review some concepts. In Section 3, we introduce the decomposition $\mathcal{D}$ of $\mathcal{P}$, and discuss the segment queries. We present our preliminary result for quickest visibility queries in Section 4, and the improved result is discussed in Section 5. Due to the space limit, we only sketch the main idea and all details can be found in the full paper [15].

2 Preliminaries

For any subset A of $\mathcal{P}$, we say that a point p is *(weakly) visible* to A if p is visible to at least one point of A. For any point $t \in \mathcal{P}$, we use $\pi(s, t)$ to denote a shortest path from s to t in $\mathcal{P}$, and in the case where the shortest path is not unique, $\pi(s, t)$ may refer to an arbitrary such path. With a little abuse of notation, for any subset A of $\mathcal{P}$, we use $\pi(s, A)$ to denote a shortest path from s to all points of A; we use $d(s, A)$ to denote the length of $\pi(s, A)$, i.e., $d(s, A) = \min_{t \in A} d(s, t)$. Let $\mathcal{V}$ denote the set of all vertices of $\mathcal{P}$.

The shortest path map. The shortest path map $\mathit{SPM}(s)$ is a decomposition of $\mathcal{P}$ into regions (or cells) such that in each cell σ, the sequence of obstacle vertices along $\pi(s, t)$ is fixed for all t in σ [10, 14]. Further, the *root* of σ, denoted by $r(\sigma)$, is the last vertex of $\mathcal{V} \cup \{s\}$ in $\pi(s, t)$ for any point $t \in \sigma$ (hence $\pi(s, t) = \pi(s, r(\sigma)) \cup \overline{r(\sigma)t}$; note that $r(\sigma)$ is s if s is visible to t). We classify each edge of a cell σ into three types: a portion of an edge

of $\mathcal{P}$, *an extension segment*, which is a line segment extended from $r(\sigma)$ along the opposite direction from $r(\sigma)$ to the vertex of $\pi(s,t)$ preceding $r(\sigma)$, and *a bisector curve/edge* that is a hyperbolic arc. For each point t on a bisector edge of $SPM(s)$, t is on the common boundary of two cells and there are two different shortest paths from s to t through the roots of the two cells, respectively. The *vertices* of $SPM(s)$ include $\mathcal{V} \cup \{s\}$ and all intersections of edges of $SPM(s)$. The intersection of two bisector edges is called a *triple point*, which has more than two shortest paths from s. The map $SPM(s)$ has $O(n)$ vertices, edges, and cells [10, 14].

For differentiation, we call the vertices and edges of $\mathcal{P}$ the *obstacle vertices* and the *obstacle edges*, respectively. The holes and the outer polygon of $\mathcal{P}$ are also called *obstacles*.

The *shortest path tree* $SPT(s)$ is the union of shortest paths from s to all obstacle vertices of $\mathcal{P}$. $SPT(s)$ has $O(n)$ edges [10, 14]. Given $SPM(s)$, $SPT(s)$ can be obtained in $O(n)$ time.

For ease of exposition, we make a general position assumption that no obstacle vertex has more than one shortest path from s and no point of $\mathcal{P}$ has more than three shortest paths from s. Hence, no bisector edge of $SPM(s)$ intersects an obstacle vertex and no three bisector edges intersect at the same point.

For any polygon P, we use $|P|$ to denote the number of vertices of P and use ∂P to denote the boundary of P.

Ray-shooting queries in simple polygons. Let P be a simple polygon. With $O(|P|)$ time and space preprocessing, each ray-shooting query in P (i.e., given a ray in P, find the first point on ∂P hit by the ray) can be answered in $O(\log |P|)$ time [2, 9]. The result can be extended to curved simple polygons or splinegons [12].

The canonical lists and cycles of planar trees. We will often talk about certain planar trees in $\mathcal{P}$ (e.g., $SPT(s)$). Consider a tree T with root r. A leaf v is called a *base leaf* if it is the leftmost leaf of a subtree rooted at a child of r. Denote by $\mathcal{L}(T,v)$ the post-order traversal list of T starting from such a base leaf v, and we call it a *canonical list* of T. The root r must be the last node in $\mathcal{L}(T,v)$. We remove r from $\mathcal{L}(T,v)$ and make the remaining list a cycle by connecting its rear to its front, and let $\mathcal{C}(T)$ denote the circular list. Although T may have multiple base leaves, $\mathcal{C}(T)$ is unique and we call $\mathcal{C}(T)$ the *canonical cycle* of T. We further use $\mathcal{L}_l(T,v)$ to denote the list of the leaves of T following their relative order in $\mathcal{L}(T,v)$ and use $\mathcal{C}_l(T)$ to denote the circular list of $\mathcal{L}_l(T,v)$. One reason we introduce these notation is the following. Let e be any edge of T. All nodes of T whose paths to r in T contain e are consecutive in $\mathcal{L}(T,v)$ and $\mathcal{C}(T)$. Similarly, all leaves of T whose paths to r in T contain e must be consecutive in $\mathcal{L}_l(T,v)$ and $\mathcal{C}_l(T)$.

The following observation on shortest paths will be frequently referred to in the paper.

▶ **Observation 1.**

1. *Suppose π_1 and π_2 are two shortest paths from s to two points in $\mathcal{P}$, respectively; then π_1 and π_2 do not cross each other.*
2. *Suppose π_1 is a shortest path from s to a point in $\mathcal{P}$ and τ is a line segment in $\mathcal{P}$; then the intersection of π_1 and τ is a sub-segment of τ (which may be a single point or empty).*

3 The Decomposition $\mathcal{D}$ and the Segment Queries

In this section, we introduce a decomposition $\mathcal{D}$ of $\mathcal{P}$ and use it to solve the segment query problem. The decomposition $\mathcal{D}$ will also be useful for solving the quickest visibility queries.

We first define a set V of points. Let p be an intersection between a bisector edge of $SPM(s)$ and an obstacle edge. Since p is on a bisector edge, it is in two cells of $SPM(s)$ and

has two shortest paths from s. We make two copies of p in the way that each copy belongs to only one cell (and thus corresponds to only one shortest path from s). We add the two copies of p to V. We do this for all intersections between bisector edges and obstacle edges. Consider a triple point p, which is in three cells of $SPM(s)$ and has three shortest paths from s. Similarly, we make three copies of p that belong to the three cells, respectively. We add the three copies of p to V. We do this for all triple points. This finishes the definition of V.

By definition, each point of V has exactly one shortest path from s. Let Π_V denote the set of shortest paths from s to all points of V. Let T_V be the union of all shortest paths of Π_V. We consider points of V distinct although some of them are copies of the same physical point. In this way, we can consider T_V as a "physical" tree rooted at s.

▶ **Definition 1.** Define $\mathcal{D}$ to be the decomposition of $\mathcal{P}$ by the edges of T_V.

In the following, we assume the shortest path map $SPM(s)$ has already been computed. We have the following lemma about the decomposition $\mathcal{D}$.

▶ **Lemma 2.**

1. *The size of the set V is $O(h)$.*
2. *The combinatorial size of $\mathcal{D}$ is $O(n)$.*
3. *Each cell of $\mathcal{D}$ is simply connected.*
4. *For any segment τ in $\mathcal{P}$, τ can intersect at most $O(h)$ cells of $\mathcal{D}$. Further, for each cell Δ of $\mathcal{D}$, the intersection τ and Δ consists of at most two (maximal) sub-segments of τ.*
5. *After $O(n)$ time preprocessing, for any segment τ' in a cell Δ of $\mathcal{D}$, the shortest path from s to τ' can be computed in $O(\log |\Delta|)$ time, where $|\Delta|$ is the combinatorial size of Δ.*
6. *For each cell Δ of $\mathcal{D}$, Δ has at most two vertices r_1 and r_2 (both in $\mathcal{V} \cup \{s\}$), called "super-roots", such that for any point $t \in \Delta$, $\pi(s,t)$ is the concatenation of $\pi(s,r)$ and the shortest path from r to t in Δ, for a super-root r in $\{r_1, r_2\}$.*
7. *Given the shortest path map $SPM(s)$, $\mathcal{D}$ can be computed in $O(n)$ time.*

Using $\mathcal{D}$, we can easily answer each segment query in $O(h \log \frac{n}{h})$ time by a "pedestrian" algorithm, similar in spirit to the ray-shooting algorithm of Hershberger and Suri [9].

▶ **Theorem 3.** *Given the shortest path map $SPM(s)$, we can build a data structure of $O(n)$ size in $O(n)$ time, such that each segment query can be answered in $O(h \log \frac{n}{h})$ time.*

4 The Quickest Visibility Queries: The Preliminary Result

In this section, we give our preliminary result on quickest visibility queries, which sets the stage for our improved result.

For any subset A of $\mathcal{P}$, a point $p \in A$ is called a *closest point* of A (with respect to s) if $d(s, A) = d(s, p)$. Given any query point q in $\mathcal{P}$, our goal is to find a shortest path from s to $Vis(q)$. Let q^* be a closest point of $Vis(q)$. To answer the query, it is sufficient to determine q^*. Thus we will focus on finding q^*. Note that if q is visible to s, then $q^* = s$. We can determine whether s is visible to q in $O(\log n)$ time by checking whether q is in the cell of $SPM(s)$ whose root is s. In the following, we assume that s is not visible to q.

We define the *windows* of q and $Vis(q)$. Consider an obstacle vertex u that is visible to q such that the two incident obstacle edges of u are on the same side of the line through q and u (e.g., see Fig. 1). Let $q(u)$ denote the first point on $\partial\mathcal{P}$ hit by the ray from u along the direction from q to u. Then $\overline{uq(u)}$ is called a *window* of q; we say that the window is *defined by* u. Further, we call $\overline{qq(u)}$ the *extended window* of $\overline{uq(u)}$.

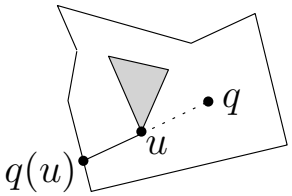


Figure 1 Illustrating a window $\overline{uq(u)}$ of q.

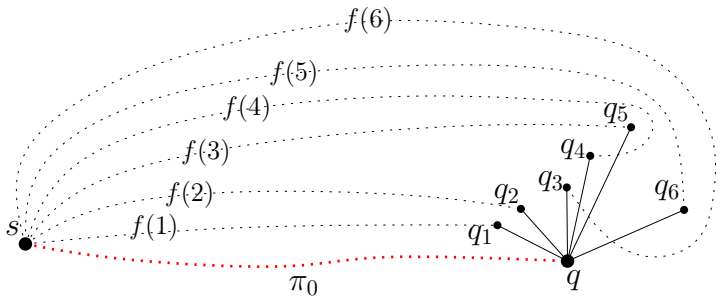


Figure 2 Illustrating the map $f(\cdot)$: $f(1) = 1$, $f(2) = 2$, $f(3) = 5$, $f(4) = 4$, $f(5) = 6$, and $f(6) = 3$. Note that the paths could be "below" π_0, but for ease of exposition, we "flip" them above π_0, and this flip operation does not change the topology of these paths.

Each window of q is an edge of $Vis(q)$, and thus the number of windows of q is $O(K)$, where $K = |Vis(q)|$. Further, there must be a closest point q^* that is on a window of q [1]. Hence, as in [1], a straightforward algorithm to compute q^* is to compute shortest paths from s to all windows of s and the path of minimum length determines q^*. To compute shortest paths from s to all windows, if we apply our segment queries on all windows using Theorem 3, then the total time would be $O(K \cdot h \cdot \log \frac{n}{h})$. In the rest of this section, we present an algorithm that can compute q^* in $O((K + h) \log h \log n)$ time, without having to compute shortest paths to all windows.

4.1 The Algorithm Overview

As the first step, we compute $Vis(q)$, which can be done in $O(K \log n)$ time after $O(n+h^2 \log h)$ time and $O(n + h^2)$ space preprocessing [4]. Then, we can find all windows and extended-windows in $O(K)$ time. For ease of exposition, we make a general position assumption for q that q is not collinear with any two obstacle vertices. The assumption implies that q is in the interior of $\mathcal{P}$ and no two windows are collinear.

Let u_0 be the root of the cell of $SPM(s)$ containing q (if q is on the boundary of multiple cells, then we take an arbitrary such cell). Hence, $\pi(s, u_0) \cup \overline{u_0 q}$ is a shortest path $\pi(s, q)$ from s to q. Note that u_0 must define a window $\overline{u_0 q(u_0)}$ of q [13]. Let $\overline{u_0 q(u_0)}, \overline{u_1 q(u_1)}, \ldots, \overline{u_k q(u_k)}$ be all windows of q ordered *clockwise* around q. Clearly, $k = O(K)$. For each $0 \leq i \leq k$, let $q_i = q(u_i)$. Note that the window $\overline{u_0 q_0}$ is special in the sense that u_0 is in $\pi(s, q)$. So we first apply our algorithm in Theorem 3 on $\overline{u_0 q_0}$ to compute a closest point q_0^* of $\overline{u_0 q_0}$. Clearly, if $q^* \in \overline{u_0 q_0}$, then $q^* = q_0^*$. In the following, we assume $q^* \notin \overline{u_0 q_0}$. Let $Q = \{q, q_1, q_2, \ldots, q_k\}$. Note that Q does not contain q_0 but q. If $q^* \in Q$, then we can find q^* by computing $d(s, p)$ for all $p \in Q$, in $O(k \log n)$ time using $SPM(s)$. Below, we assume $q^* \notin Q$. Note that the above assumption that $q^* \notin \overline{u_0 q_0} \cup Q$ is only for arguing the correctness of our following algorithm, which actually proceeds without knowing whether the assumption is true or not.

For each $0 \leq i \leq k$, let $w_i = \overline{q q_i}$, i.e., the extended window of $\overline{u_i q_i}$. Let $W = \{w_i \mid 1 \leq i \leq k\}$. For convenience of discussion, we assume that each w_i of W does not contain its two endpoints q and q_i (but the endpoints of w_i still refer to q and q_i). Since $q^* \notin \overline{u_0 q_0} \cup Q$, q^* must be on an extended window in W. Clearly, q^* is also a closest point of W. Since no two windows of q are collinear, no extended-window of W contains another. We assign each window $w_i \in W$ a direction from q to q_i, so that we can talk about its left or right side.

Suppose q^* is on $w_i \in W$. Since w_i is an open segment, by the definition of q^*, the shortest path $\pi(s, q^*)$ must reach q^* from either the left side or the right side of w_i. Formally,

we say that $\pi(s, q^*)$ reaches q^* from the left side (resp., right side) of w_i if there is a small neighborhood of q^* such that all points of $\pi(s, q^*)$ in the neighborhood are on the left side (resp., right side) of w_i. Let w_i^l (resp., w_i^r) denote the set of points p on w_i whose shortest path from s to p is from the left (resp., right) side of w_i. Hence, q^* is either on w_i^l or on w_i^r.

Our algorithm will find two points q_l^* and q_r^* such that if q^* is on w_i^l for some $i \in [1, k]$, then $q^* = q_l^*$, and otherwise (i.e.,q^* is in w_i^r for some $i \in [1, k]$), $q^* = q_r^*$.

In the following, we will only present our algorithm for finding q_l^* since the case for q_r^* is symmetric. In the following discussion, we assume q^* is on w_i^l for some $i \in [1, k]$.

The rest of this section is organized as follows. In Section 4.2, we discuss some observations, based on which we describe our pruning algorithm in Section 4.3 to prune some (portions of) segments of W such that q^* $(= q_l^*)$ is still in the remaining segments of W. In Section 4.4, we will finally compute q_l^* on the remaining segments of W. As will be clear later, our algorithm uses extended windows instead of windows because extended windows can help us with the pruning.

4.2 Observations

For any point $t \in \mathcal{P}$ with $s \neq t$, and its shortest path $\pi(s, t)$, we use t^+ to denote a point on $\pi(s, t)$ arbitrarily close to t (but $t^+ \neq t$). If t is on w_i^l for some $i \in [1, k]$, then t^+ must be on the left side of w_i. For any segment w of W, we say that w or a sub-segment of w can be *pruned* if it does not contain q^*. Our pruning algorithm, albeit somewhat involved, is based on the following simple observation.

▶ **Observation 2.** *For any point $t \in w_i^l$ for some $i \in [1, k]$, if $\pi(s, t^+)$ intersects any segment $w \in W$ or an endpoint of it, then t can be pruned (i.e., t cannot be q^*).*

Proof. Let t' be a point on $\pi(s, t^+)$ that is a point on any segment $w \in W$ or an endpoint of it. Clearly, $t' \in \mathit{Vis}(s)$ and $d(s, t') < d(s, t)$. Thus, t cannot be q^*. ◀

Consider the shortest paths $\pi(s, q_i)$ for $i = 1, 2, \ldots, k$. To simplify the notation, let $\pi_i = \pi(s, q_i)$ for each $i \in [1, k]$. In particular, let $\pi_0 = \pi(s, q)$ (not $\pi(s, q_0)$). Recall that $Q = \{q, q_1, \ldots, q_k\}$. The union of all paths π_i for $0 \leq i \leq k$ forms a planar tree, denoted by T_Q, with root at s. Consider the canonical cycle $\mathcal{C}(T_Q)$ as defined in Section 2. Let $\mathcal{C}_Q$ be the circular list of the points of Q following their relative order in $\mathcal{C}(T_Q)$. We further break $\mathcal{C}_Q$ into a list $\mathcal{L}_Q$ at q, such that $\mathcal{L}_Q$ starts from q and all other points of $\mathcal{L}_Q$ follow the counterclockwise order in $\mathcal{C}_Q$. Assume $\mathcal{L}_Q$ is $\{q, q_{f(1)}, q_{f(2)}, \ldots, q_{f(k)}\}$, i.e., the $(i+1)$-th point of the list is $q_{f(i)}$ (e.g., see Fig. 2). So $f(\cdot)$ essentially maps each point of $Q \setminus \{q\}$ from its position in $\mathcal{L}_Q$ to its position in the list $\{q_1, q_2, \ldots, q_k\}$. Hence, $f(1) \ldots, f(k)$ is a permutation of $1, \ldots, k$, and $f(i) \neq f(j)$ if $i \neq j$. The reason we introduce the list $\mathcal{L}_Q$ is that intuitively, for any $1 \leq i < j \leq k$, the path $\pi_{f(j)}$ is *counterclockwise* from $\pi_{f(i)}$ with respect to π_0 around s. For convenience, we let $f(0) = 0$.

Given $\mathit{SPM}(s)$, after $O(n)$ time preprocessing, we can compute the list $\mathcal{L}_Q$ and thus determine the map $f(\cdot)$ in $O(k \log n)$ time. The details are omitted.

We can show that for any $i \in [1, k]$, π_0 does not contain q_i and π_i does not contain q. The following lemma is proved based on Observation 2.

▶ **Lemma 4.** *Suppose π_j contains q_i with $i \neq j$ and $i, j \in [1, k]$. If $i < j$, then w_j can be pruned; otherwise, w_i can be pruned.*

In $O(k \log n)$ time, we can remove all extended-windows of W that can be pruned by Lemma 4. The details are omitted. But to simplify the notation, we assume that none of the

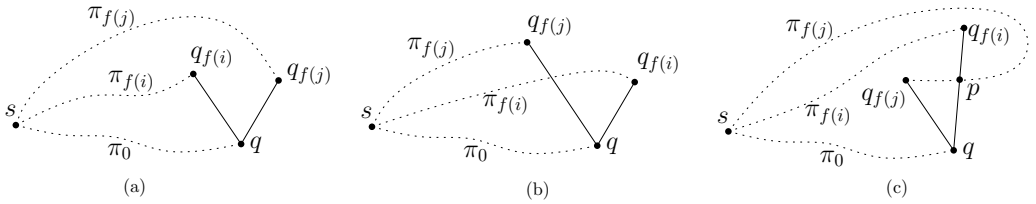


Figure 3 Illustrating Lemma 5.

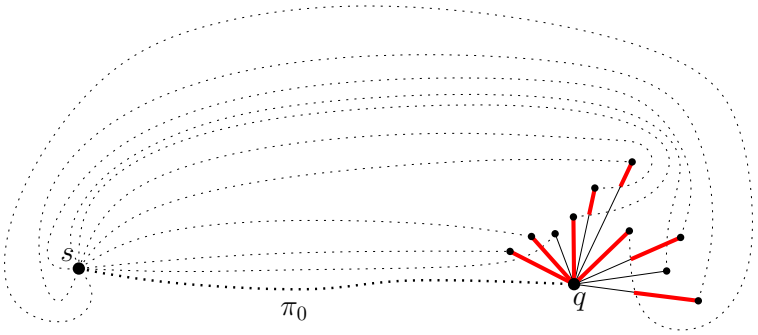


Figure 4 The thick (red) segments are the remaining parts of the segments of W after the pruning algorithms (so that q_l^* must be on the left side of a red segment). Again, we "flip" all paths above π_0.

segments of W is pruned since otherwise we could re-index all segments of W. So now W has the following property: For any $i \in [1, k]$, q_i is not contained in any π_j with $j \in [0, k]$ and $j \neq i$.

For each $i \in [1, k]$, since π_0 does not cross π_i, $\pi_0 \cup \pi_i \cup w_i$ forms a closed curve that separates the plane into two regions, one locally on the left of w_i and the other locally on the right w_i. We let D_i denote the region locally on the left side of w_i including $\pi_0 \cup \pi_i \cup w_i$ as its boundary (it is possible that D_i is unbounded). If $\pi_0 \cap \pi_i$ is a sub-path including at least one edge, then it is also considered to be in D_i. We can show that if $q^* \in w_i^l$, then $\pi(s, q^*)$ must be in D_i.

Our pruning algorithm mainly relies on Lemma 5, which is based on Observation 2.

▶ **Lemma 5.** *Suppose i and j are two indices with $1 \leq i < j \leq k$.*

1. *If $f(i) < f(j)$, then $\pi_{f(i)}$ does not cross $w_{f(j)}$ and $\pi_{f(j)}$ does not cross $w_{f(i)}$, and further, $D_{f(i)}$ is contained in $D_{f(j)}$ (e.g., see Fig. 3(a)).*
2. *If $f(i) > f(j)$, then either $\pi_{f(i)}$ crosses $w_{f(j)}$ or $\pi_{f(j)}$ crosses $w_{f(i)}$. Further, in the former case (see Fig. 3(b)), $w_{f(i)}$ can be pruned, and in the latter case (see Fig. 3(c)), the sub-segment $\overline{qp}$ of $w_{f(i)}$ can be pruned, where p is the point at which $\pi_{f(j)}$ crosses $w_{f(i)}$.*

For any $1 \leq i < j \leq k$, we say π_i and π_j are *consistent* if $f(i) < f(j)$. By Lemma 5, if π_i and π_j are not consistent, then we can do some pruning, based on which we present our pruning algorithm in Section 4.3. Figure 4 gives an example showing the remaining parts of the segments of W after the pruning.

4.3 A Pruning Algorithm for Pruning the Segments of W

We process the paths $\pi_{f(1)}, \pi_{f(2)}, \ldots, \pi_{f(k)}$ in this order. Assume that $\pi_{f(i-1)}$ has just been processed and we are about to process $\pi_{f(i)}$. Our algorithm maintains a sequence of *bundles*,

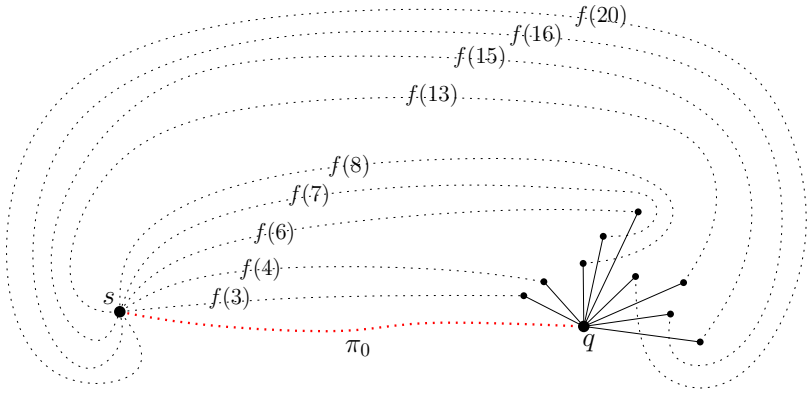


Figure 5 Illustrating the shortest paths corresponding to the indices in the current bundle sequence $\mathbb{B} = \{\underline{\{3\}}, \underline{\{4\}}, \underline{\{\{6\}, \{7\}\}}, \underline{\{8\}}, \underline{\{\{13\}, \{\{15\}, \{16\}\}, \{20\}\}}\}$, where each underline indicates a bundle of $\mathbb{B}$. For example, the last bundle is a composite bundle consisting of three children bundles with 20 as its wrap index. In the figure, the indices of the paths are labeled. Again, we "flip" all paths above π_0.

denoted by $\mathbb{B} = \{B_1, B_2, \ldots B_g\}$. Each *bundle* $B \in \mathbb{B}$ is defined recursively as follows. Essentially B is a list of sorted indices of a subset of $\{1, 2, \ldots, i-1\}$, but the indices are grouped in a special and systematic way.

There are two types of bundles: *atomic* and *composite*. If B has only one index, then it is an atomic bundle. Otherwise, B is a composite bundle consisting of a sequence of at least two bundles $B'_1, \ldots, B'_{g'}$ (with $g' \geq 2$) such that the last bundle $B'_{g'}$ must be atomic (others can be either atomic or composite), and we call the index contained in $B'_{g'}$ the *wrap index* of B. We consider the bundles $B'_1, \ldots, B'_{g'}$ as the *children bundles* of B.

Let $f_{\min}(B)$ and $f_{\max}(B)$ denote the smallest and largest $f(j)$ of all indices j of B, respectively. If B is composite, then B further has the following three *bundle-properties*. (1) The indices of B are distinct and sorted increasingly by their order in B. (2) For any $1 \leq b < g' - 1$, $f_{\max}(B'_b) < f_{\min}(B'_{b+1})$. (3) If j is the wrap index of B, then $f_{\min}(B) = f(j)$ and $\pi_{f(j)}$ crosses $w_{f(j')}$ for every $j' \in B \setminus \{j\}$ (intuitively, $\pi_{f(j)}$ "wraps" the point $q_{f(j')}$, and this is why we call j a "wrap" index). Refer to Fig. 5 for an example.

For convenience, if the context is clear, we also consider a bundle B as a set of sorted indices. So if an index j is in B, we can write "$j \in B$". We use the word "bundle" because each index j of B refers to the path $\pi_{f(j)}$. Therefore, B is a "bundle" of shortest paths.

In addition, the bundle sequence $\mathbb{B} = \{B_1, B_2, \ldots, B_g\}$ maintained by our algorithm has two $\mathbb{B}$-*properties*. (1) The indices in all bundles are distinct in $[1, i-1]$ and are sorted increasingly by their order in the sequence. (2) For any $1 \leq b < g$, $f_{\max}(B_b) < f_{\min}(B_{b+1})$.

▶ **Observation 3.**

1. *For any $1 \leq b < b' \leq g$ and any indices $j \in B_b$ and $j' \in B_{b'}$ (both B_b and $B_{b'}$ are from $\mathbb{B}$), the two shortest paths $\pi_{f(j)}$ and $\pi_{f(j')}$ are consistent (see Fig. 5).*
2. *For any composite bundle $B = \{B'_1, \ldots, B'_{g'}\}$, for any $1 \leq b < b' \leq g' - 1$ and any indices $j \in B'_b$ and $j' \in B'_{b'}$, the two shortest paths $\pi_{f(j)}$ and $\pi_{f(j')}$ are consistent (see Fig. 5).*

In the following, we describe our algorithm for processing the shortest path $\pi_{f(i)}$, during which $\mathbb{B}$ will be updated. Initially when $i = 1$, $\mathbb{B}$ contains the only atomic bundle $B = \{1\}$ and this finishes our processing for $\pi_{f(1)}$. In general when $i > 1$, we do the following.

We first find the index β such that $f_{\max}(B_\beta) < f(i) < f_{\max}(B_{\beta+1})$. We can maintain the bundle sequence $\mathbb{B}$ in a data structure so that β can be found in $O(\log n)$ time. The

details are omitted. If $\beta = g$ (so $B_{\beta+1}$ does not exist in this case), then we add a new atomic bundle $B_{g+1} = \{i\}$ to the rear of $\mathbb{B}$ and this finishes the processing of $\pi_{f(i)}$.

If $\beta \neq g$, we check whether $f_{\min}(B_{\beta+1}) < f(i)$. If $f_{\min}(B_{\beta+1}) < f(i)$, we can show that $w_{f(i)}$ can be pruned. Hence, in this case, we simply ignore $\pi_{f(i)}$ and finish the processing of $\pi_{f(i)}$. In the following, we assume $f(i) < f_{\min}(B_{\beta+1})$ (note that $f(i) \neq f_{\min}(B_{\beta+1})$ since $i \notin \mathbb{B}$). Next, we are going to find all such indices j of $\mathbb{B}$ that $\pi_{f(j)}$ crosses $w_{f(i)}$. To this end, the following two lemmas are crucial.

▶ **Lemma 6.**

1. *For any index j in B_b for any $b \in [1, \beta]$, $\pi_{f(j)}$ does not cross $w_{f(i)}$.*
2. *For any index j in B_b for any $b \in [\beta+1, g]$, if $\pi_{f(j)}$ crosses $w_{f(i)}$, then $w_{f(j)}$ can be pruned; otherwise, $\pi_{f(i)}$ must cross $w_{f(j)}$.*
3. *If j is in B_b for some $b \in [\beta+2, g]$ and $\pi_{f(j)}$ crosses $w_{f(i)}$, then $\pi_{f(j')}$ crosses $w_{f(i)}$ for any $j' \in B_{b'}$ and any $b' \in [\beta+1, b-1]$.*
4. *If j is in B_b for some $b \in [\beta+1, g-1]$ and $\pi_{f(j)}$ does not cross $w_{f(i)}$, then $\pi_{f(j')}$ does not cross $w_{f(i)}$ for any $j' \in B_{b'}$ and any $b' \in [b+1, g]$.*

For any bundle B in $\{B_{\beta+1}, B_{\beta+2}, \ldots, B_g\}$, if B has two indices j and j' such that $w_{f(i)}$ crosses $\pi_{f(j)}$ but does not cross $\pi_{f(j')}$, then we say that B is a *mixed* bundle, which is necessarily a composite bundle.

▶ **Lemma 7.** *For any mixed bundle $B = \{B'_1, B'_2, \ldots, B'_{g'}\}$, the following holds.*

1. *The path $\pi_{f(r)}$ must cross $w_{f(i)}$, where r is the wrap index of B, i.e., $B'_{g'} = \{r\}$.*
2. *If an index j is in B'_b for some $b \in [2, g'-1]$ and $\pi_{f(j)}$ crosses $w_{f(i)}$, then $\pi_{f(j')}$ crosses $w_{f(i)}$ for any $j' \in B'_{b'}$ and any $b' \in [1, b-1]$.*
3. *If an index j is in B'_b for some $b \in [1, g'-2]$ and $\pi_{f(j)}$ does not cross $w_{f(i)}$, then $\pi_{f(j')}$ does not cross $w_{f(i)}$ for any $j' \in B'_{b'}$ and any $b' \in [b+1, g'-1]$.*
4. *If a bundle B' of B has two indices j and j' such that $w_{f(i)}$ crosses $\pi_{f(j)}$ but does not cross $\pi_{f(j')}$, then B' is also a* mixed *bundle. This lemma applies to B' recursively.*

In light of the preceding two lemmas, in the following we will find the indices j of $\mathbb{B}$ such that $\pi_{f(j)}$ crosses $w_{f(i)}$ and then prune $w_{f(j)}$ by Lemma 6(2) (i.e., remove j from $\mathbb{B}$); we say that such an index j is *prunable*.

Before describing our algorithm, we discuss an operation that will be used in the algorithm. Consider a composite bundle $B = \{B'_1, B'_2, \ldots, B'_{g'}\}$ of $\mathbb{B}$. Let r be a wrap index of B, i.e., $B'_{g'} = \{r\}$. Suppose $w_{f(i)}$ crosses $\pi_{f(r)}$. Our algorithm will remove r from B and thus from $\mathbb{B}$. This is done by a *wrap-index-removal* operation. Further, suppose B is the j-th bundle of $\mathbb{B}$, i.e., $B = B_j$. After r is removed, the operation will implicitly insert the bundles $B'_1, B'_2, \ldots, B'_{g'-1}$ into the position of B in $\mathbb{B}$, i.e., after the operation, $\mathbb{B}$ becomes $B_1, \ldots, B_{j-1}, B'_1, \ldots, B'_{g'-1}, B_{j+1}, \ldots, B_g$. Note that this new bundle list still has the two $\mathbb{B}$-properties. Indeed, $f_{\max}(B_{j-1}) < f_{\min}(B) = f(r) < f_{\min}(B'_1)$ and $f_{\max}(B'_{g'-1}) \leq f_{\max}(B) < f_{\min}(B_{j+1})$. We can maintain the bundles of $\mathbb{B}$ in a data structure so that each wrap-index-removal operation can be performed in $O(\log n)$ time. The details are omitted.

Another operation that is often used in the algorithm is the following. Given any $i, j \in [1, k]$, we want to determine whether $w_{f(i)}$ crosses $\pi_{f(j)}$. We call it the *shortest path segment intersection* (or *SP-segment-intersection*) query. Our full paper presents an algorithm that can answer each such query in $O(\log h \log n)$ time, after $O(n \log h)$ time and space preprocessing.

We are ready to describe our algorithm for removing all prunable indices from $\mathbb{B}$. By Lemma 6(1), each bundle B_b of $\mathbb{B}$ for $1 \leq b \leq \beta$ does not contain any prunable index.

For each bundle B of $B_{\beta+1}, B_{\beta+2}, \ldots, B_g$ in order, we call a procedure *prune(B)* until the procedure returns "false".

If all indices of B are prunable, then *prune(B)* will return "true" and the entire bundle B will be removed from $\mathbb{B}$. Otherwise, the procedure will return false. Further, if B is a mixed bundle, then all prunable indices of B will be removed (and the procedure returns false).

The procedure *prune*(B) works as follows. It is a recursive procedure. As a base case, if B is an atomic bundle $\{j\}$, then we call an SP-segment-intersection query to check whether $\pi_{f(j)}$ crosses $w_{f(i)}$. If yes, we remove B and return true; otherwise, return false. If B is a composite bundle $\{B'_1, B'_2, \ldots, B'_{g'}\}$ with r as the wrap index (i.e., $B'_{g'} = \{r\}$), then we first call an SP-segment-intersection to check whether $\pi_{f(r)}$ crosses $w_{f(i)}$. If not, by Lemma 7(1), B does not have any prunable index and thus we simply return false. If yes, then we call a wrap-index-removal operation to remove $B'_{g'}$. Afterwards, for each $b' = 1, 2, \ldots, g' - 1$ in order, we call *prune*$(B'_{b'})$ recursively. If *prune*$(B'_{b'})$ returns false, then we return false (without calling *prune*$(B'_{b'+1})$). If it returns true, we remove $B'_{b'}$ (in fact all children bundles of $B'_{b'}$ have been removed by *prune*$(B'_{b'})$). If $b' = g' - 1$, then we return true (since all bundles of B have been removed); otherwise, we proceed on calling *prune*$(B'_{b'+1})$.

If *prune*(B_b) returns true for every b with $\beta + 1 \leq b \leq g$, then we add a new atomic bundle $\{i\}$ at the end of $\mathbb{B}$, which now becomes $\{B_1, B_2, \ldots, B_\beta, \{i\}\}$. This also finishes our preprocessing for $\pi_{f(i)}$. Otherwise, *prune*(B_b) returns false for some b with $\beta + 1 \leq b \leq g$. In this case, as a final step, we create a new composite bundle B, consisting of all bundles of $\mathbb{B}$ after B_β (not including B_β) and the atomic bundle $\{i\}$ as the last child bundle of B. This is done by a *bundle-creation* operation, which can be implemented in $O(\log n)$ time (the details are omitted). Afterwards, the new bundle sequence $\mathbb{B}$ becomes $\{B_1, B_2, \ldots, B_\beta, B\}$. It can be shown that the new bundle B is a "valid" composite bundle and the updated $\mathbb{B}$ maintains the two $\mathbb{B}$-properties.

To analyze the running time of the above algorithm, let m be the number of indices that have been removed from $\mathbb{B}$. Then, the algorithm makes at most $m+1$ SP-segment-intersection queries. To see this, once the query discovers an index j that is not prunable, the algorithm will stop without making any more such queries. On the other hand, each wrap-index-removal operation removes an index, and thus the number of such operations is at most m. Further, observe that for each bundle B, whenever we make a recursive call on a child bundle of B, the wrap index of B is guaranteed to be removed. Therefore, the number of total recursive calls is at most m as well. Hence, the running time of the algorithm is $O((m+1) \log h \log n)$.

This finishes our algorithm for processing the path $\pi_{f(i)}$. The total time for processing $\pi_{f(i)}$ is $O((m+1) \log h \log n)$. Since once an index is removed from $\mathbb{B}$, it will never be inserted into $\mathbb{B}$ again, the sum of all such m in the entire algorithm for processing all paths $\pi_{f(i)}$ for $i = 1, 2, \ldots, k$ is at most k. Hence, the total time of the entire algorithm is $O(k \log h \log n)$.

4.4 Computing the Closest Point q^*

Recall that we have assumed that q^* is on w_i^l for some $i \in [1, k]$, i.e., $q^* = q_l^*$. According to our pruning algorithm for computing the bundle sequence $\mathbb{B}$, q^* must be on $w_{f(j)}^l$ for some $j \in \mathbb{B}$. In this section, we will compute q^* by using the bundle sequence $\mathbb{B}$. For example, in Fig 4, our goal is to compute q^* on the left sides of those (red) thick segments.

4.4.1 The Set of Regions $\mathcal{R}$

Our algorithm for computing q^* uses a set $\mathcal{R}$ of regions of $\mathcal{P}$, which is introduced below.

Let $\mathcal{O}$ denote the obstacle space, which is the complement of the free space of $\mathcal{P}$. More specifically, $\mathcal{O}$ consists of the $h-1$ simple polygonal holes of $\mathcal{P}$ and the (unbounded) region outside the outer boundary of $\mathcal{P}$. Let $\mathcal{B}$ denote the union of all bisector edges of $SPM(s)$. Mitchell [13] proved that $\mathcal{O} \cup \mathcal{B}$ is simply connected and $\mathcal{P} \setminus \mathcal{B}$ is also simply connected (e.g., see Fig.1 in the appendix). We consider $\mathcal{O} \cup \mathcal{B}$ as a planar graph G. Specifically, the vertex set of G consists of all obstacles of $\mathcal{O}$ and all triple points of $SPM(s)$. For any two vertices of G, if they are connected by a chain of bisector edges in $SPM(s)$ such that the chain does not contain any other vertex of G, then G has an edge connecting the two vertices, and further, we call the above chain of bisector edges a *bisector super-curve*. It can be shown that G is a simple graph with $O(h)$ vertices, edges, and faces.

Since $|V| = O(h)$ (by Lemma 2(1)), Π_V is a set of $O(h)$ shortest paths. Recall that T_V is the union of all shortest paths of Π_V and T_V is considered as a "physical" tree rooted at s. Note that each edge of any path of Π_V except the last edge (i.e., the one connecting a point of V) is an edge of $SPT(s)$. Hence, the total number of edges of the tree T_V is $O(n)$. Throughout the paper, let $h^* = |V|$. Thus, $h^* = O(h)$.

It is known that $\mathcal{P} \setminus \mathcal{B}$ is simply connected and $\pi(s,t)$ is in $\mathcal{P} \setminus \mathcal{B}$ for any point $t \in \mathcal{P}$ [13]. To simplify the discussion, together with the copies of the points of V, we consider $\mathcal{P}' = \mathcal{P} \setminus \mathcal{B}$ as a simple polygon (with some curved edges) by making two copies for each interior point of every bisector super-curve such that they respectively belong to the two sides of the curve.

Since T_V is a planar tree, we can define its canonical lists as discussed in Section 2. Let v_1 be an arbitrary base leaf of T_V. Let the leaf list $\mathcal{L}_l(T_V, v_1)$ be $v_1, v_2, \ldots, v_{h^*}$, which follow the counterclockwise order along $\partial\mathcal{P}'$.

For each $1 \leq i \leq h^*$, let α_i denote the portion of $\partial\mathcal{P}'$ counterclockwise from v_i to v_{i+1} (let v_{h^*+1} refer to v_1). Note that α_i is either a bisector super-curve or a chain of obstacle edges. Suppose we move a point t on α_i from v_i to v_{i+1}. The shortest path $\pi(s,t)$ will continuously change with the same topology since $\pi(s,t)$ is always in $\mathcal{P}'$ (which is simply connected). Let R_i be the region of $\mathcal{P}'$ that is "swept" by $\pi(s,t)$ during the above movement of t. More specifically, let p_i be the common point on $\pi(s,v_i) \cap \pi(s,v_{i+1})$ that is farthest to s. Then, R_i is bounded by $\pi(p_i,v_i)$, $\pi(p_i,v_{i+1})$, and α_i. For convenience of discussion, we let R_i also contain the common sub-path $\pi(s,p_i) = \pi(s,v_i) \cap \pi(s,v_{i+1})$ and we call $\pi(s,p_i)$ the *tail* of R_i. We call the region bounded by $\pi(p_i,v_i)$, $\pi(p_i,v_{i+1})$, and α_i the *cell* of R_i. We consider $\pi(s,v_i)$, $\pi(s,v_{i+1})$, and α_i as the three portions of the boundary ∂R_i of R_i. The definition implies that for any point t in R_i, $\pi(s,t)$ is in R_i. In fact, if t is in the cell of R_i, then $\pi(s,t)$ is the concatenation of $\pi(s,p_i)$ and the shortest path from p_i to t in the cell. Clearly, $\mathcal{P}'$ is the union of $R_1, R_2, \ldots, R_{h^*}$. Roughly speaking, the regions $R_1, \ldots, R_{h^*}$ are counterclockwise around s. Define $\mathcal{R} = \{R_1, R_2, \ldots, R_{h^*}\}$.

4.4.2 The Algorithm for Computing q^*

Let τ be any segment in $\mathcal{P}$ such that a region $R_i \in \mathcal{R}$ contains $\pi(s,\tau)$. Suppose R_i is known. With the help of the decomposition $\mathcal{D}$ proposed in Section 3, we give a *region-processing* algorithm in the full paper to compute $\pi(s,\tau)$ in $O(\log h \log n)$ time.

Recall that $\mathcal{R} = \{R_1, R_2, \ldots, R_{h^*}\}$. Due to our general position assumption that q is not collinear with any two obstacle vertices, none of $\{q, q_1, \ldots, q_k\}$ is an obstacle vertex. Then, for each $k' \in [0,k]$, there is a unique region R_i of $\mathcal{R}$ whose cell contains $q_{f(k')}$, such that the shortest path $\pi_{f(k')}$ is contained in R_i, and we let $z(k')$ refer to the index i of R_i. Computing the indices $z(0), z(1), \ldots, z(k)$ can be done in $O(k \log n)$ time by point location queries on the cells of the regions of $\mathcal{R}$.

For any two indices k_1 and k_2 in $[1,h^*]$, if $k_1 \leq k_2$, then let $[k_1,k_2]_R$ denote the set of all integers $k' \in [k_1,k_2]$; otherwise, let $[k_1,k_2]_R$ denote the set of all integers $k' \in [k_1,h^*] \cup [1,k_2]$.

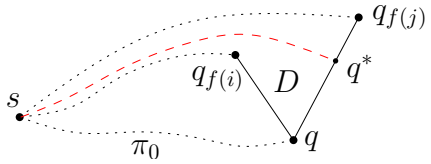


Figure 6 Illustrating Observation 4.

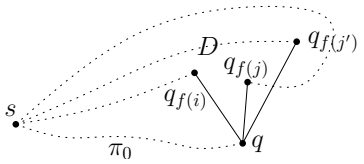


Figure 7 j is the wrap index of B_b and j' is another index of B_b with $j' \neq j$; $\pi_{f(j')}$ is in the region D.

Recall that the regions $R_1, \ldots, R_{h^*}$ are counterclockwise around s. We actually use $[k_1, k_2]_R$ to refer to the set of indices of the regions of $\mathcal{R}$ from R_{k_1} to R_{k_2} counterclockwise around s.

Next we compute q^* on $w^l_{f(j)}$ for $j \in \mathbb{B}$. Consider the bundles of $\mathbb{B} = \{B_1, B_2, \ldots, B_g\}$. For each b with $1 \leq b \leq g$, we call a procedure $path(B_b, z(i))$, where i is the last index of B_{b-1} if $b \geq 2$ and $i = 0$ otherwise. Note that $i < j$ for any index $j \in B_b$. The procedure $path(B_b, z(i))$ works as follows. Depending on whether B_b is atomic, there are two cases.

The atomic case. If B_b is atomic, let j be the only index of B_b. According to the bundle-properties, $i < j$ and $f(i) < f(j)$. So $\pi_{f(j)}$ and $\pi_{f(i)}$ are consistent. By Lemma 5(1), D_i is contained in D_j. Let D be D_j minus the interior of D_i.

▶ **Observation 4.** *If q^* is on $w^l_{f(j)}$, then $\pi(s, q^*)$ must be in D (see Fig. 6).*

▶ **Lemma 8.** *If q^* is on $w^l_{f(j)}$, then $\pi(s, q^*)$ is in $R_{k'}$ for some index $k' \in [z(i), z(j)]_R$, and further, any shortest path $\pi(s, w_{f(j)})$ from s to $w_{f(j)}$ is $\pi(s, q^*)$.*

For each $k' \in [z(i), z(j)]_R$, we apply our region-processing algorithm on $R_{k'}$ and $w_{f(j)}$ to obtain a path, and we keep the shortest path π among all such paths; let $q^l_{f(j)}$ be the endpoint of π on $w_{f(j)}$. According to Lemma 8, if q^* is on $w^l_{f(j)}$, then q^* must be $q^l_{f(j)}$.

For analyzing the total running time of our algorithm, as will be seen later, for each $k' \in [z(i), z(j)]_R$ with $k' \neq z(i)$ and $k' \neq z(j)$, the region-processing algorithm will not be called on $R_{k'}$ again in the entire algorithm for computing q^*_l. On the other hand, we charge the two algorithm calls on $R_{k'}$ for $k' = z(i)$ and $k' = z(j)$ to the index j of $\mathbb{B}$. In this way, the total number of calls to the region-processing procedure in the entire algorithm is $O(h^* + k)$ since the total number of indices of $\mathbb{B}$ is at most k and the total number of regions $R_{k'}$ is h^*.

The composite case. If B_b is composite, the algorithm is more complicated. Let j be the wrap index of B_b. Observation 4 and Lemma 8 still hold on j. However, since now the region D contains a portion of $w_{f(j')}$ for each $j' \in B_b \setminus \{j\}$ (see Fig. 7), D may also contain the shortest path from s to $w_{f(j')}$. In order to avoid calling the region-processing procedure on the same region of $\mathcal{R}$ too many times, we use the following approach to process $w_{f(j)}$.

For any two different indices of k' and k'' in a range $[k_1, k_2]_R$ of indices of the regions of $\mathcal{R}$, we say that k'' is *ccw-larger* than k' if $[k', k'']_R$ is a subset of $[k_1, k_2]_R$ (e.g., if $k_1 < k_2$, then $k' < k''$). Define z_{ij} to be the ccw-largest index in $[z(i), z(j)]$ such that $w_{f(j)}$ crosses $\partial R_{z_{ij}}$ (if no such index exists, then let $z_{ij} = z(i)$).

We first compute z_{ij} (to be discussed later). Then, we call the region-processing procedure on $R_{k'}$ for all $k' \in [z(i), z_{ij}]$ and return the shortest path π that is found; let $q^l_{f(j)}$ be the endpoint of π on $w_{f(j)}$. By the following lemma, if q^* is on $w^l_{f(j)}$, then $q^l_{f(j)}$ is q^*.

▶ **Lemma 9.** *If q^* is on $w^l_{f(j)}$, then $\pi(s, q^*)$ is in $R_{k'}$ for some index $k' \in [z(i), z_{ij}]_R$, and further, any shortest path $\pi(s, w_{f(j)})$ from s to $w_{f(j)}$ is $\pi(s, q^*)$.*

The following lemma makes sure that when we process $w_{f(j')}$ for any other index j' of B_b with $j' \neq j$, we do not need to consider the regions $R_{k'}$ for $k' \in [z(i), z_{ij} - 1]$ if $z_{ij} \neq z(i)$.

▶ **Lemma 10.** *Suppose $z_{ij} \neq z(i)$. If q^* is on $w^l_{f(j')}$ for some $j' \in B_b$ and $j' \neq j$, then $\pi(s, q^*)$ is in $R_{k'}$ for some $k' \in [z_{ij}, z(j')]_R$.*

In order to compute the index z_{ij}, we will use a $\mathcal{R}$-*region range* query. Namely, given the index range $[z(i), z(j)]_R$ as well as $w_{f(j)}$, the query can be used to compute z_{ij}. In the full paper, we give a data structure that can answer each such query in $O(\log h \log n)$ time, after $O(n \log h)$ time and space preprocessing.

After $w_{f(j)}$ is processed as above, $q^l_{f(j)}$ is computed. By Lemma 10, to process $w_{f(j')}$ for other indices j' of $B_b \setminus \{j\}$, we only need to consider the indices of the regions of $\mathcal{R}$ after z_{ij}. Let $B'_1, B'_2, \ldots, B'_{g'-1}$ be the bundles in B_b other than the last one. For each $1 \leq b' \leq g' - 1$, if $b' = 1$, we call $path(B'_{b'}, z_{ij})$ recursively; otherwise, we call $path(B'_{b'}, z(i'))$ recursively, where i' is the last index of $B'_{b'-1}$.

After $w_{f(j)}$ is processed for each $j \in \mathbb{B}$, $q^l_{f(j)}$ is computed for every $j \in \mathbb{B}$; among these at most k points, we return the point q' whose value $d(s, q')$ is the smallest as q^*_l, which is q^* based on our above analysis (and also due to our assumption that q^* is on w^l_i for some $i \in [1, k]$). The total number of calls on the region-processing procedures is $O(k + h^*)$. The total number of $\mathcal{R}$-region range queries is $O(k)$ since each such query is for a composite bundle and there are at most k bundles in total. Hence, the total time of the algorithm is $O((h + k) \log h \log n)$. Recall that $k \leq K$.

We summarize our overall algorithm in the following theorem.

▶ **Theorem 11.** *Given $SPM(s)$, we can build a data structure of $O(n \log h + h^2)$ size in $O(n \log h + h^2 \log h)$ time, such that each quickest visibility query can be answered in $O((K + h) \log h \log n)$ time, where K is the size of the visibility polygon of the query point q.*

Proof. In the preprocessing, we compute the visibility polygon query data structure in [4] for computing $Vis(q)$, which is of $O(n + h^2)$ size and can be built in $O(n + h^2 \log h)$ time. The rest of the preprocessing work includes building the decomposition $\mathcal{D}$ and the segment query data structure of Theorem 3, performing the preprocessing for computing the map $f(\cdot)$, for the region-processing algorithms, for answering SP-segment-intersection queries, for answering $\mathcal{R}$-region range queries, etc; these work takes $O(n \log h)$ time and space in total.

Given any query point q, we first compute $Vis(q)$ in $O(K \log n)$ time by the query algorithm in [4]. Then, we obtain the extended window set W. Let $k = |W|$, which is $O(K)$. Next, we compute a closest point q^* on a segment of W in $O(k \log h \log n)$ time. To this end, we compute a set S of $O(k)$ candidate points as follows. We first add $q, q_1, \ldots, q_k$ to S. Then, we compute the closest point q^*_0 of $\overline{u_0 q_0}$ and add q^*_0 to S. Next we compute the point q^*_l in $O((k + h) \log h \log n)$ time by using our pruning algorithm in Sections 4.3 and 4.4. By a symmetric algorithm, we can also compute q^*_r. We add both q^*_l and q^*_r to S. By our analysis, q^* must be one of the points of S. Since $|S| = O(k)$, we can find q^* in S in additional $O(k \log n)$ time by using the shortest path map $SPM(s)$. ◀

In fact, we have the following more general result, which might have independent interest.

▶ **Corollary 12.** *Given $SPM(s)$, we can build a data structure of $O(n \log h)$ size in $O(n \log h)$ time, such that given $k = O(n)$ segments in $\mathcal{P}$ intersecting at the same point, we can compute a shortest path from s to all these segments in $O((k + h) \log h \log n)$ time.*

Proof. The preprocessing step is the same as in Theorem 11 except that the visibility polygon query data structure [4] is not necessary any more. Hence, the total preprocessing time and

space is $O(n \log h)$. Given a set S of k segments intersecting at the same point, denoted by p, we break each segment at p to obtain two segments and we still use S to denote the new set of at most $2k$ segments. Next we compute a closest point p^* on the segments of S. To do so, we can apply the same algorithm as in Theorem 11 for computing q^* on the extended-windows of W. Indeed, the only key property of the segments of W we need is that all segments of W have a common endpoint at q. Now that all segments of S have a common endpoint p, the same algorithm still works. ◀

5 The Quickest Visibility Queries: The Improved Result

To further reduce the query time of Theorem 11 to $O(h \log h \log n)$, independent of K, the key idea is the following. First, we show that for any query point q, there exists a subset $\mathcal{S}(q)$ of $O(h)$ windows such that a closest point q^* is on a segment of $\mathcal{S}(q)$. This is done by making use of the extended corridor structure [3, 5]. Second, we give an algorithm that can compute $\mathcal{S}(q)$ in $O(h \log n)$ time, without computing $Vis(q)$, after $O(n \log h + h^2)$ space and $O(n \log h + h^2 \log h)$ time preprocessing. The result is obtained by modifying the query algorithm for computing $Vis(q)$ in [4]. Refer to our full paper [15] for all these details.

▶ **Theorem 13.** *Given $SPM(s)$, we can build a data structure of $O(n \log h + h^2)$ size in $O(n \log h + h^2 \log h)$ time, such that each quickest visibility query can be answered in $O(h \log h \log n)$ time.*

References

1. E. M. Arkin, A. Efrat, C. Knauer, J. S. B. Mitchell, V. Polishchuk, G. Rote, L. Schlipf, and T. Talvitie. Shortest path to a segment and quickest visibility queries. *Journal of Computational Geometry*, 7:77–100, 2016.
2. B. Chazelle, H. Edelsbrunner, M. Grigni, L. Guibas, J. Hershberger, M. Sharir, and J. Snoeyink. Ray shooting in polygons using geodesic triangulations. *Algorithmica*, 12(1):54–68, 1994.
3. D. Z. Chen and H. Wang. L_1 shortest path queries among polygonal obstacles in the plane. In *Proc. of 30th Symposium on Theoretical Aspects of Computer Science*, pages 293–304, 2013.
4. D. Z. Chen and H. Wang. Visibility and ray shooting queries in polygonal domains. *Computational Geometry: Theory and Applications*, 48:31–41, 2015.
5. D. Z. Chen and H. Wang. Computing the visibility polygon of an island in a polygonal domain. *Algorithmica*, 77:40–64, 2017.
6. Y. K. Cheung and O. Daescu. Approximate point-to-face shortest paths in $\mathcal{R}^3$. arXiv:1004.1588, 2010.
7. Y.-J. Chiang and R. Tamassia. Optimal shortest path and minimum-link path queries between two convex polygons in the presence of obstacles. *International Journal of Computational Geometry and Applications*, 7:85–121, 1997.
8. L. J. Guibas, J. Hershberger, D. Leven, M. Sharir, and R. E. Tarjan. Linear-time algorithms for visibility and shortest path problems inside triangulated simple polygons. *Algorithmica*, 2(1-4):209–233, 1987.
9. 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.
10. J. Hershberger and S. Suri. An optimal algorithm for Euclidean shortest paths in the plane. *SIAM Journal on Computing*, 28(6):2215–2256, 1999.

11 R. Khosravi and M. Ghodsi. The fastest way to view a query point in simple polygons. In *Proc. of the 24th European Workshop on Computational Geometry*, pages 187–190, 2005.

12 E. Melissaratos and D. Souvaine. Shortest paths help solve geometric optimization problems in planar regions. *SIAM Journal on Computing*, 21(4):601–638, 1992.

13 J. S. B. Mitchell. A new algorithm for shortest paths among obstacles in the plane. *Annals of Mathematics and Artificial Intelligence*, 3(1):83–105, 1991.

14 J. S. B. Mitchell. Shortest paths among obstacles in the plane. *International Journal of Computational Geometry and Applications*, 6(3):309–332, 1996.

15 H. Wang. Quickest visibility queries in polygonal domains. arXiv:1703.03048, 2017.

Zapping Zika with a Mosquito-Managing Drone: Computing Optimal Flight Patterns with Minimum Turn Cost*

Aaron T. Becker[1], Mustapha Debboun[2], Sándor P. Fekete[3], Dominik Krupke[4], and An Nguyen[5]

1 **Department of Electrical and Computer Engineering, University of Houston, Houston, TX, USA**
atbecker@uh.edu
2 **Mosquito & Vector Control Division, Harris County Public Health, Houston, TX, USA**
mdebboun@hcphes.org
3 **Dept. of Computer Science, TU Braunschweig, Braunschweig, Germany**
s.fekete@tu-bs.de
4 **Dept. of Computer Science, TU Braunschweig, Braunschweig, Germany**
d.krupke@tu-bs.de
5 **Department of Electrical and Computer Engineering, University of Houston, Houston, TX, USA**
an.nguyen.vn@ieee.org

Abstract

We present results arising from the problem of sweeping a mosquito-infested area with an Unmanned Aerial Vehicle (UAV) equipped with an electrified metal grid. This is related to the Traveling Salesman Problem, the Lawn Mower Problem and, most closely, Milling with Turn Cost. Planning a good trajectory can be reduced to considering penalty and budget variants of covering a grid graph with minimum turn cost. On the theoretical side, we show the solution of a problem from The Open Problems Project that had been open for more than 15 years, and hint at approximation algorithms. On the practical side, we describe an exact method based on Integer Programming that is able to compute provably optimal instances with over 500 pixels. These solutions are actually used for practical trajectories, as demonstrated in the video.

1998 ACM Subject Classification F.2.2 [Nonnumerical Algorithms and Problems] Geometrical Problems and Computations, I.2.9 [Robotics] Autonomous Vehicles

Keywords and phrases Covering tours, turn cost, complexity, exact optimization

Digital Object Identifier 10.4230/LIPIcs.SoCG.2017.62

Category Multimedia Contribution

1 Introduction

Consider an outdoor setting that is populated by swarms of mosquitoes, with a number of known hotspots. How can we lower the danger of diseases by zapping the mosquitos with a flying drone, such as the one shown in Fig. 1?

* This work was supported by the National Science Foundation Grant No. [CNS-1646607].

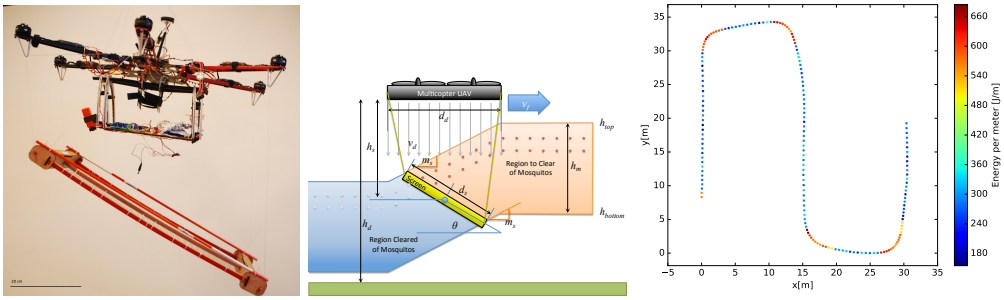


Figure 1 (Left) A drone equipped with an electrical grid for killing mosquitoes. (Middle) Physical aspects of the flying drone. (Right) Making turns is expensive.

Visiting a set of points by an optimal tour is a natural and important problem, both in theory and practice. If we are only concerned with minimizing the total distance traveled for visiting all points, we get the classic Traveling Salesman Problem (TSP). However, for path planning by a flying robot, we also incur a cost for changing direction, as illustrated in Fig. 1 (Right). This is related to the Angular-Metric TSP (AM-TSP), in which the objective is to minimize the total turn cost. In addition, we may want to focus on a subset of the points in order to provide better coverage, incurring a penalty for the uncovered ones.

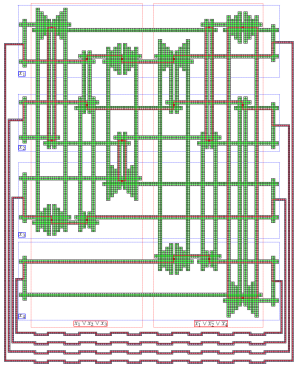
2 Related Work

The classic Traveling Salesman Problem (TSP) has enjoyed a huge amount of attention; see Cook [7] for a general overview, and Applegate et al. [2] for a more advanced textbook on computing provably exact solutions. For theoretical work on the Lawn Mower Problem, see Arkin, Fekete and Mitchell [5, 6]. Angle-restricted tour problems were studied by Fekete and Woeginger [10]. Touring points in the plane with minimal continuous turn cost was considered by Aggarwal et al. [1]. Arkin et al. [4] consider different grid-based versions of covering with turn cost, and provide a spectrum of approximation algorithms. They pose the complexity of finding a cycle cover with minimum total turn cost as an open problem, first published in the conference version in 2001 [3]; this problem gained additional attention by becoming part of The Open Problems Project [8] as Problem #53 in 2003.

3 Problems

We are given a grid graph, which arises as the dual graph from a set of pixels. We consider several different covering tour problems and their cycle cover relaxations. We identify three different ways that a pixel can be traversed, each with a different cost: straight, by a simple turn, and by a U-turn. The ratio between straight traversal and simple turns is arbitrary but fixed, while the cost of a U-turn is twice as much as for a simple turn. The following emerge for full coverage, cheap coverage of a subset, and coverage with a budget constraint.

- Given a grid graph, find a minimum-cost tour/cycle cover that covers all vertices.
- Given a grid graph and an individual penalty per vertex, find a minimum-cost tour/cycle cover in which instead of covering a vertex, its penalty may be paid.
- Given a grid graph, an individual value per vertex and a total budget, find a maximum-value tour/cycle cover of a subset of vertices, subject to the budget constraint.



	2D	3D	Hexagonal	General grids
Full cycle cover	4^*	6	6	2ω
Full tour	6^*	12	12	4ω
Subset cycle cover	4	6	6	2ω
Subset tour	10	14	14	$4\omega + 2$
Penalty cycle cover	6	8	8	$2(\omega + 1)$
Penalty tour	16	20	20	$4(\omega + 1) + 4$

Figure 2 (Left) The NP-completeness construction for the 1-in-3 3SAT instance. $(x_1 \vee x_2 \vee x_3) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_4})$. The provided solution is $x_1 = 0, x_2 = 1, x_3 = 0, x_4 = 1$. (Right) Approximation factors of our approximation techniques. ω is the maximum number of distinct full strips (or orientations) by which a grid point is covered. $(*)$ Note that Arkin et al. [4] achieve a factor of 2.5 on the number of turns for cycle cover and a factor of 3.75 on the number of turns for tours in 2-dimensional grid graphs. With distance costs it becomes a 4-approximation for cycle cover and tour.

4 Complexity and Approximation

We can prove that computing a minimum-turn cycle cover in 2-dimensional grid graphs is NP-hard. This solves Problem #53 in The Open Problems Project [8]. The hardness of the problem is not obvious: usually large parts of a solution can be easily deduced by local information and 2-factor techniques.

▶ **Theorem 1.** *It is NP-hard to find a cycle cover with a minimum number of turns in a 2-dimensional grid graph.*

See Fig. 2 (Left) for an idea of the construction. For a full proof, see Krupke [11] and the upcoming paper [9]. In addition, we can give a number of approximation algorithms; see Fig. 2 (Right) for an overview, and [11, 9] for details.

5 Integer Programming

The following is an Integer Programming formulation for finding an optimal budget cycle cover with turn cost.

$$\max \sum_{v \in V} d_v(1 - y_v) \tag{1}$$

$$\text{s.t.} \quad 1 \leq 4y_v + \sum_{\{u,w\} \subseteq N(v)} x_{v,\{u,w\}} \leq 4 \qquad \forall v \in V \tag{2}$$

$$2x_{v,\{w\}} + \sum_{u \in N(v), u \neq w} x_{v,\{w,u\}} = 2x_{w,\{v\}} + \sum_{u \in N(w), u \neq v} x_{w,\{u,v\}} \qquad \forall \{v,w\} \in E \tag{3}$$

$$\sum_{v \in V} \sum_{\{u,w\} \subseteq N(v)} \text{cost}(uvw) x_{v,\{u,w\}} \leq B \tag{4}$$

$$x_{v,\{u,w\}} \in \mathbb{N}_0, y_v \in \{0,1\} \qquad \forall v \in V, \{u,w\} \subseteq N(v)$$

Variable $x_{v,\{u,w\}}$ counts the traversals of v with end points in u and w, while y_v indicates whether a variable is left uncovered. The objective function in Eq. (1) sums up the densities

d_v of all pixels for which the *not covered* variable is false. Eq. (2) enforces a pixel to be covered or the *not covered* variable to be set to true. Arkin et al. [4] showed that no pixel needs to be visited more than four times, otherwise a simple local optimization can be performed. Eq. (3) enforces the transitions between two adjacent pixels to match. Eq. (4) limits the costs of the tour to the budget B (the battery runtime).

The separation of subtours is more complicated than for the classic TSP, because there may be subtours that cross but are not connected (due to turn cost); moreover, instead of connecting two subtours, one subtour can also be discarded. The following can separate any given solution with multiple subtours. Let C be the pixels of a selected subtour. Let p be a pixel in C not traversed by other subtours, and another covered pixel $p' \notin C$. C_s are the pixels that are traversed by the subtour without turning. $T(v)$ describes the 90°-turn variables of a pixel v. x' refers to the variable assignment in the current solution.

$$y_p + y_{p'} + \sum_{\substack{\{u,w\}\subseteq N(p) \\ x'_{p,\{u,w\}}=0}} x_{p,\{u,w\}} + \sum_{\substack{t\in T(v) \\ v\in C_s\setminus\{p\}}} t + \sum_{\substack{v\in C\setminus(C_s\cup\{p\}) \\ u\neq w\in V \\ x'_{v,\{u,w\}}=0}} x_{v,\{u,w\}} \geq 1 \tag{5}$$

6 The Video

The video opens with an introduction of the challenge of controlling mosquitoes with a UAV, followed by a discussion of geometric modeling aspects, leading to finding minimum-turn covering tours and the closely related minimum-turn cycle covers. The complexity of the latter is a long-standing open problem, whose solution is stated. For purposes of coverage with flying drones, the further variants with penalties and budget constraint are introduced, followed by a sketch of an approximation approach. An exact method based on Integer Programming is described next, which is able to solve relevant real-world instances to provable optimality. The video concludes by showing how such trajectories are used by the drone.

References

1 Alok Aggarwal, Don Coppersmith, Sanjeev Khanna, Rajeev Motwani, and Baruch Schieber. The angular-metric Traveling Salesman Problem. *SIAM Journal on Computing*, 29(3):697–711, 2000.

2 David L. Applegate, Robert E. Bixby, Vašek Chvátal, and William J. Cook. *The Traveling Salesman Problem: A computational study.* Princeton University Press, 2011.

3 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 Ann. ACM-SIAM Symp. Disc. Algorithms (SODA 2001)*, pages 138–147. SIAM, 2001.

4 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 Journal on Computing*, 35(3):531–566, 2005.

5 Esther M. Arkin, Sándor P. Fekete, and Joseph S. B. Mitchell. The lawnmower problem. In *Proc. 5th Canad. Conf. Sympos. Geom. (CCCG93)*, pages 461–466, 1993.

6 Esther M. Arkin, Sándor P. Fekete, and Joseph S. B. Mitchell. Approximation algorithms for lawn mowing and milling. *Computational Geometry*, 17(1):25–50, 2000.

7 William Cook. *In pursuit of the traveling salesman: Mathematics at the limits of computation.* Princeton University Press, 2012.

8 Erik D. Demaine, Joseph S. B. Mitchell, and O'Rourke Joseph. The open problems project. URL: `http://cs.smith.edu/~orourke/TOPP/`.

9 Sándor P. Fekete and Dominik Krupke. Covering tours and cycle covers with turn costs: Hardness and approximation. Manuscript, 2017.

10 Sándor P. Fekete and Gerhard J. Woeginger. Angle-restricted tours in the plane. *Computational Geometry*, 8(4):195–218, 1997.

11 Dominik Krupke. Algorithmic methods for complex dynamic sweeping problems. Master's thesis, Department of Computer Science, TU Braunschweig, Braunschweig, Germany, 2016.

Ruler of the Plane – Games of Geometry

Sander Beekhuis[1], Kevin Buchin[2], Thom Castermans[3], Thom Hurks[4], and Willem Sonke[5]

1 **Eindhoven University of Technology, Eindhoven, The Netherlands**
2 **Eindhoven University of Technology, Eindhoven, The Netherlands**
`k.a.buchin@tue.nl`
3 **Eindhoven University of Technology, Eindhoven, The Netherlands**
`t.h.a.castermans@tue.nl`
4 **Eindhoven University of Technology, Eindhoven, The Netherlands**
5 **Eindhoven University of Technology, Eindhoven, The Netherlands**
`w.m.sonke@tue.nl`

Abstract

Ruler of the Plane is a set of games illustrating concepts from combinatorial and computational geometry. The games are based on the art gallery problem, ham-sandwich cuts, the Voronoi game, and geometric network connectivity problems like the Euclidean minimum spanning tree and traveling salesperson problem.

1998 ACM Subject Classification F.2.2 [Nonnumerical Algorithms and Problems] Geometrical Problems and Computations

Keywords and phrases art gallery problem, ham-sandwich cuts, Voronoi game, traveling salesperson problem

Digital Object Identifier 10.4230/LIPIcs.SoCG.2017.63

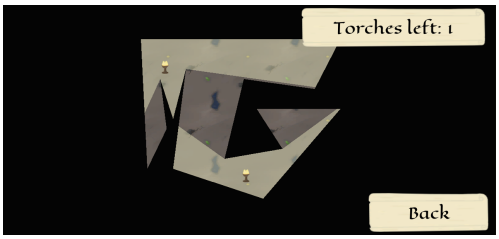
Category Multimedia Contribution

1 Concept

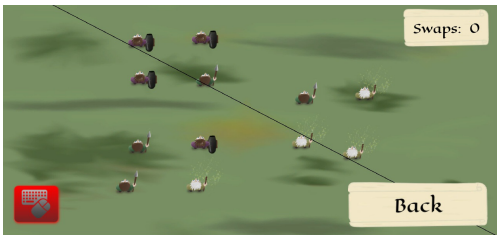
Geometry being inherently tangible, lends itself to be the base of puzzles and games. *Ruler of the Plane* is a set of four games with a medieval theme illustrating concepts from combinatorial and computational geometry. The games are based on the art gallery problem, ham-sandwich cuts, the Voronoi game, and geometric network connectivity problems like the Euclidean minimum spanning tree and traveling salesperson problem (TSP), see Figure 1.

The games also aim at providing the interested player with background on the geometric algorithms and data structures needed to implement such games. They do so by providing some pointers to geometric concepts in the game explanations, and by allowing to visualize some of the underlying data structures. For instance, the game on the ham-sandwich cuts can show the dual arrangements of the different color classes, the Voronoi game allows to show the Delaunay triangulation and empty circles. Furthermore, the games are open source and implemented using *C#* in the game engine Unity, and therefore provide the possibility to explore the underlying algorithms and data structures.

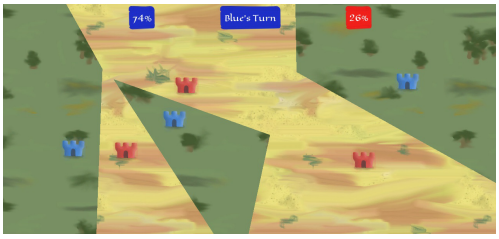
The geometric problems and the underlying algorithms and data structures of the games are common content of a Computational Geometry course. We developed the game primarily to introduce students taking such a course to these concepts in an entertaining way. An additional goal is to provide a stepping stone to introduce Combinatorial and Computational Geometry and also other algorithmic concepts like NP-hardness problems to a wider audience.



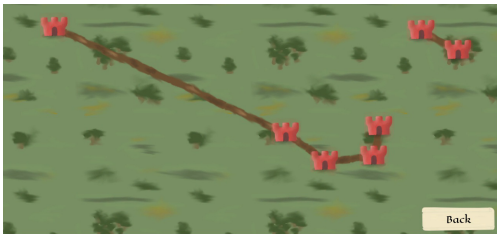
(a) *illuminate* a dungeon



(b) *divide* forces into equal units



(c) *conquer* new lands



(d) *connect* your new settlements

Figure 1 Games in *Ruler of the Plane*.

2 The Games

The game *conquer* implements the classical Voronoi game [1]: Two players place castles in turn, and the player whose Voronoi regions occupy the most area at the end wins.

The Voronoi diagram is implemented as dual of the Delaunay triangulation. The Delaunay triangulation is constructed using an implementation of a textbook randomized incremental construction [6]. Out of the four games, this is the only two-player game. To demonstrate the underlying geometry the game allows to toggle the Voronoi diagram, empty circumcircles and the dual Delaunay triangulation (see Figure 2a).

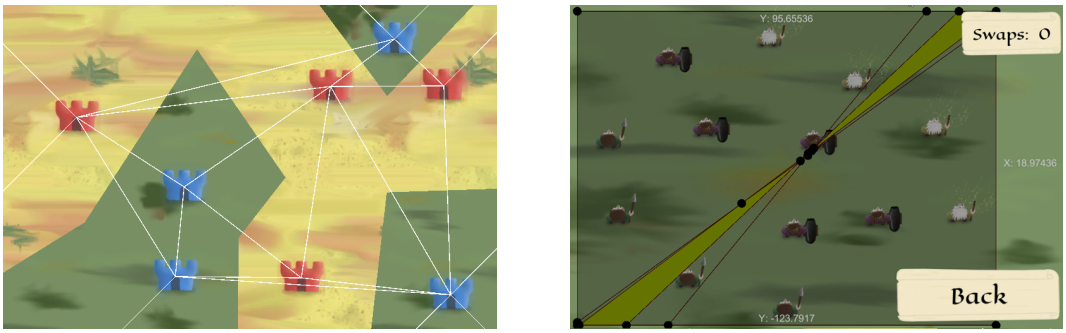
The game *divide* implements the two-dimensional ham sandwich cuts [9], but with three types of points. That is, the player needs to find a line that splits all three types of points in half. Some levels also ask to swap positions of points, before drawing a cut.

In a course on computational geometry, ham sandwich cuts are commonly covered as an application of duality and arrangements. In this context typically a simple $O(n^2)$-time algorithm is discussed: dualizing the points, computing the line arrangements and intersecting the $n/2$ levels. This is also the algorithm implemented in the game. The game allows to toggle possible cuts and the dual arrangements (see Figure 2b).

The game *connect* consists of three separate games. In the first the player has to find the Euclidean minimum spanning tree, in the second a Euclidean traveling salesperson tour, and in the third a 1.5-spanner [11] of short length. While in the first game the player has to find the exact tree, in the two other games the player has to beat an approximation computed by the game, namely Christofides algorithm [7] and the greedy spanner [4].

We included the TSP with Christofides algorithm and minimum spanning trees, since they are very natural geometric problems, suitable to discuss computational complexity with a wider audience, and since they often feature in other algorithms courses. spanners are often discussed in the context of well-separated pair decompositions.

The spanner game also provides a limited number of 'hints' in the form of the next edge the greedy spanner would add. After exhausting the base levels, the game continues with levels that ask to connect randomly generated sites.



(a) Delaunay triangulation of the castles **(b)** cells in the dual arrangement

Figure 2 Visualizing the underlying algorithms and data structures.

The game *illuminate* is an implementation of art gallery problem [2] with point guards in a simple polygon. In a Computational Geometry course, the art gallery problem with vertex guards is often discussed as a motivation for polygon triangulation, but is also interlinked with other topics, like visibility computation and boolean operations on polygons. The game computes visibility regions by a circular sweep. To remove duplicate regions it then uses the Weiler–Atherton algorithm [13]. The implementation is not yet robust, and therefore only small levels are included in the game.

3 Educational Context

As described above the games are intended for demonstration purposes for students of Computational Geometry and for a wider audience. However, also the game development was embedded in an educational context.

Various concepts for games where first implemented and tested as course projects in Computational Geometry. Some of these concepts where then integrated into the game. Most of *Ruler of the Plane* was then implemented by Master students after taking a course in Computational Geometry, partially as practical component to a reading course on algorithm engineering [10] and robust geometric algorithms [12], partially as student assistantship. The task to extend the games may result in engaging future course projects. Ruler of the Plane is open-source using *C#* in the game engine Unity, and therefore lends itself to such extensions.

4 Future Work

The games leave many opportunities for future work from designing interesting levels and variants, to improving and providing alternative implementations, to designing games on other geometric and algorithmic topics. In the following we discuss some more concrete ideas.

The Voronoi game allows for a very simple, effective strategy, which in the basic variant makes the game *conquer* less interesting. Including other variants would make the game more multi-facetted. These could include castles with different magnitudes/ranges of influence, or region of different worth, or restrictions on where castles can be placed. Also a puzzle variant where castles of one color are already placed, and the player only places castles of the other color could be challenging.

Currently the game *divide* has a small number of levels. It would be possible to generate additional levels based on random instances, but the question of generating challenging levels remains open. And again, more variants could bring more variation to the game.

In the game *connect* Christofides algorithm may be instructive, but the results are quite easy to beat. To demonstrate the NP-hardness of the problem, it would be interesting to include small, difficult instances. Other TSP heuristics would be instructive as 'hints'.

Generating interesting levels for the art gallery problem in the game *illuminate* seems challenging. Possible starting points could be gadgets used in NP-hardness constructions [8] and families of polygons used in experimental evaluations [5]. The art gallery problem has many variants, and more of these would again bring more variation to the game. In particular vertex guards would provide a game more closely to the art gallery problem as motivation to polygon triangulation.

So far games about four topics have been implemented. Obviously a course on Computational Geometry [6] leaves room for more games on other topics. For some topics, for instance orthogonal range searching, it might be more challenging to design an interesting game. And then there are other topics, which are intuitively accessible and seem to lend themselves as a base of a game, e.g, man-and-dog problems [3].

5 Resources

The games can be played online at `http://www.win.tue.nl/~kbuchin/proj/ruler/webgl/`. The game can be downloaded from `http://www.win.tue.nl/~kbuchin/proj/ruler/` and its sources from `https://github.com/kbuchin/ruler`. A video demonstrating the game is available at `http://www.win.tue.nl/~kbuchin/proj/ruler/video/PlaneRuler.mp4`.

Acknowledgments. The development of the game was supported by the TU/e Educational award.

References

1 Hee-Kap Ahn, Siu-Wing Cheng, Otfried Cheong, Mordecai Golin, and Rene Van Oostrum. Competitive facility location: the Voronoi game. *Theoretical Computer Science*, 310(1-3):457–467, 2004.

2 Martin Aigner and Günter M. Ziegler. *Proofs from THE BOOK*. Springer, 4th edition, 2009.

3 Helmut Alt and Michael Godau. Computing the Fréchet distance between two polygonal curves. *Int. J. Comput. Geom. Appl.*, 5:75–91, 1995.

4 Ingo Althöfer, Gautam Das, David Dobkin, Deborah Joseph, and José Soares. On sparse spanners of weighted graphs. *Discrete & Computational Geometry*, 9(1):81–100, 1993.

5 Yoav Amit, Joseph S. B. Mitchell, and Eli Packer. Locating guards for visibility coverage of polygons. *Int. J. Comput. Geom. Appl.*, 20:601–630, 2010.

6 Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark Overmars. *Computational Geometry: Algorithms and Applications*. Springer, 3rd edition, 2008.

7 Nicos Christofides. Worst-case analysis of a new heuristic for the travelling salesman problem. Technical Report 388, Graduate School of Industrial Administration, Carnegie Mellon University, 1976.

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

9 Chi-Yuan Lo, Jiří Matoušek, and William Steiger. Algorithms for ham-sandwich cuts. *Discrete & Computational Geometry*, 11(4):433–452, 1994.

10 Matthias Müller-Hannemann and Stefan Schirra, editors. *Algorithm engineering: bridging the gap between algorithm theory and practice*, volume 5971 of *LNCS*. Springer, 2010.

11 Giri Narasimhan and Michiel Smid. *Geometric Spanner Networks*. Cambridge University Press, New York, NY, USA, 2007.

12 Jonathan Richard Shewchuk. Lecture notes on geometric robustness, 2013.

13 Kevin Weiler and Peter Atherton. Hidden surface removal using polygon area sorting. *ACM SIGGRAPH computer graphics*, 11(2):214–222, 1977.

Folding Free-Space Diagrams: Computing the Fréchet Distance between 1-Dimensional Curves*

Kevin Buchin[1], Jinhee Chun[2], Maarten Löffler[3], Aleksandar Markovic[4], Wouter Meulemans[5], Yoshio Okamoto[6], and Taichi Shiitada[7]

1 **Eindhoven University of Technology, Eindhoven, The Netherlands**
k.a.buchin@tue.nl
2 **Tohoku University, Sendai, Japan**
jinhee@dais.is.tohoku.ac.jp
3 **Utrecht University, Utrecht, The Netherlands**
m.loffler@uu.nl
4 **Eindhoven University of Technology, Eindhoven, The Netherlands**
a.markovic@tue.nl
5 **Eindhoven University of Technology, Eindhoven, The Netherlands**
w.meulemans@tue.nl
6 **University of Electro-Communications, Chofu, Japan**
okamotoy@uec.ac.jp
7 **University of Electro-Communications, Chofu, Japan**
shiitada@gmail.com

Abstract

By folding the free-space diagram for efficient preprocessing, we show that the Fréchet distance between 1D curves can be computed in $O(nk \log n)$ time, assuming one curve has ply k.

1998 ACM Subject Classification F.2.2 [Nonnumerical Algorithms and Problems] Geometrical Problems and Computations

Keywords and phrases Fréchet distance, ply, k-packed curves

Digital Object Identifier 10.4230/LIPIcs.SoCG.2017.64

Category Multimedia Contribution

1 Introduction

The Fréchet distance is a popular similarity metric in computational geometry. Computing this distance between curves is mostly well understood: slightly super-quadratic time algorithms for 2D (or higher) are known [1, 5], with a nearly matching conditional lower bound: a $O(n^{2-\delta})$-time algorithm with $\delta > 0$ would imply that the strong exponential time hypothesis (SETH) fails [2]. For the discrete variant in 1D, this same lower bound is also known [4]. However, in the continuous case in 1D, no non-trivial lower bound is known; the fastest known algorithm runs in quadratic time [6]. A near-linear time algorithm is known [3] when the curves are separated: essentially a greedy strategy works. Thus we ask: can we compute the Fréchet distance in 1D in subquadratic time?

* K.B. is supported by NWO (612.001.207); W.M. by NLeSC (027.015.G02) and NWO (639.023.208); and Y.O. by Kayamori Foundation of Informational Science Advancement, JST CREST (JPMJCR1402), and JSPS KAKENHI (JP24106005, JP15K00009).

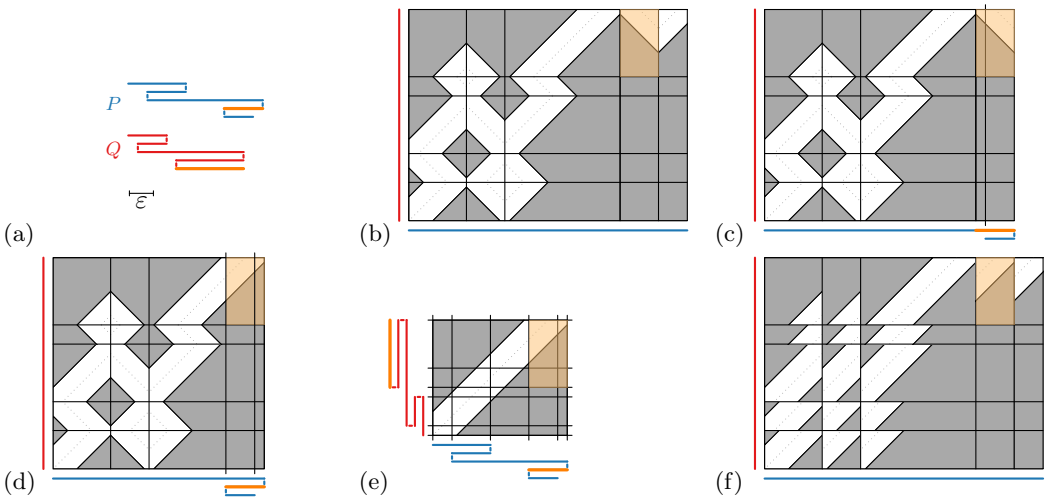


Figure 1 (a) Two 1D curves, lifted to 2D for legibility. (b) The corresponding free-space diagram $\mathcal{F}$. (c–d) Folding up the last two columns. (e) The result of folding all rows and columns. (f) View of each cell in $\mathcal{F}$ from the folded diagram. One orange cell is high-lighted to track through the process.

Contributions. While we leave the question open in general, we answer positively for the case that the number k of times one of the curves revisits the same point on $\mathbb{R}$ is bounded; no restrictions are posed on the other curve. This gives us a running time of $O(nk \log n)$. The crucial ingredient is a structural insight in the free-space diagram for 1D curves: we can fold it to find a simpler representation and solve the decision version in $O(n(k + \log n))$ time.

In 1D, a k-packed curve has a ply of $\Theta(k)$, and vice versa. Near-linear approximation algorithms for k-packed curves exist [3, 7], with matching conditional lower bounds in higher dimensions [3]. Our results show that efficient exact algorithms exist for 1D k-packed curves.

Preliminaries. A 1D curve P is described by $n + 1$ vertices $\langle p_0, \ldots, p_n \rangle$, with $p_i \in \mathbb{R}$ and n edges $p_{i-1}p_i$. We assume that the direction of the curve changes at every vertex. $|P|$ denotes the geometric length of P; we use $|p_i| = \sum_{j=1}^{i} |p_{j-1} - p_j|$ to denote the length up to vertex p_i. Note that $|p_0| = 0$ and $|p_n| = |P|$. We also consider P as a continuous function $P\colon [0, |P|] \to \mathbb{R}$ with $P(|p_i|) = p_i$ and linearly interpolated between vertices. The *ply* of a curve P is $\max_{r \in \mathbb{R}} |P^{-1}(r)| = \max_{r \in \mathbb{R}} |\{t \mid t \in [0, |P|] \wedge P(t) = r\}|$.

For a definition of the Fréchet distance d_{F}, see [1]. The free-space diagram $\mathcal{F}$ represents the parameter space of two curves [1]. It is a $[0, |P|] \times [0, |Q|]$ diagram and a point $(s, t) \in \mathcal{F}$ represents a pair of curve points: $P(s)$ and $Q(t)$. $(s, t) \in \mathcal{F}$ is *free* if $|P(s) - Q(t)| \leq \varepsilon$; the union of free points is the *free space* of $\mathcal{F}$ (Fig. 1(b)). If a monotone path from $(0, 0)$ to (s, t) exists in the free space, $(s, t) \in \mathcal{F}$ is *reachable*. Then, $d_{\mathrm{F}}(P, Q) \leq \varepsilon$ holds if and only if $(|P|, |Q|)$ is reachable in $\mathcal{F}$ [1]. Each edge $p_{i-1}p_i$ of P is a column of width $|p_{i-1} - p_i|$ in the free-space diagram; edges of Q define rows. A combination of two edges defines a cell.

2 Folding free-space diagrams

The central idea for our results is that we can *fold* the free-space diagram, see Fig. 1. This works for the following reason. First, pick a vertex p_i of P and a point q on Q. Now, consider points p on $p_{i-1}p_i$ and p' on p_ip_{i+1} that are equidistant to p_i. Due to minimality, $p = p'$, and thus $|p - q| \leq \varepsilon$ if and only if $|p' - q| \leq \varepsilon$. Considering the vertical line in $\mathcal{F}$ represents p_i, the free space before and after this line are thus a reflection of each other.

Folding up all columns and rows (Fig. 1(c–d)), we find one basic "cell" with lines specifying certain views that the edges of the curves give on this space (Fig. 1(e)). We can compose all these views to recover $\mathcal{F}$ (Fig. 1(f)): assuming both curves start in a rightward direction, we need to only reflect cells in even rows vertically and in even columns horizontally.

With a physical sheet of paper, the size of an unfolded free-space diagram, we only need to fold each row and column, and cut along the two remaining boundaries of the free space. If we unfold it, we have the free space of $\mathcal{F}$. Computing this quadratically sized structure can thus be done in a linear number of operations, using the "parallelism" of the cut.

3 One curve with bounded ply

Here, we sketch a proof for the decision algorithm, which uses folding, in the theorem below.

▶ **Theorem 1.** *Let P and Q be two 1D curves of complexity n; let the ply of Q be k. There is an algorithm to decide in $O(n(k+\log n))$ time whether $d_{\mathrm{F}}(P,Q) \leq \varepsilon$ and an algorithm to compute $d_{\mathrm{F}}(P,Q)$ in $O(nk \log n)$ time.*

We fold all columns (but not the rows) to obtain the *free-space structure* between the infinite line and Q. Consider the points q on Q for which $\|q - r\| \leq \varepsilon$ for some $r \in \mathbb{R}$: the free space on the vertical line at r in the free-space structure. We focus on the *maximal intervals* of free space along such a line. Each interval is bounded by two edges $q_{i-1}q_i$ and $q_{j-1}q_j$ such that $i < j$, $|q_{i-1} - r| > \varepsilon$ and $|q_j - r| > \varepsilon$. Our algorithm consists of the following three steps.

1. Decompose the free-space structure into vertical slabs. Each slab has the same structure of free space, that is, defined by the same edges of Q. (Lemma 3)
2. Precompute reachability information for the free-space structure. (Lemma 4)
3. Walk through the free-space diagram on a column-by-column basis, keeping track of intervals on Q that are reachable at vertex p_i. (Lemma 6)

▶ **Lemma 2.** *There are at most $k+1$ maximal intervals in Q for a given $r \in \mathbb{R}$.*

▶ **Lemma 3** (Decomposition structure)**.** *We can compute in $O(nk)$ time and space, a data structure $\mathcal{S}$ such that: (1) each slab stores the $O(k)$ edges of Q that bound the maximal intervals in order; (2) we can find the slab that contains a point $r \in \mathbb{R}$ in $O(\log n)$ time.*

Proof Sketch. We sort events (vertices of Q, $\pm\varepsilon$) in $O(nk)$ time using a linked list, starting at the previous event. Then, a sweepline decomposes the structure into slabs; only the maximal interval(s) at the event change. Query (2) is a binary search on the sorted events. ◀

▶ **Lemma 4** (Reachability structure)**.** *In $O(nk)$ time, we can determine for each edge $q_{i-1}q_i$ in Q the highest reachable point $\mathcal{H}(q_{i-1}q_i)$ in rightward direction in the free-space structure, starting from any point $(r,q) \in \mathbb{R} \times [0,|Q|]$ with $|r-q| \leq \varepsilon$, $|r-q_i| > \varepsilon$ and $q \in q_{i-1}q_i$.*

Proof Sketch. We do this by walking down the free-space structure, computing or using $\mathcal{H}$ for the boundary that we hit when we shoot a vertical ray up from (r,q). Using $\mathcal{S}$ (Lemma 3), we can answer such queries in constant time: we obtain a running time of $O(nk)$. ◀

We need the reachability structure $\mathcal{H}$ (Lemma 4) also in leftward direction, for leftward edges of P. For simplicity, we describe the algorithm only for rightward edges. For each vertex p_i of P, we compute the (maximal) reachable intervals: intervals reachable from $(0,0)$ in $\mathcal{F}$. As a maximal interval contains at most one reachable interval, a vertex has $O(k)$ reachable intervals (Lemma 2). Lemma 5 captures what is reachable at p_{i+1} from a reachable interval at p_i. We process $\mathcal{F}$ column by column, as formalized by Lemma 6 below. Afterwards, $d_{\mathrm{F}}(P,Q) \leq \varepsilon$ if and only if q_n is in a reachable interval of p_n.

▶ **Lemma 5.** *Let (r, q) be a point in $\mathcal{S}$; $r'' \geq r$; $q_{i-1}q_i$ the edge corresponding to the boundary of the free-space structure that we hit when shooting a ray up from (r, q); and (r', q') the highest reachable point in the free-space structure from (r, q) with $r' \leq r''$. Either $(r', q') = \mathcal{H}(q_{i-1}q_i)$ or the boundary slope at (r', q') is positive and $r' = r''$.*

Proof Sketch. We distinguish three cases for (r', q'): (1) If (r', q') is not on the boundary of the free space, this immediately contradicts the definition of (r', q'); (2) if (r', q') is on an upward-sloped boundary, either $r'' = r'$ or we can find a higher reachable point; (3) if (r', q') is on a downward-sloped boundary and monotonicity prevents us from going higher to the left, (r', q') must actually be equal to $\mathcal{H}(q_{i-1}q_i)$. ◀

▶ **Lemma 6.** *Given the decomposition structure $\mathcal{S}$, the reachability structure $\mathcal{H}$ and the sorted reachable intervals at p_i, we can compute the sorted reachable intervals at p_{i+1} in $O(\log n + k)$ time.*

Proof Sketch. We use the decomposition structure $\mathcal{S}$ to find the slab of p_{i+1} in $O(\log n)$ time (Lemma 3). Let A denote the maximal intervals at p_{i+1}, derived from the slab. Let E be the reachable intervals at p_i. We use $\mathcal{H}(E[j])$ as the highest reachable point in the free-space structure, stored in $\mathcal{H}$ (Lemma 4) with the edge defining the upper end of $E[j]$.

We walk over E and A simultaneously, with indices j and x respectively, to find all reachable intervals for p_{i+1}. We repeatedly do one of the following, checking them in order. (1) If the lower bound of $E[j]$ is above the upper bound of $A[x]$, we increment x by one. (2) If $\mathcal{H}(E[j])$ is below the lower bound of $A[x]$, we increment j by one. (3) If $|q - p_{j-1}| \leq \varepsilon$ where q is the point on Q represented by the lower end of $E[j]$, then we can cut across straight from $E[j]$ at q into interval $A[x]$ and we record a reachable interval starting at q and ending at $A[x]$. (4) Finally, if none of the previous cases apply, we can reach the lower end of $A[x]$ and $A[x]$ is completely reachable. At the end of (3) and (4), we increment as follows: if the upper bound of $A[x]$ is upward sloped, we increment j until we find a value such that $E[j]$ is above $A[x]$ (or run out of values in E). Then, we increment x by one.

Each case takes $O(1)$ time, thus this runs in $O(|A| + |E|) = O(k)$ time (Lemma 2). Including the query in $\mathcal{S}$, this results in a total time of $O(\log n + k)$. Correctness follows from the invariant that we collected all reachable intervals before $A[x]$, that intervals before $E[j]$ cannot reach up to $A[x]$ or later intervals, and that intervals before $A[x]$ do not limit how high $E[j]$ can reach (via Lemma 5). ◀

Acknowledgments. This research was initiated at the Dutch-Japanese Bilateral Workshop on Kinetic Geometric Networks, supported under the Japan-Netherlands Research Cooperative Program by NWO (grant 040.05.033) and JSPS.

References

1 H. Alt and M. Godau. Computing the Fréchet distance between two polygonal curves. *IJCGA*, 5(1–2):78–99, 1995.

2 K. Bringmann. Why walking the dog takes time: Fréchet distance has no strongly subquadratic algorithms unless SETH fails. In *Proc. 55th FOCS*, pages 661–670, 2014.

3 K. Bringmann and M. Künnemann. Improved approximation for Fréchet distance on c-packed curves matching conditional lower bounds. In *Proc. 26th ISAAC*, pages 517–528, 2015.

4 K. Bringmann and W. Mulzer. Approximability of the discrete fréchet distance. *JoCG*, 7(2):46–76, 2016.

5 K. Buchin, M. Buchin, W. Meulemans, and W. Mulzer. Four Soviets walk the dog: improved bounds for computing the Fréchet distance. *DCG*, 2017.

6 K. Buchin, M. Buchin, R. van Leusden, W. Meulemans, and W. Mulzer. Computing the Fréchet distance with a retractable leash. *DCG*, 56(2):315–336, 2016.

7 A. Driemel, S. Har-Peled, and C. Wenk. Approximating the Fréchet distance for realistic curves in near linear time. In *Proc. 26th SoCG*, pages 365–374, 2010.

Cardiac Trabeculae Segmentation: an Application of Computational Topology

Chao Chen*[1], Dimitris Metaxas[2], Yusu Wang[3], and Pengxiang Wu[4]

1 **City University of New York, Queens College and Graduate Center, Flushing, NY, USA**
chao.chen.cchen@gmail.com
2 **Rutgers University, New Brunswick, NJ, USA**
dnm@rutgers.edu
3 **Ohio State University, Columbus, OH, USA**
yusu@cse.ohio-state.edu
4 **Rutgers University, New Brunswick, NJ, USA**
pw241@rutgers.edu

Abstract

In this video, we present a research project on cardiac trabeculae segmentation. Trabeculae are fine muscle columns within human ventricles whose both ends are attached to the wall. Extracting these structures are very challenging even with state-of-the-art image segmentation techniques. We observed that these structures form natural topological handles. Based on such observation, we developed a topological approach, which employs advanced computational topology methods and achieve high quality segmentation results.

1998 ACM Subject Classification F.2.2 Geometric Problems and Computations

Keywords and phrases image segmentation, trabeculae, persistent homology, homology localization

Digital Object Identifier 10.4230/LIPIcs.SoCG.2017.65

Category Multimedia Contribution

1 Problem

The interior of a human cardiac ventricle is filled with fine structures including the papillary muscles and the *trabeculae*, i.e., muscle columns of various width whose both ends are attached to the ventricular wall (Figure 1). Accurately capturing these fine structures are very important in understanding the functionality of human hearts and in the diagnostic of cardiac diseases. These structures compose 23% of left ventricle (LV) end-diastolic volume in average and thus is critical in accurately estimating any volume-based metrics, e.g., ejection fraction (EF) and myocardial mass; these measures are critical in most cardiac disease diagnostics. A detailed interior surface model will also be the basis of a high quality ventricular flow simulation [10], which reveals deeper insight into the cardiac functionality of patients with diseases like hypokinesis and dyssynchrony.

With modern advanced imaging techniques, e.g., Computed Tomography (CT), we can capture details within cardiac ventricles (Fig. 1(left)). However, most state-of-the-art cardiac analysis methods [13, 12], although very efficient, can not accurately capture these complex

* Chao Chen's research is partially supported by PSC-CUNY-69844-00-47.

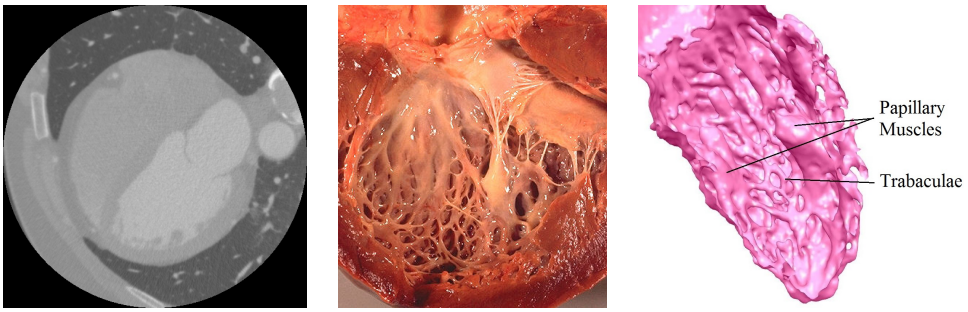


Figure 1 Left: our input CT image. Middle: interior of LV [7]. Right: our result (a 3D triangle mesh) successfully captures the trabeculae.

structures. The challenge is twofold. First, large variation of geometry and intensity of trabeculae makes it difficult to distinguish them from noise. Second, most segmentation models, e.g., region competition [14] and Markov random field [1], employ global priors, which tend to work against fine structures. A prior is certain function that measures the certain quality of the segmentation results. By optimizing such prior while fitting the segmentation result to the data, we achieve a segmentation with certain desired properties. In most segmentation models, a smoothness prior is employed, which prefers a simplified segmentation result and thus removes fine structures that we want to capture.

2 A Topological Approach

We exploit novel global information which is more suitable for the extraction of trabeculae, namely, the *topological prior*. A trabeculae is naturally a *topological handle*; both of its ends are attached to the wall, while the intermediate section is freely mobile. We propose a topological method that explicitly computes topological handles which are salient compared with their surrounding regions. The saliency is measured based on the theory of *persistent homology* [6] and can be computed efficiently. To improve the quality of the extracted handle, we further optimize the cycle representing such handle. The optimization is based on the previous methods from homology localization [3, 5, 8, 4, 2]. We propose an A$*$ search strategy to further improve the practical performance of the method.

Our system has the following modules. First, we localize the location of the left ventricle and enhance the image. This way our method could be more focused and more efficient. Second, we extract the interior surface model using traditional image segmentation methods, in particular, region competition. As show in Figure 2 Middle-Left, such method will give us a reasonable result but missing most trabeculae structures. Third, our system identifies topological handles by computing persistence homology using the intensity function of the image. Persistent dots on the diagram with high persistence (based on hand-selected threshold) are chosen as hypothetical trabeculae structures. We also filter these structures using a classifier trained on the geometric features. Fourth, we extract cycles representing these topological structures. We compute the optimal representative cycle, namely, the shortest cycle (Figure 2 Middle-Right). The remaining structures are considered the true signal and are included in the final segmentation (Figure 2 Right).

The related publications include [11, 9]. Source code for computing the shortest 1D cycle representing each persistent dot can be found at the first author's webpage: `http://eniac.cs.qc.cuny.edu/cchen`.

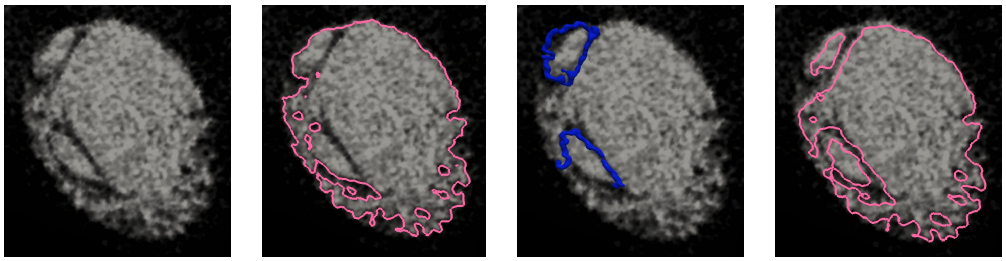


Figure 2 Left: a 2D slice of the intensity function, shaded bridges through the white regions are trabeculae. Middle-left: existing methods will miss the trabeculae completely. Middle-right: our method recovers missed trabeculae using persistent homology. Right: including these trabeculae in the final segmentation gives a better quality segmentation.

References

1 Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on pattern analysis and machine intelligence*, 23(11):1222–1239, 2001.

2 Oleksiy Busaryev, Sergio Cabello, Chao Chen, Tamal K. Dey, and Yusu Wang. Annotating simplices with a homology basis and its applications. In *Scandinavian Workshop on Algorithm Theory*, pages 189–200. Springer Berlin Heidelberg, 2012.

3 Erin W. Chambers, Jeff Erickson, and Amir Nayyeri. Minimum cuts and shortest homologous cycles. In *Proceedings of the 25th Annual Symposium on Computational Geometry*, pages 377–385. ACM, 2009.

4 Chao Chen and Daniel Freedman. Hardness results for homology localization. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1594–1604, 2010.

5 Chao Chen and Daniel Freedman. Measuring and computing natural generators for homology groups. *Computational Geometry*, 43(2):169–181, 2010.

6 H. Edelsbrunner and J. Harer. *Computational topology: an introduction*. American Mathematical Society, 2010.

7 Jr. Edwin P. Ewing. Gross pathology of idiopathic cardiomyopathy – Wikipedia, the free encyclopedia, 2016. [Online; accessed 09-December-2016].

8 Jeff Erickson and Amir Nayyeri. Minimum cuts and shortest non-separating cycles via homology covers. In *Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1166–1176. Society for Industrial and Applied Mathematics, 2011.

9 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 *Information Processing in Medical Imaging (IPMI)*, 2013.

10 Scott Kulp, Mingchen Gao, Shaoting Zhang, Zhen Qian, Szilard Voros, Dimitris Metaxas, and Leon Axel. Using high resolution cardiac CT data to model and visualize patient-specific interactions between trabeculae and blood flow. In *MICCAI*, LNCS, pages 468–475. 2011. doi:10.1007/978-3-642-23623-5_59.

11 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 *Information Processing in Medical Imaging (IPMI)*, 2017.

12 Xiantong Zhen, Heye Zhang, Ali Islam, Mousumi Bhaduri, Ian Chan, and Shuo Li. Direct and simultaneous estimation of cardiac four chamber volumes by multioutput sparse regression. *Medical Image Analysis*, 2016.

13 Yefeng Zheng, A. Barbu, B. Georgescu, M. Scheuering, and D. Comaniciu. Four-chamber heart modeling and automatic segmentation for 3D cardiac CT volumes using marginal space learning and steerable features. *TMI*, 27(11):1668–1681, nov. 2008.

14 S. C. Zhu, T. S. Lee, and A. L. Yuille. Region competition: unifying snakes, region growing, energy/Bayes/MDL for multi-band image segmentation. In *ICCV*, pages 416–423, June 1995.

MatchTheNet – An Educational Game on 3-Dimensional Polytopes

Michael Joswig[*1], **Georg Loho**[2], **Benjamin Lorenz**[3], **and Rico Raber**[4]

1 **Institut für Mathematik, MA 6-2, Technische Universität Berlin, Berlin, Germany**
joswig@math.tu-berlin.de
2 **Institut für Mathematik, MA 6-2, Technische Universität Berlin, Berlin, Germany**
loho@math.tu-berlin.de
3 **Institut für Mathematik, MA 6-2, Technische Universität Berlin, Berlin, Germany**
lorenz@math.tu-berlin.de
4 **Institut für Mathematik, MA 6-2, Technische Universität Berlin, Berlin, Germany**
raber@math.tu-berlin.de

Abstract

We present an interactive game which challenges a single player to match 3-dimensional polytopes to their planar nets. It is open source, and it runs in standard web browsers.

1998 ACM Subject Classification I.3.5 Computational Geometry and Object Modeling

Keywords and phrases three-dimensional convex polytopes; unfoldings

Digital Object Identifier 10.4230/LIPIcs.SoCG.2017.66

Category Multimedia Contribution

1 Introduction

A *polytope* is the convex hull of finitely many points in Euclidean space. While their study goes back to antiquity, polytopes are still an active research topic; see, e.g., Ziegler [6]. The *dimension* of a polytope is the dimension of its affine span. The first non-trivial class of polytopes are the 3-polytopes, i.e, those of dimension three. This includes the Platonic and Archimedean solids as their most prominent examples. The combinatorics of a 3-polytope P is determined by its (vertex–edge) graph Γ, which is planar. Our game `MatchTheNet` invites to play with these geometric objects. It is based on the infrastructure of the software system `polymake` [3].

We obtain a *planar net* of a 3-polytope by cutting the boundary along several edges. The resulting shape unfolds to a flat and connected figure, similar to Fig. 1. Given a planar net on a sheet of paper, one can cut out the shape, fold it along sketched edges and glue it

* Research by M. Joswig is supported by Einstein Foundation Berlin and Deutsche Forschungsgemeinschaft (Priority Program 1489: "Experimental methods in algebra, geometry, and number theory", SFB/TRR 109: "Discretization in Geometry and Dynamics" and SFB/TRR 195: "Symbolic Tools in Mathematics and their Application")

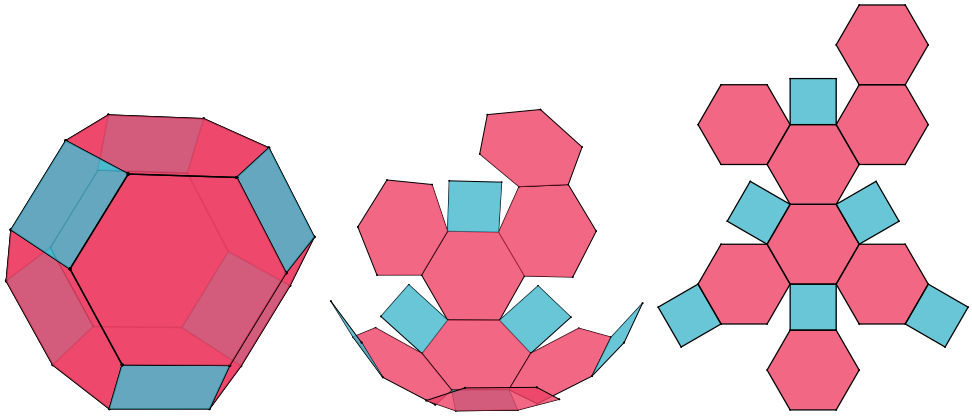


Figure 1 Unfolding the truncated octahedron, which is an Archimedean solid.

along some boundary edges to regain the original polytope. In the following, planar nets are described more formally.

The *dual graph* Δ of a 3-polytope P is the abstract graph which has the facets of P as nodes, while its edges are given by those pairs of facets which share a common edge. If we pick a spanning tree T of Δ, then, as in von Staudt's proof of Euler's formula, the edges of Γ which are not dual to any edge in T form a spanning tree T^* of Γ. In topological terms, the complementary pair (T, T^*) of spanning trees corresponds to the two critical points of an optimal Morse function of the 2-sphere. We may view the dual spanning tree T^* as a subset of the boundary ∂P. Then we obtain a map from $\pi : \partial P \setminus T^* \to \mathbb{R}^2$ as follows. We start out by picking a facet R of P and map it isometrically into the plane. Then, for each facet F adjacent to R there is a unique way of extending this map such that it is an isometry if restricted to F. Now π is defined inductively by following the unique path from any facet to the root facet R in the spanning tree T. The dual tree T^* is said to define an *edge cutting*, and the map π only depends on T^*, but not on the choice of the root facet R. If π is injective, the closure of the image $\pi(\partial P \setminus T^*)$ is called a *planar net* (or an *unfolding*) of P. It is an interesting open problem, whether or not each 3-polytope admits a planar net; see, e.g., [5] and [1] as well as the monograph [2] for an overview of topics related. Figure 2 shows that an attempt to unfold may fail.

`MatchTheNet` is a game where a single player is asked to match a set of planar nets to a set of 3-polytopes. The difficulty ranges from easy (suitable for kids in elementary school) to hard (recreational puzzle for grown-up mathematicians). The game mechanics is written in `JavaScript`, and it runs in any web browser, either locally or over the Internet. It can be played online at `www.matchthenet.de`, downloaded at `https://github.com/polymake/matchthenet`, and it is part of the `Imaginary` project.

2 Playing the Game

The front page of `MatchTheNet` explains the rules, and the player can choose the language, the level of difficulty and the number of polytopes per round. That number, which we will refer to as k here, ranges between two and five. One game lasts for five rounds. In each round the player sees k polytopes in the top row of the screen and k planar nets in the bottom row. The player swaps the planar nets with the mouse until she is confident that each polytope sits right above its planar net. Hitting the "submit" button reveals the score, which is the

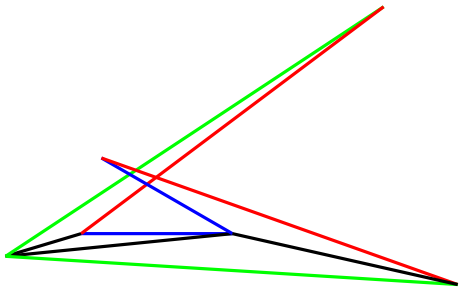


Figure 2 Tetrahedron with an attempt to unfold that fails. It arises from a spanning tree in the dual graph which is a path. Each spanning tree which has a node of degree three gives a proper planar net; this works for any tetrahedron.

total number of correct matches. Afterwards the player can either look at the solution or continue with the next round. After the fifth round the final score is displayed and compared to the current high score. During the game the polytopes can be rotated freely with mouse to look at them from all sides.

There are various ways to make the game easier or more difficult. We offer seven levels. In general, the more facets the polytope has the more difficult it is to recognize. Further, it makes a difference if the coloring of the facets gives some guidance to the combinatorics. For instance, on Level 5 there are polytopes which come from a random construction, but color helps to identify the number of vertices on each facet. On Level 6 the polytopes are the same, but all facets are green. The highest Level 7 has triplets of polytopes chosen by hand, which are very similar to one another. For this level, only $k = 3$ is available.

3 Our Collection of Polytopes

The bulk of our pre-computed 3-polytopes are regular polytopes and their generalizations. A *Platonic solid* (or *regular* 3-polytope) admits an automorphism group (of rigid motions) which acts transitively on the set of maximal flags, i.e., the triplets consisting of a vertex, an edge and a facet which are incident; there are five combinatorial types. More generally, the *Johnson solids* are the 3-polytopes whose facets are regular polygons of various gonalities. An *Archimedean solid* (or *semi-regular* 3-polytope) is a Johnson solid which admits a vertex-transitive group; there are 13 combinatorial types in addition to the regular ones. The *Catalan solids* are the duals of the Archimedean solids. There are 92 combinatorial types of *proper Johnson solids*, i.e., those which are not Archimedean [4]. Taking also the duals of the proper Johnson solids into account we arrive at five classes of 3-polytopes which are pairwise disjoint. Their numbers add up to $5 + 13 + 13 + 92 + 92 = 215$. All of them are contained in the data base of `MatchTheNet`. Figure 1 shows an unfolding of an Archimedean solid.

Additionally, we computed fifty random 3-polytopes by the following two-step procedure. In the first step we choose hyperplanes tangent to the unit sphere uniformly at random. Almost surely the resulting polytope Q is *simple*, i.e., each vertex is contained in precisely three facets. In the second step we pick a certain portion of the vertices of Q, again uniformly at random, take their convex hull, and this is our random polytope. Usually, such a polytope is neither simple nor dual to simple, i.e., simplicial.

For each level there is a subset of the entire collection from which polytopes are chosen at random during the game. The highest level is different in that certain triplets of Johnson polytopes are chosen by hand.

4 Computations in polymake

polymake is open source software for research in polyhedral geometry [3]. It deals with polytopes, polyhedra and fans as well as simplicial complexes, matroids, graphs, tropical hypersurfaces, and other objects. For MatchTheNet we use polymake as an engine to precompute all 3-polytopes and their planar nets used in our game.

To give an example we show how to produce the planar net of the truncated octahedron shown in Figure 1 to the right. This code is valid for polymake version 3.0 or higher. First we construct the polytope and its planar net. The latter employs a heuristic with backtracking.

```
polytope> $polytope = archimedean_solid('truncated_octahedron');
polytope> $net = fan::planar_net($polytope);
```

For visualization polymake offers several backends. Here, as for MatchTheNet, we use the library three.js for a direct rendering in a web browser.

```
polytope> @colors = ('green','blue','purple','red','grey');
polytope> threejs( $net->VISUAL( VertexLabels => "hidden",
                VertexColor => "black",
                FacetTransparency => 0.8,
                FacetColor => sub {
                   $colors[ min($net->MAXIMAL_POLYTOPES->[shift]->
                       size-3, @colors-1) ]
                } ));
```

In this example the color of each facet is determined by its number of vertices. So triangles become green, quadrangles blue, pentagons purple and hexagons red; all others will be shown in gray.

Acknowledgements. We are indebted to the Imaginary team for a lot of inspiration and fruitful discussions during the design of MatchTheNet.

References

1 Marshall Bern, Erik D. Demaine, David Eppstein, Eric Kuo, Andrea Mantler, and Jack Snoeyink. Ununfoldable polyhedra with convex faces. *Comput. Geom.*, 24(2):51–62, 2003. Special issue on the Fourth CGC Workshop on Computational Geometry (Baltimore, MD, 1999). doi:10.1016/S0925-7721(02)00091-3.

2 Erik D. Demaine and Joseph O'Rourke. *Geometric folding algorithms.* Cambridge University Press, Cambridge, 2007. Linkages, origami, polyhedra. doi:10.1017/CBO9780511735172.

3 Ewgenij Gawrilow and Michael Joswig. polymake: a framework for analyzing convex polytopes. In *Polytopes—combinatorics and computation (Oberwolfach, 1997)*, volume 29 of *DMV Sem.*, pages 43–73. Birkhäuser, Basel, 2000.

4 Norman W. Johnson. Convex polyhedra with regular faces. *Canad. J. Math.*, 18:169–200, 1966. doi:10.4153/CJM-1966-021-8.

5 Geoffrey C. Shephard. Convex polytopes with convex nets. *Math. Proc. Cambridge Philos. Soc.*, 78(3):389–403, 1975. doi:10.1017/S0305004100051860.

6 Günter M. Ziegler. *Lectures on polytopes*, volume 152 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 1995. doi:10.1007/978-1-4613-8431-1.

On Balls in a Hilbert Polygonal Geometry

Frank Nielsen[1] and Laëtitia Shao[2]

1 **École Polytechnique, LIX, Palaiseau, France**
`Frank.Nielsen@acm.org`
2 **École Polytechnique, Palaiseau, France**
`Laetitia.Shao@polytechnique.edu`

Abstract

Hilbert geometry is a metric geometry that extends the hyperbolic Cayley-Klein geometry. In this video, we explain the shape of balls and their properties in a convex polygonal Hilbert geometry. First, we study the combinatorial properties of Hilbert balls, showing that the shapes of Hilbert polygonal balls depend both on the center location and on the complexity of the Hilbert domain but not on their radii. We give an explicit description of the Hilbert ball for any given center and radius. We then study the intersection of two Hilbert balls. In particular, we consider the cases of empty intersection and internal/external tangencies.

1998 ACM Subject Classification I.3.5 Computational Geometry and Object Modeling

Keywords and phrases Projective geometry, Hilbert geometry, balls

Digital Object Identifier 10.4230/LIPIcs.SoCG.2017.67

Category Multimedia Contribution

1 Introduction: Hilbert geometry

Hilbert geometry is a projective geometry relying on the properties of the cross-ratio:

▶ **Definition 1** (Cross-ratio). For four collinear points a, b, c, d the cross ratio is defined as follows:

$$(a, b; c, d) = \frac{\|ac\|\|bd\|}{\|ad\|\|bc\|} \tag{1}$$

The cross-ratio is an invariant measure under perspective transformation:

▶ **Property 2** (Projective invariance of the cross-ratio). *Given four points a, b, c, d and A, B, C, D their images through a projective transformation, $(a, b; c, d) = (A, B; C, D)$.* [5]

In a Hilbert geometry, the distance between two points is defined using the cross-ratio as follows:

▶ **Definition 3** (Hilbert distance). A Hilbert distance is defined in the interior of a convex bounded domain $\mathcal{C}$. Given two distinct points, a and b of the domain, the distance is defined as follows:

$$d_{HG}(a, b) = \log((a, b; A, B)) \tag{2}$$

where $(a, b; A, B)$ is the cross-ratio where A and B denote the intersection points of line (a, b) with the domain. By definition, $d_{HG}(x, x) = 0$ for all $x \in \mathcal{C}$.

Figure 1 Left: In blue, two Hilbert balls in a circular domain. Right: In blue, three Hilbert balls in a polygonal convex domain.

▶ **Property 4** (Properties of the Hilbert distance). *Given two points a and b.*

- *The Hilbert distance is a signed distance: $d_{HG}(a,b) = -d_{HG}(b,a)$.*
- *$d_{HG}(a,a) = 0$ (law of the indiscernibles).*
- *When a is on the boundary of the convex, $\forall b \in \mathcal{C}, d_{HG}(a,b) = \infty$.*
- *$|d_{HG}|$ respects the triangular inequality and therefore $|d_{HG}|$ is a metric distance [1].*

A key property in Hilbert geometry is that shortest-path geodesics are straight lines. The Klein disk representation of hyperbolic geometry is an example of Hilbert geometry for the unit disk (convex and smooth) domain.

In this work, we consider convex polygonal Hilbert geometries, that is, Hilbert geometries defined on a convex polygonal domain. $\mathcal{C}$ now refers to a convex polygon with s vertices: $e_1, ..e_s$. The distance between two points p and q in this domain is noted $d_{\mathcal{C}}(p,q)$. The ball of radius r and center c is denoted by $\mathcal{B}(c,r)$. The sphere is denoted by $\mathcal{S}(c,r)$. See [4] for an application of Hilbert geometry to clustering in the open probability simplex.

2 Combinatorial properties of Hilbert balls

In Klein ball hyperbolic geometry or Cayley-Klein hyperbolic geometry, the balls have the shape of (Euclidean) Mahalanobis balls with displaced centers, see [2, 3]. To contrast with this smooth shape representation of balls, let us observe that when the domain is a convex polygon, the shapes of Hilbert balls are (Euclidean) polygons.

▶ **Definition 5** (Rays). Given a center point c in the domain, line $(c, e_i), i \in [s] = \{1, \ldots, s\}$ is a ray.

▶ **Lemma 6** (Description of a Hilbert ball). *$\mathcal{B}(c,r)$ is a Euclidean polygon with at most $2s$ edges and at least s edges. Each vertex of $\mathcal{B}(c,r)$ belongs to a ray.*

Proof. We first partition the polygonal domain with s to $2s$ triangles, by tracing rays $(c, e_i), i \in [s]$. We will show that each triangle induces a linear edge of the Hilbert ball.
We consider a pair of triangles (A, B, c) and (c, C, D) such that A, c, D and B, c, C are respectively collinear. Let $P \in [A, c] \cap \mathcal{B}(c,r)$ and $O = (A, B) \cap (C, D)$, we will show that line (O, P) clipped to the triangle (A, B, c) is an edge of $\mathcal{B}(c,r)$. Let U be a point on the clipped line, and M, N the intersections points of line (Uc) with the domain such that $M \in [A, B]$ and $N \in [C, D]$. Then M, U, c, N and A, P, c, D are related by the same projective transformation. Using the invariance property of the cross-ratio, we conclude that $d_{\mathcal{C}}(c, P) = d_{\mathcal{C}}(c, U) = r$. Thus, we proved Lemma 6. It is remarkable that depending on the position of the center, the number of triangles (and hence the complexity of the ball) varies. ◀

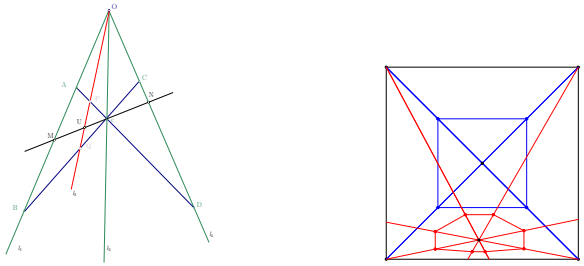


Figure 2 Left: Configuration for proof 2 (see text). Right: Varying number of rays in a square domain depending on the position of the center of the ball.

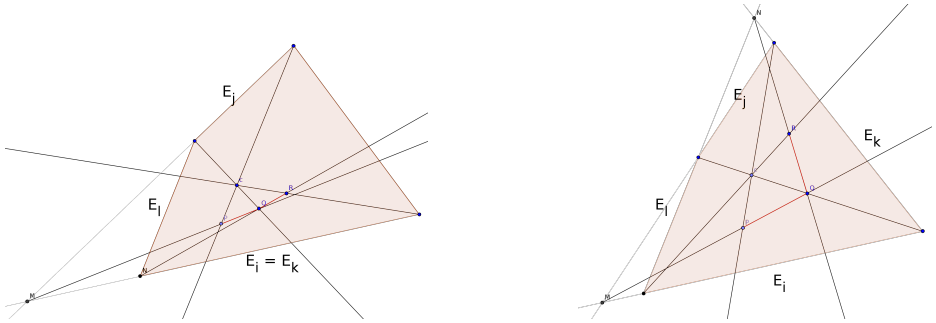


Figure 3 Left: Configuration for proof of Lemma 8 when $E_i = E_k$. Right: Configuration for proof of Lemma 8 when all edges are distinct.

▶ **Definition 7.** Given an edge $[P, Q]$ of a Hilbert ball that belongs to a pair of triangles (A, B, c) and (c, D, E), we say that $[P, Q]$ is induced by edges E_i and E_j of the domain, if $[A, B] \subset E_i$ and $[D, E] \subset E_j$.

▶ **Lemma 8** (Shape invariance with varying radius)**.** *For c a fixed center point, and r a varying radius, $\mathcal{B}(c, r)$ has the same number of edges.*

Proof. Let $[P, Q]$ and $[Q, R]$ be two adjacent edges of a Hilbert ball such that E_i, E_j induces $[P, Q]$ and E_k, E_l induces $[Q, R]$. We show that P, Q, R cannot be collinear. We note M the intersection points of the lines supported by E_i and E_j and N the intersection points of the lines supported by E_k and E_l. According to the previous proof, P, Q, M and Q, R, N are respectively collinear.

- If E_i, E_j, E_k, E_l are distinct edges, because $[P, Q]$ and $[Q, R]$ are adjacent, we can assume without loss of generality that E_i is adjacent to E_k and E_j is adjacent to E_l. If P, Q, R are collinear, then $E_i = E_k$ or $E_j = E_l$, which contradicts the previous assumption.
- Otherwise, we can assume that $E_i = E_k$. In this case, if P, Q, R are collinear, then they belong to line $(M, N) \subset E_i$. Which is impossible unless $r = \infty$.

Therefore, as the radius varies but stay finite, the number of edges remains constant. See Figure 3 for a visualization of the proof. For infinite radius, all balls fully cover the polygonal domain. ◀

▶ **Lemma 9** (Shape invariance in a simplex domain)**.** *In a simplex domain Δ, Hilbert polygonal balls do not change shape, and have a fixed complexity of $2s$ edges.*

Proof. It is a direct consequence of the two previous lemmas. ◀

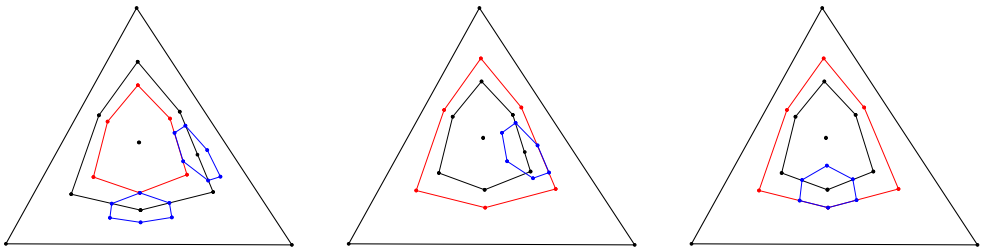


Figure 4 In a triangular domain. Left: Two cases of outer tangency. The red sphere is externally tangent to the blue spheres and share one edge with one sphere and one vertex with the other. Middle and Right: Two cases of inner tangency between the red sphere and the blue sphere. Middle: The two spheres share one edge. Right: the two spheres share two edges.

3 Intersection of Hilbert spheres

We now consider the interaction scenario of two spheres. First, let us mention a simple condition to check whether two spheres intersect or not:

▶ **Lemma 10** (Condition for empty intersection). *Given two points $c_1, c_2 \in \mathcal{C}$ and two reals $r_1, r_2 > 0$, with $r_2 >= r_1$:*

$$\mathcal{S}(c_1, r_1) \cap \mathcal{S}(c_2, r_2) \neq \emptyset \Rightarrow r_2 - r_1 \leq d_{\mathcal{C}}(c_1, c_2) \leq r_1 + r_2 \tag{3}$$

Proof. This follows from the fact that $d_{\mathcal{C}}$ respects the triangular inequality [1]. ◀

In the case of *external tangency*, *i. e.*, $d_{\mathcal{C}}(c_1, c_2) = r_1 + r_2$, if c_2 is a vertex of $\mathcal{B}(c_1, r_1 + r_2)$, the intersection of the two Hilbert spheres is reduced to a vertex. Otherwise, the two Hilbert spheres share part of an edge. In the case of *internal tangency*, *i. e.*, $d_{\mathcal{C}}(c_1, c_2) = r_2 - r_1$, if c_1 is a vertex of $\mathcal{B}(c_2, r_2 - r_1)$, the two spheres share part of two edges. Otherwise the shared part is one edge. See Figure 4 for some illustrating examples. The Java™ applet is available from `https://www.lix.polytechnique.fr/~nielsen/software.html`: The pop menu let one choose the demo to play. The online explanatory video is available at `https://www.youtube.com/watch?v=XE5x5rAK8Hk`

References

1 Curtis T. McMullen. Coxeter groups, Salem numbers and the Hilbert metric. *Publications mathématiques de l'IHÉS*, 95:151–183, 2002.

2 Frank Nielsen, Boris Muzellec, and Richard Nock. Classification with mixtures of curved Mahalanobis metrics. In *IEEE International Conference on Image Processing (ICIP)*, pages 241–245. IEEE, 2016.

3 Frank Nielsen, Boris Muzellec, and Richard Nock. Large margin nearest neighbor classification using curved Mahalanobis distances. *CoRR*, abs/1609.07082, 2016. URL: `http://arxiv.org/abs/1609.07082`.

4 Frank Nielsen and Ke Sun. Clustering in Hilbert simplex geometry. *ArXiv 1704.00454*, April 2017.

5 Jürgen Richter-Gebert. *Perspectives on Projective Geometry: A Guided Tour Through Real and Complex Geometry*. Springer Publishing Company, Incorporated, 1st edition, 2011.