Proceedings of the 31st Canadian Conference on Computational Geometry (CCCG 2019)

August 8-10, 2019
University of Alberta
Edmonton, Alberta
Canada

Logo designed by Jenne Friggstad

Preface

This volume contains the proceedings of the 31st Canadian Conference on Computational Geometry (CCCG 2019), which took place on August 8-10, 2019 at University of Alberta, Edmonton, Alberta. These proceedings are also available electronically at CCCG 2019 website: `https://sites.ualberta.ca/~cccg2019/`

We are grateful to the Program Committee, and external reviewers, for the hard work they did. They thoroughly examined all submissions and provided excellent feedback. Each submission was reviewed by at least three program committee members. Out of 51 papers submitted, the committee decided to accept 40 papers. We thank the authors of all submitted papers, all those who have registered, and in particular the invited speakers: Vida Dujmovic (Paul Erdős Memorial Lecture), Mark de Berg (Ferran Hurtado Memorial Lecture), and Joseph O'Rourke (Godfried Toussaint Memorial Lecture).

We gratefully acknowledge financial support from the Pacific Institute for the Mathematical Sciences (PIMS), Elsevier, and the University of Alberta.

August, 2019

Zachary Friggstad
Jean-Lou De Carufel

Sponsored by

Invited Speakers

Vida Dujmovic	University of Ottawa
Joseph O'Rourke	Smith College
Mark de Berg	Eindhoven University of Technology

Program Committee

Elena Arseneva	St. Petersburg University
Ahmad Biniaz	University of Windsor
Prosenjit Bose	Carleton University
Paz Carmi	Ben-Gurion University of the Negev
Timothy M. Chan	University of Illinois at Urbana-Champaign
Jean-Lou De Carufel	University of Ottawa
Zachary Friggstad	University of Alberta
Konstantinos Georgiou	Ryerson University
Joachim Gudmundsson	The University of Sydney
Michael Hoffmann	ETH Zurich
Matias Korman	Tufts University
Irina Kostitsyna	Eindhoven University of Technology
Nirman Kumar	University of Memphis
Anil Maheshwari	Carleton University
Saeed Mehrabi	Carleton University
Wolfgang Mulzer	Freie Universität Berlin
David Rappaport	Queen's University
Thomas Shermer	Simon Fraser University
Shakhar Smorodinsky	Ben-Gurion University of the Negev
André van Renssen	The University of Sydney
Carola Wenk	Tulane University

Additional Reviewers

A. Karim Abu-Affash, Hugo Akitaya, Helmut Alt, Sergey Bereg, Erin Chambers, Steven Chaplick, Anthony D'Angelo, Patrick Eades, Nicolas Grelier, Qizheng He, Darryl Hill, Kristian Hinnenthal, Hung Hoang, Bruno Jartoux, Wouter Meulemans, Majid Mirzanezhad, Ramin Mousavi, Saladi Rahul, Benjamin Raichel, Saurabh Ray, Alexandre Rok, Leonie Ryvkin, Anna Schenfisch, Patrick Schnider, Martin Seybold, Stavros Sintos, Manuel Wettstein, Charles Wolf, Sampson Wong, Yelena Yuditsky, Alena Zhukova

Local Organization

Zachary Friggstad (Co-chair)

With many thanks to Deborah Choi, Cathy Hurst, Sunrose Ko and Barb Robinson at the University of Alberta, and also Ruth Situma from PIMS.

Table of Contents

Thursday, August 8

Paul Erdős Memorial Lecture

Session 1A

Session 1B

Session 2A

Session 2B

Friday, August 9

Distinguished Lecture

Session 3A

Session 3B

Session 4A

Session 4B

Saturday, August 10

Ferran Hurtado Memorial Lecture

Session 6A

Session 6B

Graph Drawing via Layered Partitions

Vida Dujmovic*

Whether planar graphs have 3-dimensional grid drawings in linear volume was asked in 2001 by Felsner, Liotta, and Wismath. Subsequent research related this question to an older conjecture from 1992 by Heath, Leighton and Rosenberg on whether planar graphs have bounded queue-number.

I will give some history of these problems and show how they were finally resolved this year with the help of a new tool called layered partitions.

*School of Computer Science and Electrical Engineering , University of Ottawa

Maintaining a Centerpoint in the Plane with Amortized Optimal Logarithmic Insertions

Morten Eskildsen* † Matias Frank Jensen* ‡ Sebastian Kolby* § Jesper Steensgaard* ¶

Abstract

A centerpoint of a set of n points P in $\mathbb{R}^2$ is defined as a point p, not necessarily in P, such that any half-plane containing p contains at least $\lceil n/3 \rceil$ points of P. We show how to dynamically maintain a centerpoint of a planar point set, allowing insertions in amortized $O(\log n)$ time. This builds upon the work of Jadhav and Mukhopadhyay [5] who show how to find a centerpoint in linear time. Our method also suggests a way of supporting deletions in amortized $O(\log^2 n)$ time. We provide a proof of a matching lower bound for the amortized $O(\log n)$ insertion time as well as a corresponding $\Omega(\log n)$ lower bound for deletions.

1 Introduction

A centerpoint is a generalization of the median in higher dimensions and is an interesting measure due to its robustness. The notion of robustness is relevant when estimating a set of points with a single point. As an example, for a set of points in $\mathbb{R}^1$ the mean is not a robust measure: Corrupt just a single point by moving it towards infinity and the mean will do the same. However, doing the same to the median requires corrupting half of the points. This makes the median a more robust measure, and can be more interesting for data containing noisy points or outliers.

Centerpoints are defined in terms of half-space depth. Let P be a set of n points in $\mathbb{R}^d$. The half-space depth of a point $p \in \mathbb{R}^d$ with respect to P is defined to be the smallest number, t, such that there exists a closed half-space containing p that contains exactly t points of P. A centerpoint of P is defined to be any point whose depth is at least $\lceil \frac{n}{d+1} \rceil$ [6]. By Helly's theorem [4], such a point always exists. In particular, in $\mathbb{R}^1$ the median is a centerpoint. For point set P we denote the set of centerpoints as the *center* of P.

Jadhav and Mukhopadhyay [5] showed how to compute a centerpoint in $\mathbb{R}^2$ in linear time. Their method uses the ham sandwich theorem [6] and Megiddo's [7] method for finding such a partition in linear time in $\mathbb{R}^2$. We briefly explain their method in section 1.2.

We focus on the problem of dynamically maintaining a centerpoint of a planar set under insertions and deletions of points.

Centerpoints are particularly interesting due to their robustness and maintaining them under updates could be interesting for applications in which points may be added or deleted to quickly compute a centerpoint without having to do a linear time recomputation.

1.1 Related work

The problem of computing centerpoints has been extensively studied. Jadhav and Mukhopadhyay discovered an optimal linear time algorithm for the planar case in [5]. In three dimensions, Naor and Sharir introduced an efficient algorithm in [9]. In [2] Chan and Timothy M. found an $O(n^{d-1} + n \log n)$ algorithm for computing the Tukey median in the general case (and any Tukey median is a centerpoint). An efficient, randomized algorithm for computing an approximate centerpoint in $\mathbb{R}^d$ was found in [3], and a deterministic version of this algorithm developed in [8].

1.2 The Centerpoint Algorithm

The original algorithm for finding a centerpoint of a set of planar points P by Jadhav and Mukhopadhyay [5] works as follows:

1. Construct four open half-planes that each contain less than $\lceil n/3 \rceil - 1$ points, ensuring that their respective intersections are at least of size $\lceil n/3 \rceil - \lceil n/4 \rceil$. The half-planes can be seen in Figure 1 denoted by U, D, L and R. The half-planes are constructed using a generalized version of the Ham-Sandwich Cut, see section 2.2.
2. Prune points from the pairwise intersections of the U, D, L and R half-planes while maintaining that the new center is a subset of the old center.
3. Once one of the intersections is empty steps 1-2 are repeated. Repeat until the point set has a constant size.
4. Finally, a brute-force algorithm is applied to the remaining points.

*Department of Computer Science, Aarhus University, Aarhus, Denmark
†m.eskildsen@post.au.dk
‡matias.frank.jensen@post.au.dk
§sebastian.kolby@post.au.dk
¶jesper.steensgaard@post.au.dk

To prune points the algorithm chooses four points, one from each half-plane intersection. These points are then removed and a new point, called a Radon point, is inserted into the complements of the half-spaces. This ensures the center after pruning is a subset of the center before pruning.
The half-spaces are chosen such that at least a constant fraction of the points are pruned in each iteration. This results in $O(\log n)$ iterations before a constant size set of points remains.

1.3 Our Results

We were able to buffer the half-planes used in the linear time algorithm [5] for computing a centerpoint to achieve amortized $O(\log n)$ insertions and $O(\log^2 n)$ deletions while maintaining a centerpoint. Additionally, we demonstrate that $O(n)$ insertions or deletions take $\Omega(n \log n)$ time, thus amortized $\Omega(\log n)$ time per insertion or deletion, showing our $O(\log n)$ bound is tight for insertions.

2 Dynamic result

We adapt the original linear time algorithm by Jadhav and Mukhopadhyay to support updates as follows. At any time in the execution of the algorithm, we have $O(\log n)$ iterations, with the center of each iteration being a subset of the center of the previous iteration. Denote these iterations and their respective point sets and half-planes as $I_1, ..., I_m$. We let $|I_k|$ denote the number of points in the kth iteration. The *middle* of I_k is the intersection of the complements of I_k's four half-planes. We also let Middle_k and Center_k be the middle and center of I_k respectively.

When a point is inserted, it is inserted into every iteration. We maintain the invariant that for every iteration I_k, $\mathsf{Center}_k \subseteq \mathsf{Middle}_k$. When too many points have been inserted into the half-planes of I_k, this invariant can potentially be broken. At that point we throw away the iterations $I_k, ..., I_m$ and run Jadhav and Mukhopadhyay's linear time algorithm on the points in I_k generating a new set of iterations from I_k and downwards. We call this process *recomputing* I_k.

We also maintain the invariant that the number of points in the last iteration I_m is bounded by some fixed constant. Whenever a point is inserted into I_m, we recompute a centerpoint of I_m using any brute force method which is why we need $|I_m|$ to be bounded by a constant.

If too many points are inserted into a given iteration I this invariant can be broken, requiring the recomputing of I. Recomputing an iteration I takes $O(|I|)$ time, so to achieve $O(1)$ amortized cost per insertion per iteration, we construct the half-planes such that a constant fraction of $|I|$ points can be inserted before a recomputation is necessary. We call this *buffering the half-planes*.

2.1 Constructing the buffered half-planes

To buffer the half-planes, it is necessary to modify the method used in [5] with which half-planes are constructed.

We need to construct the half-planes such that they all contain a certain fraction of the points which give the half-planes the desired properties described above. The fraction must be larger than $\lceil n/4 \rceil - 1$ because $4(\lceil n/4 \rceil - 1)$ may be smaller than n, making it possible for the intersections of the half-planes to be empty.

The other bound of the the half-planes is borrowed from the static algorithm. If every half-plane contains less than $\lceil n/3 \rceil$ points, then $\mathsf{Center} \subseteq \mathsf{Middle}$ [5].

We select an x such that $3 < x < 4$. We define $V_P = \lceil |P|/x \rceil$ to be the number of points contained in the buffered half-planes. With V_P points in each half plane, their pairwise intersections must each contain the following number of points:

$$\left\lceil \frac{4V_P - |P|}{4} \right\rceil \geq \left\lceil \frac{|P|(4-x)}{4x} \right\rceil.$$

We define W_P to be the number of points in every intersection as

$$W_P = \left\lceil \frac{|P|(4-x)}{4x} \right\rceil : 3 < x < 4.$$

2.2 Adapting the algorithm

The static algorithm can be adapted such that every half-plane contains V_P points and every intersection contains W_P points. This can be done using the generalized Ham-Sandwich Cut algorithm as described in the original algorithm [5], deriving from the algorithm by Megiddo [7]. We specify that no more than $4W_P$ points may be pruned from the intersections in a given iteration so that the constant size iteration may be bounded.

For the pruning used in [5] to work, it is only required that $\mathsf{Center}_k \subseteq \mathsf{Middle}_k$ for every iteration I_k. If this is the case, then $\mathsf{Center}_{k+1} \subseteq \mathsf{Center}_k$. Remember that if all half-planes of I_k has less than $\lceil |I_k|/3 \rceil$ points, then $\mathsf{Center}_k \subseteq \mathsf{Middle}_k$.

We also maintain the invariant that a constant fraction is pruned away in each iteration which means $|I_{k+1}| \leq |I_k|/y$ for some $y > 1$. This ensures a logarithmic number of iterations.

These invariants are maintained after every insertion.

2.2.1 Notation

We denote by I'_i the ith iteration and the points that have been added to it since the last recomputation of

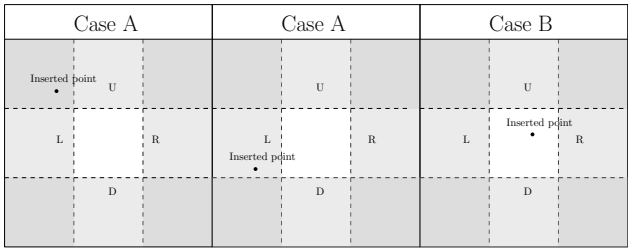


Figure 1: Examples of cases A and B. The dot represents the inserted point.

I_i. The sets α_i and β_i denote the points added to I_i since the last recomputation that fall into case A and B respectively, as described below.

The set $\mathsf{Pruned}(I_i)$ refers to the set of points that were pruned in the iteration I_i,this can range between 3 and $4W_{I_i}$

2.2.2 Inserting into the iterations

Each inserted point, q, is added to all of the iterations starting from I_1. For each iteration the point can enter one of three cases.

Case A where q is contained within at least one of the half-planes. Case B where the point is contained in the middle. Case C where the point is being added to the constant size problem of the final iteration.

2.2.3 Case A

In Case A we add q to the half-plane(s) that contain it in I_i. If this results in any of the planes having more than or equal to N_{I_i} points, the problem is recomputed from the current iteration. The subsequent iterations are discarded, and the new iterations, used to solve the problem, are saved as I_i to I_{i+j-1} where j is the number of iterations required to solve the new sub-problem.

If adding the point to the half-planes that contain it does not cause any of them to have at least N_{I_i} points, the point can be added to the point set of the iteration and then iteratively added to the subsequent iteration I_{i+1}. This maintains the $\mathsf{Center}_{i+1} \subseteq \mathsf{Center}_i$ invariant.

In this case it is also necessary to consider whether the fraction of points pruned remains greater than $1/y$, where $1/y$ is as defined in section 2.2.4. If this is not the case recomputation of the iteration is done as mentioned above.

2.2.4 Case B

In the case where the inserted point, q, is in the middle of an iteration, q may be added to the point set of the iteration. If the pruned fraction becomes smaller than $1/y$ recompute the iteration, as mentioned in case A. Otherwise the point can be added iteratively to the following iteration.

We choose the constant y so it satisfies the following conditions:

$$0 < \frac{1}{y} < \frac{3(4-x)}{4x} < \frac{3}{12} \quad \text{for } 3 < x < 4.$$

The lower bound is needed to ensure a positive number of tokens in the amortized analysis. The upper bound is necessary due to the minimum fraction of pruned points, since at least $3W_p$ points are pruned when each iteration is calculated. These $3W_p$ points are guaranteed to be pruned as the four intersections will have at least W_P points each, which can result in the inclusion of W_P Radon points in the next iteration. Therefore the minimum fraction of pruned points after insertions should be smaller than this to allow for adding a fraction of $|I|$ before recomputation is required.

If the fraction of points that have been pruned $\frac{|\mathsf{Pruned}(I_i)|}{|I_i'|}$ becomes smaller than or equal to $1/y$ the sub-problem from I_i should be recomputed and the new iterations should be saved replacing the old iterations from I_i onwards.

2.2.5 Case C

The constant size problem, in the last iteration, must be recomputed every time a point is inserted in said iteration. As a result, it must be ensured that this final iteration remains within constant bounds, so that the bruteforce calculation does not affect the running time of the algorithm. We demonstrate that the final iteration remains within constant bounds in section 2.5.4. This is guaranteed as more iterations are added if the constant iteration becomes too large.

2.3 Correctness

When an iteration is recomputed the linear time algorithm ensures that $\mathsf{Center}_i \subseteq \mathsf{Middle}_i$. Because of this, $\mathsf{Center}_{i+1} \subseteq \mathsf{Center}_i$ for all iterations $i = 1, ..., m-1$. Thus a centerpoint of I_m, calculated by any brute force method, is also a centerpoint of P.

After every insertion we have argued in case A that either the $\mathsf{Center}_i \subseteq \mathsf{Middle}_i$ invariant is maintained or otherwise the iteration I_i is recomputed. Case B and C cannot break this invariant.

Letting Center denote the center of P, we therefore conclude that after every insertion

$$\mathsf{Center}_m \subseteq \cdots \subseteq \mathsf{Center}_1 = \mathsf{Center}$$

and thus, any centerpoint of the last iteration I_m is a centerpoint of P.

2.4 Maintaining logarithmic iterations

At least a fraction $1/y$ of the points is pruned in every iteration. Hence, for every iteration the problem size

is scaled down by at least $\frac{y-1}{y}$, giving a logarithmic number of iterations of size $O(\log_{\frac{y}{y-1}}(|P|))$ where $\frac{y}{y-1}$ is a constant.

2.5 Amortized analysis

There are two cases where an insertion would require the recomputation of the current and subsequent iterations. The first case is where some half-plane would contain too many points. In this case the recomputation is payed for by the ♠ tokens. The second case is where too small a fraction of the points are pruned in the iteration. This is payed by ♡ tokens. Depending on the two cases points contribute differently to the recomputation, which is why we have chosen to distinguish these tokens from one another. It takes $O(|I_i|)$ to recompute an iteration and all its subsequent iterations. Thus we can show the amortized constant bound, if it can be demonstrated that adding a constant number of tokens for each insertion provides sufficient tokens, $|I_i|$, for this recomputation.

2.5.1 Insertions

A point is given a number of tokens when it is added to the point set of an iteration. If the point is contained in any of the half-planes, Case A, the point will be given $\frac{3x}{x-3}+1$ ♠ tokens and $\frac{4x}{12y-3xy-4x}+1$ ♡ tokens. Additionally, if the point's insertion necessitates recomputation further $2(\frac{3x}{x-3})$ ♠ tokens will be added.

If a point is added in the middle, Case B, then the point will be given one ♠ token and $\frac{4x}{12y-3xy-4x}+1$ ♡ tokens.

2.5.2 Recomputation due to overfull half-planes

First we will consider the tokens for when our recomputation is as a result of a Case A insertion where one of our half-planes contains too many points.

Let I be an iteration which is recomputed and α be the number of Case A insertions into I. In this case α has become too large and can be bounded as follows:

$$|\alpha| \geq N_I - 1 - \left\lceil \frac{|I|}{x} \right\rceil \geq \frac{|I|}{3} - \frac{|I|}{x} - 2 = \frac{|I|(x-3)}{3x} - 2$$

Each point of β has given one ♠ to account for its own recomputation. Similarly, every point in α will have one ♠ token for its own recomputation, this leaves $\frac{3x}{x-3}$♠ tokens for each point of α to contribute toward the recomputation of the original point set I. The final point that is inserted has $2(\frac{3x}{x-3})$ additional ♠ tokens to contribute toward the recomputation of I. This gives a total number of ♠ tokens on at least:

$$\frac{3x}{x-3}\left(\frac{|I|(x-3)}{3x} - 2\right) + 2\frac{3x}{x-3} = |I|$$

With these $|I|$ ♠ tokens, we have enough tokens to recompute the iteration with the inserted points as there are $|I'| = |I| + |\alpha| + |\beta|$ which are enough to recompute the current and subsequent iterations linearly.

2.5.3 Recomputation for too few pruned points

For this case, where the fraction of points pruned grows too small, it can be similarly demonstrated that a linear number of tokens will have been accumulated before recomputation. We consider the case where the fraction pruned is the smallest possible, $3W_I$. For all the points in $\alpha \cup \beta$ one of each points ♡ tokens can be used to account for their recomputation which provides $|\alpha|+|\beta|$ tokens. After this each point in $\alpha \cup \beta$ has $\frac{4x}{12y-3xy-4x}$ remaining ♡ tokens. Since

$$\frac{1}{y} = \frac{|\mathsf{Pruned}(I)|}{|I'|} = 3\frac{\lceil |I|\frac{4-x}{4x} \rceil}{|I| + |\beta| + |\alpha|}$$

we have

$$\begin{aligned} |\beta| + |\alpha| &= 3y\left\lceil \frac{|I|(4-x)}{4x} \right\rceil - |I| \\ &\geq 3y\frac{|I|(4-x)}{4x} - |I| = \frac{|I|(12y-3xy-4x)}{4x}. \end{aligned}$$

This means $\alpha \cup \beta$ will have at least $|I|$ ♡ tokens remaining. Thus, there will be at least $|I'| = |I| + |\alpha| + |\beta|$ ♡ tokens allowing for the linear recomputation of the iteration and subsequent iterations.

2.5.4 The constant size iteration

We show that the size of the final iteration I_{z+1} only increases by a constant bound between the recomputations of I_z. This means that we can recompute the final iteration in constant time.

Let the constant c denote the initial size of the constant iteration I_{z+1} and c' denote the updated value of c after one or more insertions. An upper bound can be defined as follows:

$$\begin{aligned} c' &= |I_z| - |\mathsf{Pruned}(I_z)| + |\alpha_z| + |\beta_z| \\ &\leq |I_z| - |\mathsf{Pruned}(I_z)| + y|\mathsf{Pruned}(I_z)| - |I_z| \\ &= (y-1)|\mathsf{Pruned}(I_z)| \\ &\leq (y-1)4W_{I_z} = 4(y-1)\left\lceil \frac{|I_z|(4-x)}{4x} \right\rceil \end{aligned}$$

The lower bound can be demonstrated as shown below:

$$c' > c \geq |I_z| - |\mathsf{Pruned}(I_z)| \geq |I_z| - 4W_{I_z} \geq \frac{|I_z|(2x-4)}{x} - 4$$

With these upper and lower bounds it follows that the value of c' remain within a constant factor of the penultimate iteration. The point set of the final iteration, initially bounded by some constant size would therefore remain within some constant bound.

2.5.5 Result

The number of iterations is limited by $\log_{\frac{y}{y-1}}(|P|)$ as Case B necessitates pruning at least $1/y$. Given that each insertion only requires a constant number of tokens we can conclude that the amortized running time of the algorithm is $O(\log_{\frac{y}{y-1}}|P|)$.

2.5.6 Deletions

Using our insertion algorithm it follows that it is possible to support deletions in amortized $O(\log^2 n)$ time. These deletions can be performed by considering how a point, q, was originally pruned by the centerpoint algorithm. If a new Radon point was added to the point set in the subsequent iteration, it must be deleted recursively from the following iteration. This occurs when q was pruned with three other points forming a convex quadrilateral, as described in [5]. Following this the points that were pruned with q must be inserted again, to the subsequent iteration, as if they were not pruned. The need for recursive deletions gives rise to the $O(\log^2 n)$ bound.

3 Lower bounds for centerpoint updates

In the following we will show amortized asymptotic lower bounds for dynamic insertions and deletions of points in the plane while maintaining a centerpoint. We will show that doing $O(n)$ insertions or deletions both take worst case $\Omega(n \log n)$ time, thus amortized $\Omega(\log n)$ time per insertion or deletion.

To show that $O(n)$ deletions/insertions take $\Omega(n \log(n))$ time, we construct two algorithms $\mathsf{Sort}_{\mathsf{delete}}$ and $\mathsf{Sort}_{\mathsf{insert}}$ which sort n numbers. The algorithms first transform the input numbers into a set of points in the plane, and then while maintaining a centerpoint under $O(n)$ deletions or insertions respectively, sort the original numbers.

3.1 Main planar point construction

Both $\mathsf{Sort}_{\mathsf{insert}}$ and $\mathsf{Sort}_{\mathsf{delete}}$ use the following construction. Consider a set of planar points $P = L_1 \cup L_2 \cup L_3$ where L_1, L_2, L_3 are disjoint. We refer to these subsets as *legs* and position the points in P such that all the points in one of the legs lie on the same ray starting in $(0, 0)$. More precisely, for each L_i, $i = 1, 2, 3$, L_i is a finite subset of the ray

$$R_i = \{(\cos(2i\pi/3) \cdot x,\ \sin(2i\pi/3) \cdot x) \mid x \in \mathbb{R}_+\}$$

Before we get into further details about the construction we need a bit more notation: For $i = 1, 2, 3$ and $j = 1, ..., |L_i|$ let l_{ij} be the jth closest point to the origin of the ith leg. Figure 2 illustrates this construction.

By carefully setting the number of points in the three legs, we can force a single specific point to be the only centerpoint of P, which the sorting algorithms can then utilize.

Lemma 1 *Let $P = L_1 \cup L_2 \cup L_3$ be defined as above. If $|L_1| = m + 2k + 1$, $|L_2| = m + k$ and $|L_3| = m$ for some $m, k \in \mathbb{N}_0$, then the only centerpoint of P is $l_{1,k+1}$*

Proof: First notice that $|P| = 3m + 3k + 1$ and $\lceil n/3 \rceil = m + k + 1$. We first show that $l_{1,k+1}$ is a centerpoint of P. Consider any hyperplane H containing $l_{1,k+1}$. Let H_P be $H \cap P$. Assume H contains any of the points $l_{1,1}, ..., l_{1,k}$. Then H contains all of the points $l_{1,1}, ..., l_{1,k}$ since they all lie on the same line on the same side of $l_{1,k+1}$. But then H must also contain either all of L_2 or all of L_3. From the sizes of L_2 and L_3 we get that

$$|H_P| \geq k + 1 + \min(|L_2|, |L_3|) = m + 1 + k = \lceil n/3 \rceil$$

If H does not contain any of the points $l_{1,1}, ..., l_{1,k}$ then H must contain all of the points $l_{1,k+1}, l_{1,k+2}, ..., l_{m+2k+1}$. In this case we have that

$$|H_P| \geq m + 2k + 1 - k = m + k + 1 = \lceil n/3 \rceil$$

This proves that $l_{1,k+1}$ is a centerpoint of P.

We now show that no other point p can be a centerpoint. We do this through three cases. For all three cases, Figure 2 illustrates the configuration.

Case 1: p does not lie on the line extending R_1
Consider the line l parallel with R_1 and going through p. The line l either intersects R_2 or R_3. Let H be the hyperplane with l as boundary line and containing the side of the plane with the ray R that l intersects. Then H only contains points of P lying on the ray R which is either R_2 or R_3. Thus $|H_P| \leq \max(|L_2|, |L_3|) = m + k < \lceil n/3 \rceil$ which shows p is not a centerpoint.

Case 2: p lies on the line extending R_1 and there are strictly less than $k + 1$ points of L_1 below p.
Let $L_{1,\geq p}$ be the points of L_1 above p and $L_{1,\leq p}$ the points on L_1 below p. Note that $|L_{1,\leq p}| < k + 1$.

The line l is a line through p which separates the plane such that all the points in L_2 are on the same side as the points in $L_{1,\geq p}$ and all the points in L_3 are on the same side as the points in $L_{1,\leq p}$. For the hyperplane H bounded by l and containing L_3 it then holds that $H_P = L_{1,\leq p} \cup L_3$, so $|H_P| = |L_{1,\leq p}| + |L_3| < k + 1 + m = \lceil n/3 \rceil$. Thus p is not a centerpoint in this case either.

Case 3: p lies on the line extending R_1 and there at least $k + 1$ points of L_1 below p.

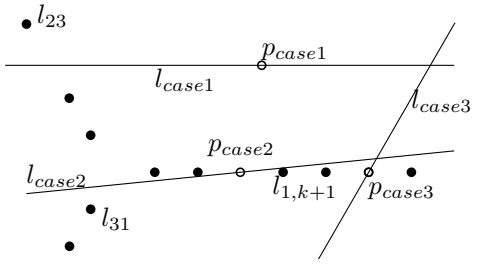


Figure 2: The main planar point construction and the three cases in the proof of lemma 1. ($m = 2$ and $k = 1$)

Let $L_{1,\geq p}$ and $L_{1,\leq p}$ have the same meaning as before. Since $p \neq l_{1,k+1}$, $|L_{1,\geq p}| \leq n+2k+1-(k+1) = n+k+m$. The line l is a boundary line through p parallel with R_2 and H the hyperplane bounded by l containing $L_{1,\geq p}$. Then H will not contain any points of either L_2 or L_3, and thus $|H_P| = |L_{1,\geq p}| \leq n + k + m < \lceil n/3 \rceil$. This shows p cannot be a centerpoint.

These three cases contain all possibilities for $p \neq l_{1,k+1}$ and completes the proof of the lemma.

3.1.1 Transformation of numbers into planar points

The sorting algorithms are given as input n real numbers $x_1, ..., x_n$. Let $x^* = 1 + \max_{i=1,...,n} x_i$. The numbers are then transformed into the following sets of planer points

$$L_1 = \{(x_i, 0) \mid i = 1, ..., n\} \cup \{(x^*, 0)\}$$
$$L_2 = \{(\cos(2\pi/3) \cdot x_i,\ \sin(2\pi/3) \cdot x_i) \mid i = 1, ..., n\}$$
$$L_3 = \{(\cos(4\pi/3) \cdot x_i,\ \sin(4\pi/3) \cdot x_i) \mid i = 1, ..., n\}$$

For the rest of this section $y_1, ..., y_n$ will be the input numbers $x_1, ..., x_n$ in sorted order, so $y_1 \leq y_2 \leq ... \leq y_n$.

3.2 Sorting using deletions

$\mathsf{Sort}_{\mathsf{delete}}$ works by first transforming the input points into $P = L_1 \cup L_2 \cup L_3$ as described above. At this point the only centerpoint of P is $(y_1, 0)$. Then we delete $c = (y_1, 0)$ and c rotated by $2\pi/3$ and $4\pi/3$ around origo. By lemma 1 the only centerpoint of P is now $(y_2, 0)$. We iteratively delete the current centerpoint and its two rotations around origo, and by lemma 1 after each deletion of these 3 points, the only centerpoint is $(y_i, 0)$ with y_i being the next number in sorted order.

$\mathsf{Sort}_{\mathsf{delete}}$ uses $O(n)$ centerpoint deletions and $O(n)$ instructions outside these.

3.3 Sorting using insertions

$\mathsf{Sort}_{\mathsf{insert}}$ works in a similar way as $\mathsf{Sort}_{\mathsf{delete}}$. It starts by transforming the input points into the same planar construction P. Again by lemma 1, the only centerpoint is $(y_1, 0)$. We then iteratively add x^* twice to leg 1 and once rotated to leg 2. After each triple of insertions lemma 1 gives that the only centerpoint is $(y_i, 0)$ with y_i being the next point in sorted order. Thus, we can do sorting using insertions.

$\mathsf{Sort}_{\mathsf{insert}}$ uses $O(n)$ insertions and $O(n)$ instructions outside these.

Theorem 2 *Given a point set P, maintaining a centerpoint of P under $O(n)$ insertions or deletions of points requires worst case $\Omega(n \log(n))$ operations.*

Proof: We consider algorithms which can be represented by an algebraic decision tree [1]. This first of all requires that the fundamental operations of the algorithm are additions, subtractions, multiplications, divisions and square root. Next, the branching of the algorithm must only depend on comparisons and data lookup by either referencing local variables directly or simple integer indexing into arrays.

The algorithms $\mathsf{Sort}_{\mathsf{delete}}$ and $\mathsf{Sort}_{\mathsf{insert}}$ implement `sort` in $O(n \cdot$ `delete`$)$ time and $O(n \cdot$ `insert`$)$ time respectively. For any implementation of `insert` and `delete` representable by an algebraic decision tree, $\mathsf{Sort}_{\mathsf{delete}}$ and $\mathsf{Sort}_{\mathsf{insert}}$ are representable by an algebraic decision tree. The complexity of `sort` is $\Omega(n \log n)$ in this model [1], thus the worst case complexity of doing $O(n)$ `delete` or `insert` operations is $\Omega(n \log n)$.

4 Conclusion

We have developed an algorithm for maintaining centerpoints under insertions and deletions in amortized $O(\log n)$ time for insertions and $O(\log^2 n)$ for deletions. We also showed an amortized $\Omega(\log n)$ lower bound for insertions and deletions, thus achieving amortized asymptotically optimal insertions of points.

It remains an open problem whether or not it is possible to support deletions in amortized $O(\log n)$ time. The limitation of our method with regards to supporting deletions stems from the need to recursively delete added Radon points. Doing so naively requires using our $O(\log n)$ insertion algorithm to insert some additional points, yielding an $O(\log^2 n)$ bound.

5 Acknowledgements

This paper was written as part of the Talent Track Program[1] for Bachelor students at the Department of Computer Science at Aarhus Univerity, Denmark. We thank Peyman Afshani and Gerth Stølting Brodal from the department for their advice as well as the anonymous referees for their valuable feedback.

[1] `http://studerende.au.dk/studier/fagportaler/datalogi/undervisning/talentforloeb/`

References

[1] M. Ben-Or. Lower bounds for algebraic computation trees. *Proceedings of the fifteenth annual ACM symposium on Theory of computing*, pages 80–86, 1983.

[2] T. M. Chan. An optimal randomized algorithm for maximum tukey depth. In *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '04, pages 430–436, Philadelphia, PA, USA, 2004. Society for Industrial and Applied Mathematics.

[3] K. L. Clarkson, D. Eppstein, G. L. Miller, C. Sturtivant, and S.-H. Teng. Approximating center points with iterative radon points. *International Journal of Computational Geometry & Applications*, 6(03):357–377, 1996.

[4] E. Helly. Uber mengen konvexer körper mit gemeinschaftlichen punkte. *Math.-Verein.*, pages 175–176, 1923.

[5] S. Jadhav and A. Mukhopadhyay. Computing a centerpoint of a finite planar set of points in linear time. *Discrete and Computational Geometry*, pages 291–312, 1994.

[6] J. Matoušek. *Lectures on discrete geometry.* Springer, 2002.

[7] N. Megiddo. Partitioning with two lines in the plane. *J. Algorithms*, pages 430–433, 1985.

[8] G. L. Miller and D. R. Sheehy. Approximate centerpoints with proofs. *Computational Geometry*, 43(8):647–654, 2010.

[9] N. Naor and M. Sharir. Computing a point in the center of a point set in three dimensions. *CCCG: Canadian Conference in Computational Geometry*, 01 1990.

I/O Optimal Data Structures for Categorical Range Skyline Queries

Arnab Ganguly* Daniel Gibney† Rahul Shah‡ Sharma V. Thankachan §

Abstract

The skyline of a set of two-dimensional points is the subset of points which are not dominated by any other point. In this paper, we deal with two-dimensional points (in rank space) which are assigned an additional category, or color. The goal is to preprocess n points so that given a three-sided region of the form $[a, b] \times [\tau, \infty]$ we can return the set of colors on the skyline of the queried region. We approach the problem in the external memory model and present an I/O optimal data structure requiring $O(n \log^* n)$ words of space.

1 Introduction

Skyline queries ask the following: given a set of points P, what are the points in P that are not dominated by any other point in P. In two dimensions a point $p_2 = (x_2, y_2)$ dominates a point $p_1 = (x_1, y_1)$ if $x_2 \geq x_1$ and $y_2 > y_1$ or $x_2 > x_1$ and $y_2 \geq y_1$. That is, p_2 is at least the value of p_1 in one dimension and larger than p_1 in the other dimension. This paper deals with a variation on this problem called **categorical range skylines**. In a categorical range skyline we begin with a set of two-dimensional points P, where every point in P has been assigned to a category, which we synonymously call a color. Given the point set P, we would like to build a data structure such that for any three sided region of the form $\mathcal{R} = [a, b] \times [\tau, \infty]$ we can answer in optimal I/O which colors are on the skyline of the region $\mathcal{R}$. A point in P is described by an x-coordinate i, y-coordinate $A[i]$, and color $C[i]$. We assume the points are in rank space, that is, the points are on an $n \times n$ grid where no two points share the same x-coordinate or y-coordinate.

Problem 1.1 *Given a set of points P in rank space of the form $(i, A[i], C[i])$, preprocess P so that given a query $\mathcal{R} = [a, b] \times [\tau, \infty]$ the query response is the distinct values of $C[\cdot]$ present on the skyline of $\mathcal{R}$.*

Theorem 1 *Under the external memory model of computation, Problem 1.1 can be answered in optimal I/O using a near-linear space index. Specifically, we present an $O(1 + \frac{k}{B})$-I/O solution using a data structure which requires $O(n \log^* n)$ words of space, where n is the number of points in P, B is the block size, k is the output size, and $\log^*$ is the iterated logarithm.*[1]

In the full version of this paper we give a data structure which uses linear space and answers queries in $O(\log^* n + \frac{k}{B})$ I/O.

2 Related Work

The proposal to incorporate the skyline operator into database systems came from Börzsönyi et al., in 2001 [3]. Since then there have been notable results, for example, an I/O optimal solution by Papadias et. al [24], and an online algorithm that allows for efficiently finding an element on the skyline, which then provides the user the ability to direct the search along the skyline [14]. In a slightly different setting, Rahul and Janardan [28] considered points in d-dimensions, where each point has t additional features. They showed how to answer skyline queries using $O(n \log^{d+t-1} n)$ space and $O((1 + occ) \log^{d+t} n)$ I/O; here, n is the number of points in the dataset and occ is the output size. They also showed that if $d \geq 2$ and $t \geq 3$, reporting the number of points on the skyline is at least as hard as the set-intersection problem. Other works on skyline include finding the skyline in uncertain databases [29], finding k-dominant skylines [6], computing categorical skyline in streaming data [30]; see [12] for a catalog of problems related to skyline computation.

The colored (or categorical) range reporting problem has also received significant attention [4, 5, 9, 10, 11, 18, 19, 22, 23]. Other variants, such as top-k color reporting [13, 20] have also been studied. The problem of approximate range counting has also been approached in [27]. In the external memory model, one-dimensional color reporting has seen a series of improvements [2, 15, 21], culminating in the I/O optimal index of Nekrich and Vitter [23]. For answering two-dimensional and three-sided categorical range queries, Larsen and Walderveen [17] presented an $O(\log_B n + \log_B^{(h)} n + \frac{k}{B})$ I/O index that occupies $O(nh)$ words of space, where $1 \leq h \leq \log^* n$. This allows for a trade-off between space and memory. For example, one can obtain either a linear space solution requiring

*University of Wisconsin - Whitewater, gangulya@uww.edu
†University of Central Florida daniel.gibney@ucf.edu
‡Louisiana State University rahul@csc.lsu.edu
§University of Central Florida sharma.thankachan@ucf.edu

[1] $\log^* n$ is the iterated logarithm. It represents the number of times log has to be applied to n before the result is ≤ 1.

$O(\log_B n + \frac{k}{B})$ I/O per query, or a $O(n \log^* n)$ space solution requiring $O(\log_B n + \frac{k}{B})$ I/O per query. To answer four-sided queries, standard techniques lead to an index with a $\log n$ multiplicative factor in space which retains the query time. Patil et al. [25] propose a solution for Categorical Range Maxima Queries with a similar space-time trade off.

We obtain our results by modifying the double chaining techniques presented in [25], allowing for the identification of unique highest instances of colors on the skyline. We also demonstrate a novel way to combine precomputed answers in the bootstrapping.

3 Preliminaries

External Memory Model: Given its applicability to large data sets, it makes sense to analyze algorithms related to Problem 1.1 under the external memory model. The external memory model has a two-level memory structure, where the main (internal) memory is fast and is of bounded size, whereas the external memory (disk) is much slower, and has unlimited storage. Although arithmetic operations can be performed on the data stored in the main memory, the data structure, owing to the huge size of the data, does not fit in the main memory and must be stored on the disk. Using one I/O operation, we can read a contiguous block of B words from disk into the main memory or write a block of B words from main memory into the external disk. All main memory operations are considered free and the algorithm complexity is measured in the number of I/O operations. Thus, this model of computation is appropriate in cases where the transfer of data between external memory and the processor dominates the computation time.

Three-dimensional Dominance Reporting: Given a set of three-dimensional points P, we can build a data structure such that for any query point $q = (q_1, q_2, q_3)$ we can report points $(x_1, x_2, x_3) \in P - \{q\}$ where $q_1 \leq x_1$, $q_2 \leq x_2$, and $q_3 \leq x_3$. That is, we can report all points dominating q. This can be done in linear space requiring $O(\log_B n + \frac{k}{B})$ I/O per query [1].

Three-sided Orthogonal Range Reporting: Given a set of two-dimensional points P in rank space, we can build a data structure such that for any query region $\mathcal{R} = [a, b] \times [\tau, \infty]$ we can report the points in $P \cap \mathcal{R}$. This can be done in linear space with an optimal $O(1 + \frac{k}{B})$ I/O per query [16].

4 Internal data structure

In this section, we outline a data structure that provides solutions to Problem 1.1 in linear space using

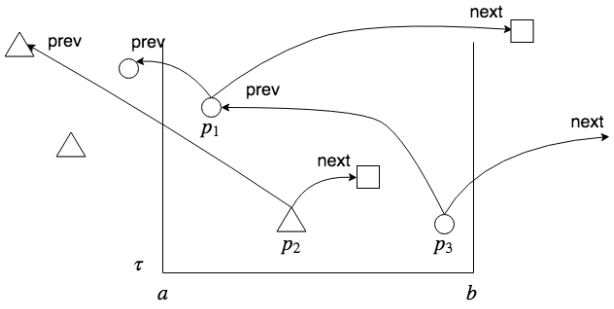


Figure 1: An example of the two sided chaining. The point p_1 is the only point that satisfies the conditions 4.1. The point p_2 has a *next* value less than b. The point p_3 has a *prev* value greater than a.

$O(\log^3 \frac{n}{B} + \frac{k}{B})$ I/O per query. Although sub-optimal in terms of I/O, it will provide us with the essential building block to arrive at the desired results (via bootstrapping technique to be discussed later). The overall idea is to first augment each data point with a pair of intervals, and then build three dimension dominance reporting structures over some subsets of the points.

We begin by describing the pair of intervals that each point is augmented with. Later at query time, we shall check if one of the intervals is stabbed by the value a and the other by the value b. Thus, finding all candidate pairs of intervals that meet this criterion can be found using nested interval trees. The remaining inequalities are checked using 3D dominance reporting structures.

4.1 Double Chaining

The notion of chaining to create intervals was introduced by Gupta et al., [8] and expanded by [25]. We present a variation on the idea. For each data point $(i, A[i], C[i])$ we define:

$$prev(i) = \max\{\{j \in [1, i) \mid C[i] = C[j] \text{ and } A[j] > A[i] \text{ and } (j, A[j]) \text{ is not dominated by any point } (k, A[k]) \text{ where } j < k \leq i\} \cup \{-\infty\}\}$$

$$next(i) = \min\{\{j \in (i, n] \mid A[i] < A[j]\} \cup \{\infty\}\}$$

See Figure 1 for a graphical illustration. The following lemma establishes the role of *next* in solving the problem. It's proof can be easily established recognizing that $next(i)$ gives the first component of the next point which dominates $(i, A[i])$.

Lemma 2 *A point $(i, A[i], C[i])$ in the region $\mathcal{R} = [a, b] \times [\tau, \infty]$ is on the skyline of $\mathcal{R}$ iff $next(i) > b$. (proof Appendix A)*

In other words, the *next* values allow us to check which points are on the skyline. The role of the *prev* value is established by the following lemma.

Lemma 3 *A point $(i, A[i], C[i])$ on the skyline of the region $\mathcal{R} = [a, b] \times [\tau, \infty]$ is the highest instance of color $C[i]$ on the skyline iff $prev(i) < a$.(proof Appendix B)*

Combining Lemmas 2 and 3, we see that a point $p = (i, A[i], C[i])$ is the highest instance of color $C[i]$ on the skyline of region $\mathcal{R} = [a, b] \times [\tau, \infty]$ iff $prev(i) < a$ and $next(i) > b$. See Figure 1 for a motivating example. Combining the remaining constraint that the points be in $\mathcal{R}$, our solutions to Problem 1.1 will be the points that satisfy the following conditions.

Condition 4.1 *The necessary and sufficient conditions for a point $(i, prev(i), next(i), A[i], C[i])$ to be in the solution set for a query $[a, b] \times [\tau, \infty]$ are: $a \leq i$, $i \leq b$, $A[i] \geq \tau$, $prev(i) < a$, and $next(i) > b$.*

Crucially, the last two constraints, can be phrased in terms of interval stabbing. The backward interval $(prev(i), i]$ must be stabbed by a. Here, by using the notation $(a, b]$ being stabbed by c, we mean that $a < c \leq b$. Similarly, the 'forward' interval $[i, next(i))$ must be stabbed by b. To notate the connection between these interval pairs and the original point $(i, A[i], C[i])$ we use the five tuple $(i, prev(i), next(i), A[i], C[i])$. This can be viewed as weighted rectangle stabbing.

It is worth mentioning, for categorical range skylines the Word-RAM model, optimal solutions for 5-sided rectangle stabbing in 3D can adapted [7]. Similarly, in the pointer machine model we can adapt near optimal solutions for 5-sided rectangle stabbing in [26]. However, we study this problem in the I/O model and our techniques are similar to the ones in [25, 26].

4.2 First level interval Tree

We first describe how to construct an interval tree which allows us to find all candidate points where $(prev(i), i]$ is stabbed by a in $O(\log \frac{n}{B})$ I/O.

We begin with a balance binary search tree using the right end points of the intervals $(prev(i), i]$ for $1 \leq i \leq n$ as the keys. For every node v_i for $1 \leq i \leq n$ we assign a set of intervals.

$$\mathcal{I}(i) = \{(prev(j), j] \mid (prev(j), j] \text{ is stabbed by } i \text{ and } (prev(j), j] \text{ is not stabbed by } k \text{ for any ancestor } v_k\}$$

Although simple, we establish the following fact because it allows us to later truncate the trees while maintaining linear space.

Lemma 4 *Every interval $(prev(i), i]$ is included in exactly one node's interval set.(proof Appendix C)*

To figure out which intervals are potentially stabbed by a, we can traverse the tree from the root until we find the vertex v_i corresponding to the predecessor of a. Then, any interval stabbed by a must belong to an interval set along the path from the root to v_i. Identifying these intervals takes $O(\log n)$ I/O.

Remembering that the target number of I/O in this section is in fact $O(\log \frac{n}{B})$, we now modify the tree slightly. Let $size(v)$ denote the number of nodes in a vertex v's subtree. We next identify the nodes, v_i, where $size(v_i) \leq B$ and the size of v_i's parent is greater than B. Using $subtree(i)$ to denote the indices belonging to the nodes of the subtree rooted at v_i, we then set $\mathcal{I}(i) = \cup_{j \in subtree(i)} \mathcal{I}(j)$. The height of each subtree removed was $\log B$. It follows that the height of the resulting tree, and thus the worst case search required to identify the interval sets of interest, is now $\log n - \log B = O(\log \frac{n}{B})$.

4.3 Second level interval tree

For each node v_i, we now have a set of intervals $\mathcal{I}(i)$. Each of these backward intervals, say $(prev(j), j] \in \mathcal{I}(i)$ has it paired forward interval $[j, next(j))$. For this set of forward intervals, which are paired with the intervals in $\mathcal{I}(i)$, we now construct an interval tree. Just as above, for each j we create a node v_j in a balanced binary search tree. Define an interval set for the node v_j,

$$\mathcal{I}(j) = \{[h, next(h)) \mid [h, next(h)) \text{ is stabbed by } j \text{ and } [h, next(h)) \text{ is not stabbed by } k \text{ for any ancestor } v_k\}$$

Again we combine the interval sets by unioning the interval sets of nodes where the size is less than B into a single interval set. Just as above, this makes the height of the tree $O(\log \frac{n}{B})$.

To query which of the forward intervals derived from $\mathcal{I}(i)$ are stabbed by b, we can traverse the tree from the root until we find the successor of b, say v_s. Then, all of the potential forward intervals are in some interval set for the nodes from the root to v_s.

To get nodes whose interval sets potentially contain intervals where the backward and the forward intervals are stabbed by a and b respectively, we traverse the interval tree described in section 4.2, gather up $O(\log \frac{n}{B})$ nodes on the first level. Then for each of these first level nodes, v_i, we traverse their interval tree, gathering $O(\log \frac{|\mathcal{I}(i)|}{B})$ nodes corresponding to $O(\log \frac{|\mathcal{I}(i)|}{B})$ subsets of $\mathcal{I}(i)$. Based on the heights of the trees, the total number of I/O for this is $O(\log^2 \frac{n}{B})$.

In terms of space, notice by Lemma 4 that every backward interval appears in only one node's interval set in the first-level interval tree. This implies the paired forward interval will only appear in one second-level interval tree. By the same argument, this paired forward interval appears only once in its second-level interval tree. Therefore, a forward and backward interval pair is only stored once in the entire structure. Hence, the total space used by this data structure is linear.

By the time we have finished traversing the nested interval tree, we have gathered a set of points (without explicitly reporting them) such that any point which is in our solution set is in this set as well. We need an additional data structure to check which of these points satisfy the first three inequalities in Condition 4.1.

4.4 Dominance Queries

Let v_i be a node in the first-level interval tree. Let v_j be a node in the second-level interval tree which is built off of the intervals in $\mathcal{I}(i)$. Let $\mathcal{I}(i,j) = \mathcal{I}(i) \cap \mathcal{I}(j)$ and suppose we obtained $\mathcal{I}(i,j)$ while querying on the values a and b. Remember that we traverse to a predecessor of a on the first-level interval tree and a successor of b on the second-level interval tree. We have the following cases to consider in order to identify which points corresponding $\mathcal{I}(i,j)$ satisfy the conditions 4.1:

Case 1. $a \leq i \leq j \leq b$: We need to find the points $(h, prev(h), next(h), A[h], C[h]) \in \mathcal{I}(i,j)$ such that $A[h] \geq \tau$, $prev(h) < a$, and $next(h) > b$.

Case 2. $i \leq a \leq j \leq b$: We need to find the points $(h, prev(h), next(h), A[h], C[h]) \in \mathcal{I}(i,j)$ such that $A[h] \geq \tau$, $a \leq h$, and $next(h) > b$.

Case 3. $a \leq i \leq b \leq j$: We need to find the points $(h, prev(h), next(h), A[h], C[h]) \in \mathcal{I}(i,j)$ such that $A[h] \geq \tau$, $prev(h) < a$, and $h \leq b$.

Case 4. $i \leq a \leq b \leq j$: We need to find the points $(h, prev(h), next(h), A[h], C[h]) \in \mathcal{I}(i,j)$ such that $A[h] \geq \tau$, $a \leq h$, and $h \leq b$.

Although we can't know prior to seeing a and b which of these four cases will apply, we do have the values i, $A[i]$, $next(i)$, $prev(i)$ at our disposal in pre-processing. We can therefore prepare each interval set $\mathcal{I}(i,j)$ corresponding points $(i, prev(i), next(i), A[i], C[i])$ for answering four types of three-dimensional dominance queries. These four queries correspond to the four cases above. We build the data structure for answering:

Q1. On points $(A[i], -prev(i), next(i))$ for query point $q = (\tau, -a, b)$ we can find all points dominating q.

Q2. On points $(A[i], i, next(i))$ for query point $q = (\tau, a, b)$ we can find all points dominating q.

Q3. On points $(A[i], -prev(i), -i)$ for query point $q = (\tau, -a, -b)$ we can find all points dominating q.

Q4. On points $(A[i], i, -i)$ for query point $q = (\tau, a, -b)$ we can find all points dominating q.

Using the data structure described in section 3 we can build each of these dominance reporting structures in linear space. From this it follows that the overall space usage of the structure remains linear. The query I/O for the dominance reporting structure is $O\left(\log_B |\mathcal{I}(i,j)| + \frac{k_{i,j}}{B}\right) = O\left(\log \frac{n}{B} + \frac{k_{i,j}}{B}\right)$, where $k_{i,j}$ is the number of solutions present in this interval set. It follows then that the total number of I/O is $O\left(\log^2 \frac{n}{B} \log \frac{n}{B} + \sum_{i,j} \frac{k_{i,j}}{B}\right) = O\left(\log^3 \frac{n}{B} + \frac{k}{B}\right)$.

4.5 Summary of internal data structure

In summary, the data structure consists of an interval tree with a nested interval tree for each node. Three-dimensional dominance structures are built for every second level node. The first level interval tree is for the backward intervals, where each node is assigned a subset of intervals. The second level interval tree of v_i is built for the forward intervals whose backward intervals are in v_i's interval set. Each second level node has three-dimensional dominance reporting structures to answer queries of the types **Q1**, **Q2**, **Q3**, and **Q4**. The total space is linear.

Querying is then done by traversing the first level interval tree until the predecessor of a is reached. All vertices along this path have their second level interval tree searched until the successor of b is reached. Then all vertices along the path on the second level trees have a three-dimensional queries applied to them, depending on which of the four cases apply. The number of I/O per query is $O\left(\log^3 \frac{n}{B} + \frac{k}{B}\right)$.

5 A $O(n \log^* n)$ space solution with optimal I/O

We are now ready to apply bootstrapping to convert our linear space, $O\left(\log^3 \frac{n}{B} + \frac{k}{B}\right)$ I/O solution to a solution with $O(n \log^* n)$ words of space and a query cost of $O(1 + \frac{k}{B})$ I/O.

The overall idea is to store the answers to subproblems in such a way that they can be efficiently recombined into the correct solution. Notice first that we only have to worry about the case where the $\log^3 \frac{n}{B}$ term dominates. Therefore, we consider in this section when it is the case that $B \log^3 \frac{n}{B} \geq k$.

5.1 Data Structure

We decompose the query range as in Figure 5. On internal sub-ranges we will use pre-computed solutions, whereas on the sub-ranges on the far ends we will use the data structure described in Section 4. Define $\Delta_j = B(\log^{(j)} \frac{n}{B})^5$ and $k_j = B(\log^{(j)} \frac{n}{B})^3$ for $j = 1, \ldots, \log^* \frac{n}{B}$, and $\Delta_0 = n$. WLOG we assume that all Δ_j values are multiples of 2.

- **Internal Structures**: First, partition the array A into Δ_j disjoint blocks for $j = 0, \ldots, \log^* n$. For each of these blocks we build the internal structure

described in section 4. The space for storing the internal structures is linear. Summing over j from 1 to $\log^* n$, the total cost is $n \log^* n$.

- **Precomputed solutions (type 1):** For each Δ_j block $A[i, i+\Delta_j]$, for $1 \le i \le \lfloor \frac{n}{\Delta_j} \rfloor, 1 \le j \le \log^* n$, create the blocks $\Delta_j, 2\Delta_j, 4\Delta_j, 8\Delta_j, \ldots$, each starting from $A[i, i+\Delta_j]$'s left boundary and continuing in powers of two until the next power of two would cause it to cross a Δ_{j-1} boundary. For each of these blocks store k_j points from the solution on that block's skyline. Do this in order from the point whose $C[\cdot]$ value is highest to the point whose $C[\cdot]$ value is lowest.

The space to store the $k_j \frac{n}{\Delta_j}$ answers per j is

$$O\left(\frac{n}{\Delta_j} k_j \log \frac{\Delta_j}{\Delta_{j-1}}\right) = O\left(n \frac{\log \frac{\Delta_j}{\Delta_{j-1}}}{(\log^{(j)} \frac{n}{B})^2}\right) = O(n)$$

Summing over all j from $0, \ldots, \log^* n$ gives us that the space overall is $O(n \log^* n)$ words.

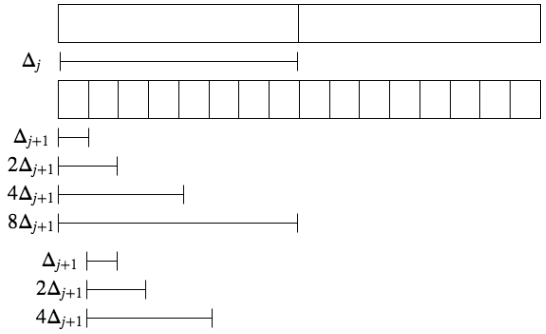


Figure 2: Type 1 precomputed solutions are stored for multiples of two for the blocks of size Δ_{j+1} until the they would cross the next Δ_j boundary.

Next we add one more type of precomputed solution.

- **Precomputed solutions (type 2):** For each block of size Δ_j we store the top k_j solution points on the skyline for the span starting on the right boundary of the block and ending at the next boundary of a block with size h for all $h < j$, for $1 \le j \le \log^* n$. Also, for each block $A[i, i+\Delta_j]$ of size Δ_j store the top k_j solution points on the skyline for the span starting at rightmost boundary of the preceding block with size h for all $h < j$ and ending at the leftmost boundary of $A[i, i+\Delta_j]$. For an illustration of the concept, consider the values $h_1, h_2 < j$ in Figure 3.

The total space taken by these precomputed solutions is k_j times the number of precomputed solutions per

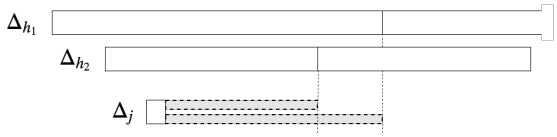


Figure 3: Here $h_1, h_2 < j$. Type 2 precomputed solutions are stored from the right boundary of a Δ_j block to first boundary of a Δ_{h_1}. They are also stored to the first boundary of a Δ_{h_2} block. The regions are shown in the grey rectangles.

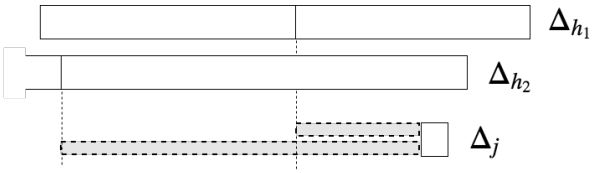


Figure 4: Here $h_1, h_2 < j$. Type 2 precomputed solutions are stored from the left boundary of a Δ_j block to the first boundary of a Δ_{h_1} block. They are also stored to the first boundary of a Δ_{h_2} block. The regions are shown in the grey rectangles.

block which is $j-1$ times the total number of blocks of size Δ_j, that is $\frac{n}{\Delta_j}$. This gives

$$O\left(\sum_{j=1}^{\log^* \frac{n}{B}} k_j \frac{n}{\Delta_j}(j-1)\right) = O\left(\sum_{j=1}^{\log^* \frac{n}{B}} n \frac{j-1}{\left(\log^{(j)} \frac{n}{B}\right)^2}\right) = O\left(n \log^* n\right).$$

5.2 Range Decomposition and Querying

We are given a query region $[a, b] \times [\tau, \infty]$. We first suppose the output size is $k_{\pi+1} < k \le k_\pi$ (we will show how to deal with not knowing the value of k in section 6). Let $[a^\pi, b^\pi]$ be the longest span of blocks of size Δ_π inside $[a, b]$. Let j be the largest value of j such that $[a^\pi, b^\pi]$ completely contains a Δ_j block. Let the maximal span of say m blocks of size Δ_j inside $[a^\pi, b^\pi]$ consist of the blocks $[a^j, a^j + \Delta_j], [a^j + \Delta_j + 1, a^j + 2\Delta_j], \ldots, [a^j + (m-1)\Delta_j + 1, a^j + m\Delta_j]$.

We claim that $[a^j, a^j + m\Delta_j]$ can be covered by two overlapping ranges of length $2^{\ell_1}\Delta_j$ and $2^{\ell_2}\Delta_j$ for some $\ell_1, \ell_2 \ge 0$. These ranges can start from different left boundaries of Δ_j blocks within $[a^j, a^j + m\Delta_j]$. To see that this is always true, take $\ell_1 = \lfloor \log m \rfloor$ with the span starting from the index i, and $\ell_2 = \lfloor \log m \rfloor$ with the span starting from the index $i + m\Delta_j - \ell_2 + 1$.

We decompose the span $[a, b]$ into six parts. They are: $R_1 = [a, a^\pi]$, $R_2 = [a^\pi, a^j]$, $R_3 = [a^j, a^j + 2^{\ell}_1 \Delta_j]$, $R_4 = [a^j + m\Delta_j - 2^{\ell_2}\Delta_j + 1, a^j + m\Delta_j]$, $R_5 = [a^j + m\Delta_j, b^\pi]$, and $R_6 = [b^\pi, b]$. See Figure 5 for an illustration.

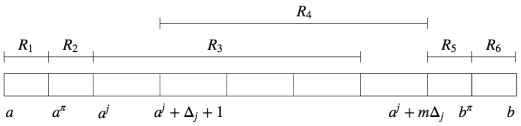


Figure 5: The six subspans in the decomposition of $[a, b]$. R_1 and R_6 are solved with the internal structure, R_2 and R_5 with type 2 precomputed solutions, and R_4 and R_3 with type 1 precomputed solutions.

We process the sub-problems in the order R_6, R_5, ..., R_1. We keep track of the highest point on the skyline seen so far. That value gets set as the τ value on the next sub-problem, see Figure 6. This is necessary to ensure that we do not output points that are not on the skyline of the original queried range.

5.2.1 Processing R_6 and R_1

For R_6 and R_1 we get the solutions from the internal data structure for the blocks $[b^\pi + 1, b^\pi + \Delta_\pi + 1]$ and $[a^\pi - \Delta_\pi, a^\pi - 1]$ respectively. The number of I/O required is

$$O\left(\log^3 \frac{\Delta_\pi}{B} + \frac{k_\pi}{B}\right) = O\left(\frac{k_{\pi+1}}{B} + \frac{k_\pi}{B}\right)$$

Under the assumption that $k_{\pi+1} < k < k_\pi$ we have that $\frac{k_\pi}{B}$ dominates $\frac{k_{\pi+1}}{B}$ asymptotically, hence this happens in $O(1 + \frac{k_\pi}{B})$ I/O.

For R_6, while getting the results from the internal structure we keep track of the largest $A[\cdot]$ seen so far and set it as the τ value for the next subrange, R_5. We iterate through the precomputed solutions stopping when $A[\cdot] < \tau$. When processing R_6 we check for every point reported by the internal structure whether its *prev* value is less than a. If it is we report it as part of the solution, otherwise we do not. For R_1 we report all solutions.

5.2.2 Processing Precomputed Solutions R_5, R_4, R_3, and R_2

When we arrive at a span we have a value τ from the previously processed span to the right. Recall that the answers are stored from the highest leftmost instance of a skyline color to the lowest. We take the first $A[\cdot]$ reported within the precomputed solution and set that as the τ value for the next span. Following this, we process each point. If a point has a $A[i]$ value less than τ, we break and move onto the next span. If a point has a $prev(i)$ value which is less than a we report it as part of the solution, otherwise we don't report it. Notice that a given color only gets its points rejected a constant number of times.

5.2.3 Summary

The essential reason for decomposing the interval in the way demonstrated in Figure 5 is that the internal structure from section 4 can be used on the far ends, where as a constant number of type 1 and type 2 solutions can be used inside the $[a^\pi, b^\pi]$ interval. Having a constant number of sub-problems implies that the number of times a solution will be rejected so as to not be reported multiple times will be a constant as well. As a result the total I/O's are $O\left(1 + \frac{k}{B}\right)$.

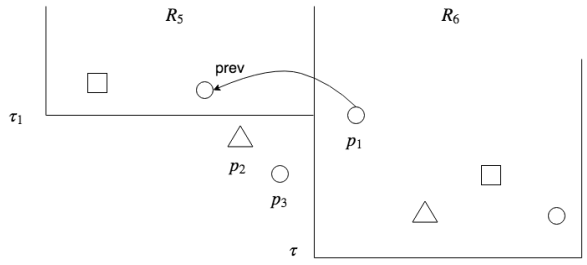


Figure 6: When processing R_6 and R_5 we first take the $A[\cdot]$ value of p_1 and set as the lower bound of the next range. This is necessary otherwise p_2 and p_3 might be included in the solution set. The point p_1 however is not included in the final output since its *prev* value is greater than a.

6 Computing Output Size

Applying the algorithms described in section 5 requires knowing the output size of the solution k. Fortunately, determining the value of k is relatively simple.

We have that there exists data structures for answering queries of size k where $k_j < k \leq k_{j+1}$ and $1 \leq j \leq \log^* \frac{n}{B} - 1$. Therefore, we can apply the procedure described below to obtain the solution.

for all $j = \log^* \frac{n}{B} - 1$ down to 1 **do**
  Run procedure (assuming $k_j < k \leq k_{j+1}$).
  if the output size is k_{j+1} **then**
    continue
  end if
  if the output size is less than k_{j+1} **then**
    return output
  end if
end for

The number of I/O used running this procedure is

$$O\left(1 + \frac{k_{\log^* \frac{n}{B}-1}}{B} + \frac{k_{\log^* \frac{n}{B}-2}}{B} + \ldots + \frac{k}{B}\right) = O\left(1 + \frac{k}{B}\right).$$

Acknowledgments Supported in part by the National Science Foundation under CCF-1527435.

References

[1] P. Afshani. On dominance reporting in 3d. In *European Symposium on Algorithms*, pages 41–51. Springer, 2008.

[2] L. Arge, V. Samoladas, and J. S. Vitter. On two-dimensional indexability and optimal range search indexing. In *Proc. 18th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*, pages 346–357, 1999.

[3] S. Borzsony, D. Kossmann, and K. Stocker. The skyline operator. In *Data Engineering, 2001. Proceedings. 17th International Conference on*, pages 421–430. IEEE, 2001.

[4] P. Bozanis, N. Kitsios, C. Makris, and A. K. Tsakalidis. New upper bounds for generalized intersection searching problems. In *Proc. 22nd International Colloquium on Automata, Languages and Programming (ICALP 95)*, pages 464–474, 1995.

[5] P. Bozanis, N. Kitsios, C. Makris, and A. K. Tsakalidis. New results on intersection query problems. *Comput. J.*, 40(1):22–29, 1997.

[6] C. Y. Chan, H. V. Jagadish, K. Tan, A. K. H. Tung, and Z. Zhang. Finding k-dominant skylines in high dimensional space. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, Chicago, Illinois, USA, June 27-29, 2006*, pages 503–514, 2006.

[7] T. M. Chan, Y. Nekrich, S. Rahul, and K. Tsakalidis. Orthogonal point location and rectangle stabbing queries in 3-d. In *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, pages 31:1–31:14, 2018.

[8] P. Gupta, R. Janardan, S. Rahul, and M. Smid. Computational geometry: Generalized (or colored) intersection searching. *Handbook of Data Structures and Applications, to appear*, 2018.

[9] P. Gupta, R. Janardan, and M. H. M. Smid. Further results on generalized intersection searching problems: counting, reporting, and dynamization. *Journal of Algorithms*, 19(2):282–317, 1995.

[10] P. Gupta, R. Janardan, and M. H. M. Smid. Algorithms for generalized halfspace range searching and other intersection searching problems. *Comput. Geom.*, 6:1–19, 1996.

[11] R. Janardan and M. A. Lopez. Generalized intersection searching problems. *International Journal of Computational Geometry and Applications*, 3(1):39–69, 1993.

[12] C. Kalyvas and T. Tzouramanis. A survey of skyline query processing. *CoRR*, abs/1704.01788, 2017.

[13] M. Karpinski and Y. Nekrich. Top-k color queries for document retrieval. In *Proc. 22nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2011)*, pages 401–411, 2011.

[14] D. Kossmann, F. Ramsak, and S. Rost. Shooting stars in the sky: An online algorithm for skyline queries. In *Proceedings of the 28th international conference on Very Large Data Bases*, pages 275–286. VLDB Endowment, 2002.

[15] K. G. Larsen and R. Pagh. I/O-efficient data structures for colored range and prefix reporting. In *Proc. 23rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 583–592, 2012.

[16] K. G. Larsen and R. Pagh. I/o-efficient data structures for colored range and prefix reporting. In *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms*, pages 583–592. Society for Industrial and Applied Mathematics, 2012.

[17] K. G. Larsen and F. Van Walderveen. Near-optimal range reporting structures for categorical data. In *Proceedings of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms*, pages 265–276. SIAM, 2013.

[18] S. Muthukrishnan. Efficient algorithms for document retrieval problems. In *Proc. 13th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 657–666, 2002.

[19] A. Nanopoulos and P. Bozanis. Categorical range queires in large databases. In *Proc. 8th International Symposium on Advances in Spatial and Temporal Databases, (SSTD 2003)*, pages 122–139, 2003.

[20] G. Navarro and Y. Nekrich. Top-k document retrieval in optimal time and linear space. In *Proc. 23rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2012)*, pages 1066–1077, 2012.

[21] Y. Nekrich. External memory range reporting on a grid. In *Proc. 18th International Symposium on Algorithms and Computation (ISAAC 2007)*, pages 525–535, 2007.

[22] Y. Nekrich. Efficient range searching for categorical and plain data. *ACM Trans. Database Syst.*, 39(1):9, 2014.

[23] Y. Nekrich and J. S. Vitter. Optimal color range reporting in one dimension. In *Proc. 21st Annual European Symposium Algorithms (ESA 2013)*, pages 743–754, 2013.

[24] D. Papadias, Y. Tao, G. Fu, and B. Seeger. An optimal and progressive algorithm for skyline queries. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 467–478. ACM, 2003.

[25] M. Patil, S. V. Thankachan, R. Shah, Y. Nekrich, and J. S. Vitter. Categorical range maxima queries. In *Proceedings of the 33rd ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 266–277. ACM, 2014.

[26] S. Rahul. Improved bounds for orthogonal point enclosure query and point location in orthogonal subdivisions in $\setminus^3$. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 200–211, 2015.

[27] S. Rahul. Approximate range counting revisited. In *33rd International Symposium on Computational Geometry, SoCG 2017, July 4-7, 2017, Brisbane, Australia*, pages 55:1–55:15, 2017.

[28] S. Rahul and R. Janardan. Algorithms for range-skyline queries. In *SIGSPATIAL 2012 International Conference on Advances in Geographic Information Systems (formerly known as GIS), SIGSPATIAL'12, Redondo Beach, CA, USA, November 7-9, 2012*, pages 526–529, 2012.

[29] N. H. M. Saad, H. Ibrahim, F. Sidi, R. Yaakob, and A. A. Alwan. Computing range skyline query on uncertain dimension. In *Database and Expert Systems Applications - 27th International Conference, DEXA 2016, Porto, Portugal, September 5-8, 2016, Proceedings, Part II*, pages 377–388, 2016.

[30] N. Sarkas, G. Das, N. Koudas, and A. K. H. Tung. Categorical skylines for streaming data. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, Vancouver, BC, Canada, June 10-12, 2008*, pages 239–250, 2008.

A Proof of Lemma 2

First, suppose $p = (i, A[i], C[i])$ is on the skyline of region $\mathcal{R}$. Then by definition of the skyline, p is not dominated by any other point in $\mathcal{R}$. Hence, the next dominating point, if it exists, has a first component value greater than b. If it doesn't exist, then its value ∞ is greater than b. Conversely, if $next(i) > b$, then the next point which dominates p must be to the right of b, which implies p is not dominated by any point in $\mathcal{R}$.

B Proof of Lemma 3

Suppose $p = (i, A[i], C[i])$ is the highest instance of $C[i]$ on the skyline of region $\mathcal{R}$, and for the sake of contradiction, let us assume that $j = prev(i) \geq a$. We will show that the point $p_{prev} = (j, A[j], C[j])$ is on the skyline of region $\mathcal{R}$. Then, the fact that $C[i] = C[j]$ and $A[j] > A[i]$ will cause a contradiction. By the definition of $prev(i)$, we have that $A[j] \geq A[k]$ for $j < k \leq i$. Combining this with the fact that $A[i] \geq A[\ell]$ for $i < \ell \leq b$, we have that $A[j] > A[h]$ for $j < h \leq b$ and p_{prev} is not dominated in $\mathcal{R}$. This implies p_{prev} is on the skyline.

In the other direction, assume $j = prev(i) < a$. Suppose for the sake of contradiction that $(i, A[i], C[i])$ is not the highest instance of $C[i]$ on the skyline for $\mathcal{R}$. Then there exists another point $q = (h, A[h], C[h])$ where $A[h] > A[i]$, $C[h] = C[i]$, $h \geq a$, and q also on the skyline for region $\mathcal{R}$. But then q must not be dominated by any of the points in $\mathcal{R}$, including the points $(k, A[k])$ for $h < k \leq i$. This implies that $prev(i) \geq h \geq b$, a contradiction.

C Proof of Lemma 4

The interval $(prev(i), i]$ is included in $\mathcal{I}(i)$ unless $(prev(i), i]$ was stabbed by some k for an ancestor v_k. Moreover, once it appears on some path from the root to the leaf on some vertex v_j it cannot appear anywhere else on that path. Suppose for the sake of contradiction that $(prev(i), i]$ appears at some other node v_h where v_h is not on this path. Without loss of generality, we assume $h < j$. Then there must exists some key value for a least common ancestor ℓ such that $h < \ell < j$. But then ℓ must stab $(prev(i), i]$ as well, a contradiction.

Bipartite and Series-Parallel Graphs Without Planar Lombardi Drawings

David Eppstein*

Abstract

We find a family of planar bipartite graphs all of whose Lombardi drawings (drawings with circular arcs for edges, meeting at equal angles at the vertices) are nonplanar. We also find families of embedded series-parallel graphs and apex-trees (graphs formed by adding one vertex to a tree) for which there is no planar Lombardi drawing consistent with the given embedding.

1 Introduction

Lombardi drawing is a style of graph drawing using curved edges. In this style, each edge must be drawn as a circular arc, and consecutive edges around each vertex must meet at equal angles. Many classes of graphs are known to have such drawings, including regular bipartite graphs and all 2-degenerate graphs (graphs that can be reduced to the empty graph by repeatedly removing vertices of degree at most two) [5]. This drawing style can significantly reduce the area usage of tree drawings [6], and display many of the symmetries of more general graphs [5].

When a given graph is planar, we would like to find a planar Lombardi drawing of it. When this is possible, the resulting drawings have simple edge shapes, no crossings, and optimal angular resolution, all of which are properties that lead to more readable drawings. It is known that all Halin graphs have planar Lombardi drawings [5], that all 3-regular planar graphs [7] and all 4-regular polyhedral graphs [8] have planar Lombardi drawings, and that all outerpaths have planar Lombardi drawings [4]. For some other classes of planar graphs, even when a Lombardi drawing exists, it might not be planar. Classes of planar graphs that are known to not always be drawable planarly in Lombardi style include the nested triangle graphs [5], 4-regular planar graphs [7], planar 3-trees [4], and the graphs of knot and link diagrams [8].

However, for several other important classes of planar graphs, the existence of a Lombardi drawing has remained open. These include the outerplanar graphs, the series-parallel graphs, and the planar bipartite graphs. All of these classes are 2-degenerate, so they always have Lombardi drawings, but these drawings might not be planar. In this paper we settle this open problem for two of these classes of graphs, the planar bipartite graphs and the (embedded) series-parallel graphs. We construct a family of planar bipartite graphs whose Lombardi drawings are all nonplanar. We also construct a family of series-parallel graphs with a given embedding such that no planar Lombardi drawing respects that embedding. Our construction for series-parallel graphs can be extended to maximal series-parallel graphs, to bipartite series-parallel graphs and to apex-trees, the graphs formed by adding a single vertex to a tree.

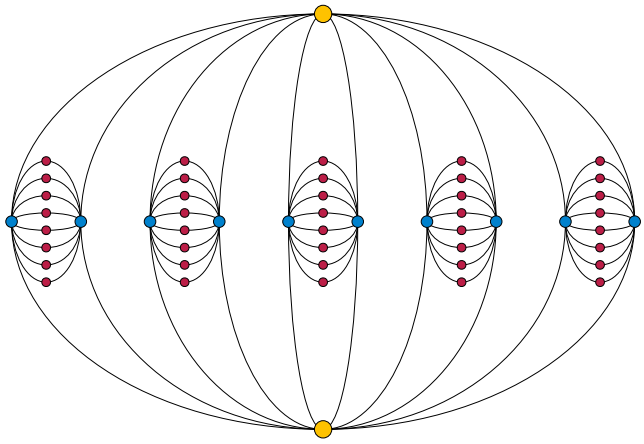


Figure 1: The bipartite graph $B(5)$ formed by our construction. Although drawn planarly with curved edges, this is not a Lombardi drawing: the edges are arcs of ellipses rather than of circles, and pairs of consecutive edges at the same vertex do not all have the same angles.

2 The graphs

We begin by describing the family of planar bipartite graphs $B(k)$ that we will prove (for sufficiently large k) do not have a planar Lombardi drawing. Each vertex in $B(k)$ has degree either 2 or $2k$. To construct $B(k)$, begin with a complete bipartite graph $K_{2,2k}$ and its unique planar embedding; in Figure 1, the two-vertex side of the bipartition of this graph is shown by the yellow vertices and the $2k$-vertex side is shown by the blue vertices. Each yellow vertex has exactly $2k$ blue neighbors.

Next, partition the blue vertices into k pairs of vertices, each sharing a face. For each pair of blue vertices in this partition, add another complete bipartite graph $K_{2,2k-2}$

*Department of Computer Science, University of California, Irvine. Research supported in part by NSF grants CCF-1618301 and CCF-1616248.

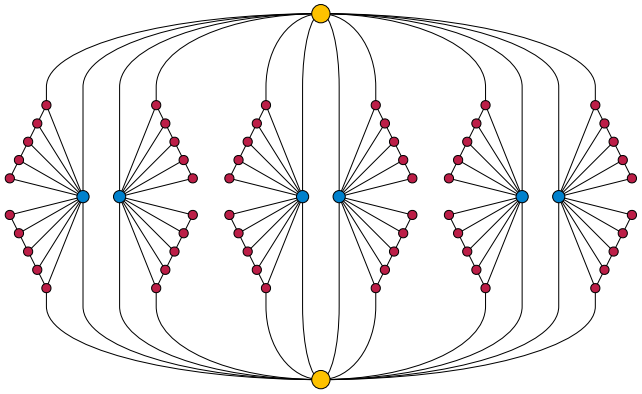


Figure 2: The embedded series-parallel graph $S(3)$ formed by our construction.

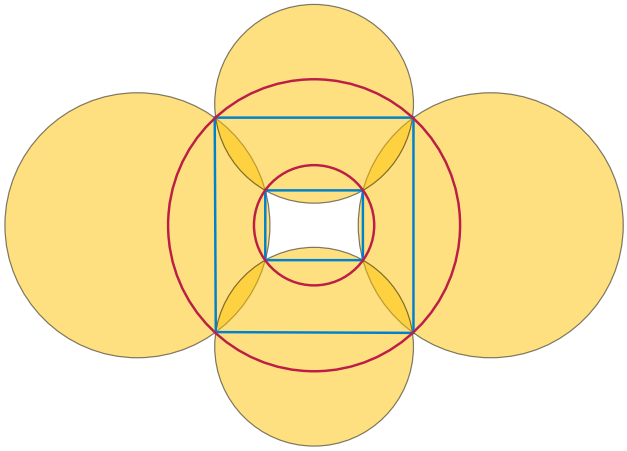


Figure 3: Illustration for Lemma 1: Four circles with centers on a rhombus, and with opposite pairs of circles having equal radii, define two rectangles of pairwise intersection points.

connecting these two blue vertices to $2k-2$ additional vertices (shown as red in the figure). After this addition, each blue vertex has exactly $2k$ neighbors, two of them yellow and the rest red. Each red vertex has exactly two neighbors. There are two yellow vertices, $2k$ blue vertices, and $k(2k-2)$ red vertices, for a total of $2k^2+2$ vertices in the overall graph.

Clearly, the graphs $B(k)$ are all planar, because they are formed by attaching together planar subgraphs (complete bipartite graphs where one side has two vertices) on pairs of vertices that are cofacial in both subgraphs. They are bipartite, with the yellow and red vertices on one side of their bipartition and the blue vertices on the other side. Although they are not 3-vertex-connected, all of their planar embeddings are isomorphic.

Analogously, we define a family of embedded series-parallel graphs $S(k)$. Again, each such graph will have two yellow vertices and $2k$ blue vertices, connected in the pattern of a complete bipartite graph $K_{2,2k}$. For each yellow–blue edge e of this graph, we add a path of $2k-1$ red vertices. We connect every vertex in this path to the blue endpoint of e, and we connect one endpoint of the path to the yellow endpoint of e. We fix an embedding of $S(k)$ in which every yellow–blue quadrilateral contains either zero or four red paths (Figure 2). The resulting graph has two yellow vertices, $2k$ blue vertices, and $4k(2k-1)$ red vertices, for a total of $8k^2-2k+2$ vertices. The yellow and blue vertices have degree $4k$, while the red vertices have degrees two or three.

We claim that, for sufficiently large values of k, the graphs $G(k)$ and $S(k)$ do not have planar Lombardi drawings. Therefore, neither every planar bipartite graph nor every series-parallel graph has a planar Lombardi drawing. In the remainder of this paper we prove this claim.

3 Equiangular arc-quadrilaterals

The key feature of both of our graph constructions $B(k)$ and $S(k)$ is the existence of many yellow–blue quadrilateral faces in which all vertices have equal and high degree (this degree is $d=2k$ in $B(k)$ and $d=4k$ in $S(k)$). If such a graph is to have a Lombardi drawing, each of these faces must necessarily be drawn as a quadrilateral with circular-arc sides and with the same interior angle $2\pi/d$ at all four of its vertices. Equiangular arc-quadrilaterals have been investigated before from the point of view of conformal mapping [2]; in this section we investigate some of their additional properties.

Our main tool is the following lemma:

Lemma 1 *Let abcd be a non-self-crossing quadrilateral in the plane with circular-arc sides and equal interior angles. Then the four points abcd lie on a circle and the quadrilateral abcd either lies entirely inside or entirely outside the circle.*

Proof. We abbreviate the conclusion of the lemma by saying that *abcd* is cyclic. The properties of being an equiangular non-self-crossing circular-arc quadrilateral and of being cyclic are both invariant under Möbius transformations, which preserve both circularity and the crossing angles of curves. Therefore, if we can find a Möbius transformation of a given equiangular circular-arc quadrilateral such that the transformed quadrilateral is cyclic, the original quadrilateral will also be cyclic, as the lemma states it to be.

Start by finding a Möbius transformations which makes two opposite arcs *ab* and *cd* come from circles with the same radius as each other. Because of the equality of crossing angles, and by symmetry, both of

the other two circular arcs bc and ad must come from circles whose centers lie on the perpendicular bisector of ab and cd. There remains a one–dimensional family of Möbius transformations that preserve the position of the circles containing the transformed copies of arcs ab and cd but that move the other two circles along the bisector of these two fixed circles. We can use this remaining degree of freedom to move the other two circles so that their centers are equidistant from the midpoint of the centers of the two fixed circles.

After this transformation, it follows from the equality of crossing angles that the circles containing the transformed copies of arcs bc and ad have the same radii as each other, and the four circles have been transformed into a position centered at the vertices of a rhombus with opposite pairs having the same radius as each other. By symmetry, the transformed copies of vertices $abcd$ must lie on one of the two rectangles defined by the crossing points of these four transformed circles. (Figure 3).

If the interior angle of $abcd$ is less than π, then the transformed copy of $abcd$ must lie within the circle that circumscribes the inner rectangle, forming the boundary of a hole in the union of the four transformed disks. If the interior angle is greater than π, it must lie outside the outer circle, forming the outer boundary of the union of the four disks. In either case, the transformed copy of $abcd$ is cyclic, so $abcd$ itself must be cyclic. □

A special case of this lemma for right-angled arc-quadrilaterals was used previously by the author to prove that some 4-regular planar graphs have no planar Lombardi drawing [7]. Another special case, for arc-quadrilaterals in which all interior angles are zero, has been used previously in mesh generation [1].

Definition 2 *We define the* tilt *of an equiangular circular-arc quadrilateral to be the maximum interior angle of any of the four circular-arc bigons between the quadrilateral and its enclosing circle.*

Each of the four bigons has equal angles at its two vertices. At each of the four vertices, the two bigon angles and the interior angle of the quadrilateral add to π. It follows that opposite bigons have the same angles as each other, and each vertex of the quadrilateral is incident to a bigon with vertices of the tilt angle.

4 Bipolar coordinates

To describe a second parameter of equiangular circular-arc quadrilaterals, it is convenient to introduce the *bipolar coordinate system*, defined from a pair of points s and t, the *foci* of the coordinate system. These coordinates are conventionally denoted σ and τ. The σ-coordinate σ_p of a point p is the (oriented) angle spt, whose level sets are the blue circular arcs through the two foci in Figure 4. The τ-coordinate τ_p of p is the logarithm of the ratio of the two distances from p to the two foci, whose level sets are the red circles separating the two foci in Figure 4. This coordinate system has the convenient property that any (orientation-preserving) Möbius transformation that preserves the location of the two foci acts by translation on the coordinates.[1]

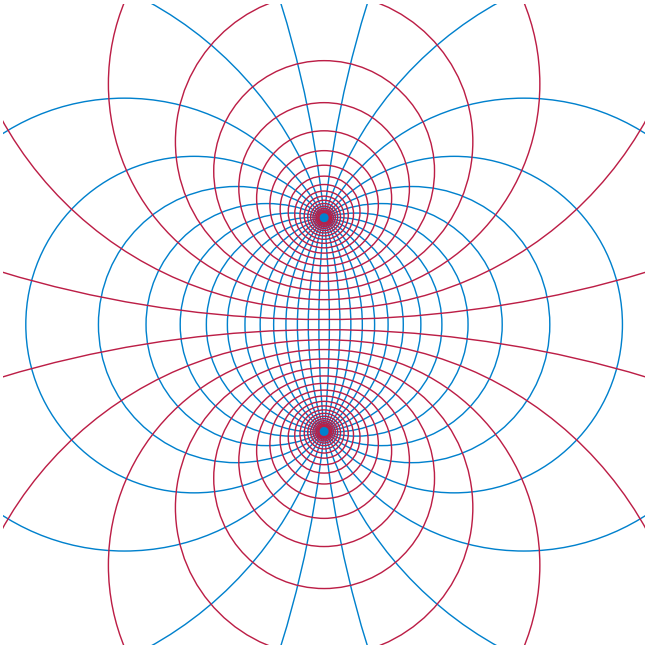


Figure 4: Curves of constant and evenly-spaced coordinate values for bipolar coordinates, forming two orthogonal pencils of circles.

Lemma 3 *Any Möbius transformation that preserves the location of the two foci acts on the bipolar coordinates of any point by adding a fixed value to its σ-coordinate (modulo 2π) and adding another fixed value to its τ-coordinate, with the added values depending on the transformation but not on the point.*

Proof. All Möbius transformations preserves circles, incidences between points and curves, and angles between pairs of incident curves. Therefore, any focus-preserving Möbius transformation takes circles through the two foci (the level sets for σ-coordinates) to other circles through the two foci, and it takes the perpendicular family of circles (the level sets for τ-coordinates) to other circles in the same family. Therefore it acts separately on the σ- and τ-coordinates. The additivity of its action on the σ-coordinates follows from the preservation of angles between pairs of circles through the two foci.

To show that the transformation acts additively on τ-coordinate (the logarithm of the ratio of distances of

[1] This property can be seen as a reflection of the fact that the bipolar coordinate system comes from a conformal mapping of a rectangular grid; see, e.g., [3].

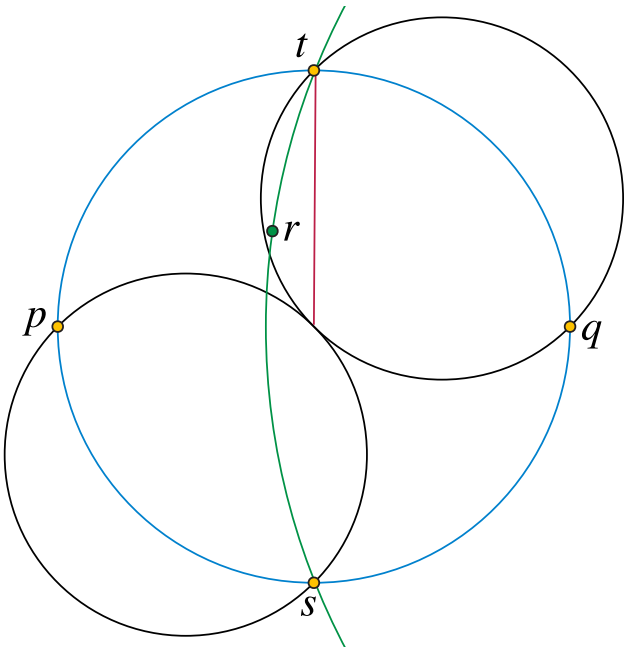


Figure 5: Illustration for Lemma 5: If quadrilateral *sptq* has high tilt, and r lies between the quadrilateral and its enclosing circle, to the left of the red arc, then r must have a higher τ-coordinate value than p.

a point from the two foci), we can assume without loss of generality (by scaling, translating, and rotating the plane if necessary) that the two foci are at the two points $q = \pm 1$ of the complex plane. Consider the general form $q \mapsto (aq + b)/(cq + d)$ of a Möbius transformation as a fractional linear transformation of the complex plane. For a transformation to fix $q = 1$ we need $a + b = c + d$ and for it to fix $q = -1$ we need $b - a = -(d - c)$. Solving these two equations in four unknowns gives $a = d$ and $b = c$. Therefore, the transformations fixing the foci take the special form $q \mapsto (aq + b)/(bq + a)$.

For a transformation of this form, and for any point x on the interval $[-1, 1]$ of real numbers with distance ratio $(1 + x)/(1 - x)$, the image of x has distance ratio

$$\frac{1 + (ax + b)(bx + a)}{1 - (ax + b)(bx + a)} = \frac{a + b}{a - b} \cdot \frac{1 + x}{1 - x},$$

multiplying the original distance ratio of x by a value that depends only on the transformation. Because the transformation acts separately on σ- and τ-coordinates, we obtain the same multiplicative action on distance ratios for any other point on the complex plane with the same τ-coordinate as x. This multiplicative action on distance ratios translates into an additive action on their logarithms. □

Another advantage of bipolar coordinates is that they provide a way of comparing angles at the two foci, that

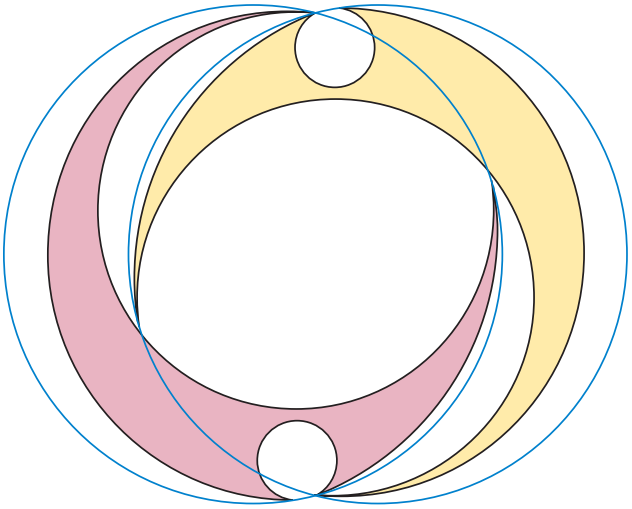


Figure 6: Two arc-quadrilaterals with sharp angles between them at their shared vertices reach into each other's pockets to touch their circumscribing circles. The two smaller pockets on the outer arcs of the circles have significantly different τ-coordinates from each other in bipolar coordinates with the shared vertices as foci.

will be convenient for relating the tilts of different quadrilaterals to each other:

Observation 4 *Let sptq be an equiangular arc-quadrilateral with interior angle θ and tilt φ. Then, in the bipolar coordinate system with foci s and t, the angle (difference between the σ-coordinates) of arc tp in the limit as it approaches t and of arc sq as it approaches s is exactly $2\theta + 2\varphi - \pi$.*

We can also use bipolar coordinates to show that heavily tilted quadrilaterals lead to an increase in τ-coordinate:

Lemma 5 *Let sptq be an equilateral arc-quadrilateral with tilt at least $3\pi/4$, such that the large angle between vertex s and the circle C containing the quadrilateral is on the clockwise side of s (the side closest to p). Let r be a point in the bigon between arc tq and C, such that circular arc srt makes an angle of at most $\pi/2$ with circular arc spt. Then, in the bipolar coordinate system for foci s and t, $\tau_r > \tau_p$.*

Proof. Because of the Möbius invariance of coordinate differences in the bipolar coordinate system, we can without loss of generality perform a Möbius transformation so that s, p, and t are the bottom, left, and topmost points of C, as shown in Figure 5. After this transformation, points above the horizontal line through p will have higher τ-coordinate than o, and points below the horizontal line through o will have lower τ-coordinate.

As the figure shows, an arc with tilt exactly $3\pi/4$ through p and s passes through the center of circle C, causing the region in which r may lie to be bounded by a vertical line segment (red) from the circle's center to t. All points within this region have higher τ-coordinate than p. For tilt values greater than $3\pi/4$, the arc from p to s with that tilt extends even farther beyond the center of C, so (although arc tq may also extend farther to the left) the region in which r may lie remains bounded within the upper left quarter of C, within which all τ-coordinates are greater than that of p. □

5 Nonplanarity

We are now ready for our main theorems.

Theorem 6 *For $k > 8$, the bipartite graph $B(k)$ does not have a planar Lombardi drawing.*

Proof. Let s and t be the two yellow vertices of the graph $B(k)$. We will consider a bipolar coordinate system with foci s and t. Note that graph $B(k)$ contains k quadrilateral faces sp_itq_i, where p_i and q_i are blue vertices. Because all four vertices of these quadrilaterals have degree $2k$, these quadrilaterals must be drawn (if a planar Lombardi drawing is to exist) as equiangular arc-quadrilaterals with interior angle π/k.

Consider the two consecutive quadrilaterals sp_itq_i and $sp_{i+1}tq_{i+1}$ whose enclosing circles C_i and C_{i+1} meet each other at the sharpest angle of any two consecutive enclosing circles. The sum of the angles between the k consecutive circles is 2π so this minimum angle is at most $2\pi/k$. In order for point q_i to lie on circle C_i, some arc of circle C_i must lie on the same side as q_i of quadrilateral $sp_{i+1}tq_{i+1}$. This arc must stay outside of quadrilateral $sp_{i+1}tq_{i+1}$ from its crossing point with the quadrilateral until terminating at either s or t; by symmetry, we can assume without loss of generality that it terminates at the lower vertex s, as shown in Figure 6. Then, near s, quadrilateral $sp_{i+1}tq_{i+1}$ lies between circles C_i and C_{i+1}, so it must have tilt at least $\pi(1 - \frac{2}{k})$. By Observation 4 and the equal spacing of angles around s and t, all quadrilaterals sp_itq_i must have the same tilt.

Because $k > 8$, this tilt is $\geq 3\pi/4$, so each quadrilateral sp_itp_i meets the precondition of having high tilt of Lemma 5. For any quadrilateral sp_itq_i, the point p_{i+1} of the next quadrilateral is connected to s by an arc of quadrilateral $sp_{i+1}tq_{i+1}$ that lies entirely within C_{i+1} and makes an angle of π/k to arc sq_i, so the arc of C_{i+1} containing p_{i+1} makes an angle of at most $3\pi/k$ to the arc of C_i containing p_i. Thus, point p_{i+1} meets the other precondition of Lemma 5 for the position of the point r with respect to the quadrilateral. By this lemma, each point p_{i+1} has a greater τ-coordinate than p_i. But it is impossible for this monotonic increase in τ-coordinates to continue all the way around the circle of quadrilaterals surrounding the two foci and back to the starting point. This impossibility shows that the drawing cannot exist. □

Theorem 7 *For $k > 8$ the series-parallel graph $S(k)$, embedded as shown in Figure 2, does not have a planar Lombardi drawing.*

Proof. As with $B(k)$, this graph contains k quadrilateral faces, sharing the same two opposite yellow vertices, in which all vertices have equal degree ($4k$ in $S(k)$ instead of $2k$ in $B(k)$). The proof of Theorem 6 used only this property of $B(k)$, and not the precise value of the interior angle of these quadrilaterals, so it applies equally well to $S(k)$. □

We remark that the construction of $S(k)$ can be adjusted in several different ways to obtain more constrained families of embedded series-parallel and related graphs that, again, have no planar Lombardi drawing:

- If we add an edge between the two yellow vertices, and adjust the lengths of the red chains to keep the yellow and blue degrees equal, we obtain a family of embedded maximal series-parallel graphs (that is, embedded 2-trees) with no planar Lombardi drawing.
- If we subdivide the yellow–red and red–red edges of $S(k)$, we obtain a family of embedded bipartite series-parallel graphs with no planar Lombardi drawing.
- If we replace the red chains of $S(k)$ by an appropriate number of degree-one red vertices, connected to the blue vertices, we obtain a family of embedded apex-trees (graphs formed by adding a single vertex to a tree) with no planar Lombardi drawing. The apex vertex (the vertex whose removal produces a tree) can be chosen to be either of the two yellow vertices.

We omit the details.

6 Conclusions

We have shown that bipartite planar graphs, and series-parallel graphs with a fixed planar embedding, do not always have planar Lombardi drawings, even though their low degeneracy implies that they always have (nonplanar) Lombardi drawings. In the question of which important subfamilies of planar graphs have planar Lombardi drawings, several important cases remain unsolved. These include the outerplanar graphs, both with and without assuming an outerplanar embedding, the cactus graphs, and the series-parallel graphs without a fixed choice of embedding. We leave these as open for future research.

References

[1] Marshall Bern, Scott Mitchell, and Jim Ruppert. Linear-size nonobtuse triangulation of polygons. *Discrete & Computational Geometry*, 14(4):411–428, 1995. doi:10.1007/BF02570715.

[2] Philip R. Brown and R. Michael Porter. Conformal mapping of circular quadrilaterals and Weierstrass elliptic functions. *Computational Methods and Function Theory*, 11(2):463–486, 2011. doi:10.1007/BF03321872.

[3] Jeng-Tzong Chen, Ming-Hong Tsai, and Chein-Shan Liu. Conformal mapping and bipolar coordinate for eccentric Laplace problems. *Computer Applications in Engineering Education*, 17(3):314–322, 2009. doi:10.1002/cae.20208.

[4] Christian A. Duncan, David Eppstein, Michael T. Goodrich, Stephen G. Kobourov, and Maarten Löffler. Planar and poly-arc Lombardi drawings. *Journal of Computational Geometry*, 9(1):328–355, 2018. Special issue on Graph Drawing Beyond Planarity. arXiv:1109.0345, doi:10.7155/jgaa.00457.

[5] Christian A. Duncan, David Eppstein, Michael T. Goodrich, Stephen G. Kobourov, and Martin Nöllenburg. Lombardi drawings of graphs. *J. Graph Algorithms & Applications*, 16(1):85–108, 2012. Special issue for GD 2010. arXiv:1009.0579, doi:10.7155/jgaa.00251.

[6] 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. arXiv:1009.0581, doi:10.1007/s00454-012-9472-y.

[7] David Eppstein. A Möbius-invariant power diagram and its applications to soap bubbles and planar Lombardi drawing. *Discrete & Computational Geometry*, 52(3):515–550, 2014. Special issue for SoCG 2013. doi:10.1007/s00454-014-9627-0.

[8] Philipp Kindermann, Stephen G. Kobourov, Maarten Löffler, Martin Nöllenburg, André Schulz, and Birgit Vogtenhuber. Lombardi drawings of knots and links. In Fabrizio Frati and Kwan-Liu Ma, editors, *Graph Drawing and Network Visualization - 25th International Symposium, GD 2017, Boston, MA, USA, September 25-27, 2017, Revised Selected Papers*, volume 10692 of *Lecture Notes in Computer Science*, pages 113–126. Springer, 2017. arXiv:1708.09819, doi:10.1007/978-3-319-73915-1_10.

Three-Coloring Three-Dimensional Uniform Hypergraphs

Ahmad Biniaz* Prosenjit Bose† Jean Cardinal‡ Michael S. Payne§

Abstract

We study the chromatic number of hypergraphs whose vertex-hyperedge incidence poset has dimension at most three. Schnyder (1989) showed that graphs with this property are planar and thus four-colorable. Results of Keszegh and Pálvölgyi (2015) imply that k-uniform hypergraphs with dimension at most three are two-colorable for $k \geq 9$. In this paper we show that k-uniform hypergraphs with dimension at most three are three-colorable for $k \geq 6$. This implies three colorability of k-uniform triangle Delaunay hypergraphs and k-uniform hypergraphs induced by points and octants in 3-space. We also observe that the chromatic number of k-uniform hypergraphs with dimension $d \geq 4$ is not bounded by any function of k and d.

1 Introduction

A *hypergraph* G consists of a set of *vertices* and a collection of non-empty subsets of vertices called *hyperedges*. The *incidence poset* of G is the partially ordered set (poset) describing the vertex-hyperedge containment relationship. The order dimension of a poset $\mathcal{P}$ is defined as the minimum size of a set of total orders on the elements of $\mathcal{P}$ whose intersection is $\mathcal{P}$. The *dimension* of G is the order dimension of its incidence poset. A *k-uniform* hypergraph (or *k-graph*) is a hypergraph in which all hyperedges have cardinality k. A (simple) *graph* is one for which k is 2. A *c-coloring* of G is to color each vertex by one of the colors $\{1, \ldots, c\}$ such that no edge of G has all vertices of the same color. A hypergraph is *c-colorable* if it admits a c-coloring.

In 1989, Schnyder showed that a graph has dimension at most three if and only if it is planar [13]. Therefore, all such graphs are 4-colorable by the Four Colour Theorem. We study the problem of coloring k-graphs of dimension at most three. We will refer to hypergraphs of dimension at most three as *three-dimensional hypergraphs*. It follows from the seminal work of Keszegh and Pálvölgyi [10] that three-dimensional k-graphs are 2-colorable for $k \geq 9$. In this note we adapt their approach and show the following result.

*University of Waterloo, ahmad.biniaz@gmail.com
†Carleton University, jit@scs.carleton.ca
‡Université Libre de Bruxelles (ULB), jcardin@ulb.ac.be
§La Trobe University, m.payne@latrobe.edu.au

Theorem 1 *Every three-dimensional k-uniform hypergraph is 3-colorable, for $k \geq 6$.*

2 Background

The dimension of a hypergraph can be determined by representing the incidence poset as the intersection of a number of total orders on vertices. The following is a well-known characterization of hypergraphs of dimension d that we will rely upon often [13].

Proposition 2 (Schnyder 1989) *A hypergraph H has dimension at most d if and only if there exist d total orders $<_1, \ldots, <_d$ on the vertices of H such that*

- *the intersection of all the orders is empty, and*
- *for each hyperedge e of H and each vertex $z \notin e$ there exists i such that $x <_i z$ for every $x \in e$.*

This characterization implies that any hyperedge e is uniquely determined by its maximum vertices in the d total orders. Every vertex not in e must be above at least one of these maxima. See Figure 6 for an illustration of a hyperedge.

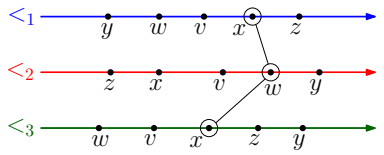


Figure 1: Illustration of a hyperedge $e = \{v, w, x\}$.

Our work is inspired by the work of Keszegh and Pálvölgyi [10] on coloring octant k-graphs (a subclass of k-graphs). Given any finite set P of points in $\mathbb{R}^3$, take as hyperedges every set of k points that is the intersection of P with some axis-parallel octant (which is an open set of the form $(-\infty, x) \times (-\infty, y) \times (-\infty, z)$ with apex point (x, y, z)). They showed that any octant k-graph is 2-colorable for $k \geq 9$, and there are octant 4-graphs that are not 2-colorable. It is implied by Proposition 2 that three-dimensional k-graphs are a subclass of octant k-graphs—just use the three total orders to give coordinates to the vertices. (Octant hypergraphs are more general because the coordinates need not satisfy the first property in Proposition 2.)

Another relevant class of geometric hypergraphs are triangle Delaunay k-graphs: Given a finite set P of

points in general position in $\mathbb{R}^2$ and a triangle T, a set of k points form a hyperedge if there exists a homothet[1] of T containing just those k points. The classical Delaunay graph of a point set has a similar construction but with respect to a circle instead of a triangle.

With Proposition 2, it can readily be seen that triangle Delaunay hypergraphs have dimension at most three. The three necessary total orders can be obtained by sweeping P with three lines parallel to the three sides of T, as in Figure 2. Combining this with Theorem 1 we get the following corollary.

Corollary 3 *Every k-uniform triangle Delaunay hypergraph is 3-colorable, for $k \geq 6$.*

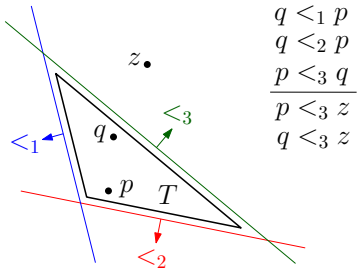


Figure 2: Obtaining three total orders that satisfy the conditions of Proposition 2. The set $\{p, q\}$ is an edge of triangle Delaunay 2-graph.

Returning to graphs, another (perhaps lesser known) result of Schnyder [13, Corollary 5.4] implies that every three-dimensional graph (and thus every planar graph) can be represented as a subgraph of a triangle Delaunay graph. Schnyder called these representations 'barycentric embeddings'. One might wonder whether every three-dimensional k-graph is also a subgraph of a triangle Delaunay k-graph. We note that there exist three-dimensional 10-graphs that are not realizable as a triangle Delaunay 10-graph (Stefan Felsner, private communication). To sum up, we can see that the class of triangle Delaunay hypergraphs is a proper subset of the class of three-dimensional hypergraphs, which is in turn a proper subset of the class of octant hypergraphs.

There exist a large number of fascinating coloring problems for geometric hypergraphs that are closely related to the problem studied here; e.g. [1, 2, 3, 4, 5, 6, 8, 9, 11, 12]. In particular, the question of whether k-uniform Delaunay hypergraphs (induced by points and circles in the plane) are 3-colorable for some k remains open [1], while the analogous questions for homothets of a convex n-gon have relatively loose bounds on k [11]. On the other hand, it is known that there is no finite k such that k-graphs induced by axis parallel rectangles on points in the plane are 2-colorable [6]. For further related problems and results see the discussion in [11].

[1]Homothets of T include translations and scalings but not rotations or reflections.

3 Further results

The result about axis parallel rectangles just mentioned, which is due to Chen, Pach, Szegedy, and Tardos [6], implies that there can be no analogue of Theorem 1 in higher dimensions (see Corollary 5). The statement for $k = 2$ (graphs) was first proved by Ossona de Mendez and Rosenstiehl [12], then rediscovered by Trotter and Wang [14].

Theorem 4 (Chen et al. [6]) *For any positive integers c and k, there is a finite point set in the plane with the property that no matter how we color its elements with c colors, there always exists an axis-parallel rectangle containing at least k points, all of which have the same color.*

Corollary 5 *For any triple of integers $c \geq 1$, $k \geq 1$, and $d \geq 4$, there exists a d-dimensional k-uniform hypergraph that is not c-colorable.*

Proof. Let H_1 be a hypergraph whose vertex set is a finite point set P in the plane that satisfies the conditions of Theorem 4, and whose edge set contains all k-subsets of points that can be obtained by intersecting P with an axis-aligned rectangle. By considering the four total orders obtained by sweeping P horizontally and vertically in both directions, and using Proposition 2, we observe that H_1 is a k-uniform hypergraph with dimension at most 4. Theorem 4 implies that H_1 is not c-colorable.

Let H_2 be any d-dimensional k-uniform hypergraph. Then the disjoint union of H_1 and H_2 is a k-uniform hypergraph that is d-dimensional due to the dimension of H_2, and not c-colorable because H_1 requires more than c colors. □

Finally we note an extension of Theorem 1. Despite the important role of graph planarity in the proof of Theorem 1, the analogous result for octant k-graphs follows as a corollary.

Corollary 6 *Every k-uniform octant hypergraph is 3-colorable, for $k \geq 6$.*

Proof. An octant k-graph has vertex set $P \subset \mathbb{R}^3$. We consider the three coordinate directions as three total orders on P. Unlike the case of three-dimensional k-graphs, the intersection of the three orders may not be empty. Consider the poset B on P with the order relation being the intersection of these three orders. Note that if u dominates v in this partial order then every octant containing u also contains v.

Start with the subset $S \subset P$ consisting of all the minimal elements in B. As S is an antichain in B, it

induces a three-dimensional k-graph, and so we may 3-color it by Theorem 1.

Now add in a point x that is minimal in $B \setminus S$, and notice that x dominates some point(s) of S. This means that some hyperedges disappear, and some others are created with x and $k-1$ points of S. We need only ensure that these new hyperedges are properly colored, and this can be done by giving x any of the 3 colors that is distinct from the color of a point dominated by x.

By iteratively adding minimal elements from the remaining points, we can build up a 3-coloring for the whole set P. □

4 Proof of Theorem 1

Let H denote the three-dimensional k-graph that we want to color. Following Keszegh and Pálvölgyi, the proof strategy involves constructing a graph F that has an edge in every hyperedge of H. Thus a proper coloring of F is a proper coloring of H. In the proof of Keszegh and Pálvölgyi, F is a forest, and therefore 2-colorable. In our proof, F is a triangle-free three-dimensional graph, and thus planar by Schnyder's theorem and hence 3-colorable by Grötzsch's theorem which says that triangle-free planar graphs are 3-colorable [7].

Let V be the vertex set of H, and let $<_1, <_2, <_3$ be the three total orders on V with empty intersection. For every ordered triple (x, y, z) of (not necessarily distinct) elements of V, we define the *combinatorial triangle* Δxyz as the subset of V determined by three maxima x, y, z:

$$\Delta xyz = \{v \in V \mid v \leq_1 x \ \wedge \ v \leq_2 y \ \wedge \ v \leq_3 z\}.$$

Triangles containing k elements are precisely the hyperedges of H. If two elements of V are reversed by $<_1$ and $<_2$ we say they are *incomparable*, otherwise they are *comparable*. If for two comparable elements x and y we have $x <_1 y$ and $x <_2 y$, then we say y *dominates* x. See Figure 3 for an illustration. Note that if y dominates x then we must have $y <_3 x$ because the intersection of $<_1, <_2, <_3$ is empty.

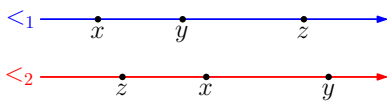


Figure 3: The elements x and z are incomparable, while x and y are comparable and y dominates x.

Without loss of generality we assume that H is edge maximal, that is, H contains all hyperedges (with k vertices) that satisfy the second property in Proposition 2. Let G be the edge-maximal planar graph obtained by the same three orders. We describe an iterative algorithm that processes V in the order of $<_3$ and creates a planar triangle-free subgraph F of G such that every hyperedge of H contains an edge of F. By slightly abusing notation, in the rest of description we refer to F as an edge set (the graph F is induced by this edge set).

We maintain a sequence Y of vertices and a set F of edges that satisfy the following invariants after each iteration of the algorithm:

- Elements of Y are pairwise incomparable and no element in Y dominates a processed element. (By the definition of incomparability, Y is ordered forwards by $<_1$ and backwards by $<_2$.)
- The set F is a subset of G, has no 3-cycles, has no edge between two vertices of Y, and has an edge in every hyperedge formed by processed vertices.

The sequence Y plays the role of "staircase" in [10] that separates the processed vertices from unprocessed vertices. Initially, F is empty and Y contains the least element in $<_3$. The algorithm processes the next vertex m in $<_3$ as follows:

(1) While there exists $v \in Y$ that dominates m, then add vm to F and remove v from Y.

(2) Add m to Y.

(3) While there exist three consecutive vertices $u, v, w \in Y$ such that $u <_2 v <_2 w$ and Δuwm does not contain any vertex outside Y, then add uv and vw to F and remove v from Y.

Since we apply step (3) greedily, any triple considered in this step contains m, that is $m \in \{u, v, w\}$. Moreover, since $u <_2 v <_2 w$ and the elements of Y are pairwise incomparable, we have $w <_1 v <_1 u$. This in turn implies that Δuwm contains u, v, and w.

It remains to show that each of the claimed properties for Y and F holds at the end of every iteration. In the proof of these properties we use the fact that m is the maximum element in $<_3$ that is processed so far, without further mentioning. Let X denote the set of processed elements that are not in Y.

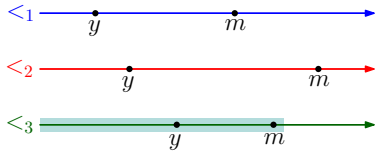


Figure 4: The processed elements (in the order $<_3$) are shaded; m is processed in the current iteration.

– *Elements of Y are pairwise incomparable:* Before we add a new vertex m to Y in step (2), we remove all vertices that dominate m in step (1). The vertex m does not dominate any vertex $y \in Y$ because otherwise we would have $y <_1 m$, $y <_2 m$, and $y <_3 m$ which

contradicts the intersection of total orders being empty; see Figure 4 for an illustration. Therefore, the elements of Y are pairwise incomparable after each iteration.

– *Elements of Y do not dominate elements of X:* We need to ensure this only when we add the current element m to Y. If m dominates an element of X, then similar to the previous claim (as in Figure 4) we get a contradiction to the emptiness of the intersection of total orders.

– *F has no edge between two vertices of Y:* In step (1) after adding the edge vm, we remove v from Y. In step (3) after adding the edges uv and vw, we remove v from Y. Therefore, this claim follows.

– *F is a subset of G:* Consider an edge vm added to F in step (1). We show that the triangle Δvvm contains only v and m; this implies that vm is an edge of G. This triangle contains m because $m <_1 v$ and $m <_2 v$ (as v dominates m) and contains v because $v <_3 m$ (as m is the largest element of $<_3$ processed so far). Now we show, by contradiction, that Δvvm does not contain any other point. Recall that before adding vm, the vertex v belongs to Y. If Δvvm contains another element $y \in Y$ (as in Figure 5) then v dominates y; this contradicts the fact that elements of Y are pairwise incomparable. If Δvvm contains an element $x \in X$ (as in Figure 5) then v dominates x; this contradicts the fact that elements of Y do not dominate elements of X.

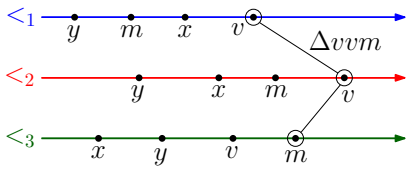


Figure 5: The triangle Δvvm contains m, v, x, and y.

Now consider edges uv and vw added to F in step (3), and recall that due to greedily application of this step we have $m \in \{u, v, w\}$. Our choices of u, v, w (as three consecutive elements of Y) and Δuwm (as having no element of X) ensures that Δuwm contains only u, v, w. In this setting the triangle $\Delta uv*$ contains only u, v, and the triangle $\Delta vw*$ contains only v, w ($*$ represents the maximum of the first two elements with respect to $<_3$). Thus, uv and vw are edges of G. See Figure 6 for an illustration.

– *F has no 3-cycle:* Since there are no edges between elements of Y, the edges added in step (1)—between m and elements of Y—do not create any 3-cycle. Consider edges uv and vw added in step (3). Since there was no edge between u and w which belong to Y, the three vertices u, v, w cannot form a 3-cycle. If uv creates a 3-cycle then u and v were joined by a path of length two

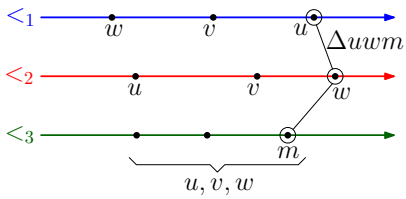


Figure 6: The triangle Δuwm contains only u, v, w.

through x say which now belongs to X. For x to have two neighbors in Y, the edges ux and xv must come from a prior application of step (3), and thus we must have $u <_2 x <_2 v$ and $v <_1 x <_1 u$, as in Figure 7. In this setting the triangle Δuwm contains x, which contradicts the current application of step (3). Thus uv does not create a 3-cycle. A similar argument applies for vw.

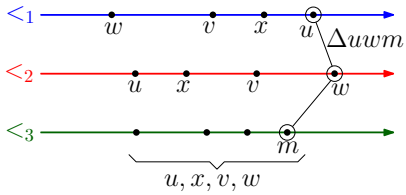


Figure 7: The triangle Δuwm contains u, v, w and x.

– *Every hyperedge formed by processed vertices contains an edge in F:* It suffices to show this only for hyperedges containing m. Consider any such hyperedge h, and recall that $|h| = k \geq 6$. Consider the state of Y and F at the end of current iteration, and set $i := |h \cap X|$. Depending on i, we consider three cases.

If $i = 0$, then all elements of h belong to Y and an edge would be added inside h in step (3).

Now suppose that $i = 1$, and let v be the only element in $h \cap X$ (here is the place where we use $k \geq 6$). Then $|h \cap Y| \geq 5$. Let $a <_1 b <_1 c <_1 d <_1 e$ be five consecutive elements of $h \cap Y$, and note that $e <_2 d <_2 c <_2 b <_2 a$. Thus $\Delta eam \subseteq h$, as in Figure 8. Let m_1 be the greatest of a, b, c and let m_2 be the greatest of c, d, e both with respect to $<_3$. In this setting either Δecm_2 or Δcam_1 does not contain v because otherwise $v <_1 c$, $v <_2 c$, and $v <_3 c$ (as in Figure 8) which contradicts the emptiness of the intersection of total orders. Therefore, in step (3) we get edges ed, dc or cb, ba, and thus h contains an edge of F.

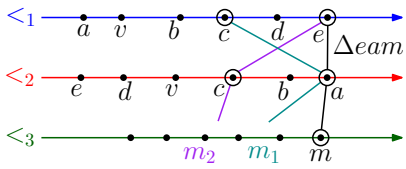


Figure 8: The triangle Δeam is a subset of h.

Now suppose that $i \geq 2$. Then $h \cap X$ contains two elements that are either comparable or incomparable.

First suppose that $h \cap X$ contains two comparable elements v and t. We may assume that t dominates v (this implies that $t <_3 v$). Moreover, we may assume that v was added to X in step (3) because otherwise by step (1) we get an edge in Δttm which is a subset of h. Since step (3) has been applied, as its prerequisites v has two incomparable neighbors u, w such that $u <_2 v <_2 w$ and Δuwm_1 does not contain t, where $m_1 \leq_3 m$ is the maximum of u, v, w with respect to $<_3$; see Figure 9. In step (3) the edges uv and vm were added to F. Recall that $t <_3 v$, and thus $t <_3 m_1$. In this setting either $u, v <_1 t$ (as in Figure 9) or $v, w <_2 t$. In the first case u and v are contained in Δttm_1 while in the second case v and w are contained in Δttm_1. Since Δttm_1 is a subset of Δttm which is in turn a subset of h, we get an edge of F in h.

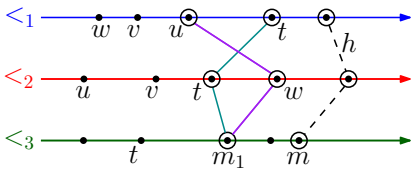


Figure 9: The triangle Δuwm_1 does not contain t.

Now suppose that $h \cap X$ contains two incomparable elements v and t. We may assume that both were added to X in step (3) because otherwise by step (1) we get an edge in Δvvm or in Δttm which are subsets of h. Without loss of generality assume that $t <_2 v$ and that v was added to X after t. As prerequisite of step (3) the vertex v has two incomparable neighbors $u <_2 v <_2 w$ in the triangle Δuwm_1 that does not contain t, where $m_1 \leq_3 m$ is the maximum of u, v, w with respect to $<_3$. In step (3) the edges uv and vm were added to F. We show (by contradiction) that Δtvm contains u and v, or v and w; this would imply our claim because Δtvm is a subset of h.

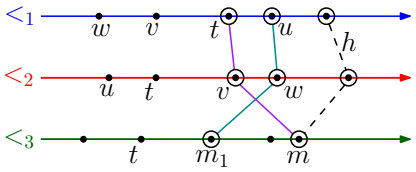


Figure 10: The triangle Δuwm_1 does not contain t.

Observe that Δtvm contains v. For the sake of contradiction assume that Δtvm does not contain any of u and w, and thus $t <_1 u$ and $v <_2 w$, as in Figure 10. Recall that v was added to X when we were processing the greatest element of $\{u, v, w\}$ in $<_3$, which is m_1. At that time, t was already in X which means that t was processed before m_1, i.e., $t <_3 m_1$, as in Figure 10. In this setting Δuwm_1 contains t, which contradicts the application of step (3) on u, v, w. This completes the proof.

Acknowledgement. This work initiated at the *Fifth Annual Workshop on Geometry and Graphs*, March 5-10, 2017, at the Bellairs Research Institute of McGill University, Barbados. We are grateful to the organizers and to the participants for a wonderful workshop.

Ahmad Biniaz is upported by NSERC Postdoctoral Fellowship. Prosenjit Bose is supported by NSERC. Jean Cardinal is supported by the F.R.S.-FNRS under grant CDR J.0146.18. Michael S. Payne is supported by a Discovery Early Career Researcher Award funded by the Australian Government.

References

[1] E. Ackerman, B. Keszegh, and D. Pálvölgyi. Coloring hypergraphs defined by stabbed pseudo-disks and abab-free hypergraphs. 2019. `http://arxiv.org/abs/1902.08468`.

[2] A. Asinowski, J. Cardinal, N. Cohen, S. Collette, T. Hackl, M. Hoffmann, K. B. Knauer, S. Langerman, M. Lason, P. Micek, G. Rote, and T. Ueckerdt. Coloring hypergraphs induced by dynamic point sets and bottomless rectangles. In *Proceedings of the 13th International Symposium on Algorithms and Data Structures (WADS)*, pages 73–84, 2013.

[3] J. Cardinal, K. B. Knauer, P. Micek, and T. Ueckerdt. Making triangles colorful. *Journal of Computational Geometry*, 4(1):240–246, 2013.

[4] J. Cardinal, K. B. Knauer, P. Micek, and T. Ueckerdt. Making octants colorful and related covering decomposition problems. *SIAM Journal on Discrete Mathematics*, 28(4):1948–1959, 2014.

[5] J. Cardinal and M. Korman. Coloring planar homothets and three-dimensional hypergraphs. *Computational Geometry: Theory and Applications*, 46(9):1027–1035, 2013.

[6] X. Chen, J. Pach, M. Szegedy, and G. Tardos. Delaunay graphs of point sets in the plane with respect to axis-parallel rectangles. *Random Structures Algorithms*, 34(1):11–23, 2009.

[7] H. Grötzsch. Zur Theorie der diskreten Gebilde, VII: Ein Dreifarbensatz für dreikreisfreie Netze auf der Kugel. *Wiss. Z. Martin-Luther-Univ. Halle-Wittenberg. Math.-Nat. Reihe*, 8:109–120, 1959.

[8] B. Keszegh and D. Pálvölgyi. Octants are cover-decomposable. *Discrete & Computational Geometry*, 47(3):598–609, 2012.

[9] B. Keszegh and D. Pálvölgyi. Octants are cover-decomposable into many coverings. *Computational Geometry: Theory and Applications*, 47(5):585–588, 2014.

[10] B. Keszegh and D. Pálvölgyi. More on decomposing coverings by octants. *Journal of Computational Geometry*, 6(1):300–315, 2015.

[11] B. Keszegh and D. Pálvölgyi. Proper coloring of geometric hypergraphs. In *Proceedings of the 33rd International Symposium on Computational Geometry* (*SoCG*), pages 47:1–47:15. 2017.

[12] P. Ossona de Mendez and P. Rosenstiehl. Homomorphism and dimension. *Combinatorics, Probability and Computing*, 14(5-6):861–872, 2005.

[13] W. Schnyder. Planar graphs and poset dimension. *Order*, 5(4):323–343, 1989.

[14] W. T. Trotter and R. Wang. Incidence posets and cover graphs. *Order*, 31(2):279–287, 2014.

Redundant Persistent Acyclic Formations for Vision-Based Control of Distributed Multi-Agent Formations

Alyxander Burns[*] Peter Klemperer[†] Jaemarie Solyst[‡] Audrey St. John[§]

Abstract

We present theoretical and experimental results on the application of acyclic persistent leader-follower formations with redundancy to a distributed multi-agent system. A leader-follower formation is defined on a set of point agents constrained by fixed distance assignments for following other agents; if satisfying the constraints results in the distances between all pairs of agents being maintained, the formation is *persistent*. The (generic) persistence of a leader-follower formation in 2D is combinatorially characterized by a directed graph with one "leader" vertex having no out-edges, one "co-leader" vertex having exactly one out-edge (to the leader), all other "follower" vertices having out-degree at least 2, and an underlying minimally rigid (undirected) graph.

We provide theoretical results for three types of persistent formations with redundancy, including an inductive construction for generating redundantly persistent graphs (the strongest notion of redundancy). We apply redundant persistence to multi-robot systems as a mechanism for incorporating robustness to sensing failure. In particular, we implement the approach on a vision-based distributed multi-robot platform. Using acyclic orientations permits a simple, "wave"-based control that converges reliably, and redundant edges allow the control to recover effectively from sensing limitations (e.g., a camera's limited field of view or obstruction by another robot).

1 Introduction

In applications such as collective transport [12], a formation of robots may be required to maintain a single "shape" or rigid structure. We focus on *leader-follower* formations in the plane, where two pre-specified agents determine the formation's trajectory (e.g., via teleoperation): a *leader* with two degrees of freedom and a *co-leader*[1] with one degree of freedom. The remaining robots are called *followers* and compute their trajectories independently using local geometric conditions.

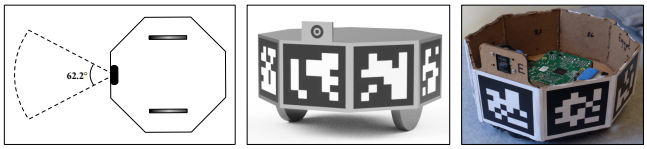


Figure 1: From left to right is a schematic of bottom view of robot (horizontal field of view is 62.2°), 3D model of robot design, picture taken of built robot without cover to show hardware inside.

By modeling these geometric constraints as directed edges, *persistence theory* [8] can be applied to distributed multi-agent systems as a way of controlling a leader-follower formation so that it maintains global structure through local sensing and actuation. Persistence theory is analogous to rigidity theory, where the combinatorics of bar-and-joint frameworks (defined by fixed distance constraints between points) can capture the rigidity or flexibility of the system. When applied to a multi-agent formation, where agents are modeled as points, the source of a directed edge can be assigned the **sensing** and **maintenance** of the distance constraint, reducing overall sensor and actuation costs.

Sensing on a mobile robotic platform is susceptible to hardware and environmental limitations (for instance, a hardware component may fail or the field of view of a camera may be obstructed or limited by lighting). To address this aspect, we build robustness into the theoretical model using constraint redundancy. While a combinatorial characterization and an efficient algorithm for redundantly rigid 2D bar-and-joint frameworks are both known, we are only aware of redundancy in persistence being studied in our previous work [3]. Furthermore, the behavior of persistent formations does not always mimic the corresponding notions in rigid frameworks. Particularly relevant to redundancy is that, while adding an edge to a rigid framework maintains rigidity, **adding an edge to a persistent formation may cause persistence to be lost** [**8**]. In [3], it was shown that every *rigidity circuit* can be oriented

[*]College of Information and Computer Sciences, UMass Amherst, alyxanderbur@cs.umass.edu, *partially supported by NSF IIS-1253146*

[†]Department of Computer Science & Engineering, Mount Holyoke College, pklemper@mtholyoke.edu

[‡]Department of Computer Science, Mount Holyoke College, solys22j@mtholyoke.edu

[§]Department of Computer Science, Mount Holyoke College, astjohn@mtholyoke.edu, *partially supported by NSF IIS-1253146*

[1]The term "first-follower" is also used in related work.

to create a persistent leader-follower formation (with a weak notion of redundancy), and a subsequent recursive algorithm was presented.

In a distributed system, each mobile robot must have a control scheme for actuation that satisfies its set of assigned distance constraints. We restrict our focus to *acyclic* persistent formations, captured by a directed graph with no cycles; a simple, "wave"-based control approach can then be used to ensure reliable convergence.

1.1 Related work.

Results from (undirected) rigidity theory have been applied in similar domains, including network localization [4], decentralized approaches to rigid network constructions [16, 19] and control of multi-robot formations in 3D [20]. A necessary condition for global rigidity is redundant rigidity (drop any edge and remain rigid), leading to the adaptation of the "pebble game" algorithm of [11] to a distributed network. In [5], redundancy is noted as an obstacle to formation control due to real-world sensing inconsistencies.

A directed application of rigidity theory to leader-follower architectures appeared in [5], with persistence theory developed in [8]. A combinatorial characterization for acyclic persistent leader-follower formations was given in [8], along with an inductive "vertex addition" construction technique; the vertex addition step is also known in the rigidity theoretic literature as a 0-extension or a Henneberg I step. The work of [15] uses this construction approach to generate minimally persistent leader-follower formations; a control scheme based on "target points" is proposed and analyzed through simulation and proof of convergence. Given an undirected graph, the existence of an acyclic persistent orientation can be checked in polynomial time [1].

We apply the concept of "cooperative positioning," where follower robots sense other neighboring robots, which are momentarily stationary and serve as "landmarks" [13] for pose calculations. Distance or range measurements can be sensed via an external motion capture system [20], a combination of GPS and computer vision [9], or even infrared [10, 18]. We use popular onboard computer vision techniques for sensing local information, allowing a completely distributed approach.

This paper builds upon the work of [3]; to the best of our knowledge, the use of redundancy as a tool for robustness had not been studied previously in the literature[2]. In this work, we address additional notions of redundancy in persistence, left open in [3], and demonstrate the application on a vision-based distributed multi-robot platform (the simulation results of [3] relied on idealized beacon sensors).

[2]Research in this area is scattered, appearing in various mathematical, computational and engineering settings.

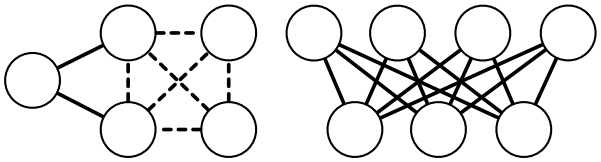


Figure 2: (a) A rigid graph that is not minimally rigid (removing a dashed edge does not result in a flexible framework) and is not redundantly rigid (removing a solid edge does result in a flexible framework). The K_4 subgraph is a rigidity circuit, as removing any edge results in a minimally rigid graph. (b) The complete bipartite graph $K_{3,4}$ is a rigidity circuit.

1.2 Contributions.

We provide theoretical and applied results for acyclic redundant persistence. Analogous to rigidity circuits, we define a *persistence circuit* to be a directed graph such that the removal of any edge results in a minimally persistent graph. We give an algorithm for constructing a persistence circuit from a rigidity circuit before proving that a persistence circuit cannot be a leader-follower formation. By relaxing the notion of redundancy to sets of redundant edges (where any edge in the set may be dropped without impacting persistence), the work of [3] gives an algorithm for constructing a persistent leader-follower formation from a rigidity circuit. In this paper, we show that not every rigidity circuit has an *acyclic* persistent leader-follower formation. Finally, we work with a strong notion of redundancy by defining *redundantly persistent* leader-follower formations, where almost any edge can be dropped without losing persistence, and give an inductive construction algorithm.

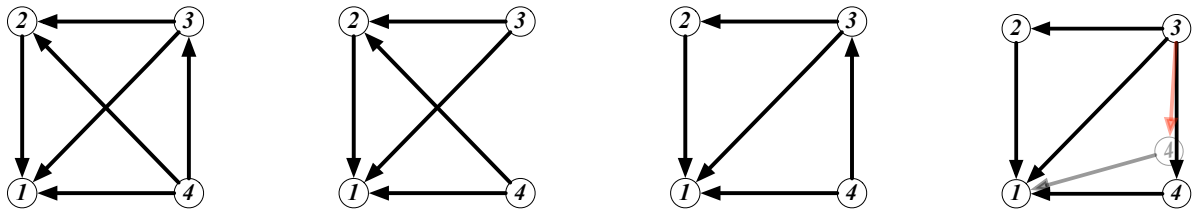
We conclude by applying the theoretical framework to a homogeneous formation of non-holonomic (differential drive) robots, each equipped with a single camera for sensing (see Figure 1). We provide experimental results that validate the ability for a fully distributed multi-robot formation to move and converge to a global structure, where redundancy provides robustness in the presence of limited sensing capabilities.

2 Preliminaries

Our work relies on persistence theory [8] and rigidity theory (see, e.g., [7, 17]). For containment, we give a high-level overview of the relevant concepts here.

2.1 Rigidity theory

Let $G = (V, E)$ be an undirected graph with n vertices and $\ell : E \to \mathbb{R}$ an assignment of distances (or lengths) to each edge; we refer to (G, ℓ) as a (bar-and-joint) *framework*. If $\mathbf{p} \in (\mathbb{R}^2)^n$ assigns positions to each vertex such that the distance constraints are satisfied,



(a) An acyclic persistent formation with redundancy; any edge from vertex 4 can be dropped without losing persistence.

(b) Without an edge, the formation from 3a is minimally persistent; its acyclicity permits an ordering with 3 waves: ({1}, {2}, {3,4}).

(c) Dropping a different edge gives a different minimally persistent formation with an ordering requiring 4 waves: ({1}, {2}, {3},{4}).

(d) The underlying (rigid) graph remains the same, but reversing edge 43 in the formation from 3c results in a formation that is not persistent.

Figure 3: Examples of persistent and not persistent graphs on 4 vertices.

i.e., $||\mathbf{p}(i) - \mathbf{p}(j)|| = l(ij)^2$ for each edge $ij \in E$, we refer to $\mathbf{p}$ as a *realization* of the framework (G, ℓ). A graph is said to be (generically[3]) *rigid* if the distances between all pairs of vertices is determined by the distances specified by edges; otherwise, it is *flexible*. A graph is *minimally rigid* if the removal of any edge results in a flexible graph. Rigid graphs that are not minimally rigid include redundant edges; such a graph is called *redundantly rigid* if the removal of *any* edge results in a rigid graph. Minimality of redundant rigidity is captured through the notion of a *rigidity circuit* (corresponding to the circuits of the rigidity matroid), a rigid graph where removing any edge results in a minimally rigid graph. Figure 2(a) depicts a rigid graph that is neither minimally rigid nor redundantly rigid; removing any dashed edge maintains rigidity, but removing a solid edge results in a flexible graph. The complete graph K_4 (dashed edges) and complete bipartite graph $K_{3,4}$ (also in Figure 2) are rigidity circuits.

Rigidity is defined via undirected graphs, so a straightforward application to a multi-robot formation could require both robots to sense and maintain the constraint dictated by an edge, incurring unnecessary sensing, computation and actuation costs. Therefore, the notion of persistence provides the analogous definition in the directed setting: a directed graph is *persistent* if each vertex can be assigned a position satisfying the distance constraints dictated by its out-going edges and the pairwise distances between all vertices is determined.

Figure 3 highlights the distinct behavior of persistence. The formations of Figures 3c and 3d share the same underlying undirected (rigid) graph, but only the formation of Figure 3c is persistent. The formation depicted in Figure 3d does not have the property that every vertex can satisfy its assigned constraints. Intuitively, vertex 4 can move anywhere on a circle about vertex 1. Vertex 3 can find a position that satisfies two of the three out-going constraints; however, for almost all positions of vertex 4, such as the faded gray position, the third constraint (red edge $\overrightarrow{34}$) will be violated.

The formal definition of persistence requires technical overhead that is outside the scope of this paper. Instead, we will use the following characterization of persistence from Theorem 3 of [8]: a directed graph is persistent if and only if the underlying undirected graph of every subgraph obtained by removing out-edges from vertices with degree ≥ 2 until all vertices have out-degree ≤ 2 is rigid. In particular, a directed graph is *minimally persistent* (i.e., removing any constraint results in a loss of persistence) if and only if its underlying undirected graph is (minimally) rigid and every vertex has out-degree at most 2. For example, the graph in Figure 3a is persistent; without an edge, as in Figures 3b and 3c, it is minimally persistent. We are interested in persistent *leader-follower* formations, where there is a "leader" vertex with out-degree 0, a "co-leader" vertex with out-degree 1 incident to the leader and all other vertices having out-degree at least 2. Since the leader and co-leader vertices have 3 degrees of freedom between them, their positions can be used to determine the coordinates of a persistent formation.

3 Results

We present results on three types of redundancy for persistence theory, analogous to those found in rigidity theory: (1) persistence circuits, (2) persistent formations with redundant edges, and (3) redundantly persistent formations. Due to space constraints, we refer the reader to the Appendix for proofs of the results (including proofs of algorithm correctness) in this section.

3.1 Persistence circuits

In rigidity theory, minimal redundancy is captured by rigidity circuits: removing any edge results in a mini-

[3]The technical definition of genericity is outside of the scope of this paper, but can be thought of as applying to "almost all" realizations (i.e., those not in a special position).

(a) First, add vertices 1 and 2 to V and edge $\overrightarrow{21}$ to A.

(b) Add vertex 3 to V and edges $\overrightarrow{31}$ and $\overrightarrow{32}$ to A.

(c) Add vertex 4 to V and edges $\overrightarrow{41}$, $\overrightarrow{42}$, and $\overrightarrow{43}$ to A.

(d) Add vertex 5 to V and edges $\overrightarrow{52}$, $\overrightarrow{53}$, and $\overrightarrow{54}$ to A.

Figure 4: Construction of an acyclic redundantly persistent formation using Algorithm 2.

mally rigid graph. Analogously, we define a *persistence circuit* to be a graph such that any edge's removal gives a minimally persistent graph.

A persistence circuit must have an underlying rigidity circuit. Since rigidity circuits are well-understood combinatorially and can be constructed inductively [2], Algorithm 1 produces a *persistence circuit* from a rigidity circuit. The $(2,3)$-pebble game algorithm of [11, 14], which appears as a subroutine, determines rigidity of an undirected graph in quadratic time by constructing a directed graph. If the input graph is rigid, the output will be a directed graph with out-degree at most 2 and an underlying undirected minimally rigid graph, i.e., a minimally persistent graph. Throughout the algorithm, pebbles are used as a mechanism for controlling the out-degree of a vertex; "pebble collection moves" rely on depth-first search to re-orient directed edges.

Algorithm 1 Construction of a persistence circuit from a rigidity circuit

Given a rigidity circuit $G = (V, E)$:

1. Remove any edge $e = ij \in E$.
2. Play the $(2,3)$-pebble game on $G' = (V, E \setminus \{e\})$ to obtain a directed graph H.
3. If i has 0 pebbles, use pebble collection moves on H to collect a single pebble on i.
4. Output the resulting directed graph with the additional edge $\overrightarrow{ij}$.

While persistent, the class of graphs output by Algorithm 1 are not leader-follower formations. In fact, we can show the following:

Lemma 1 *A persistent leader-follower formation cannot be a persistence circuit.*

Although we cannot have leader-follower formations which are persistence circuits, [3] presents an algorithm that orients a rigidity circuit to be a persistent leader-follower formation with a more restricted notion of redundancy. The graphs produced contain sets of *redundant* edges; for example, the set of out-edges from vertex 4 in Figure 3a is redundant, as any edge may be dropped without losing persistence.

3.2 Acyclic persistent formations with redundant edge sets

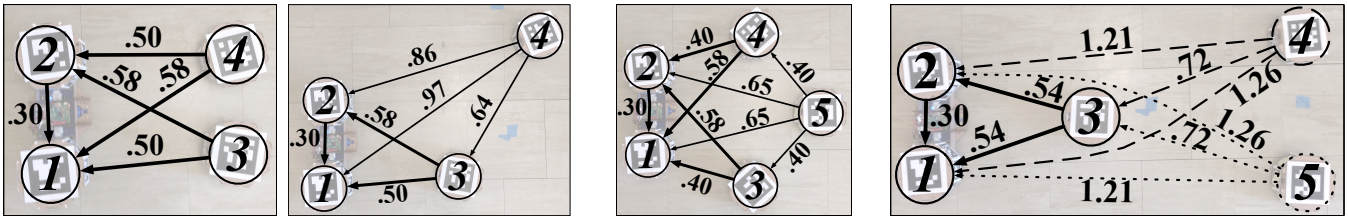
We now restrict our focus to **acyclic** persistent formations, which permit a "wave"-based control approach to satisfying the constraints. These are characterized combinatorially in Theorem 5 of [8]: an acyclic graph is persistent if and only if it has (1) one "leader" vertex with out-degree 0, (2) one "co-leader" vertex with out-degree 1, incident to the "leader," (3) all other "follower" vertices with out-degree 2 or larger. Then there exists a *Henneberg sequence* for the vertices (corresponding to an inductive "Henneberg"-type construction of the underlying undirected rigid graph) such that each vertex only has out-going edges to vertices earlier in the sequence. The first two vertices in the sequence are the leader and co-leader. We can group the subsequent vertices into a sequence of k *waves* $(w_0, w_1, \ldots, w_{k-1})$, where $w_i \subset V$, such that the vertices in a wave only have out-going edges to vertices in earlier waves. For example, Figure 3b depicts a graph with 3 waves, while Figure 3c requires 4 waves. We use the waves to control a formation so that it converges reliably, as shown through the experimental results of Section 4.

Note that the algorithm from [3] can be used to find an acyclic persistent leader-follower orientation of a rigidity circuit, if one exists, by brute-force consideration of the removal of every edge. This naturally leads to the question of whether there are rigidity circuits for which no acyclic persistent leader-follower orientations exist. The following gives the answer in the negative.

Lemma 2 *There are no acyclic leader-follower persistent orientations of the rigidity circuit $K_{3,4}$.*

3.3 Inductive constructions for acyclic redundantly persistent leader-follower formations

Following the results from the previous section, it is of interest to understand properties of acyclic persistent leader-follower formations. Theorem 5 of [8] implies that the out-edges of any follower vertex with out-



(a) Rectangle Formation (b) Kite Formation (c) Star Formation (d) Butterfly Formation

Figure 5: Screenshots of persistent formations taken from overhead during an experiment. The robot IDs and specified distance constraint (in meters) label the graph.

degree larger than 2 form a redundant set of constraints: any subset of edges such that at least 2 remain can be dropped without losing persistence. For example, consider the Butterfly graph of Figure 5d. Vertex 1 is the leader, vertex 2 the co-leader and vertices 3, 4 and 5 are followers. Since vertices 4 and 5 have out-degree 3, their out-edge sets (dashed and dotted, respectively) each form a redundant set; any out-edge can be dropped from either set without losing persistence.

Algorithm 2 constructs graphs that are acyclic persistent leader-follower formations which satisfy a version of the (strong) notion of being redundantly persistent. We define this notion in the context of an acyclic leader-follower formation, which always contains a base triangle, so we do not require redundancy within it. Let $G = (V, E)$ be an acyclic persistent leader-follower formation and $v_L, v_C, v_3, \ldots, v_{|V|}$ be a Henneberg ordering for G; let $E_b = \{\overrightarrow{v_C v_L}, \overrightarrow{v_3 v_L}, \overrightarrow{v_3 v_C}\}$ be the *base triangle* edge set. We call G *redundantly persistent* if the removal of any edge e not in the base triangle results in a persistent graph. We extend the trilateration approach of [4] to persistence to inductively construct acyclic redundantly persistent leader-follower formations. Refer to Figure 4 for an example run of the construction. Note that any set of vertices can be chosen in Step 2(b); the choice could be random or constrained by additional criteria (such as visibility determined by the geometry of a formation).

Algorithm 2 Inductive construction of a redundantly persistent acyclic formation

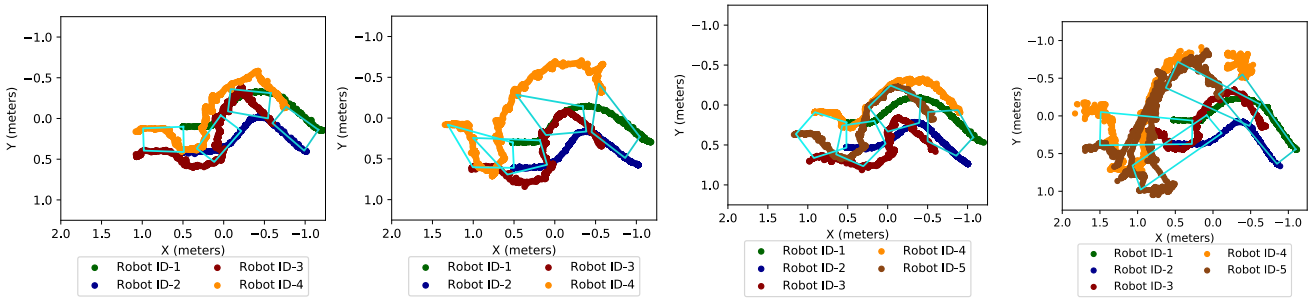
1. Initialize a set of vertices $V = \{v_L, v_C\}$ and a set of edges $A = \{\overrightarrow{v_C v_L}\}$
2. For $i \in [3..n]$:
   (a) If $i = 3$, add edges $\overrightarrow{v_i v_L}$ and $\overrightarrow{v_i v_C}$ to A
   (b) Else select a set $V' \subseteq V$ of at least 3 distinct vertices and add edges $\overrightarrow{v_i v'}$, for all $v' \in V'$, to A
   (c) Add v_i to V
3. Output the directed graph $H = (V, A)$

4 Application to multi-robot formations

Acyclic persistent leader-follower formations can be controlled with reliable convergence by having each wave (in order) sense and satisfy constraints; after the final wave is completed, the resulting positions form a realization for the underlying framework. We validate this approach through experiments on four acyclic leader-follower formations with varying levels of redundancy, depicted in Figure 5. The persistent graphs are overlaid on the multi-robot formation, with distances (in meters) labeling the edges. We chose these formations to analyze the impact of both combinatorial and geometric properties.

The Rectangle formation is minimally persistent, composed of four robots geometrically positioned at the corners of a rectangle; the two robots (vertices 3 and 4) are constrained to follow leader vertex 1 and co-leader vertex 2. The addition of edge $\overrightarrow{43}$ gives the Kite formation, where robot 4 has a set of 3 redundant out-edges; any edge may be lost without impacting the persistence of the formation. The Star formation is persistent with the out-edges of vertex 5 forming a redundant set; any pair of out-edges may be dropped without impacting persistence. Note that the Star formation is not redundantly persistent, as dropping an out-edge from vertex 4 will result in the loss of persistence. Finally, the Butterfly formation, produced by Algorithm 2 on 5 vertices, is redundantly persistent.

In all formations, vertices 1 and 2 are the leader and co-leader, together determining the global positioning of the formation. For ease of control, we simplify the implementation using a single leader robot platform containing the points for the leader and co-leader and use the same predetermined path for the leader throughout our experiments. Followers sense their positions relative to neighbors using a dictionary of OpenCV-Aruco [6] computer vision markers. When redundant edge sets are present, the follower has a prioritized list of target pairs to use for constraint maintenance. This results in a



(a) Rectangle Formation (b) Kite Formation (c) Star Formation (d) Butterfly Formation

Figure 6: Scatter plots illustrating the tracks of each robot for each of four formations. We outline the convex hull (cyan) of each formation during every tenth measurement to emphasize the formation.

more robust multi-agent system than using a minimally persistent formation, as the formation can recover from sensing failure, e.g., due to environmental factors, such as lighting, or occlusion of targets by other robots or objects. Given the relative positions of two targets, the follower computes a point satisfying both constraints (choosing the closer point of intersection of the two corresponding circles) and moves directly to that position. We refer the reader to the Appendix for more details on the hardware setup and wave-based control approach.

4.1 Experimental results

An overhead camera was used to obtain the coordinates of each robot for analysis. Figure 6 illustrates the movement tracks of the robots in each of the tested formations. Note that, for the Butterfly formation, the initial sharp turn of the leader caused Robot ID-5 to move out of the view of the overhead camera. We overlay the convex hull at timed intervals to highlight the maintenance of the overall structure.

Formation	From	To Robot 1	To Robot 2	To Robot 3
Rectangle	Robot 3	0.012±0.020, 0.168	0.004±0.019, 0.136	NA
	Robot 4	**0.009±0.012 ≤ 0.163**	**0.005±0.012, 0.166**	(-0.027±0.009 ≤ 0.057)
Kite A	Robot 3	0.005±0.008 ≤ 0.142	-0.001±0.008 ≤ 0.097	NA
	Robot 4	**0.005±0.012 ≤ 0.130**	**-0.001±0.013, 0.122**	-0.020±0.010 ≤ 0.145
Kite B	Robot 3	0.008±0.015, 0.161	0.004±0.019 ≤ 0.230	NA
	Robot 4	**0.007±0.014 ≤ 0.149**	0.001±0.013 ≤ 0.194	**-0.018±0.018 ≤ 0.139**
Kite C	Robot 3	0.005±0.019≤ 0.136	-0.002±0.018 ≤ 0.142	NA
	Robot 4	0.007±0.028 ≤ 0.181	**-0.014±0.032 ≤ 0.206**	**0.004±0.025 ≤ 0.132**

Table 1: Constraint accuracy for four formations: the Rectangle formation and three variations of Kite.

For the Rectangle and Kite formations, Table 1 provides more precise validation of constraint maintenance. Since the Kite formation has a set of redundant edges from vertex 4, we ran 3 experiments, varying the priority list of pairs of vertices to follow: Kite A used ({1,2},{1,3},{2,3}), Kite B used ({1,3},{2,3},{1,2}) and Kite C used ({2,3},{1,3},{1,2}). The results were compiled by calculating the mean error for each pair-wise constraint within each formation, then calculating the minimum, mean and maximum of those means; they are formatted as: *mean error during camera wave (m)* $\pm$*standard deviation* $\leq$ *max error from constraint.* The maximum single-wave leader movement is 0.112 m and 10 degrees, which affects the maximum error from constraint throughout the movement waves. The results in parentheses were not directly controlled, but instead maintained by the formation. Additional results and data may be found in the Appendix.

5 Remarks

Rigidity and persistence theory are typically applied as static analysis tools, but control of a multi-agent mobile formation must include approaches for dynamic movement. By working with acyclic formations, we are able to exploit Henneberg sequences (developed primarily as a tool for inductive proof techniques) to implement a wave-based control scheme with reliable convergence.

Performing experiments through a low-cost hardware platform can give insight into the challenges to creating a robust control for multi-robot formations; popular vision-based sensing is susceptible to geometric and environmental factors, including viewing angle, lighting and occlusion. Our results confirmed the robustness brought to the system through the incorporation of redundancy in the theoretical model, allowing the control to recover from sensing loss.

While redundancy in rigidity theory is well-understood, the analogous concepts in persistence theory have not been well-studied. Our results reveal distinct behaviors of redundancy in the directed graphs of persistence theory that do not arise in the undirected graphs of rigidity theory.

We are grateful to the anonymous reviewers for their detailed and helpful comments.

References

[1] J. Bang-Jensen and T. Jordan. On persistent directed graphs. *Networks*, 52:271–276, 2008.

[2] A. R. Berg and T. Jordán. A proof of connelly's conjecture on 3-connected circuits of the rigidity matroid. *J. Comb. Theory Ser. B*, 88(1):77–97, May 2003.

[3] A. Burns, B. Schulze, and A. St. John. Persistent multi-robot formations with redundancy. In R. Groß, A. Kolling, S. Berman, E. Frazzoli, A. Martinoli, F. Matsuno, and M. Gauci, editors, *Distributed Autonomous Robotic Systems: The 13th International Symposium*, pages 133–146. Springer International Publishing, 2018.

[4] T. Eren, O. K. Goldenberg, W. Whiteley, Y. R. Yang, A. S. Morse, B. D. O. Anderson, and P. N. Belhumeur. Rigidity, computation, and randomization in network localization. In *IEEE INFOCOM 2004*, volume 4, pages 2673–2684 vol.4, 2004.

[5] T. Eren, W. Whiteley, and P. N. Belhumeur. Rigid formations with leader-follower architecture. Technical report, Columbia University, CUCS-010-05, March 2005.

[6] S. Garrido-Jurado, R. M. noz Salinas, F. Madrid-Cuevas, and M. Marín-Jiménez. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, 47(6):2280 – 2292, 2014.

[7] J. Graver, B. Servatius, and H. Servatius. *Combinatorial Rigidity*, volume 2 of *Graduate Studies in Mathematics*. American Mathematical Society, 1993.

[8] J. M. Hendrickx, B. D. O. Anderson, J.-C. Delvenne, and V. D. Blondel. Directed graphs for the analysis of rigidity and persistence in autonomous agent systems. *International Journal of Robust and Nonlinear Control*, 17(10-11):960–981, 2007.

[9] K. Hou, H. Sun, Q. Jia, and Y. Zhang. An autonomous positioning and navigation system for spherical mobile robot. *Procedia Engineering*, 29:2556 – 2561, 2012. 2012 International Workshop on Information and Electronics Engineering.

[10] R. Islam, I. Mobin, M. N. Shakib, and M. M. Rahman. An approach of ir-based short-range correspondence systems for swarm robot balanced requisitions and communications. *CoRR*, abs/1608.03610, 2016.

[11] D. Jacobs and B. Hendrickson. An Algorithm for Two-Dimensional Rigidity Percolation: The Pebble Game. *Journal of Computational Physics*, 137(CP975809):346 – 365, 1997.

[12] C. R. Kube and E. Bonabeau. Cooperative Transport By Ants and Robots. Working Papers 99-01-008, Santa Fe Institute, Jan. 1999.

[13] R. Kurazume, S. Nagata, and S. Hirose. Cooperative positioning with multiple robots. In *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, pages 1250–1257 vol.2, 5 1994.

[14] A. Lee and I. Streinu. Pebble game algorithms and sparse graphs. *Discrete Mathematics*, 308(8):1425–1437, 2008.

[15] S. Mou, M. Cao, and A. Stephen Morse. Target-point formation control. *Automatica*, 61(C):113–118, Nov. 2015.

[16] A. Priolo, R. K. Williams, A. Gasparri, and G. S. Sukhatme. Decentralized algorithms for optimally rigid network constructions. In *IEEE International Conference on Robotics and Automation*, pages 5010–5015, 2014.

[17] M. Sitharam, A. St. John, and J. Sidman, editors. *Handbook of Geometric Constraint Systems Principles*. Chapman and Hall/CRC, 2018.

[18] H. Wei, N. Li, M. Liu, and J. Tan. A novel autonomous self-assembly distributed swarm flying robot. *Chinese Journal of Aeronautics*, 26(3):791 – 800, 2013.

[19] R. K. Williams, A. Gasparri, M. Soffietti, and G. S. Sukhatme. Redundantly rigid topologies in decentralized multi-agent networks. In *IEEE Conference on Decision and Control*, pages 6101–6108, 2015.

[20] D. Zelazo, A. Franchi, H. H. Bülthoff, and P. R. Giordano. Decentralized rigidity maintenance control with range measurements for multi-robot systems. *International Journal of Robotics Research*, 34:128, 01 2015.

Appendix

5.1 Proofs for theoretical results

We include here the proofs for correctness of Algorithm 1, Lemma 1, Lemma 2 and Algorithm 2.

Proof. [of correctness of Algorithm 1] Let $H = (V, A)$ be the output of Algorithm 1. By construction and Invariant (1) of Lemma 10 of [14] (for any vertex v, the sum of the number of pebbles on v and the out-degree of v is 2), H has either 1 vertex with out-degree 0 or 2 vertices with out-degree 1; all other vertices have out-degree 2. We show that H is a persistence circuit. Let $e \in A$ and $J = (V, A \backslash e)$. Then J either has (1) 3 vertices with out-degree 1, and all other vertices with out-degree 2; or (2) 1 vertex with out-degree 0, 1 vertex with out-degree 1, and all others with out-degree 2. Since the underlying (undirected) graph of H is a rigidity circuit, the underlying graph of J is minimally rigid. Thus, by Theorem 3 of [8], J is minimally persistent and H is a persistence circuit. □

Proof. [of Lemma 1] Let $H = (V, A)$ be a persistent leader-follower formation with v_C, v_L the leader and co-leader vertices having out-degree 0 and 1, respectively. Then there must be exactly one vertex x with out-degree 3 and all other vertices with out-degree 2. Assume, for a contradiction, that H is a persistence circuit; since the underlying graph of H must be a rigidity circuit, $|V| \geq 4$. Let $y \in V$ be a vertex with out-degree 2 and $e = \overrightarrow{yz}$ be one of the out-going edges. By assumption, $J = (V, A \setminus e)$ is minimally persistent and its underlying graph is minimally rigid. Since x has out-degree 3, dropping any out-going edge must result in a persistent graph J' by Theorem 3 of [8]. However, the underlying graph of J becomes flexible after the removal of any edge, so J' cannot be persistent, giving the contradiction. □

Proof. [of Lemma 2] Consider any persistent leader-follower orientation $G = (U, V, E)$ of $K_{3,4}$, where $|U| = 3$ and $|V| = 4$ are disjoint vertex sets. We show it must have a cycle. Suppose, for a contradiction, it is acyclic. Since $K_{3,4}$ is a rigidity circuit, $|E| = 2n - 2$ edges, there must be one vertex v_L with out-degree 0, one vertex v_C with out-degree 1, one vertex x with out-degree 3 and all other vertices with out-degree 2. Since G is an acyclic persistent orientation, there exists an ordering of vertices $v_L, v_C, v_3, \ldots, v_7$ such that the out-edges of each v_i are directed to vertices with smaller indices. Then $x = v_7 \in V$ and $v_3, \ldots, v_6$ must have out-degree 2. However, v_6 is incident to 3 vertices of the set $\{v_L, v_c, v_3, \ldots, v_5\}$ and must be oriented towards them; thus, v_6 has out-degree 3, giving the contradiction. □

Proof. [of Algorithm 2] Let G be the output of an execution of Algorithm 2. By construction, G is an acyclic graph with (1) one "leader" vertex with out-degree 0, (2) one "co-leader" vertex with out-degree 1, incident to the "leader," (3) all other "follower" vertices with out-degree 3. By Theorem 5 of [8], G is persistent. Let $e \notin E_b = \{\overrightarrow{v_C v_L}, \overrightarrow{v_3 v_L}, \overrightarrow{v_3 v_C}\}$; then e is an out-edge from a vertex of degree 3. By Theorem 3 of [8], the graph obtained by removing e is persistent. Thus, G is redundantly persistent. □

Formation	{1,2}	{1,3}	{2,3}	Search
Kite A	**98%**	0%	0%	2%
Kite B	0%	**99%**	1%	0%
Kite C	0%	76%	**23%**	1%
Star Robot 3	95%	NA	NA	5%
Star Robot 4	100%	NA	NA	0%
Star Robot 5	**96%**	0%	0%	4%
Butterfly Robot 3	100%	NA	NA	0%
Butterfly Robot 4	**4%**	71%	$< 1\%$	25%
Butterfly Robot 5	**7%**	1%	75%	17%

Table 2: Summary of the landmark-pair selections taken by robots during experiments. Followers require two landmarks and choose from a prioritized list of pairs (column labels) when more are available; the top priority pair is boldface in each row.

5.2 Experimental details

This section contains a more detailed description of our experimental setup as well as additional data and analysis.

Setup. Figure 5 provides a depiction of the hardware design. Robots are based on a Pololu Romi[4] wheeled differential-drive chassis and electronics fitted within a octagon camera target body. The leader is fitted with two camera targets 30 cm apart, centered over the Romi Chassis. On-board computation is provided by a Raspberry Pi version 3 with connected Raspberry Pi Camera (v2, 8 MP version). These robots are relatively low-cost, with a total bill-of-materials under $200 USD.

The robot shells are designed as an octagonal prism, with 2.7" edges, and 2.3" square Aruco vision markers centered within each face; a unique marker was assigned to each robot, allowing it to be used as an identifier. We chose 8 sides as it experimentally produced the best results. Shells with fewer sides were more susceptible to marker identification loss due to viewing angle or occlusion factors. Using more sides (while keeping the overall shell dimensions constant) forced a decrease in the size of the markers, resulting in marker recognition loss susceptible to distance factors; the octagonal prism allowed recognition up to 3 meters away.

Formation control. The overall movement of the formation is dictated by the leader robot; individual follower robots move to maintain their specified constraints. By working with acyclic persistent formations, we can use a simple "wave" control that converges reliably. Robots only move during their assigned waves, with the specific wave assigned to a robot using a Henneberg ordering for the formation. Each wave ends only when all robots assigned to that wave have finished moving and satisfied their constraints.

Separating the formation into waves creates several advantages beyond reliable convergence: the landmarks (Aruco markers) tracked by the followers are not moving, resulting in improved camera accuracy, followers need only minimize

[4]Pololu Corporation: https://www.pololu.com/category/203/romi-chassis-kits

constraint deviation with their landmarks, and fewer robots moving reduces the risk of collisions. However, the wave-motion procedure suffers from the disadvantage that the minimum deviation from the range-constraints is bounded by the distance traveled by the leader, and the cascading effects as the following waves move increase the error.

Redundancy data and analysis. Table 2 summarizes the landmark usage of each robot in determining movement targets throughout 5 experiments on the formations with redundancy: 3 with the Kite formation (varying the top landmark-pair in the priority list), one for the Star and one for the Butterfly formation. Each robot made approximately one-hundred movements, and the results are reported as a percentage of total movements (including those required rotate while searching for markers). Note that sensing constraints play a role in the experiment for Kite C, where (Robot 2,Robot 3) is the prioritized pair, but (Robot 1,Robot 3) determines 76% of the movements. The geometry of the formation places landmarks Robot 2 and Robot 3 at the extreme opposite edges of Robot 4's field-of-vision, making them less likely to be viewed simultaneously.

Geometric Systems of Unbiased Representatives

Aritra Banik* Bhaswar B. Bhattacharya† Sujoy Bhore‡ Leonardo Martínez-Sandoval§

Abstract

Let P be a set of points in $\mathbb{R}^d$, B a bicoloring of P and $\mathcal{O}$ a family of geometric objects (that is, intervals, boxes, balls, etc). An object from $\mathcal{O}$ is called balanced with respect to B if it contains the same number of points from each color of B. For a collection $\mathcal{B}$ of bicolorings of P, a geometric system of unbiased representatives (G-SUR) is a subset $\mathcal{O}' \subseteq \mathcal{O}$ such that for any bicoloring B of $\mathcal{B}$ there is an object in $\mathcal{O}'$ that is balanced with respect to B.

We study the problem of finding G-SURs. We obtain general bounds on the size of G-SURs consisting of intervals, size-restricted intervals, axis-parallel boxes and Euclidean balls. We show that the G-SUR problem is NP-hard even in the simple case of points on a line and interval ranges. Furthermore, we study a related problem on determining the size of the largest and smallest balanced intervals for points on the real line with a random distribution and coloring.

Our results are a natural extension to a geometric context of the work initiated by Balachandran et al. on arbitrary systems of unbiased representatives.

1 Introduction

Let P be a set of size n. A *bicoloring* B of P is a color assignment (red or blue) of the points in P, that is, $B : P \to \{\text{Red, Blue}\}$, where B contains at least one red and at least one blue point. For a bicoloring B, a subset of points $P' \subseteq P$ is called *balanced with respect to B* if P' contains the same number of red and blue points, with respect to B. Given a set P and a set of bicolorings $\mathcal{B}$, a *system of unbiased representatives (SUR)* consists of a collection $\mathcal{S}$ of subsets of P such that for every bicoloring $B \in \mathcal{B}$, there is at least one subset in $\mathcal{S}$ that is *balanced* with respect to B.

Balachandran et al. [2] studied various problems related to finding SURs, with the motivation that SURs are useful for product testing over a large population. For example, suppose the effectiveness of a drug on patients is studied with respect to a large set of binary attributes related to physical characteristics, such as body weight, height, age. It is desirable to choose few families of test subjects that help to represent these attributes in a balanced manner.

Now, consider an instance where in addition we are given specific geographic locations for our test subjects and we are asked to pick them close to each other to save costs in sampling. In this situation, we cannot choose arbitrary families of test subjects: we would be required to impose some geometric constrains on them.

A natural way to model this restriction is to represent the population by a point set P in Euclidean space of some dimension and to sample using ranges from some fixed family of geometric objects, that is, intervals, boxes, balls, etc. This leads to the following definitions.

For a bicoloring B of P, we say that a geometric range is *balanced with respect to B* if the subset of points of P that it contains is balanced with respect to B. Given a set P, a set of bicolorings $\mathcal{B}$ and a family of allowed geometric ranges $\mathcal{O}$, a *geometric system of unbiased representatives (G-SUR)* consists of a subfamily $\mathcal{O}' \subseteq \mathcal{O}$ such that for every bicoloring $B \in \mathcal{B}$, there is at least one object in $\mathcal{O}'$ that is balanced with respect to B.

Problem 1 *(**G-SUR**) Given a set $P \subset \mathbb{R}^d$ of n points, a set of bicolorings $\mathcal{B}$ of P, and a collection of allowed geometric ranges $\mathcal{O}$, find a G-SUR of minimal size.*

For a specific attribute, it is desirable to understand how big a balanced range (for this attribute) can be. Assuming attributes are uniformly distributed over the population, leads to the following problem:

Problem 2 *(**Balanced Random Covering**) Given a set P of n points and a random bicoloring B of P (chosen uniformly at random from a collection of bicolorings $\mathcal{B}$ of P), what can be said about the behavior of the size of the largest/smallest balanced interval as n goes to infinity?*

In addition to the practical motivation, Problem 1 and

*School of Computer Sciences, National Institute of Science Education and Research, HBNI, Bhubaneswar, India. aritrabanik@gmail.com

†Department of Statistics, University of Pennsylvania, Philadelphia, USA. bhaswar@wharton.upenn.edu

‡Algorithms and Complexity Group, TU Wien, Vienna, Austria. sujoy.bhore@gmail.com. Research supported by the grant P31119 of the Austrian Science Fund (FWF).

§Institut de Mathématiques de Jussieu-Paris Rive Gauche (UMR 7586), Sorbonne Université, France. leomtz@im.unam.mx. Research supported by the grant ANR-17-CE40-0018 of the French National Research Agency ANR (project CAPPS)

Problem 2 are related to the vast literature on colorings of geometric objects in which a balanced property is desired. This includes classical results as the ham-sandwich theorem and its algoritmic version by Lo et al. [6]. Other relevant results on balanced coloring of point sets include the balanced island problem studied by Aichholzer et al. [1], balanced partitions problem for 3-colored planar sets by Bereg et al. [4], and balanced 4-holes in bichromatic point set [3].

1.1 Our Results

As an introduction to the subtleties of the geometric context, we study the G-SUR problem for n points on a line and interval ranges in Section 2. We show, given a set of n points on a line and a collection of interval ranges, there is G-SUR of size $n-1$. Moreover, this bound is tight, that is, there are a set of bicolorings for which $n-1$ intervals are required to obtain a balanced interval (Theorem 3). Motivated by statistical significance, we then focus on G-SURs where the set of ranges are intervals of size $2k$. Here, we show that for any set of bicolorings $\mathcal{B}$, where each bicoloring in $\mathcal{B}$ contains more than $\lfloor \frac{n}{2k}+1 \rfloor (k-1)$ red and $\lfloor \frac{n}{2k}+1 \rfloor (k-1)$ blue points such a G-SUR exists (Theorem 4). Next, for $m < n/2$, we give bounds on the size of G-SURs for when each bicoloring of $\mathcal{B}$ has at least m red and m blue points. More precisely, we show that $n-m$ intervals are always sufficient and sometimes necessary (Theorem 6). All these results extend to higher dimensions to point sets in $\mathbb{R}^d$ and G-SURs consisting of axis-parallel boxes. Section 3 provides the hardness results. We show that the problem of finding a minimal size G-SUR is NP-hard even in the simple case of points on the real line and interval ranges (Theorem 7). To do this we provide a reduction from the Set Cover problem.

In Section 4, we study the problem for points in $\mathbb{R}^d$ and G-SURs consisting of Euclidean balls. Once more, we show $n-1$ balls are sometimes necessary and always sufficient to give a G-SUR (Theorem 10).

Finally, in Section 5, we study the Balanced Random Covering problem, where we compute the asymptotic size of the largest/smallest balanced interval for uniformly random bicolorings of points on a line in a discrete model (Theorem 11) and a continuous one (Theorem 12).

2 Points on a Line and Interval G-SURs

Let $P = \{p_1 \ldots, p_n\}$ be a set of points on the real line $\mathbb{R}$. Throughout this section we assume that $\{p_1 \ldots, p_n\}$ is sorted from left to right on the real line. Here our goal is to find a minimum size G-SUR consisting of interval ranges for a given family of bicolorings $\mathcal{B}$ of P.

2.1 Lower and Upper Bounds

In this section we show that $n-1$ intervals are always sufficient and sometimes necessary.

Theorem 3 *Let $P = \{p_1, \ldots, p_n\}$ be a set of n points on a line. Then, the following hold:*

(a) There exists a set of $n-1$ bicolorings $\mathcal{B}$, for which any G-SUR consisting of intervals has size at least $n-1$ and

(b) There exists a set $\mathcal{I}$ of $n-1$ intervals such that for any bicoloring B of P there is at least one balanced interval in $\mathcal{I}$ with respect to B.

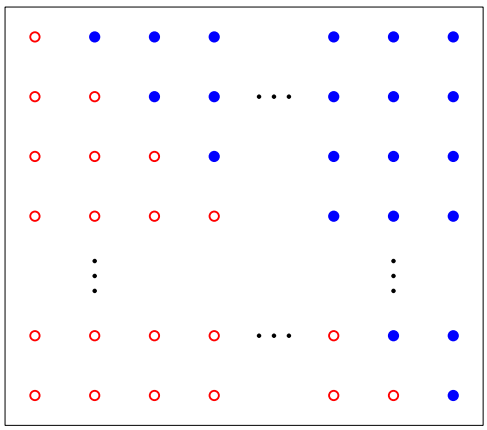


Figure 1: An illustration of the case where $n-1$ intervals are necessary.

Proof. We prove the first part of the theorem by constructing an example where $n-1$ intervals are necessary. Without loss of generality, we can assume $P = \{1, 2, \ldots, n\}$. We consider the set of bicolorings $\mathcal{B} = \{B_1, \ldots, B_{n-1}\}$ where the bicoloring B_i colors the first i-points red and the remaining $(n-i)$ points blue (see Figure 1).

A balanced interval with respect to B_i may be shortened until its endpoints are integers. Thus, we may choose a minimal G-SUR that consists only of intervals with integral endpoints. In such a system, an interval that is balanced for B_i must be symmetric around $\frac{2i+1}{2}$. Different bicolorings need intervals symmetric around different points, so a G-SUR for $\mathcal{B}$ requires at least $n-1$ intervals.

For the second part of the theorem consider

$$\mathcal{I} = \{[p_1, p_2], [p_2, p_3], \ldots, [p_{n-1}, p_n]\}.$$

Let B be any bicoloring of P. We claim that there exists an interval in $\mathcal{I}$ which is balanced with respect to

B. Indeed, if this is not the case then all the intervals in $\mathcal{I}$ are monochromatic, so

$$B(p_1) = B(p_2) = \ldots = B(p_n),$$

which is a contradics the fact that B contains at least one red and one blue point. This concludes the proof of (b). □

Theorem 3 is already evidence of the contrast between the geometric and the abstract setting. While Balachandran et al. proved that $n-1$ arbitrary sets are sometimes necessary, their example consists of all 2^n-2 possible bicolorings. The theorem above shows that only taking $(n-1)$ bicolorings are enough for the necessity, in the geometric context. Theorem 3 says that this is still the case even if we further restrict the allowed ranges to be intervals.

An analogous proof shows that the $n-1$ bound carries to point sets in $\mathbb{R}^d$ and G-SURs consisting of axis-parallel boxes.

2.2 G-SURs Consisting of Intervals of Size $2k$

In this section we fix a positive integer k and consider the case where the G-SUR consists of intervals of length exactly $2k$. Certainly, with this restriction it is not possible to have a G-SUR for every possible set of bicolorings. For example, consider any bicoloring B that contains exactly one red point. No interval of size greater than 2 is balanced with respect to B. In the following lemma we show that if each color is large enough, then there exists a G-SUR consisting of intervals of length $2k$.

Theorem 4 *Given $n \geq 2k$, a set $P = \{p_1, \ldots, p_n\}$ of points on the real line and a set of bicolorings $\mathcal{B}$, where each bicoloring in $\mathcal{B}$ contains more than $\left\lfloor \frac{n}{2k} + 1 \right\rfloor (k-1)$ red and $\left\lfloor \frac{n}{2k} + 1 \right\rfloor (k-1)$ blue points, there exist a G-SUR for $\mathcal{B}$ consisting of intervals of size $2k$.*

Proof. Let B be a bicoloring of P containing more than $\left\lfloor \frac{n}{2k} + 1 \right\rfloor (k-1)$ red and $\left\lfloor \frac{n}{2k} + 1 \right\rfloor (k-1)$ blue points. Consider now the set of consecutive disjoint intervals of size $2k$. More formally,

$$\mathcal{I} = \{I_j | I_j = [p_j, p_{j+2k-1}] \text{ where } 1 \leq j \leq n - 2k + 1\}.$$

Let r_j (resp. b_j) be the number of red (resp. blue) points in the interval I_j. We say that an interval in $\mathcal{I}$ is *red* (resp. *blue*) if $r_j > b_j$ (resp. $b_j > r_j$).

We claim that $\mathcal{I}$ is a G-SUR. For the sake of contradiction, we suppose that every interval from $\mathcal{I}$ is either red or blue. We may assume I_1 is red.

Claim 5 *Every interval $I_j \in \mathcal{I}$ is red.*

Proof. We prove this by induction on j. By assumption, I_1 is red. Now, suppose that I_j is red, that is, $r_j > b_j$, so $r_j \geq k+1$. As we shift from I_j to I_{j+1} we lose or gain at most one red point, so $r_{j+1} \geq r_j - 1 \geq k$. It is then impossible for I_{j+1} to be blue. Since each interval is either red or blue, I_{j+1} must be red. □

Consider now the set of disjoint intervals

$$\mathcal{I}' = \{I_j | j \equiv 1 \bmod 2k, 1 \leq j \leq n - 2k + 1\}.$$

The intervals from $\mathcal{I}'$ and the interval I_{n-2k+1} cover the entire set P. Since every interval from $\mathcal{I}$ is red, it has at most $k-1$ blue points. Therefore, the coloring has at most

$$\left\lfloor \frac{n}{2k} + 1 \right\rfloor (k-1)$$

blue points. This yields a contradiction to the number of blue points of B given by the hypothesis, so $\mathcal{I}$ must have a balanced interval with respect to B. □

The result in Theorem 4 is tight, in the sense that if we have fewer points of either color we cannot guarantee the existence of a G-SUR of size $2k$. This can be witnessed by an example on $n = 3k-1$ points on the real line and the coloring B whose first $k-1$ and last $k-1$ points are blue, and the middle $k+1$ ones are red.

An analogous proof shows that Theorem 4 extends to point sets in $\mathbb{R}^d$ and G-SURs consisting of axis-parallel boxes of size $2k$.

2.3 G-SURs for m-restricted Bicolorings

For $m \leq n/2$, a bicoloring B of P is *m-restricted* if it contains at least m points of each color. In this section we study the size of G-SURs for m-restricted colorings.

If n is even and $m = n/2$, then there is a G-SUR of size 1: the one consisting of the interval $[p_1, p_n]$. Otherwise, we have the following result.

Theorem 6 *Let $P = \{p_1, \ldots, p_n\}$ be a set of n points on the real line and $m < n/2$ a positive integer. Then,*

(a) There exists a set of $n-m$ bicolorings $\mathcal{B}$, for which any G-SUR consisting of intervals has size at least $n-m$.

(b) There exists a set $\mathcal{I}$ of $n-m$ intervals such that for any bicoloring B of P there is at least one balanced interval in $\mathcal{I}$ with respect to B.

Proof. To prove the second part of the theorem consider the set of points

$$P' = \{p_1, \ldots, p_{n-m+1}\},$$

and let $\mathcal{I}$ be the G-SUR given in Theorem 3 for P', which is of size $n-m$. Let B be any m-restricted bicoloring of P. Note that $P \setminus P'$ is of size $m-1$, so by definition it is impossible that all the red or blue points are completely contained in $P \setminus P'$. Then P' contains both blue and red points, so the restriction B' of B to P' is also a valid coloring for P'. We may then take a balanced interval $I \in \mathcal{I}$ with respect to B'. This interval is also balanced with respect to B, so $\mathcal{I}$ is a G-SUR for all the m-restricted bicolorings of P.

Now we prove that $n-m$ intervals are sometimes necessary. We may assume $P = \{1, 2, \ldots, n\}$. We consider the set of m-restricted bicolorings $\mathcal{B} = \{B_1, \ldots, B_{n-m}\}$ defined as follows. For $i = 1, 2, \ldots, n-2m+1$, the bicoloring B_i has the leftmost $m+i-1$ points colored red and the rest blue. For $i = n-2m+2, \ldots, n-m$, the bicoloring B_i has the points in the interval

$$\{i-n+2m, \ldots, i+m-1\}$$

colored red and the remaining m points colored blue. See Figure 2 for an example.

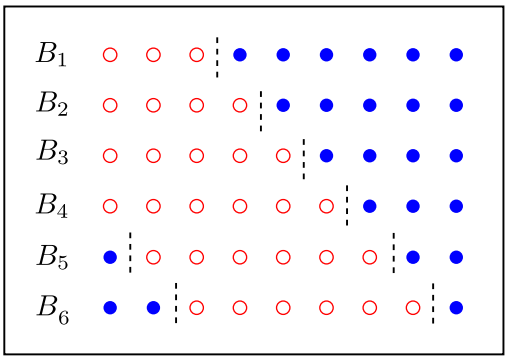


Figure 2: An example of the construction in which $n-m$ intervals are needed in Theorem 6 for $n = 9$ and $m = 3$. Vertical dashed segments indicate where the symmetries must hold.

As in the proof of Theorem 3, we may assume that a G-SUR consists of intervals with integral endpoints. For $i = 1, 2, \ldots, n-2m+1$, an interval balanced with respect to B_i must be symmetric around $\frac{2m+2i-1}{2}$.

For $i \geq n-2m+2$, an interval balanced with respect to B_i cannot have blue points from both the left and right sides, as otherwise it would have $n-m > n/2 > m$ red points, but at most m blue points. So it has to be symmetric either around the $\frac{2i-2n+4m-1}{2}$ or around $\frac{2i+2m-1}{2}$.

Regardless of these final choices, we obtain intervals symmetric around $n-m$ different points, so they must all be different. This finishes the proof.

□

3 Hardness Results

In this section we study the computational aspect of finding minimal size G-SURs for points on the real line using interval ranges. We receive as input a set of n points $P = \{p_1, \ldots, p_n\}$ and a family of bicolorings $\mathcal{B} = \{B_1, \ldots, B_m\}$ of P. We expect as output the size of the minimal G-SUR consisting of interval ranges. We denote this problem as Minimal Interval G-SUR.

Theorem 7 *The* Minimal Interval G-SUR *problem is NP-hard.*

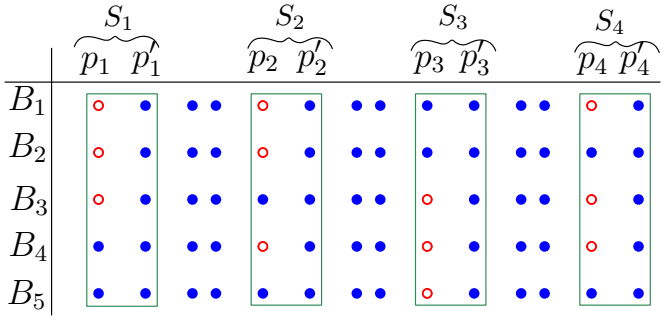


Figure 3: An illustration of the construction used in Theorem 7. In each bicolring B_i, the blue points between two consecutive pairs $\{p_i, p'_i\}$ and $\{p_{i+1}, p'_{i+1}\}$ are dummy points.

Proof. We give a reduction from the Set Cover problem. In the Set Cover problem, we are given a set of elements $X = \{x_1, \ldots, x_n\}$ (called the universe), and a collection $\mathcal{S} = \{S_1, \ldots, S_m\}$ of m subsets, where each $S_i \subseteq X$, and $\bigcup_{1 \leq i \leq m} S_i = X$. The goal is to identify the smallest sub-collection of $\mathcal{S}$ whose union equals the universe.

From an instance $(X, \mathcal{S})$ of the Set Cover problem we create an instance $(\mathcal{B}, P)$ of the Minimal Interval G-SUR problem in the following manner. For every set S_i, we create a pair of consecutive points $\{p_i, p'_i\}$. The points are in the following order $\{\{p_1, p'_1\}, \{p_2, p'_2\}, \ldots, \{p_m, p'_m\}\}$ on the line. Besides for any two consecutive pair of points $\{p_i, p'_i\}$ and $\{p_{i+1}, p'_{i+1}\}$, we introduce two dummy points between them. For every $x_j \in X$ we construct a bicoloring B_j as follows: the points p_i and p'_i are colored red and blue, respectively, if $x_j \in S_i$. Otherwise, the points p_i and p'_i are colored blue. Furthermore, we color all the dummy points blue in any bicoloring. This completes the construction. We illustrate an instance $(\mathcal{B}, P)$ that is reduced from the set system $S_1 = \{x_1, x_2, x_3\}, S_2 = \{x_1, x_2, x_4\}, S_3 = \{x_3, x_4, x_5\}$, $S_4 = \{x_1, x_3, x_4\}$ (see Figure 3).

In the forward direction, we show that if there is a solution of the Set Cover problem of size k then there

is a solution of Minimal Interval G-SUR problem of size k. Assume that the Set Cover problem has a solution $\mathcal{S}^*$ where $|\mathcal{S}^*| \leq k$. We construct a set $\mathcal{I}^*$ of intervals cardinality k as follows. $\mathcal{I}^* = \{(p_i, p_i') | S_i \in \mathcal{S}^*\}$. The claim is $\mathcal{I}^*$ is a solution of Minimal Interval G-SUR problem. Otherwise, there exists a bicoloring $B_i \in \mathcal{B}$ such that there is no balanced interval in $\mathcal{I}^*$ with respect to B_i. However we know there exists a set $S_j \in \mathcal{S}^*$ that contains x_i. Thus the interval (p_j, p_j') is already taken in $\mathcal{I}^*$, and by construction it is balanced with respect to B_i. Hence the claim holds.

Conversely, suppose $\mathcal{I}^*$ is a solution of the Minimal Interval G-SUR of cardinality at most k. Observe that, due to the construction of $(\mathcal{B}, P)$, any interval that is not either of the form (p_i, p_i') or (d, p_i) (where d is a dummy point) is not balanced with respect to any B_i. Hence we may assume that any interval in $\mathcal{I}^*$ has one of these two forms. Next, we construct the following set $\mathcal{S}^* = \{S_j | (p_j, p_j') \in \mathcal{I}^* \text{ or } (d, p_j) \in \mathcal{I}^*\}$. Observe that $|\mathcal{S}^*| \leq k$. We claim $\mathcal{S}^*$ is a set cover for the set system $(X, \mathcal{S})$. If not, there exists an element $x_i \in X$ that is not covered. Consider the corresponding bicoloring B_i. We know there is a balanced interval $I \in \mathcal{I}^*$ with respect to B_i. By construction, I can be either (p_j, p_j') or (d, p_j). Hence we know $S_j \in \mathcal{S}^*$, and $x_i \in S_j$. This contradicts our assumption and the claim holds. □

4 Points in $\mathbb{R}^d$ and Ball G-SURs

Let $P = \{p_1 \ldots, p_n\}$ be a set of points in $\mathbb{R}^d$. Given a family of bicolorings $\mathcal{B}$, here the goal is to find a G-SUR $\mathcal{O}^*$ consisting of Euclidean balls. We show general bounds for the size of $\mathcal{O}^*$.

To give a lower bound, we embed the one-dimensional example from Lemma 3 in a line ℓ of $\mathbb{R}^d$ and note that a ball is balanced if and only if the interval resulting from intersecting the ball with ℓ is balanced. Any ball creates at most one such interval on ℓ, so a G-SUR must have size at least $n - 1$.

We now show that $n - 1$ balls always suffice. For this we consider the *Gabriel graph* $G(P)$ whose vertex set is P and there is an edge (x, y) when the closed ball with diameter on the line segment xy contains no other point of P.

Lemma 8 *The Gabriel graph is connected.*

This result is well-known on the plane (see e.g. [5]). For completeness, here we provide a proof which works in higher dimensions.

Proof. It is enough to show that for any partition $P = Q \cup R$ of the vertices of $G(P)$ there is an edge between a vertex of Q and a vertex of R. Let (q, r) be a pair of closest points $q \in Q$ and $r \in R$, that is, that minimize $d(q, r)$.

If the ball with diameter qr contains another point r' from, say, R, then $d(q, r') < d(q, r)$, a contradiction. Similarly, this ball cannot contain another point from Q. So qr is an edge of $G(P)$, and the proof is complete. □

Lemma 9 *For any set P of n points in $\mathbb{R}^d$ and any family $\mathcal{B}$ of bicolorings of P, there exists a G-SUR consisting of $n - 1$ Euclidean balls.*

Proof. Consider the set of $n - 1$ edges E of a spanning tree T of the Gabriel graph $G(P)$. For every edge $e = (p, q) \in E$, let O_e be the ball with diameter pq. Let $\mathcal{O} = \{O_e | e \in E\}$. We claim that $\mathcal{O}$ is a G-SUR.

For any bicoloring B of P there is at least a red point r and a blue point b. Since T is connected, there is a path on T that connects r to b, and thus there is an edge in this path with endpoints of opposite colors with respect to B. The ball corresponding to this particular edge is balanced, as it only contains r and b. □

Thereby we conclude the following theorem.

Theorem 10 *Let $d \geq 1$ and $n \geq 2$ be positive integers. Then the following hold:*

(a) There exists a set P of n points in $\mathbb{R}^d$ and a family of $n - 1$ bicolorings $\mathcal{B}$ for P, for which any G-SUR consisting of Euclidean balls has size at least $n - 1$.

(b) For any set of n points P in $\mathbb{R}^d$ there exists a set $\mathcal{O}$ of $n - 1$ Euclidean balls such that for any bicoloring B of P there is at least one balanced ball in $\mathcal{O}$ with respect to B.

5 Balanced Covering on Random Points on a Line

In this section we study the properties of balanced intervals for random bicolorings of points on a line. Consider a set $P = \{p_1, p_2, \ldots, p_{n+m}\}$ on a line, pick a subset of P of size m uniformly at random, color these points red. Color the remaining n points blue. Define the random variables:

- $\mathcal{T}_{m,n}$ = the size of the smallest balanced interval.
- $\mathcal{S}_{m,n}$ = the size of the largest balanced interval,

We are interested on the asymptotic behaviour of $\mathcal{T}_{m,n}$ and $\mathcal{S}_{m,n}$ as m and n become large. Due to space constraints, here we focus only in the case in which m is much smaller compared to n. In this situation we have the following result.

Theorem 11 *For $m = o(\sqrt{n})$,*

$$\mathcal{S}_{m,n} = \mathcal{T}_{m,n} = 2,$$

with high probability.[1]

Proof. The equality $\mathcal{T}_{m,n} = 2$ is direct because there are always two consecutive points of different color.

Next, we consider $\mathcal{S}_{m,n}$. Without loss of generality, we may assume that $n > 3(m+2)$. Let E be the event that are at least three blue points between each pair of red points, before the first red point and after the last red point. Note that the event $\mathcal{S}_{m,n} \geq 4$ is impossible if E happens.

We can calculate the probability of E happening using the following argument. Consider *blocks* of colors of type *bbbrbbb*, *rbbb* and *b*. Each situation in which E happens corresponds to placing a *bbbrbbb* block, then $m-1$ blocks *rbbb* to its right, and then distributing the remaining *b*'s in between these blocks. Therefore, the number of situations in which the event happens is

$$\binom{m+n-3(m-1)-6}{m} = \binom{n-2m-3}{m}$$

The probability space has size $\binom{m+n}{m}$. Therefore, by Bernoulli's inequality,

$$\begin{aligned}
\mathbb{P}(E) = \frac{\binom{n-2m-3}{m}}{\binom{m+n}{m}} &= \frac{(n-2m-3)!n!}{(n-3m-3)!(m+n)!} \\
&= \prod_{j=0}^{m-1} \left(1 - \frac{3m+3}{m+n-j}\right) \\
&\geq \left(1 - \frac{3m+3}{n+1}\right)^m \\
&\geq 1 - m\left(\frac{3m+3}{n+1}\right).
\end{aligned}$$

Since $m = o(\sqrt{n})$, the right hand side converges to 1 as n goes to infinity. This means that with high probability the event E happens, and therefore, with high probability $\mathcal{S}_{m,n} = 2$. □

In the discrete model presented above the points are equally spaced. In practical applications this is not always the case. Thus we can also study an analogous problem in the following continuous model which takes into consideration the distance between random samples.

We independently and uniformly sample m points from the interval $[0, 1]$ and color them red, and, similarly, sample n independent and uniform points and color them blue. By symmetry, any of the red/blue discrete orderings are equally probable, and thus they distribute as in the discrete model above. Therefore, as before, $\mathcal{S}_{m,n} = \mathcal{T}_{m,n} = 2$, with high probability. Furthermore, in this case, we can also consider the length of the balanced intervals. More precisely,

- $\mathcal{M}_{m,n}$ = the length of the shortest balanced interval,
- $\mathcal{L}_{m,n}$ = the length of the longest balanced interval.

Once more, suppose that $m = o(\sqrt{n})$. Since $\mathcal{S}_{m,n} = 2$ with high probability, the largest balanced interval must have as endpoints two consecutive points with high probability. Moreover, as n increases, the maximum spacing between consecutive blue points converges to 0 almost surely. These two remarks give a sketch of the proof for the following theorem.

Theorem 12 *For $m = o(\sqrt{n})$, $\mathcal{M}_{m,n}$ and $\mathcal{L}_{m,n}$ converge to 0 almost surely.*

[1] Here we use the usual convention that $X_n = x$ *with high probability* if $\lim_{n\to\infty} \mathbb{P}(X_n = x) = 1$.

References

[1] O. Aichholzer, N. Atienza, J. M. Díaz-Báñez, R. Fabila-Monroy, D. Flores-Peñaloza, P. Pérez-Lantero, B. Vogtenhuber, and J. Urrutia. Computing balanced islands in two colored point sets in the plane. *Information Processing Letters*, 135:28–32, 2018.

[2] N. Balachandran, R. Mathew, T. K. Mishra, and S. P. Pal. Induced-bisecting families of bicolorings for hypergraphs. *Discrete Mathematics*, 341(6):1732–1739, 2018.

[3] S. Bereg, J. M. Díaz-Báñez, R. Fabila-Monroy, P. Pérez-Lantero, A. Ramírez-Vigueras, T. Sakai, J. Urrutia, and I. Ventura. On balanced 4-holes in bichromatic point sets. *Computational Geometry*, 48(3):169–179, 2015.

[4] S. Bereg, F. Hurtado, M. Kano, M. Korman, D. Lara, C. Seara, R. I. Silveira, J. Urrutia, and K. Verbeek. Balanced partitions of 3-colored geometric sets in the plane. *Discrete Applied Mathematics*, 181:21–32, 2015.

[5] M. De Berg, M. Van Kreveld, M. Overmars, and O. Schwarzkopf. Computational geometry. In *Computational geometry*, pages 1–17. Springer, 1997.

[6] C.-Y. Lo, J. Matoušek, and W. Steiger. Algorithms for ham-sandwich cuts. *Discrete & Computational Geometry*, 11(4):433–452, 1994.

Chirotopes of Random Points in Space are Realizable on a Small Integer Grid

Jean Cardinal* Ruy Fabila-Monroy† Carlos Hidalgo-Toscano†

Abstract

We prove that with high probability, a uniform sample of n points in a convex domain in $\mathbb{R}^d$ can be rounded to points on a grid of step size proportional to $1/n^{d+1+\epsilon}$ without changing the underlying chirotope (oriented matroid). Therefore, chirotopes of random point sets can be encoded with $O(n \log n)$ bits. This is in stark contrast to the worst case, where the grid may be forced to have step size $1/2^{2^{\Omega(n)}}$ even for $d = 2$.

This result is a high-dimensional generalization of previous results on order types of random planar point sets due to Fabila-Monroy and Huemer (2017) and Devillers, Duchon, Glisse, and Goaoc (2018).

1 Introduction

Chirotopes, order types, and oriented matroids. Many interesting properties of planar point sets in general position are captured by the combinatorial abstraction consisting of the orientation – clockwise or counterclockwise – of every triple of points. This generalizes naturally to d-dimensional point sets. Consider a set $S \subset \mathbb{R}^d$ of n points in general position (no $d+1$ on a hyperplane). Let us denote by $\Lambda(S, k)$ the set of all ordered k-tuples of distinct points of S. With every ordered $(d+1)$-tuple $P = (p_1, p_2, \ldots, p_{d+1}) \in \Lambda(S, d+1)$ of points we can associate a binary value $\chi(P)$ indicating the orientation of the corresponding simplex. This can be expressed as the sign of a determinant:

$$\chi(P) = \operatorname{sgn} \begin{vmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,d} & 1 \\ p_{2,1} & p_{2,2} & \cdots & p_{2,d} & 1 \\ & & \ddots & & 1 \\ p_{d+1,1} & p_{d+1,2} & \cdots & p_{d+1,d} & 1 \end{vmatrix}.$$

The map χ is referred to as the *chirotope* of the point set S. The values of $\chi(P)$ obey the chirotope axioms, in particular the Grassmann-Plücker relations, and completely characterize the rank-$(d+1)$ *oriented matroid* defined by the point set. Note that not all maps satisfying the chirotope axioms are chirotopes of point sets. The *Topological Representation Theorem*, however, ensures that they always have a representation as a collection of pseudohyperplanes. For $d = 2$, such a representation is a *pseudoline arrangement*. We refer the reader to the classical text from Björner, Las Vergnas, Sturmfels, White, and Ziegler [4] for more background on oriented matroids and chirotopes.

We say that two sets of points have the same *order type* whenever there exists a bijection between them such that the chirotope is preserved. By extracting the purely combinatorial features of a set of points, oriented matroids and order types are useful tools in discrete and computational geometry. In computational geometry, it is the case for instance that the chirotope of a point set is sufficient to compute its convex hull, and therefore provides a simple query-based computation model for this task, in a way that is reminiscent to comparison-based sorting. Knuth explored such a model and several generalizations in his book *Axioms and Hulls* [19]. As for discrete geometry, Eppstein's recent book on forbidden configurations [9] contains a thorough, unified treatment of major results in geometry of planar point sets through the lens of monotone properties of chirotopes. The *Order Type Database* from Aichholzer et al. contains all equivalence classes of chirotopes realized by sets of up to 10 points in the plane [3].

Algebraic universality and bit complexity. It is not clear, however, how much information is contained in a chirotope. For $d = 2$, it is known that a chirotope can be encoded using $O(n^2)$ bits [14, 25, 11, 12]. Recently, it has even been shown that chirotopes induced by sets of n points in $\mathbb{R}^d$ could be stored using $O(n^d (\log \log n)^2 / \log n)$ bits, in such a way that the orientation of every $(d+1)$-tuple can be recovered in $O(\log n / \log \log n)$ time on a word RAM [6].

These representations, however, are distinct from the natural representation of a set of points by d-tuples of coordinates. The reason is that such a representation can have exponential bitsize: for every n there exists a chirotope of a set of n points in the plane, every geometric realization of which requires coordinates that are doubly exponential in n [15]. This is only one consequence of a more general phenomenon, known as *algebraic universality* of rank-three oriented matroids, and

*Université libre de Bruxelles (ULB), Brussels, Belgium. jcardin@ulb.ac.be

†CINVESTAV, CDMX, Mexico. Partially supported by CONACyT grant 253261 ruyfabila@math.cinvestav.mx, cmhidalgo@math.cinvestav.mx

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 734922.

proved by Mnëv [21, 22] and Richter-Gebert [23]. In a nutshell, it states that for any semialgebraic set A, there exists an oriented matroid whose realization space is stably equivalent, in particular homotopic, to A. Algebraic universality holds for other discrete geometric structures such as unit disk graphs [20, 16], 4-dimensional polytopes [24], simplicial polytopes [1], and d-dimensional Delaunay triangulations [2].

It is likely, however, that this worst-case exponential coordinate bitsize is irrelevant for "typical" point sets, or for point sets occurring in applications. It is therefore natural to wonder what is the required coordinate bitsize for chirotopes of *random* point sets. The model we will assume here is that of a uniform distribution on a fixed compact, convex domain.

Related works. The question of random order types has first been tackled by Bokowski, Richter-Gebert, and Schindler [5]. They attribute the question of estimating the probability of an order type to Goodman and Pollack, and investigate it under the assumption of a uniform distribution on the Grassmannian. They discuss the problem of finding an efficient random generator on the Grassmann manifold from a generator for the unit interval. They also perform experiments supporting the conjecture that the maximum probability is reached by the oriented matroid corresponding to the cyclic polytope.

Recently, Fabila-Monroy and Huemer [10] proved that with high probability, a uniform sample of points in the plane can be rounded to a $n^{3+\epsilon} \times n^{3+\epsilon}$ grid without altering its chirotope. Even more recently, Devillers, Duchon, Glisse, and Goaoc [7] investigated the number of bits that need to be read from the coordinates of random points to know their order type, and obtain the same result in a slightly more general setting. They also raise the question of whether uniform samples yield a vanishing fraction of order types. The difficulty of sampling order types uniformly was also discussed by Goaoc, Hubard, de Joannis de Verclos, Sereni, and Volec [13].

Another closely related line of work is that of random alignment and shape distribution of triangles [18, 17]. Probabilistic analyses supporting the idea that near-alignment of points occur naturally in random sets were applied in particular to alignments of quasars [8], and debunking pseudoscientific claims on mysterious alignments between archaeological sites in Great Britain [26]. It is known from these works, for instance, that for a uniform random sample of n points in the unit square, the expected number of triples contained in a slab of width w is proportional to wn^3. Hence unless the width is chosen to be at most proportional to n^{-3}, the sample contains near-aligned points, whose orientation is likely to be flipped by a rounding procedure.

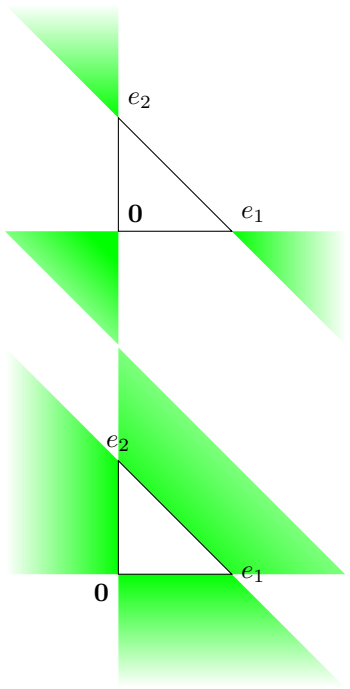


Figure 1: The regions defined in Lemma 1 for $d = 2$. No line can intersect all the three R_i or all the three S_i regions simultaneously.

Our contribution. We generalize the result of Fabila-Monroy and Huemer [10] and Devillers et al. [7] to d-dimensional point sets. We prove that in a uniform sample of n points in $\mathbb{R}^d$, points can be rounded to a $n^{d+1+\epsilon} \times \ldots \times n^{d+1+\epsilon}$ grid without altering their chirotope. We believe that the proof is simpler than the previous ones, even in the case $d = 2$.

2 Chirotopes

We begin with a simple observation on the structure of cells in an arrangement of $d+1$ hyperplanes in $\mathbb{R}^d$.

Lemma 1 *Let $P := \{p_0, \ldots, p_d\}$ be a set of $d+1$ points in general position in $\mathbb{R}^d$. Let $\mathcal{H}$ be the hyperplane arrangement generated by all the hyperplanes passing through d points of P. Let*

- *$R_1, \ldots, R_{d+1}$ be the unbounded cells of $\mathcal{H}$ that do not contain a facet of $\mathrm{conv}(P)$ in their boundary; and*
- *$S_1, \ldots, S_{d+1}$ be the unbounded cells of $\mathcal{H}$ that do contain a facet of $\mathrm{conv}(P)$ in their boundary.*

Then there is no hyperplane that simultaneously intersects all the S_i or all the R_i.

Proof. By doing an affine transformation we may assume that $p_0 = 0$ and p_i is the vector with 1 in its i-th

coordinate and 0 in all other coordinates. Note that the R_i and S_i are defined by

$$\begin{aligned} R_i &:= \{x \in \mathbb{R}^d : x_j < 0 \ \forall j \in [d] \setminus \{i\} \wedge \sum_{j\in[d]} x_j > 1\}, \\ R_{d+1} &:= \{x \in \mathbb{R}^d : x_j < 0 \ \forall j \in [d]\} \end{aligned}$$

and

$$\begin{aligned} S_i &:= \{x \in \mathbb{R}^d : \\ &\quad x_i < 0 \wedge x_j > 0 \ \forall j \in [d] \setminus \{i\} \wedge \sum_{j\in[d]} x_j < 1\}, \\ S_{d+1} &:= \{x \in \mathbb{R}^d : x_j > 0 \ \forall j \in [d] \wedge \sum_{j\in[d]} x_j > 1\} \end{aligned}$$

For $d = 2$, the observation is direct and illustrated on Figure 1. For $d > 2$, we proceed by induction and suppose that the result holds for $d-1$. Consider a hyperplane h intersecting the regions $R_1, R_2, \ldots, R_d$. In both cases, if h intersects R_{d+1}, then it must intersect $R_{d+1} \cap h'$, where h' is one of the hyperplanes of equation $x_j = 0$ for $j \in [d]$. The intersection of the whole arrangement with h' yields a similar situation in dimension $d-1$, for which the statement holds by induction. Therefore, h cannot intersect R_{d+1}. The proof for the S_i is similar. □

We now give a sufficient condition for two order tuples to have the same orientation after a perturbation.

Lemma 2 *Consider two ordered $(d+1)$-tuples $P := (p_1, p_2, \ldots, p_{d+1})$ and $Q := (q_1, q_2, \ldots, q_{d+1})$ of points in $\mathbb{R}^d$. For every $1 \le i \le d+1$, let f_i be the hyperplane passing through the facet of $\mathrm{conv}(P)$ opposite to p_i;*

Suppose that for every $1 \le i \le d+1$ the following two conditions hold.

1) *p_i and q_i are on the same open halfspace bounded by f_i; and*
2) *the distances from q_i to f_i and from p_i to f_i are both larger than the distance from q_j to f_i, for all $j \neq i$.*

Then P and Q have the same orientation.

Proof. By 1) and 2), for every $1 \le i \le d+1$ there exists a hyperplane h_i parallel to f_i that separates p_i and q_i from the other q_j $(j \neq i)$. Let $\mathcal{H}$ be the hyperplane arrangement generated by the h_i. Note that for every $1 \le i \le d+1$, p_i and q_i lie in the same cell of $\mathcal{H}$. Since the hyperplanes h_i are parallel to the facets of $\mathrm{conv}(P)$, $\mathcal{H}$ can be of one of two types, depending on whether the unique bounded cell of $\mathcal{H}$ has the same or the opposite orientation as $\mathrm{conv}(P)$, see Figure 2. Let $C_1, \ldots, C_{d+1}$ be the cells of $\mathcal{H}$ containing p_i and q_i, respectively. Note that the C_i are either the R_i or the

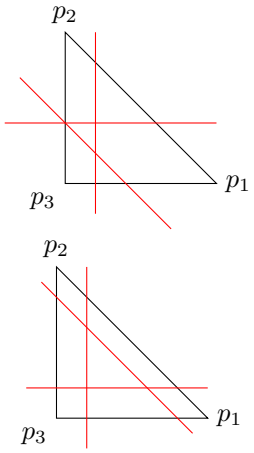


Figure 2: The hyperplanes h_i in Lemma 2.

S_i defined in Lemma 1. In both cases no hyperplane can intersect all the C_i simultaneously. For every $1 \le i \le d+1$, let f_i' be the hyperplane passing through the q_j different from q_i. Note, f_i' intersects all the C_j' with $j \neq i$. Thus, f_i' does not intersect C_i, and p_i and q_i are on the same open halfspace defined by f_i. Therefore, P and Q have the same orientation. □

We now prove our main result by showing that if S is a random point set and S' is obtained by rounding S on a sufficiently dense grid, then the conditions of the lemma hold for every pair composed of a d-simplex in S and its corresponding rounded version in S'.

Theorem 3 *Let S be a uniform sample of n points in the d-dimensional unit ball. Then for every $\epsilon > 0$, with probability at least $1 - O\left(\frac{1}{n^\epsilon}\right)$, the points of S can be rounded to a grid of step size $1/(n^{d+1+\epsilon})$ without changing their chirotope.*

Proof. Let $M := n^{d+1+\epsilon}$. Let S' be the image of S after rounding each point to its nearest neighbor on a grid of step size $1/M$. Consider a $(d+1)$-tuple of points $P := (p_1, \ldots, p_{d+1})$ in S and the corresponding $(d+1)$-tuple of rounded points $Q := (q_1, \ldots, q_{d+1})$ in S'. As in Lemma 2, let f_i be the hyperplane passing through the facet of $\mathrm{conv}(P)$ opposite to p_i. We prove that the conditions of Lemma 2 hold with high probability. By definition, for any given j, the absolute difference between p_j and q_j is at most $\sqrt{d}/M$. Thus the conditions of Lemma 2 hold if the distance from p_i to f_i is at least $2\sqrt{d}/M$. Let B_d be the unit d-dimensional ball. The $d-1$-volume of the intersection of B_d and the hyperplane containing f_i is at most $\mathrm{vol}(B_d - 1)$. Thus, the probability that for a given $1 \le i \le d+1$ the distance from p_i to f_i is less or equal to $2\sqrt{d}/M$ is at most

$(2\sqrt{d}/M)\text{vol}(B_{d-1})/\text{vol}(B_d)$. We have that

$$\begin{aligned}\frac{2\sqrt{d}}{M}\cdot\frac{\text{vol}(B_{d-1})}{\text{vol}(B_d)} &= \frac{2\sqrt{d}}{\sqrt{\pi}M}\cdot\frac{\Gamma(\frac{d}{2}+1)}{\Gamma(\frac{d-1}{2}+1)}\\ &< \frac{2\sqrt{d}}{\sqrt{\pi}M}\cdot\frac{\Gamma(\frac{d}{2}+1)}{\Gamma(\frac{d}{2})}\\ &= \frac{d^{3/2}}{\sqrt{\pi}M}.\end{aligned}$$

We apply the union bound over all such bad events. There are $(d+1)\binom{n}{d+1}$ such events to consider. Thus, the probability that no $(d+1)$-tuple has a different orientation as the corresponding $(d+1)$-tuple in S' is at least

$$1-\binom{n}{d+1}\frac{(d+1)d^{3/2}}{\sqrt{\pi}M} = 1-O\left(\frac{1}{n^{\epsilon}}\right).$$

□

Note that we considered the uniform distribution on the unit ball for convenience. The same analysis holds for any fixed convex body, where the probability of a bad event happening depends on the discrepancies in the distributions of the projections in different directions.

References

[1] Karim A. Adiprasito and Arnau Padrol. The universality theorem for neighborly polytopes. *Combinatorica*, 37(2):129–136, 2017.

[2] Karim A. Adiprasito, Arnau Padrol, and Louis Theran. Universality theorems for inscribed polytopes and Delaunay triangulations. *Discrete & Computational Geometry*, 54(2):412–431, 2015.

[3] Oswin Aichholzer, Franz Aurenhammer, and Hannes Krasser. Enumerating order types for small point sets with applications. *Order*, 19(3):265–281, 2002.

[4] Anders Björner, Michel Las Vergnas, Bernd Sturmfels, Neil White, and Günter Ziegler. *Oriented Matroids*. Number 46 in Encyclopedia of Mathematics and its Applications. Cambridge University Press, 2nd edition, 1999.

[5] Jürgen Bokowski, Jürgen Richter-Gebert, and Werner Schindler. On the distribution of order types. *Comput. Geom.*, 1:127–142, 1991.

[6] Jean Cardinal, Timothy M. Chan, John Iacono, Stefan Langerman, and Aurélien Ooms. Subquadratic encodings for point configurations. In *34th International Symposium on Computational Geometry, SoCG 2018, June 11-14, 2018, Budapest, Hungary*, pages 20:1–20:14, 2018.

[7] Olivier Devillers, Philippe Duchon, Marc Glisse, and Xavier Goaoc. On order types of random point sets. *CoRR*, abs/1812.08525, 2018.

[8] Michael G. Edmunds and Glyn H. George. Random alignment of quasars. *Nature*, 290:481–483, 04 1981.

[9] David Eppstein. *Forbidden Configurations in Discrete Geometry*. Cambridge University Press, 2018.

[10] Ruy Fabila-Monroy and Clemens Huemer. Order types of random point sets can be realized with small integer coordinates. In *XVII Spanish Meeting on Computational Geometry: book of abstracts, Alicante, June 26-28*, pages 73–76, 2017.

[11] Stefan Felsner. On the number of arrangements of pseudolines. *Discrete & Computational Geometry*, 18(3):257–267, 1997.

[12] Stefan Felsner and Pavel Valtr. Coding and counting arrangements of pseudolines. *Discrete & Computational Geometry*, 46(3):405–416, 2011.

[13] Xavier Goaoc, Alfredo Hubard, Rémi de Joannis de Verclos, Jean-Sébastien Sereni, and Jan Volec. Limits of order types. In *31st International Symposium on Computational Geometry, SoCG 2015, June 22-25, 2015, Eindhoven, The Netherlands*, pages 300–314, 2015.

[14] Jacob E. Goodman and Richard Pollack. Multidimensional sorting. *SIAM J. Comput.*, 12(3):484–507, 1983.

[15] Jacob E. Goodman, Richard Pollack, and Bernd Sturmfels. Coordinate representation of order types requires exponential storage. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing, May 14-17, 1989, Seattle, Washington, USA*, pages 405–410, 1989.

[16] Ross J. Kang and Tobias Müller. Sphere and dot product representations of graphs. *Discrete & Computational Geometry*, 47(3):548–568, 2012.

[17] David G. Kendall. Exact distributions for shapes of random triangles in convex sets. *Advances in Applied Probability*, 17(2):308–329, 1985.

[18] David G. Kendall and Wilfrid S. Kendall. Alignments in two-dimensional random sets of points. *Advances in Applied Probability*, 12(2):380–424, 1980.

[19] Donald E. Knuth. *Axioms and Hulls*, volume 606 of *Lecture Notes in Computer Science*. Springer, 1992.

[20] Colin McDiarmid and Tobias Müller. Integer realizations of disk and segment graphs. *Journal of Combinatorial Theory, Series B*, 103(1):114–143, 2013.

[21] Nicolai E. Mnëv. Varieties of combinatorial types of projective configurations and convex polyhedra. *Dokl. Akad. Nauk SSSR*, 283(6):1312–1314, 1985.

[22] Nicolai E. Mnëv. The universality theorems on the classification problem of configuration varieties and convex polytopes varieties. In *Topology and geometry – Rohlin seminar*, pages 527–543. Springer, 1988.

[23] Jürgen Richter-Gebert. Mnëv's universality theorem revisited. In *Proceedings of the Séminaire Lotharingien de Combinatoire*, pages 211–225, 1995.

[24] Jürgen Richter-Gebert. *Realization Spaces of Polytopes*. Lecture Notes in Mathematics. Springer Berlin Heidelberg, 1996.

[25] Ileana Streinu. Clusters of stars. In *Proceedings of the Thirteenth Annual Symposium on Computational Geometry, Nice, France, June 4-6, 1997*, pages 439–441, 1997.

[26] Tom Williamson and Liz Bellamy. *Ley lines in question*. Tadworth, Surrey : World's Work, 1983.

Coordinatizing Data With Lens Spaces and Persistent Cohomology

Luis Polanco* Jose A. Perea †

Abstract

We introduce here a framework to construct coordinates in *finite* Lens spaces for data with nontrivial 1-dimensional $\mathbb{Z}_q$ persistent cohomology, for $q > 2$ prime. Said coordinates are defined on an open neighborhood of the data, yet constructed with only a small subset of landmarks. We also introduce a dimensionality reduction scheme in $S^{2n-1}/\mathbb{Z}_q$ (Lens-PCA: LPCA), and demonstrate the efficacy of the pipeline $\mathbb{Z}_q$-persistent cohomology $\Rightarrow S^{2n-1}/\mathbb{Z}_q$ coordinates $\Rightarrow$ LPCA, for nonlinear (topological) dimensionality reduction.

1 Introduction

One of the main questions in Topological Data Analysis (TDA) is how to use topological signatures like persistent (co)homology [11] to infer spaces parametrizing a given data set [3, 1, 4]. This is relevant in nonlinear dimensionality reduction since the presence of nontrivial topology—e.g., loops, voids, non-orientability, torsion, etc—can prevent accurate descriptions with low-dimensional Euclidean coordinates.

Here we seek to address this problem motivated by two facts. The first: If G is a topological abelian group, then one can associate to it a contractible space, EG, equipped with a free right G-action. For instance, if $G = \mathbb{Z}$, then $\mathbb{R}$ is a model for $E\mathbb{Z}$, with right $\mathbb{Z}$-action $\mathbb{R} \times \mathbb{Z} \ni (r, n) \mapsto r + n \in \mathbb{R}$. The quotient $BG := EG/G$ is called the classifying space of G [8]. In particular $B\mathbb{Z} \simeq S^1$, $B\mathbb{Z}_2 \simeq \mathbb{R}\mathbf{P}^\infty$, $BS^1 \simeq \mathbb{C}\mathbf{P}^\infty$ and $B\mathbb{Z}_q \simeq S^\infty/\mathbb{Z}_q$; here $\simeq$ denotes homotopy equivalence. The second fact: If B is a topological space and $\mathscr{C}_G$ is the sheaf over B (defined in [9]) sending each $U \subset B$ open to the abelian group of continuous maps from U to G, then $\check{H}^1(B; \mathscr{C}_G)$—the first Čech cohomology group of B with coefficients in $\mathscr{C}_G$—is in bijective correspondence with $[B\,,\, BG]$—the set of homotopy classes of continuous maps from B to the classifying space BG. This bijection is a manifestation of the Brown representability theorem [2], and implies, in so many words, that Čech cohomology classes can be represented as coordinates with values in a classifying space (like S^1 or $S^\infty/\mathbb{Z}_q$).

For point cloud data—i.e., for a finite subset X of an ambient metric space (M, d)—one does not compute Čech cohomology, but rather *persistent cohomology*. Specifically, the persistent cohomology of the Rips filtration on the data set X (or a subset of landmarks L). The first main result of this paper contends that steps one through three below mimic the bijection $\check{H}^1(B; \mathscr{C}_{\mathbb{Z}_q}) \cong [B, S^\infty/\mathbb{Z}_q]$ for $B \subset M$ an open neighborhood of X:

1. Let (M, d) be a metric space and let $L \subset X \subset M$ be finite. X is the data and L is a set of landmarks.
2. For a prime $q > 2$ compute $PH^1(\mathcal{R}(L); \mathbb{Z}_q)$; the 1-dim $\mathbb{Z}_q$-persistent cohomology of the Rips filtration on L. If the corresponding persistence diagram $\mathsf{dgm}(L)$ has an element (a, b) so that $2a < b$, then let $a \leq \epsilon < b/2$ and choose a representative cocycle $\eta \in Z^1(R_{2\epsilon}(L); \mathbb{Z}_q)$ whose cohomology class has (a, b) as birth-death pair.
3. Let $B_\epsilon(l)$ be the open ball in M of radius ϵ centered at $l \in L = \{l_1, \ldots, l_n\}$, and let $\varphi = \{\varphi_l\}_{l \in L}$ be a partition of unity subordinated to $\mathcal{B} = \{B_\epsilon(l)\}_{l \in L}$. If $\zeta_q \neq 1$ is a q-th root of unity, then the cocycle η yields a map $f : \bigcup \mathcal{B} \longrightarrow L_q^n$ to the Lens space $L_q^n = S^{2n-1}/\mathbb{Z}_q$, given in homogeneous coordinates by the formula

$$B_\epsilon(\ell_j) \ni b\ ,\ f(b) = \left[\sqrt{\varphi_1(b)}\zeta_q^{\eta_{j1}} : \cdots : \sqrt{\varphi_n(b)}\zeta_q^{\eta_{jn}}\right]$$

where $\eta_{jk} \in \mathbb{Z}_q$ is the value of the cocycle η on the edge $\{l_j, l_k\} \in R_{2\epsilon}(L)$.

If $X \subset \bigcup \mathcal{B}$, then $f(X) = Y \subset L_q^n$ is the representation of the data—in a potentially high dimensional Lens space—corresponding to the cocycle η. The second contribution of this paper is a dimensionality reduction procedure in L_q^n akin to Principal Component Analysis, called LPCA. This allows us to produce from Y, a family of point clouds $P_k(Y) \subset L_q^k$, $1 \leq k \leq n$, $P_n(Y) = Y$, minimizing an appropriate notion of distortion. These are the Lens coordinates of X induced by the cocycle η.

This work, combined with [10, 12], should be seen as one of the final steps in completing the program of using the classifying space BG, for G abelian and finitely generated, to produce coordinates for data with nontrivial underlying 1^{st} cohomology. Indeed, this follows from the fact that $B(G \oplus G') \simeq BG \times BG'$, and that if G is finitely generated and abelian, then it is ismorphic to $\mathbb{Z}^n \oplus \mathbb{Z}_{n_1} \oplus \cdots \oplus \mathbb{Z}_{n_r}$ for unique integers $n, n_1, \ldots, n_r \geq 0$.

*Department of Computational Mathematics, Science and Engineering, Michigan State University, polanco2@msu.edu

†Department of Computational Mathematics, Science and Engineering, Department of Mathematics, Michigan State University, joperea@msu.edu

2 Preliminaries

2.1 Persistent Cohomology

A family $\mathcal{K} = \{K_\alpha\}_{\alpha\in\mathbb{R}}$ of simplicial complexes is called a filtration if $K_\alpha \subset K_{\alpha'}$ whenever $\alpha \leq \alpha'$. If $\mathbb{F}$ is a field and $i \geq 0$ is an integer, then the direct sum $PH^i(\mathcal{K};\mathbb{F}) := \bigoplus_\alpha H^i(K_\alpha;\mathbb{F})$ of cohomology groups is called the i-th dimensional **$\mathbb{F}$-persistent cohomology** of $\mathcal{K}$. A theorem of Crawley-Boevey [5] contends that if $H^i(K_\alpha;\mathbb{F})$ is finite dimensional for each α, then the isomorphism type of $PH^i(\mathcal{K};\mathbb{F})$—as a persistence module—is uniquely determined by a multiset (i.e., a set whose elements may appear with repetitions)

$$\mathsf{dgm} \subset \{(\alpha,\alpha') \in [-\infty,\infty]^2 : \alpha \leq \alpha'\}$$

called the **persistence diagram** of $PH^i(\mathcal{K};\mathbb{F})$. Pairs (α,α') with large persistence $\alpha' - \alpha$, are indicative of stable topological features throughout the filtration $\mathcal{K}$.

Persistent cohomology is used in TDA to quantify the topology underlying a data set. There are two widely used filtrations associated to a subset X of a metric space (M,d), the **Rips filtration** $\mathcal{R}(X) = \{R_\alpha(X)\}_\alpha$ and the **Čech filtration** $\check{\mathcal{C}}(X) = \{\check{C}_\alpha(X)\}_\alpha$. Specifically, $R_\alpha(X)$ is the set of nonempty finite subsets of X with diameter less than α, and $\check{C}_\alpha(X)$ is the nerve of the collection $\mathcal{B}_\alpha$ of open balls $B_\alpha(x) \subset M$ of radius α, centered at $x \in X$. In other words, $\check{C}_\alpha(X) = \mathcal{N}(\mathcal{B}_\alpha)$. Generally $\mathcal{R}(X)$ is more easily computable, but $\check{\mathcal{C}}(X)$ has better theoretical properties (e.g., the Nerve theorem [6, 4G.3]). Their relative weaknesses are ameliorated by noticing that

$$R_\alpha(X) \subset \mathcal{N}(\mathcal{B}_\alpha) \subset R_{2\alpha}(X)$$

for all α, and using both filtrations in analyses: Rips for computations, and Čech for theoretical inference.

2.2 Lens Spaces

Let $q \in \mathbb{N}$ and let $\zeta_q \in \mathbb{C}$ be a primary q-th root of unity. Fix $n \in \mathbb{N}$ and let $q_1,\ldots,q_n \in \mathbb{N}$ be relatively prime to q. We define the **Lens space** $L^n_q(q_1,\ldots,q_n)$ as the quotient of $S^{2n-1} \subset \mathbb{C}^n$ by the $\mathbb{Z}_q$ right action

$$[z_1,\ldots,z_n]\cdot g := \left[z_1\zeta_q^{q_1 g},\ldots,z_n\zeta_q^{q_n g}\right]$$

with simplified notation $L^n_q := L^n_q(1,\ldots,1)$. Notice that when $q = 2$ and $q_1 = \cdots = q_n = 1$, then the right action described above is the antipodal map of S^{2n-1}, and therefore $L^n_2 = \mathbb{R}\mathbf{P}^{2n-1}$. Similarly, the infinite Lens space $L^\infty_q = L^\infty_q(1,1,\ldots)$ is defined as the quotient of the infinite unit sphere $S^\infty \subset \mathbb{C}^\infty$, by the action of $\mathbb{Z}_q$ induced by scalar-vector multiplication by powers of ζ_q.

2.2.1 A Fundamental domain for $L^2_q(1,p)$

In what follows we describe a convenient model for both $L^2_q(1,p)$ and a fundamental domain thereof. This model will allow us to provide visualizations in Lens spaces towards the end of the paper. Let D^3 be the set of points $\mathbf{x} \in \mathbb{R}^3$ with $\|\mathbf{x}\| \leq 1$, and let D_+ (D_-) be the upper (lower) hemisphere of ∂D^3, including the equator. Let $r_{p/q} : D_+ \longrightarrow D_+$ be counterclockwise rotation by $2\pi p/q$ radians around the z-axis, and let $\rho : D_+ \longrightarrow D_-$ be the reflection $\rho(x,y,z) = (x,y,-z)$. Then, $L^2_q(1,p)$ is homeomorphic to $D^3/\sim$, where $\mathbf{x} \sim \mathbf{y}$ if and only if $\mathbf{x} \in D_+$ and $\mathbf{y} = \rho \circ r_{p/q}(\mathbf{x})$.

2.3 Principal Bundles

Let B be a topological space with base point $b_0 \in B$. One of the most transparent methods for producing an explicit bijection between $\check{H}^1(B;\mathscr{C}_{\mathbb{Z}_q})$ and $[B, L^\infty_q]$ is via the theory of Principal bundles. We present a terse introduction here, but direct the interested reader to [7] for details. A continuous map $\pi : P \longrightarrow B$ is said to be a **fiber bundle** with fiber $F = \pi^{-1}(b_0)$ and total space P, if π is surjective, and every $b \in B$ has an open neighborhood $U \subset B$ as well as a homeomorphism $\rho_U : U \times F \longrightarrow \pi^{-1}(U)$, so that $\pi \circ \rho_U(x,e) = x$ for every $(x,e) \in U \times F$.

Let $(G,+)$ be an abelian topological group. A fiber bundle $\pi : P \longrightarrow B$ is said to be a **principal G-bundle** over B, if P comes equipped with a free right G-action $P \times G \ni (e,g) \mapsto e\cdot g \in P$ which is transitive in $\pi^{-1}(b)$ for every $b \in B$. Moreover, two principal G-bundles $\pi : P \longrightarrow B$ and $\pi' : P' \longrightarrow B$ are isomorphic, if there exits a homeomorphism $\Phi : P \longrightarrow P'$, with $\pi' \circ \Phi = \pi$ and so that $\Phi(e\cdot g) = \Phi(e)\cdot g$ for all $(e,g) \in P\times G$. Given an open cover $\mathcal{U} = \{U_j\}_{j\in J}$ of B, a **Čech cocycle**

$$\eta = \{\eta_{jk}\} \in \check{Z}^1(\mathcal{U};\mathscr{C}_G)$$

is a collection of continuous maps $\eta_{jk} : U_j \cap U_k \longrightarrow G$ so that $\eta_{jk}(b) + \eta_{kl}(b) = \eta_{jl}(b)$ for every $b \in U_j \cap U_k \cap U_l$. Given such a cocycle, one can construct a principal G-bundle over B with total space

$$P_\eta = \left(\bigcup_{j\in J} U_j \times \{j\} \times G\right)/\sim$$

where $(b,j,g) \sim (b,k,g+\eta_{jk}(b))$ for every $b \in U_j \cap U_k$, and $\pi : P_\eta \longrightarrow B$ sends the class of (b,j,g) to $b \in B$.

Theorem 1 *If $\mathsf{Prin}_G(B)$ denotes the set of isomorphism classes of principal G-bundles over B, then*

$$\begin{array}{ccc} \check{H}^1(B;\mathscr{C}_G) & \longrightarrow & \mathsf{Prin}_G(B) \\ [\eta] & \mapsto & [P_\eta] \end{array}$$

is a bijection.

Proof. See 2.4 and 2.5 in [10] □

Now, let us see describe the relation between principal G-bundles over B, and maps from B to the classifying space BG. Indeed, let $\jmath : EG \longrightarrow BG = EG/G$ be the quotient map. Given $h : B \longrightarrow BG$ continuous, the pullback h^*EG is the principal G-bundle over B with total space $\{(b, e) \in B \times EG : h(b) = \jmath(e)\}$, and projection map $(b, e) \mapsto b$. Moreover,

Theorem 2 *Let $[B, BG]$ denote the set of homotopy class of maps from B to the classifying space BG. Then, the function*

$$\begin{array}{ccc} [B, BG] & \longrightarrow & \mathsf{Prin}_G(B) \\ [h] & \mapsto & [h^*EG] \end{array}$$

is a bijection.

Proof. See [7], Chapter 4: Theorems 12.2 and 12.4. □

In summary, given a principal G-bundle $\pi : P \longrightarrow B$, or its corresponding Čech cocycle η, there exists a continuous map $h : B \longrightarrow BG$ so that h^*EG is isomorphic to (π, P), and the choice of h is unique up to homotopy. Any such choice is called a classifying map for $\pi : P \longrightarrow B$.

3 Main Theorem: Explicit Classifying Maps for L_q^∞

The goal of this section is to show how one can go from a singular cocycle $\eta \in Z^1(\mathcal{N}(\mathcal{U}); \mathbb{Z}_q)$ to an explicit map $f : \bigcup \mathcal{U} \longrightarrow L_q^\infty$. All proofs are included in the Apendix. Let $J = \{1, \ldots, n\}$, let $\mathcal{U} = \{U_j\}_{j\in J}$ be an open cover for B, and let $\{\varphi_j\}_{j\in J}$ be a partition of unity dominated by $\mathcal{U}$. If $\eta = Z^1(\mathcal{N}(\mathcal{U}); \mathbb{Z}_q)$ and ζ_q is a primitive q-th root of unity, let $f_j : U_j \times \{j\} \times \mathbb{Z}_q \longrightarrow S^{2n-1} \subset \mathbb{C}^n$ be

$$f_j(b, j, g) = \left[\sqrt{\varphi_1(b)}\zeta_q^{(g+\eta_{j1})}, \ldots, \sqrt{\varphi_n(b)}\zeta_q^{(g+\eta_{jn})}\right]$$

If $b \in U_j \cap U_k$, then $f_j(b, j, g) = f_k(b, k, g + \eta_{jk})$ and we get an induced map $\Phi : P_\eta \longrightarrow S^{2n-1} \subset S^\infty$ taking the class of (b, j, g) in the quotient P_η to $f_j(b, j, g)$.

Proposition 3 *Φ is well defined and $\mathbb{Z}_q$-equivariant.*

Let $p : S^{2n-1} \longrightarrow L_q^n$ be the quiotient map. Since $\Phi : P_\eta \longrightarrow S^{2n-1} \subset S^\infty$ is $\mathbb{Z}_q$-equivariant, it induces a map $f : B \longrightarrow L_q^n \subset L_q^\infty$ such that $p \circ \Phi = f \circ \pi$. By construction of $\pi : P_\eta \longrightarrow B$, $f(\pi([b, j, g])) = f(b)$ for any $g \in \mathbb{Z}_q$. In particular for $0 \in \mathbb{Z}_q$

$$U_j \ni b \ , \quad f(b) = \left[\sqrt{\varphi_1(b)}\zeta_q^{\eta_{j1}} : \cdots : \sqrt{\varphi_n(b)}\zeta_q^{\eta_{jn}}\right] \quad (1)$$

Remark 4 *The notation $[a_1 : \cdots : a_n]$ corresponds to homogeneous coordinates in $S^{2n-1}/\mathbb{Z}_q$. In other words, $[a_1 : \cdots : a_n] = \{[a_1 \cdot \alpha, \ldots, a_n \cdot \alpha] \in S^{2n-1} : \alpha \in \mathbb{Z}_q\}$.*

Theorem 5 *The map f classifies the $\mathbb{Z}_q$-principal bundle P_η associated to the cocycle $\eta \in Z^1(\mathcal{N}(\mathcal{U}); \mathbb{Z}_q)$.*

4 Lens coordinates for data

Let (M, d) be a metric space and let $L \subset M$ be a finite subset. We will use the following notation from now on: $B_\epsilon(l) = \{y \in M : d(y, l) < \epsilon\}$, $\mathcal{B}_\epsilon = \{B_\epsilon(l)\}_{l\in L}$, and $L^\epsilon = \bigcup \mathcal{B}_\epsilon$. Given a data set $X \subset M$, our goal will be to choose $L \subset X$, a suitable ϵ such that $X \subset L^\epsilon$, and a cocycle $\eta \in Z^1(\mathcal{N}(\mathcal{B}_\epsilon); \mathbb{Z}_q)$. Equation (1) yields a map $f : L^\epsilon \to L_q^\infty$ defined for every point in X, but constructed from a much smaller subset of landmarks. Next we describe the details of this construction.

4.1 Landmark selection

We select the landmark set $L \subset X$ either at random or through `maxmin` sampling. The latter proceeds inductively as follows: Fix $n \leq |X|$, and let $l_1 \in X$ be chosen at random. Given $l_1, \ldots, l_j \in X$ for $j < n$, we let $l_{j+1} = \underset{x\in X}{\operatorname{argmax}} \min\{d(x, l_1), \ldots, d(x, l_j)\}$.

4.2 A Partition of Unity subordinated to $\mathcal{B}_\epsilon$

Defining f requires a partition of unity subordinated to $\mathcal{B}_\epsilon$. Since $\mathcal{B}_\epsilon$ is an open cover composed of metric balls, then we can provide an explicit construction. Indeed, for $r \in \mathbb{R}$ let $|r|_+ := \max\{r, 0\}$, then

$$\varphi_l(x) := |\epsilon - d(x, l)|_+ \Big/ \sum_{l'\in L} |\epsilon - d(x, l')|_+ \quad (2)$$

is a partition of unity subordinated to $\mathcal{B}_\epsilon$.

4.3 From Rips to Čech to Rips

As we alluded to in the introduction, a persistent cohomology calculation is an appropriate vehicle to select a scale ϵ and a candidate cocycle η. That said, determining $\eta \in Z^1(\mathcal{N}(\mathcal{B}_\epsilon), \mathbb{Z}_q)$ would require computing $\mathcal{N}(\mathcal{B}_\epsilon)$ for all ϵ, which in general is an expensive procedure. Instead we will use the homomorphisms

$$H^1(\mathcal{R}_{2\epsilon}(L)) \xrightarrow{i^*} H^1(\mathcal{N}(\mathcal{B}_\epsilon)) \longrightarrow H^1(\mathcal{R}_\epsilon(L)), \qquad \iota$$

induced by the appropriate inclusions. Indeed, let $\tilde{\eta} \in Z^1(\mathcal{R}_{2\epsilon}(L); \mathbb{Z}_q)$ be such that $[\tilde{\eta}] \notin \ker(\iota)$. This is where we use the persistent cohomology of $\mathcal{R}(L)$. Since the previous diagram commutes, then $[\tilde{\eta}] \notin \ker(i^*)$, so $i^*([\tilde{\eta}]) \neq 0$ in $H^1(\mathcal{N}(\mathcal{B}_\epsilon); \mathbb{Z}_q)$. We will let $[\eta] = i^*([\tilde{\eta}])$ be the class that we use in Theorem 5. However,

Proposition 6 *If $b \in \mathcal{B}_\epsilon(l_j)$ and $1 \leq k \leq n$, then*

$$\sqrt{\varphi_k(b)}\zeta_q^{\eta_{jk}} = \sqrt{\varphi_k(b)}\zeta_q^{\tilde{\eta}_{jk}}.$$

That is, we can compute Lens coordinates using only the Rips filtration on the landmark set.

5 Dimensionality Reduction in L_q^n via Principal Lens Components

Equation (1) gives an explicit formula for the classifying map $f : B \longrightarrow L_q^n$. By construction, the dimension of L_q^n depends on the number n of landmarks selected, which in general can be large. The main goal of this section is to construct a dimensionality reduction procedure in L_q^n to address this shortcoming. To this end, we define the distance $d_L : L_q^n \times L_q^n \longrightarrow [0, \infty)$ as

$$d_L([x],[y]) := d_H(x \cdot \mathbb{Z}_q \,,\, y \cdot \mathbb{Z}_q) \tag{3}$$

where d_H id the Hausdorff distance for subsets of S^{2n-1}.

We will now describe a notion of **projection in L_q^n** onto lower-dimensional Lens spaces. Indeed, let $u \in S^{2n-1}$. Since $\zeta_q^k w \in \mathsf{span}_{\mathbb{C}}(u)^{\perp}$ for any $k \in \mathbb{Z}_q$ and $w \in \mathsf{span}_{\mathbb{C}}(u)^{\perp}$, then

$$L_q^{n-1}(u) := (\mathsf{span}_{\mathbb{C}}(u)^{\perp} \cap S^{2n-1})/\mathbb{Z}_q$$

is isometric to L_q^{n-1}. Let $P_u^{\perp}(v) = v - \langle v, u\rangle_{\mathbb{C}} u$ for $v \in \mathbb{C}^n$, and if $v \notin \mathsf{span}_{\mathbb{C}}(u)$, then we let

$$\mathcal{P}_u([v]) := \left[P_u^{\perp}(v)/\|P_u^{\perp}(v)\|\right] \in L_q^{n-1}(u)$$

It readily follows that $\mathcal{P}_u$ is well defined, and that

Lemma 7 *For $u \in S^{2n-1}$ and $v \notin \mathsf{span}_{\mathbb{C}}(u)$, we have*

$$d_L([v], \mathcal{P}_u([v])) = d\left(v \,,\, P_u^{\perp}(v)/\|P_u^{\perp}(v)\|\right)$$

where d is the distance on S^{2n-1}. Furthermore, $\mathcal{P}_u([v])$ is the point in $L_q^{n-1}(u)$ closest to $[v]$ with respect to d_L.

This last result suggests that a PCA-like approach is possible for dimensionality reduction in Lens spaces. Specifically, for $Y = \{[y_1], \ldots, [y_N]\} \subset L_q^n$, the goal is to find $u \in S^{2n-1}$ such that $L_q^{n-1}(u)$ is the best $(n-1)$-Lens space approximation to Y, then project Y onto $L_q^{n-1}(u)$ using $\mathcal{P}_u$, and repeat the process iteratively reducing the dimension by 1 each time. At each stage, the appropriate constrained optimization problem is

$$\begin{aligned} u^* &= \operatorname*{argmin}_{u\in\mathbb{C}^n, \|u\|=1} \sum_{j=1}^{N} d_L([y_j], \mathcal{P}_u([y_i]))^2 \\ &= \operatorname*{argmin}_{u\in\mathbb{C}^n, \|u\|=1} \sum_{j=1}^{N} \left(\frac{\pi}{2} - \arccos(|\langle y_i, u\rangle|)\right)^2 \end{aligned}$$

which can be linearized using the Taylor series expansion of $\arccos(\theta)$ around 0. Indeed, $|\frac{\pi}{2} - \arccos(\theta)| \approx |\theta|$ to third order, and thus

$$u^* \approx \operatorname*{argmin}_{u\in\mathbb{C}^n, \|u\|=1} \sum_{j=1}^{N} |\langle y_i, u\rangle|^2.$$

This approximation is a linear least square problem whose solution is given by the eigenvector corresponding to the smallest eigenvalue of the covariance matrix

$$\mathsf{Cov}\,(y_1, \ldots, y_N) = \begin{bmatrix} | & & | \\ y_1 & \cdots & y_N \\ | & & | \end{bmatrix} \begin{bmatrix} - & \overline{y}_1 & - \\ & \vdots & \\ - & \overline{y}_N & - \end{bmatrix}.$$

Moreover, for any $\alpha_1, \ldots, \alpha_N \in S^1 \subset \mathbb{C}$ we have that $\mathsf{Cov}\,(\alpha_1 y_1, \ldots, \alpha_N y_N) = \mathsf{Cov}\,(y_1, \ldots, y_N)$, so $\mathsf{Cov}(Y)$ is well defined for $Y \subset L_q^n$.

5.1 Inductive construction of LPCA

Let $v_n = \mathsf{LastLensComp}(Y)$ be the eigenvector of $\mathsf{Cov}(Y)$ corresponding to the smallest eigenvalue. Assume that we have constructed $v_{k+1}, \ldots, v_n \in S^{2n-1}$ for $1 < k < n$, and let $\{u_1, \ldots, u_k\}$ be an orthonormal basis for $\mathsf{span}_{\mathbb{C}}(v_{k+1}, \ldots, v_n)^{\perp}$. Let $U_k \in \mathbb{C}^{n\times k}$ be the matrix with columns $u_1, \ldots, u_k$, and let $U_k^{\dagger}$ be its conjugate transpose. We define the **k-th Lens Principal component** of Y as the vector

$$v_k \;:=\; U_k \cdot \mathsf{LastLensComp}\left(\frac{U_k^{\dagger} y_1}{\|U_k^{\dagger} y_1\|}, \ldots, \frac{U_k^{\dagger} y_N}{\|U_k^{\dagger} y_N\|}\right)$$

This inductive procedure yields a collection $[v_2], \ldots, [v_n] \in L_q^n$, and we let $v_1 \in S^{2n-1}$ be such that $\mathsf{span}_{\mathbb{C}}\{v_1\} = \mathsf{span}_{\mathbb{C}}\{v_2, \ldots, v_n\}^{\perp}$. Finally

$$\mathsf{LPCA}(Y) := \{[v_1], \ldots, [v_n]\}$$

are the **Lens Principal Components** of Y. Let $V_k \in \mathbb{C}^{n\times k}$ be the n-by-k matrix with columns $v_1, \ldots, v_k$, and let $P_k(Y) \subset L_q^k$ be the set of classes $\left[\frac{V_k^{\dagger} y_j}{\|V_k^{\dagger} y_j\|}\right]$, $1 \le j \le N$. The point clouds $P_k(Y)$, $k = 1, \ldots, n$, are the **Lens Principal Coordinates** of Y.

5.2 Choosing a target dimension.

The **variance recovered** by the first k Lens Principal Components $[v_1], \ldots, [v_k] \in L_q^n$ is defined as

$$\mathsf{var}_k(Y) := \frac{1}{N} \sum_{l=2}^{k} \sum_{j=1}^{N} d_L\left(\left[\frac{V_l^{\dagger} y_j}{\|V_l^{\dagger} y_j\|}\right], L_q^{l-1}(e_{l-1})\right)^2$$

where V_l is the n-by-l matrix with columns $v_1, \ldots, v_l$, $1 < l \le k$, and $e_{l-1} \in \mathbb{C}^l$ is the vector $[0, \ldots, 0, 1, 0]$.

Therefore, the **percentage of cumulative variance** $p.\mathsf{var}(k) := \mathsf{var}_k(Y)/\mathsf{var}_n(Y)$, can be interpreted as the portion of total variance of Y along $\mathsf{LPCA}(Y)$, explained by the first k components.

Thus we can select the target dimension as the smallest k for which $p.\mathsf{var}_k(Y)$ is greater than a predetermined value. In other words, we select the dimension that recovers a significant portion of the total variance. Another possible guideline to choose the target dimension is as the minimum value of k for which $p.\mathsf{var}(k) - p.\mathsf{var}(k+1) < \gamma$ for a small $\gamma > 0$.

5.3 Independence of the cocycle representative.

Let $\eta \in Z^1(\mathcal{N}(\mathcal{B}_\epsilon); \mathbb{Z}_q)$ be such that $[\eta] \neq 0$ in $H^1(\mathcal{N}(\mathcal{B}_\epsilon); \mathbb{Z}_q)$, and let $\eta' = \eta + \delta^0(\alpha)$ with $\alpha \in C^0(\mathcal{N}(\mathcal{B}_\epsilon); \mathbb{Z}_q)$. If $b \in U_j$, then

$$f_{\eta'}(b) = [\sqrt{\phi_1(b)}\zeta_q^{\eta_{j1}+\alpha_1} : \cdots : \sqrt{\phi_n(b)}\zeta_q^{\eta_{jn}+\alpha_n}]$$

If Z_α is the square diagonal matrix with entries $\zeta_q^{\alpha_1}, \zeta_q^{\alpha_2}, \ldots, \zeta_q^{\alpha_n}$, then $f_{\eta'}(b) = Z_\alpha \cdot f(b)$. Moreover, after taking classes in L_q^n, this implies that $f_{\eta'}(X) = Z_\alpha \cdot f(X)$. Since $\mathsf{Cov}(Z_\alpha \cdot f(X)) = Z_\alpha \mathsf{Cov}(f(X)) Z_\alpha^\dagger$ and Z_α is orthonormal, then if v is an eigenvector of $\mathsf{Cov}(f(X))$ with eigenvalue σ, we also have that $Z_\alpha v$ is an eigenvector of $\mathsf{Cov}(Z_\alpha \cdot f(X))$ with the same eigenvalue. Therefore

$$\mathsf{LastLensComp}(f_{\eta'}(X)) = Z_\alpha \mathsf{LastLensComp}(f(X)).$$

Since each component in LPCA is obtained in the same manner, we have that $\mathsf{LPCA}(f_{\eta'}(X)) = Z_\alpha \mathsf{LPCA}(f(X))$. Thus, the lens coordinates from two cohomologous cocycles η and $\eta + \delta^0(\alpha)$ (i.e., representing the same cohomology class) only differ by the isometry of L_q^n induced by the linear map Z_α.

6 Examples

6.1 The Circle S^1

Let $S^1 \subset \mathbb{C}$ be the unit circle, and let X a random sample around S^1, with 10,000 points and Gaussian noise in the normal direction. $L \subset X$ is a landmark set with 10 points obtained as described in Section 4.1.

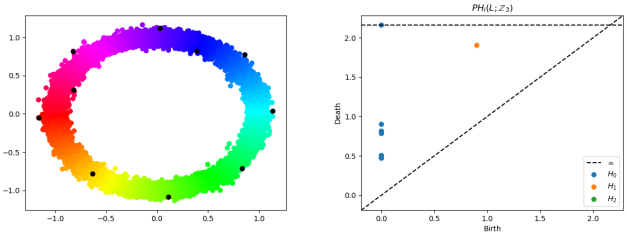


Figure 1: **Left:** Sample X, in black landmark set $L \subset X$. **Right:** $PH^i(\mathcal{R}(L); \mathbb{Z}_3)$ for $i = 0, 1, 2$.

Let a be the cohomological death of the most persistent class $PH^1(\mathcal{R}(L); \mathbb{Z}_q)$. For $\epsilon := a + 10^{-5}$ and $\eta = i^*(\eta') \in Z^1(\mathcal{N}(\mathcal{B}_\epsilon); \mathbb{Z}_q)$ we define the map $f : B_\epsilon \to L_3^{10}$ as in Equation (1).

After computing LPCA for $f(X) \subset L_3^{10}$ and the percentage of cumulative variance $p.\mathsf{var}_Y(k)$ we obtain the row in Table 1 with label S^1 (see Figure 7 for more details). We see that dimension 1 recovers $\sim 60\%$ of the variance. Moreover, Figure 2 shows $P_2(f(X)) \subset L_3^2$

Dim. (n)	**1**	**2**	**3**	**4**	**5**
S^1	0.62	0.75	0.81	0.86	0.89
$M(\mathbb{Z}_3, 1)$	0.56	0.7	0.76	0.8	0.83
L_3^2	0.47	0.62	0.67	0.71	0.73

Table 1: Percentage of recovered variance in L_3^n.

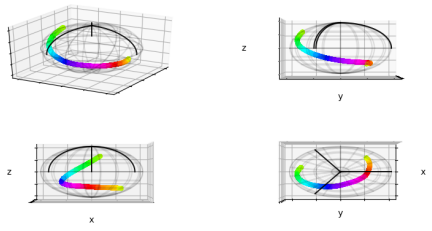


Figure 2: Visualization $P_2(f(X)) \subset L_3^2$.

in the fundamental domain described in Section 2.2.1 trough the map in Equation (4).

One key aspect of LC (Lens coordinates) is that it is designed to highlight the cohomology class η used on Equation (1). This is easily observed in this example; we selected the most persistent class in $PH^1(\mathcal{R}(L); \mathbb{Z}_3)$ and as a consequence in Figure 2 we see how this class is preserved while all the information in the normal direction is lost in the process.

6.2 The Moore space $M(\mathbb{Z}_3, 1)$.

Let G be an abelian group and $n \in \mathbb{N}$. The Moore space $M(G, n)$ is a CW-complex such that $H_n(M(G, n), \mathbb{Z}) = G$ and $\tilde{H}_i(M(G, n), \mathbb{Z}) = 0$ for all $i \neq n$. A well known construction for $M(\mathbb{Z}_3, 1)$ can be found in [6]. Equation (5) defines a metric on $M(\mathbb{Z}_3, 1)$.

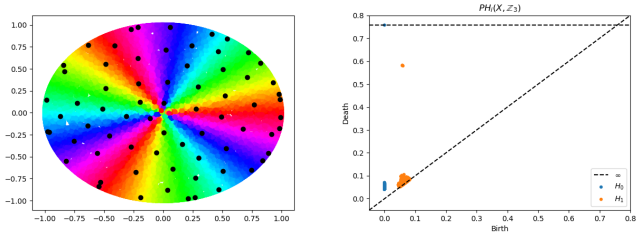


Figure 3: **Left:** $X \subset M(\mathbb{Z}_3, 1)$ with landmarks in black. **Right:** $PH^i(\mathcal{R}(L); \mathbb{Z}_3)$ for $i = 0, 1$.

Figure 3, on the left, shows a sample $X \subset M(\mathbb{Z}_3, 1)$ with $|X| = 15,000$ and 70 landmarks. The landmarks were obtained by minmax sampling after feeding the algorithm with an initial set of 10 point on the boundary on the disc. Figure 4 shows the persistent cohomology of $\mathcal{R}(L)$ with coefficients in $\mathbb{Z}_2$ and $\mathbb{Z}_3$ side-by-side.

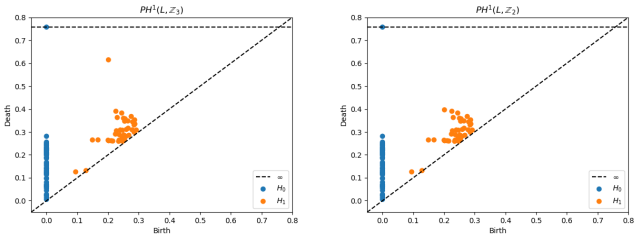


Figure 4: $PH^i(R(L);\mathbb{F})$ for $i = 0, 1$ and $\mathbb{F} = \mathbb{Z}_2, \mathbb{Z}_3$.

We compute $f : M(\mathbb{Z}_3, 1) \longrightarrow L_3^{70}$ analogously to the previous example and obtain a point cloud $f(X) \subset L_3^{70}$. The profile of recovered variance is shown in Table 1. Dimension 2 provides a low dimensional representation of $f(X)$ inside L_3^2 with 70% of recovered variance (Figure 8).

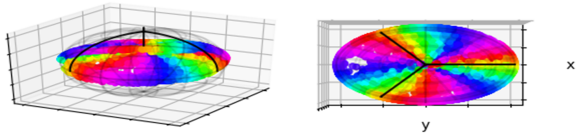


Figure 5: Visualization of the resulting $P_2(f(X)) \subset L_3^2$.

Since f classifies the principal $\mathbb{Z}_3$-bundle P_η over $M(\mathbb{Z}_3, 1)$, then f must be homotopic to the inclusion of $M(\mathbb{Z}_q, 1)$ in L_q^∞. Figure 5 shows $X \subset M(\mathbb{Z}_3, 1)$ mapped by f in L_3^2. Notice the identifications on X are handled by the identification on $S^1 \times \{0\} \subset D^3$ from the fundamental domain on Section 2.2.1. See https://youtu.be/_Ic730_xFkw for a more complete visualization.

6.3 The Lens space $L_3^2 = S^3/\mathbb{Z}_3$.

We use the metric defined in Equation (3) on L_3^2 and randomly sample 15,000 points to create $X \subset L_3^2$. Figure 6(left) shows the sample set using the fundamental domain from section 2.2.1.

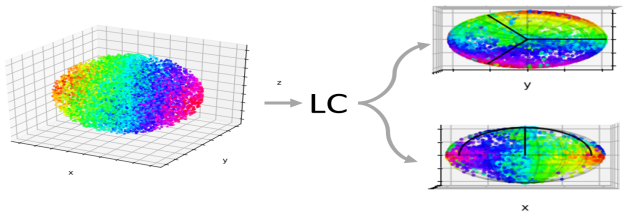


Figure 6: **Left:** $X \subset L_3^2$. **Right:** Lens coordinates.

We can use $PH^i(\mathcal{R}(X);\mathbb{Z}_2)$ and $PH^i(\mathcal{R}(X);\mathbb{Z}_3)$ to verify that the sampled metric space has the expected topological features. Figure 10 contains the corresponding persistent diagrams.

Just as in the previous examples define $f : L_3^2 \to L_3^\infty$ using the most persistent class in $PH^1(\mathcal{R}(L);\mathbb{Z}_3)$. The homotopy class of f must be the same as that of the inclusion $L_3^2 \subset L_3^\infty$, since f classifies the $\mathbb{Z}_3$-principal bundle P_η. Thus we expect L_3^2 to be preserved up to homotopy under LPCA. Figure 6 offers a side and top view of $P_2(f(X)) \subset L_3^2$. Here we clearly see how the original data set X is transformed while preserving the identifications on the boundary of the fundamental domain. Finally in Table 1 we show the variance profile for the dimensionality reduction problem. We see that for dimension 4 we have recovered more than 70% of the total variance as seen in Table 1 and Figure 9.

6.4 Isomap dimensionality reduction

We conclude this section by providing evidence that Lens coordinates (LC) preserve topological features when compared to other dimensionality reduction algorithms. For this purpose we use Isomap ([13]) as our point of comparison.

The Isomap algorithm consist of 3 main steps. The first step determines neighborhoods of each point using k-th nearest neighbors. The second step estimates the geodesic distances between all pairs of points using shortest distance path, and the final step applies classical MDS to the matrix of graph distances.

Let dgm be a persistent diagram. Define per_1 to be the largest persistence of an element in dgm, and let per_2 be the second largest persistence of an element dgm.

$\text{per}_1/\text{per}_2$		$\mathbb{Z}_2$	$\mathbb{Z}_3$
$M(\mathbb{Z}_q, 1)$	Isomap	1.0105	1.0105
	LC	1.7171	3.6789
L_3^2	Isomap	1.0080	1.0080
	LC	1.1592	2.8072

Table 2: In green we highlight the fraction that indicates which method better identifies the topological features.

For both $M(\mathbb{Z}_3, 1)$ and L_3^2 it is clear that the Isomap projection fails to preserve the difference between the cohomology groups with coefficients in $\mathbb{Z}_2$ and $\mathbb{Z}_3$. On the other hand the LC projections maintains this difference in both examples (see Table 4 for more details).

Acknowledgements

This work was partially supported by the NSF under grant DMS-1622301.

References

[1] J. A. Perea and G. Carlsson. A klein-bottle-based dictionary for texture representation. *International Journal of Computer Vision*, 107:75–97, 03 2014. doi: 10.1007/s11263-013-0676-2.

[2] E. H. Brown. Cohomology theories. *Annals of Mathematics*, pages 467–484, 1962.

[3] G. Carlsson. Topological pattern recognition for point cloud data. *Acta Numerica*, 23:289368, 2014. doi: 10.1017/S0962492914000051.

[4] G. Carlsson, T. Ishkhanov, V. Silva, and A. Zomorodian. On the local behavior of spaces of natural images. *Int. J. Comput. Vision*, 76(1): 1–12, Jan. 2008. ISSN 0920-5691. doi: 10.1007/s11263-007-0056-x. URL http://dx.doi.org/10.1007/s11263-007-0056-x.

[5] W. Crawley-Boevey. Decomposition of pointwise finite-dimensional persistence modules. *Journal of Algebra and its Applications*, 14(05):1550066, 2015.

[6] A. Hatcher. *Algebraic topology*. Cambridge University Press, 2002.

[7] D. Husemoller and D. Husemöller. *Fibre Bundles*. Graduate Texts in Mathematics. Springer, 1994. ISBN 9780387940878. URL https://books.google.com/books?id=DPr_BSH89cAC.

[8] J. Milnor. Construction of universal bundles, ii. *Annals of Mathematics*, pages 430–436, 1956.

[9] R. Miranda. *Algebraic curves and Riemann surfaces*, volume 5. American Mathematical Soc., 1995.

[10] J. A. Perea. Sparse Circular Coordinates via Principal $\mathbb{Z}$-Bundles. *arXiv e-prints*, art. arXiv:1809.09269, Sep 2018.

[11] J. A. Perea. A brief history of persistence. *preprint arXiv:1809.03624*, 2018. https://arxiv.org/abs/1809.03624.

[12] J. A. Perea. Multiscale projective coordinates via persistent cohomology of sparse filtrations. *Discrete & Computational Geometry*, 59(1):175–225, 2018.

[13] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319, 2000.

Appendix

Proof. [of Proposition 3] Take $[b, j, g] \in P_\eta$ and consider a different representative of the class. Namely, an element $(b, k, g + \eta_{jk})$ such that $b \in U_j \cap U_k$. By definition of Φ, we have $\Phi([b, j, g]) = f_j(b, j, g)$ and $\Phi([b, k, g + \eta_{jk}]) = f_k(b, k, g + \eta_{jk})$. And since $f_j(b, j, g) = f_k(b, k, g + \eta_{jk})$, we have that

$$\Phi([b, j, g]) = \Phi([b, k, g + \eta_{jk}]),$$

which shows that Φ is well defined.

To see that Φ is $\mathbb{Z}_q$-equivariant, take $m \in \mathbb{Z}_q$ for any $m = 0, \ldots, q-1$ and compute

$$\begin{aligned}
&\Phi([b, j, g]) \cdot m \\
&= \left[\sqrt{\varphi_1(b)}\zeta_q^{(g+m+\eta_{j1})}, \ldots, \sqrt{\varphi_n(b)}\zeta_q^{(g+m+\eta_{jn})}\right] \\
&= f_j(b, j, g + m) = \Phi([b, j, g + m]) \\
&= \Phi([b, j, g] \cdot m).
\end{aligned}$$

□

Proof. [of Theorem 5] First we need to see that f is well defined. Let $b \in U_j \cap U_k$, therefore

$$\begin{aligned}
p(\Phi([b, j, 0])) &= \left[\sqrt{\varphi_1(b)}\zeta_q^{\eta_{j1}} : \cdots : \sqrt{\varphi_n(b)}\zeta_q^{\eta_{jn}}\right] \\
&= p(\Phi([b, k, 0)).
\end{aligned}$$

This shows that $f(b)$ is independent of the open set containing b.

Hence $(\Phi, f) : (P_\eta, \pi, B) \to (S^{2n-1}, \pi, L_q^n)$ is a morphism of principal $\mathbb{Z}_q$-bundles, and by [[7], Chapter 4: Theorem 4.2] we conclude that P_η and $f^*(S^{2n-1})$ are isomorphic principal $\mathbb{Z}_q$-bundles over B. □

Proof. [of Proposition 6] First of all, $\mathcal{R}_{2\epsilon}(L)^{(0)} = \mathcal{N}(\mathcal{B}_\epsilon)^{(0)} = L$. If $b \notin B_\epsilon(l_k)$, then $\varphi_k(b) = 0$ and therefore the equality holds. If on the other hand $b \in B_\epsilon(l_k) \cap B_\epsilon(l_j)$, then $\{j, k\} \in \mathcal{N}(\mathcal{B}_\epsilon)^{(1)} \subset \mathcal{R}_{2\epsilon}(L)^{(1)}$. In which case, by definition of i^*, we have $\tilde{\eta}_{jk} = \eta_{jk}$. □

Proposition 8 *Let $[x], [y] \in L_q^n$, then*

$$d_L([x], [y]) = d(x, y \cdot \mathbb{Z}_q) = \min_{g \in \mathbb{Z}_q} d(x, y \cdot g).$$

Proof. For $x, y \in \mathbb{C}^n$ let $\langle x, y\rangle_{\mathbb{R}} := \mathsf{real}(\langle x, y\rangle_{\mathbb{C}})$. By definition of Hausdorff distance, we have that

$$\begin{aligned}
d_L([x], [y]) = \max\Big\{ &\max_{g \in \mathbb{Z}_q} \min_{h \in \mathbb{Z}_q} \arccos(\langle x \cdot g, y \cdot h\rangle_{\mathbb{R}}), \\
&\max_{h \in \mathbb{Z}_q} \min_{g \in \mathbb{Z}_q} \arccos(\langle x \cdot g, y \cdot h\rangle_{\mathbb{R}})\Big\}.
\end{aligned}$$

Notice that

$$\begin{aligned}
\langle x \cdot g, y \cdot h\rangle_{\mathbb{R}} &= \mathsf{real}\left(\left\langle \zeta_q^g x, \zeta_q^h y\right\rangle_{\mathbb{C}}\right) \\
&= \mathsf{real}\left(\left\langle x, \zeta_q^{(h-g)} y\right\rangle_{\mathbb{C}}\right) \\
&= \langle x, y \cdot (h - g)\rangle_{\mathbb{R}}
\end{aligned}$$

And since $\mathbb{Z}_q$ is Abelian, then

$$\begin{aligned}\max_{h\in\mathbb{Z}_q}\min_{g\in\mathbb{Z}_q}&\arccos(\langle x\cdot g, y\cdot h\rangle_{\mathbb{R}})\\ &=\max_{h\in\mathbb{Z}_q}\min_{g\in\mathbb{Z}_q}\arccos(\langle x\cdot(g-h), y\rangle_{\mathbb{R}})\\ &=\max_{h\in\mathbb{Z}_q}\min_{g\in\mathbb{Z}_q}\arccos(\langle x\cdot(-h), y\cdot(-g)\rangle_{\mathbb{R}})\\ &=\max_{h'\in\mathbb{Z}_q}\min_{g'\in\mathbb{Z}_q}\arccos(\langle x\cdot h', y\cdot g'\rangle_{\mathbb{R}}).\end{aligned}$$

Thus

$$d_L([x],[y])=\max_{g\in\mathbb{Z}_q}\min_{h\in\mathbb{Z}_q}\arccos(\langle x\cdot g, y\cdot h\rangle_{\mathbb{R}}).$$

Furthermore $d_L([x],[y]) = \max_{g\in\mathbb{Z}_q} d(x\cdot g, y\cdot\mathbb{Z}_q) = \max_{g\in\mathbb{Z}_q} d(x, y\cdot(-g)\mathbb{Z}_q)$. Since $y\cdot\big((-g)\mathbb{Z}_q\big) = y\cdot\mathbb{Z}_q$ for any $g\in\mathbb{Z}_q$, we obtain $d_L([x],[y]) = \max_{g\in\mathbb{Z}_q} d(x, y\cdot\mathbb{Z}_q) = d(x, y\cdot\mathbb{Z}_q) = \min_{h\in\mathbb{Z}_q} d(x, y\cdot h)$. □

Proof. [of Lemma 7] From Theorem 8 we know that

$$\begin{aligned}d_L([v],P_u^{\perp}([v])) &= \min_{g\in\mathbb{Z}_q} d(v, P_u^{\perp}([v])\cdot g)\\ &= \min_{g\in\mathbb{Z}_q} d\left(v, \frac{P_u^{\perp}(v)}{\|P_u^{\perp}(v)\|}\cdot g\right).\end{aligned}$$

Let $g^* := \operatorname{argmin}_{g\in\mathbb{Z}_q} d\left(v, \frac{P_u^{\perp}(v)}{\|P_u^{\perp}(v)\|}\cdot g\right)$, so we have

$$d_L([v],P_u^{\perp}([v])) = \arccos\left(\left\langle v, \frac{P_u^{\perp}(v)}{\|P_u^{\perp}(v)\|}\cdot g^*\right\rangle_{\mathbb{R}}\right).$$

Notice that the argument of the arccos can be simplified as follows

$$\begin{aligned}\left\langle v, \frac{P_u^{\perp}(v)}{\|P_u^{\perp}(v)\|}\cdot g^*\right\rangle_{\mathbb{R}} &= \left\langle \langle v,u\rangle_{\mathbb{C}}u + P_u^{\perp}(v), \frac{P_u^{\perp}(v)}{\|P_u^{\perp}(v)\|}\cdot g^*\right\rangle_{\mathbb{R}}\\ &= \left\langle \langle v,u\rangle_{\mathbb{C}}u, \frac{P_u^{\perp}(v)}{\|P_u^{\perp}(v)\|}\cdot g^*\right\rangle_{\mathbb{R}}\\ &\quad + \left\langle P_u^{\perp}(v), \frac{P_u^{\perp}(v)}{\|P_u^{\perp}(v)\|}\cdot g^*\right\rangle_{\mathbb{R}}.\end{aligned}$$

since u and $P_u^{\perp}(v)$ are orthogonal in $\mathbb{C}^n$ then they are also orthogonal in $\mathbb{R}^{2n}$, making the then the firs summand on the right hand side equal to zero. Additionally since arccos as a real valued function is monotonically decreasing we have

$$g^* = \operatorname*{argmax}_{g\in\mathbb{Z}_q} \frac{1}{\|P_u^{\perp}(v)\|}\left\langle P_u^{\perp}(v), P_u^{\perp}(v)\cdot g\right\rangle_{\mathbb{R}}.$$

Using the fact that the action of $\mathbb{Z}_q$ is an isometry (and therefore an operator of norm one) as well as the Cauchy-Schwartz inequality we obtain

$$\begin{aligned}\frac{\left\langle P_u^{\perp}(v), P_u^{\perp}(v)\cdot g\right\rangle_{\mathbb{R}}}{\|P_u^{\perp}(v)\|} &\le \left|\frac{1}{\|P_u^{\perp}(v)\|}\left\langle P_u^{\perp}(v), P_u^{\perp}(v)\cdot g\right\rangle_{\mathbb{R}}\right|\\ &\le \frac{1}{\|P_u^{\perp}(v)\|}\|P_u^{\perp}(v)\|\|P_u^{\perp}(v)\cdot g\|\\ &= \|P_u^{\perp}(v)\cdot g\| = \|P_u^{\perp}(v)\|.\end{aligned}$$

And the equality holds whenever $g = e \in \mathbb{Z}_q$, so we must have $g^* = e$.

Let $[w]\in L_q^{n-1}(u)$, so $w\in \mathsf{span}_{\mathbb{C}}^{\perp}(u)$ which implies that for any $h\in\mathbb{Z}_q$

$$\langle u, w\cdot h\rangle_{\mathbb{C}} = \sum_k u_k(\overline{\zeta_q^h w_k}) = \zeta_q^{-h}\sum_k u_k\overline{w_k} = \zeta_q^{-h}\langle u,w\rangle = 0.$$

In other words $w\cdot h\in\mathsf{span}_{\mathbb{C}}^{\perp}(u)$ for any $h\in\mathbb{Z}_q$.

Thus by the Cauchy-Schwartz inequality

$$\begin{aligned}\langle v, w\cdot h\rangle_{\mathbb{R}} &= \langle\langle v,u\rangle_{\mathbb{C}}u + P_u^{\perp}(v), w\cdot h\rangle_{\mathbb{R}} = \langle P_u^{\perp}(v), w\cdot h\rangle_{\mathbb{R}}\\ &\le |\langle P_u^{\perp}(v), w\cdot h\rangle_{\mathbb{R}}| \le \|P_u^{\perp}(v)\|\|w\cdot h\|\\ &= \|P_u^{\perp}(v)\|\|w\| = \|P_u^{\perp}(v)\|,\end{aligned}$$

since the action of $\mathbb{Z}_q$ is an isometry and $w\in S^{2n-1}$.

Finally since arccos is decreasing

$$d_L([v],P_u^{\perp}([v])) = \arccos(\|P_u^{\perp}(v)\|) \le \arccos(\langle v, w\cdot h\rangle_{\mathbb{R}})$$

for all $h\in\mathbb{Z}_q$, thus $d_L([v],P_u^{\perp}([v])) \le d_L([v],[w])$. □

Visualization map for L_3^2. Given $v_1,\dots,v_n\in S^{2n-1}$ representatives for the classes in $\mathsf{LPCA}(Y)$. We want to visualize $P_2(Y)\subset L_3^2$ in the fundamental domain described in Section 2.2.1. Let

$$P_2(Y) = \left\{\left[\langle y_i,v_1\rangle_{\mathbb{C}}, \langle y_i,v_2\rangle_{\mathbb{C}}\right]\in S^3\subset\mathbb{C}^2 : [y_i]\in Y\right\}$$

and define $G: P_2(Y)\longrightarrow S^3\subset\mathbb{C}^2$ to be

$$G(z,w) := \left(\zeta_3^{-k}z, \left(\arg(w)-\frac{\pi}{3}\right)\sqrt{1-|z|^2}\right) \qquad (4)$$

where $\arg(w)\in\left[0,\frac{2\pi}{3}\right)$, and k an integer such that

$$\arg(z) = k\frac{2\pi}{3}+\theta,$$

where θ is the remainder after division by $\frac{2\pi}{3}$.

Metric on the Moore space $M(\mathbb{Z}_3,1)$. For $x,y\in\mathbb{C}$ with $|x|,|y|\le 1$, we let

$$d(x,y) = \begin{cases}\sqrt{|\langle x,y\rangle_{\mathbb{R}}|} & \text{if } |x|,|w|<1\\ \min\limits_{\zeta\in\mathbb{Z}_3}\sqrt{|\langle x,\zeta y\rangle_{\mathbb{R}}|} & \text{if } |x|=1 \text{ or } |w|=1\\ \min\limits_{\zeta\in\mathbb{Z}_3}\arccos(|\langle x,\zeta y\rangle_{\mathbb{R}}|) & \text{if } |x|=1 \text{ and } |w|=1\end{cases}. \qquad (5)$$

Profiles of recovered variance.

Recovered variance of LPCA on S^1.

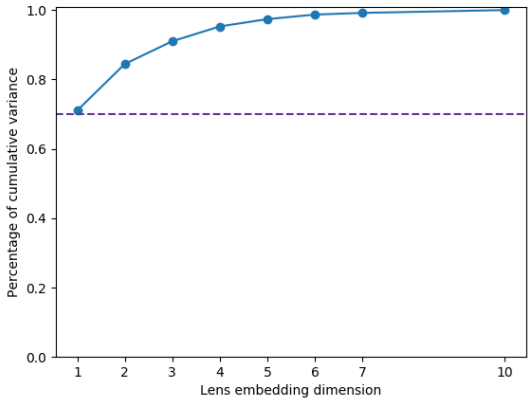


Figure 7: Profile of recovered variance on S^1.

Recovered variance of LPCA on $M(\mathbb{Z}_3, 1)$.

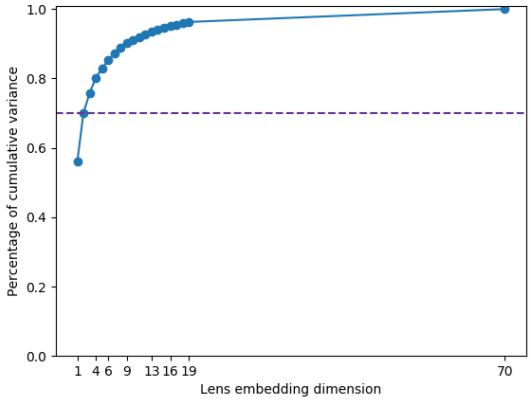


Figure 8: Profile of recovered variance on $M(\mathbb{Z}_3, 1)$.

Recovered variance of LPCA on L_3^2.

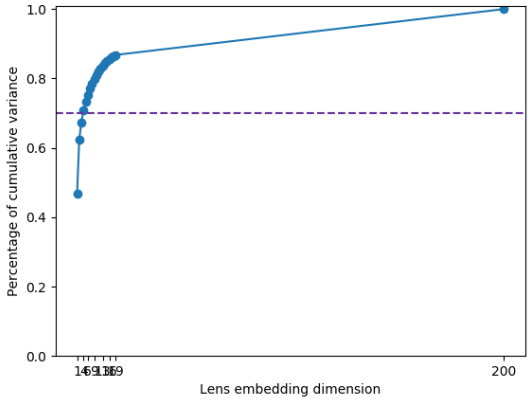


Figure 9: Profile of recovered variance on L_3^2.

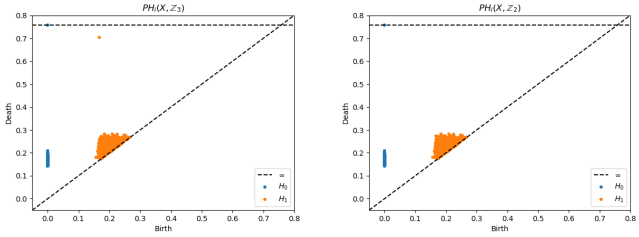


Figure 10: $PH^i(\mathcal{R}(L); \mathbb{Z}_3)$ for $i = 0, 1$. $PH^i(\mathcal{R}(L); \mathbb{Z}_2)$ for $i = 0, 1$.

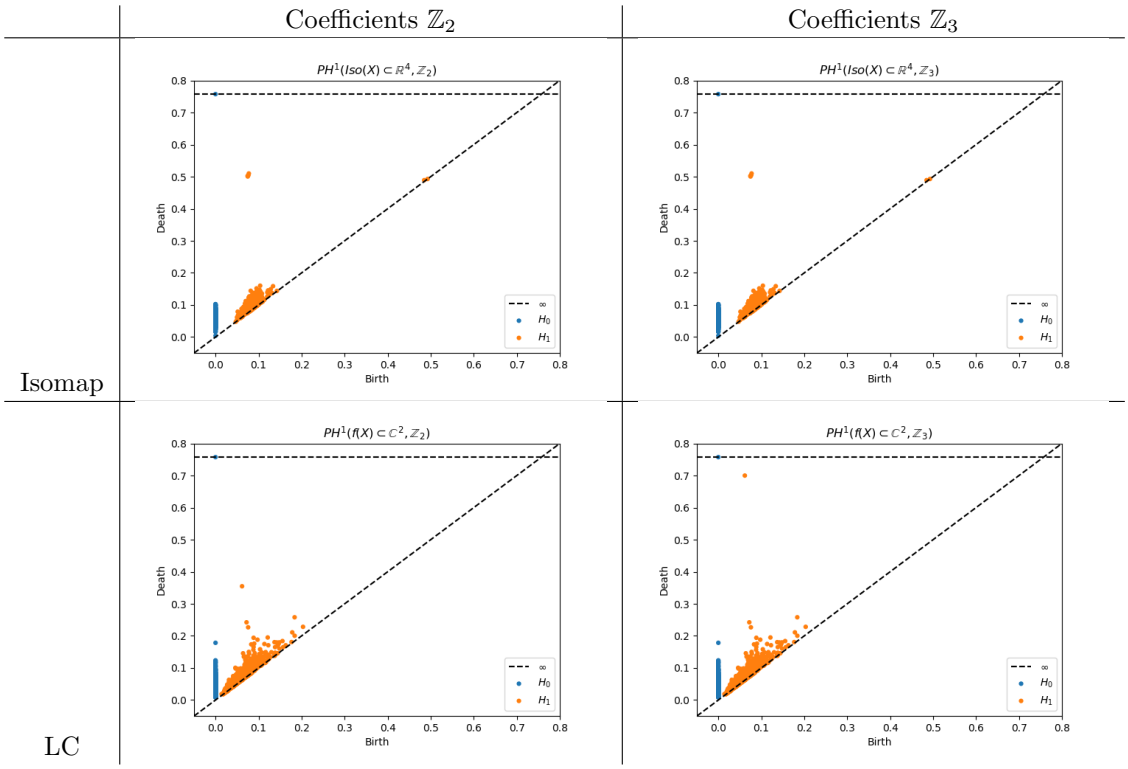


Table 3: Persistent homology of the Isomap vs. LPCA for $M(\mathbb{Z}_3, 1)$ into a 4 dimensional space.

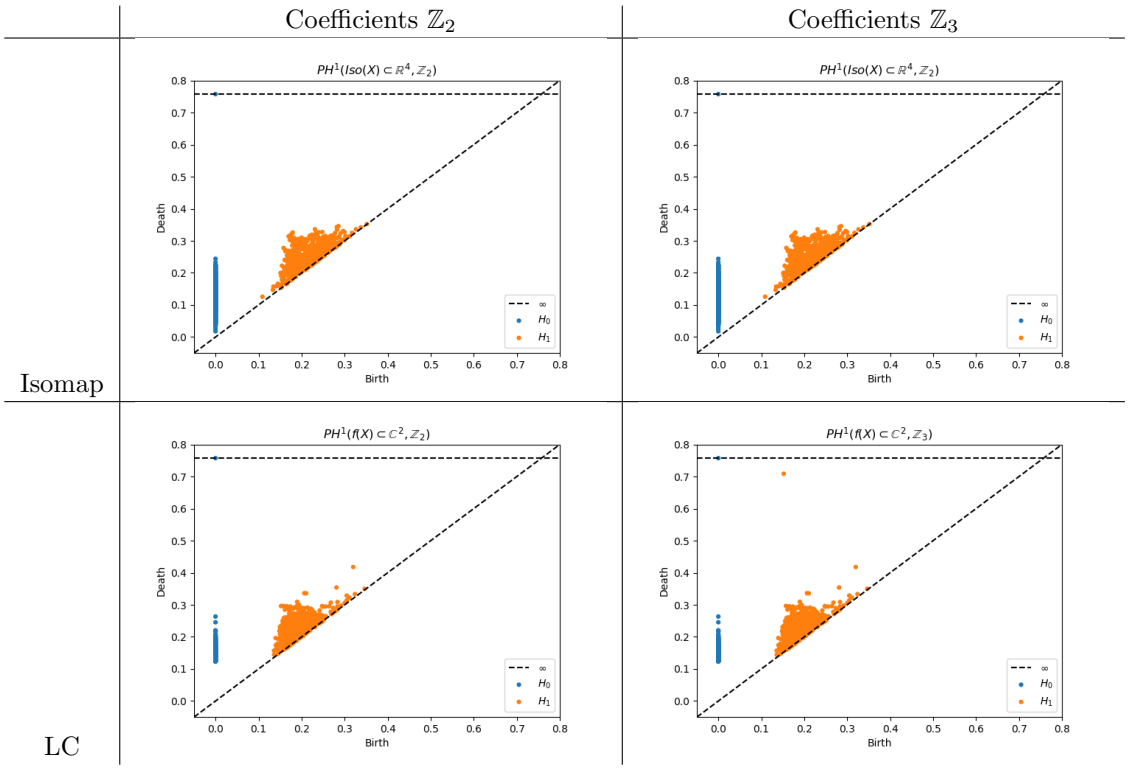


Table 4: Persistent homology of the Isomap vs. LPCA for L_3^2 into a 4 dimensional space.

Computing Feasible Trajectories for an Articulated Probe in Three Dimensions

Ovidiu Daescu* Ka Yaw Teo*

Abstract

An articulated probe is modeled as two line segments ab and bc connected at point b. Line segment ab can be arbitrarily long, while bc is of a given length r. The input consists of a set of n disjoint triangular obstacles in $\mathbb{R}^3$ and a target point t in the free space, all enclosed by a large sphere S centered at t. Initially, the probe is located outside S and assumes an *unarticulated* configuration, in which line segments ab and bc are collinear and $b \in ac$. The goal is to find a feasible (obstacle avoiding) probe trajectory to reach t, with the condition that the probe is constrained by the following sequence of moves – a straight-line insertion of the unarticulated probe into S, possibly followed by a rotation of bc at b for at most $\pi/2$ radians, so that c coincides with t.

We prove that if there exists a feasible probe trajectory, then a set of *extremal* feasible trajectories must be present. Through careful case analysis, we show that these extremal trajectories can be represented by $O(n^4)$ combinatorial events. We present a solution approach that enumerates and verifies these combinatorial events for feasibility in overall $O(n^{4+\epsilon})$ time using $O(n^{4+\epsilon})$ space, for any constant $\epsilon > 0$. The enumeration algorithm is highly parallel, considering that each combinatorial event can be generated and verified for feasibility independently of the others. In the process of deriving our solution, we design the first data structure for addressing a special instance of circular sector emptiness queries among polyhedral obstacles in three dimensional space, and provide a simplified data structure for the corresponding emptiness query problem in two dimensions.

1 Introduction

We address the three-dimensional (3D) version of the articulated probe trajectory planning problem introduced in [4]. An articulated probe is modeled as two line segments ab and bc joined at point b. The length of line segment ab can be arbitrarily large, while line segment bc has a fixed length r. Line segment bc may rotate at point b. A 3D workspace contains a set P of n interior disjoint triangular obstacles and a target point t in the free space, all within a large sphere S centered at t (see Figure 1).

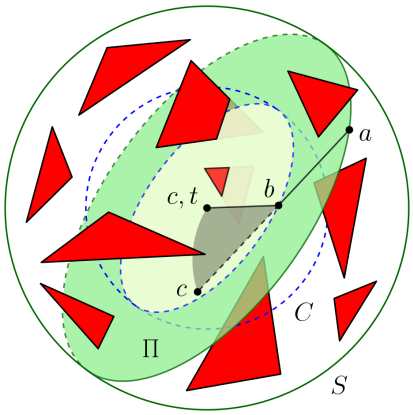


Figure 1: Articulated probe trajectory in 3D. After inserting line segment abc into sphere S, in order to reach target point t, line segment bc may be required to rotate from its intermediate position (dashed line segment) to the final position (solid line segment).

The probe is said to have an *unarticulated* configuration if line segments ab and bc are collinear, and $b \in ac$. Otherwise, the probe has an *articulated* configuration – that is, bc has been rotated at b, and bc is no longer collinear with ab.

The probe begins in an unarticulated configuration outside sphere S. The unarticulated probe, represented by straight line segment abc, is then inserted along a straight line into sphere S. After completing the insertion, line segment bc may be rotated at point b up to $\pi/2$ radians in order to reach t. Hence, the final configuration of the probe could be either unarticulated or articulated. The *intermediate* configuration of the probe is the unarticulated configuration after the insertion but before the rotation.

A *feasible* probe trajectory consists of an initial insertion of straight line segment abc into sphere S, possibly followed by a rotation of line segment bc at point b, such that point c ends at target point t, while avoiding the obstacles in the process of insertion and rotation.

The objective of the problem is to determine a feasible probe trajectory, if one exists.

As illustrated in Figure 1, a feasible probe trajectory

*Department of Computer Science, University of Texas at Dallas, Richardson, TX, USA. {ovidiu.daescu, ka.teo}@utdallas.edu

is planar – that is, its motion always lies in a plane Π passing through point t. Since line segment bc may only rotate as far as $\pi/2$ radians at point b, for any feasible probe trajectory, point b is the first intersection between segment ab and the sphere C of radius r centered at t. When line segment bc rotates about point b, the area swept by bc is a sector σ of a circle D in plane Π (i.e., a portion of a disk enclosed by two radii and a circular arc), where D is of radius r and centered at a point on sphere C (specifically, a point b on the intersection of C and Π). Circular sector σ always has an endpoint of one of its bounding radii located at t.

Motivation

Besides its pointed relevance in robotics, the outlined problem arises particularly from planning for minimally invasive surgeries. In fact, surgical instruments that can be modeled by our simple articulated probe are already in clinical use (e.g., da Vinci EndoWrist by Intuitive Surgical), given their enhanced capability in reaching remote targets while circumventing surrounding critical structures [13]. In our problem setting, a human body cavity can be viewed as (a subset of) workspace S, and any critical organ/tissue can be represented by using a triangle mesh. Despite its importance and relevance, the problem has never been investigated in three dimensions from a theoretical viewpoint, and only a handful of results in two dimensions have been reported [3, 4, 5].

Related work

Daescu, Fox, and Teo [4] originally proposed the aforementioned trajectory planning problem in two dimensions (2D), and they presented an $O(n^2 \log n)$-time, $O(n \log n)$-space algorithm for finding a feasible trajectory amidst n line segment obstacles. The algorithm was based on computing extremal trajectories that are tangent to one or two obstacle vertices. This algorithmic approach was later extended to finding a feasible trajectory of a given clearance δ from the obstacles, for any $\delta > 0$, in $O(n^2 \log n)$ time using $O(n^2)$ space [3]. In addition, Daescu and Teo [5] showed that the feasible solution space for the two-dimensional trajectory planning problem can be characterized by a simple-curve arrangement of complexity $O(k)$, and the arrangement can be constructed in $O(n \log n + k)$ time using $O(n+k)$ space, where $k = O(n^2)$ is the number of vertices in the arrangement.

Results and contributions

We describe an algorithm that computes a feasible probe trajectory in 3D, if one exists, in $O(n^{4+\epsilon})$ time using $O(n^{4+\epsilon})$ space, for any constant $\epsilon > 0$. First, we prove that if there exists a feasible probe trajectory, then some *extremal* feasible trajectories must be present. An extremal trajectory is characterized by its intersections or tangencies with a combination of obstacle edges, vertices, and/or surfaces. Through careful case analysis, we show that these extremal trajectories can be represented by $O(n^4)$ combinatorial events. Our algorithm is based on enumerating and verifying these combinatorial events for feasibility. As an alternative, an $O(n^5)$-time algorithm with $O(n)$-space usage is achievable by performing a simple $O(n)$-check on each of the $O(n^4)$ events.

While deriving our solution approach, we develop the first data structure for solving a special case of the circular sector emptiness query problem in 3D, where the query circular sector has a fixed radius r and an endpoint of its arc located at fixed point t. We present a data structure of size $O(n^{4+\epsilon})$ for answering a query of the sort in $O(\log n)$ time. When mapped to the plane, this result yields a new data structure for solving the corresponding circular sector emptiness query problem in 2D. Our new $\mathbb{R}^2$ query data structure simplifies the two-part approach formerly proposed in [4] while maintaining the same time and space complexity. Sharir and Shaul [12] proposed a solution based on semialgebraic range searching for circular cap (larger than a semidisk) emptiness queries in 2D. These circular sector emptiness queries, in 2D and 3D, are considered to be of independent interest. To the best of the authors' knowledge, there has been no published data structure for general circular sector emptiness queries in 3D or even 2D.

2 Extremal feasible trajectories

In this section, we prove that if there exists a feasible probe trajectory, then a set of extremal feasible trajectories must also be present.

Let ℓ denote a line segment of the probe. In addition, let σ and γ denote a circular sector (i.e., area swept by bc, as previously described) and its arc, respectively.

Let τ be a triangular obstacle of P in $\mathbb{R}^3$, and let e denote en edge of τ. Without loss of generality, assume that τ is not co-planar with t, and ℓ is not parallel to e. ℓ may intersect e at an endpoint of e, an interior point of e, or none. An endpoint of e is a *support vertex* of ℓ if ℓ intersects the endpoint of e. e is a *support edge* of ℓ if ℓ intersects e at an interior point. Thus, a *support* of ℓ is either a support vertex or a support edge.

Suppose, without loss of generality, that σ and e lie in different planes. σ is said to be supported by e if σ intersects an endpoint of e. γ is supported by e if γ intersects an interior point of e. If γ is tangent to the surface of τ, then the surface is a *support surface* of γ. For an illustration of the various types of supports just described, see Figure 2.

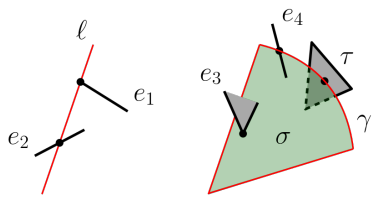


Figure 2: Support vertex, edge, and surface. The endpoint of edge e_1 is a support vertex of line segment ℓ. Edge e_2 is a support edge of ℓ. The endpoint of e_3 is a support vertex of circular sector σ. Edge e_4 supports circular arc γ at an interior point. Triangle τ is a support surface of γ.

A line segment is *extremal* (or *isolated*) with respect to a set of support vertices and edges if the line segment cannot be moved continuously while maintaining its intersections with these vertices and edges. Analogously, a probe trajectory is isolated by a set of supports if the trajectory cannot be altered without losing any of its supports. Note that the term "isolated" has the same meaning as "extremal," and they will be used interchangeably hereafter.

Lemma 1 *Assume that a feasible* ***unarticulated*** *trajectory exists – that is, the unarticulated probe can be inserted into S to reach t while avoiding the obstacles (i.e., t can see to infinity without obstruction). Then, there exists an extremal feasible unarticulated trajectory, in which the probe is isolated by either one support vertex or two support edges.*

Lemma 2 *Assume that a feasible* ***articulated*** *trajectory exists – that is, the unarticulated probe can be inserted into S, and a subsequent rotation of bc at b can be performed to reach t, all while avoiding the obstacles. Then, there exists either*

i) an extremal feasible unarticulated trajectory or

ii) an extremal feasible trajectory such that the probe assumes an articulated final configuration, and the trajectory is isolated with respect to at most four supports (edges, vertices, surfaces, or a combination of the three).

The existence of extremal feasible probe trajectories can be proven by using a series of perturbation arguments. The full proofs of Lemmas 1 and 2 are given in Appendices A and B, respectively.

3 Finding and validating extremal trajectories

Based on Lemmas 1 and 2, we can compute the set of extremal probe trajectories and verify them for feasibility. The algorithms and data structures required are detailed next.

Unarticulated trajectories

Let V and E denote the set of vertices and edges of the triangles of P, respectively. We compute a set R of rays, each of which i) originates at point t, ii) is either passing through a vertex of V or isolated with respect to two support edges of E, and iii) does not intersect any triangle of P in its interior. Each ray of R represents an extremal feasible unarticulated trajectory. Set R can be obtained by computing the visibility polyhedron from point t in $O(n^2)$ time using $O(n^2)$ space [9]. In fact, this approach yields all feasible unarticulated solutions, since the visibility polyhedron gives all unbounded rays from t to infinity (i.e., feasible unarticulated trajectories).

Lemma 3 *The set of all feasible unarticulated probe trajectories can be found in $O(n^2)$ time using $O(n^2)$ space.*

Articulated trajectories

We at first compute the set of extremal articulated trajectories, which are characterized by $O(n^4)$ combinatorial events (see Lemma 2). These extremal articulated trajectories can be enumerated in $O(n^4)$ time either geometrically or algebraically (see Appendix C).

An extremal articulated trajectory is deemed feasible if and only if i) segment ab and ii) circular sector σ do not intersect with any triangular obstacle in its interior. Checking for these scenarios can be reduced to the following two query problems – i) ray shooting query and ii) circular sector emptiness query.

Ray shooting queries

A ray shooting query (among n interior disjoint triangles) can be performed to determine whether a query segment ab intersects with any triangular obstacle in its interior (i.e., whether a query segment stabs through the interior of an obstacle triangle). According to de Berg et al. [6] and Pellegrini [10], such a query can be answered in $O(\log n)$ time using $O(n^{4+\epsilon})$ preprocessing time and space, for any constant $\epsilon > 0$. Alternatively, by using the data structure proposed by Agarwal and Matousek [2], which provides a trade-off between space and time, a ray shooting query (amidst n triangles) can be answered in $O(n^{1+\epsilon}/m^{1/4})$ time using $O(m)$ storage and $O(m^{1+\epsilon})$ preprocessing time, for any $n \leq m \leq n^4$. By employing this fairly complex data structure, given that we have $O(n^4)$ queries in our case, when $m = n$, we obtain $O(n^{3/4+\epsilon})$ time per query and $O(n^{19/4+\epsilon})$ total time.

Circular sector emptiness queries

In this section, we address a special instance of the circular sector emptiness query problem in 3D.

Given a set P of n triangles in $\mathbb{R}^3$, preprocess it so that, for a query circular sector σ with a fixed radius r and an endpoint of its arc located at fixed point t, one can quickly determine whether σ intersects P.

Let Π be a plane passing through point t. We can parameterize Π by using two variables (I, Ω) (see Figure 3, where Π_0 is the plane with $(I = 0, \Omega = 0)$).

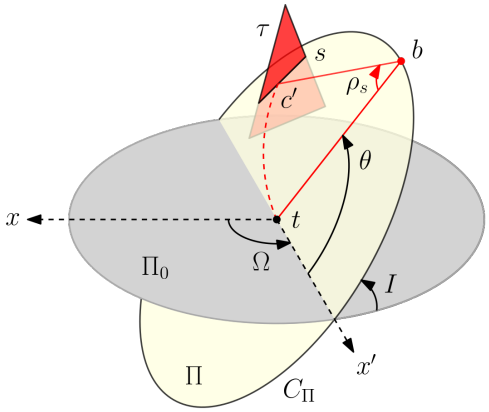


Figure 3: Characterization of ρ_s as function of θ in Π.

Let $C_\Pi{}'$ be the circle in Π with radius $\sqrt{2}r$ and the center at t. Let τ be a triangle that intersects Π in a line segment s. Observe that a query circular sector σ in Π, as described above, can only intersect with τ if s lies inside $C_\Pi{}'$.

Let C_Π denote the circle of radius r centered at t in Π. For any point $b \in C_\Pi$, let $\theta \in [0, 2\pi)$ be the angle of tb relative to the x-axis of Π (called the x'-axis, as depicted in Figure 3). Let bc' denote the farthest radius from tb (with respect to the circle of radius r centered at b in Π) before the minor circular sector bounded by radii tb and bc' intersects with s. Let ρ_s be the angle of bc' with respect to tb. Recall that segment bc of the probe, after the insertion of the probe has completed, may only rotate up to $\pi/2$ radians in either direction in plane Π. Thus, $\rho_s \in [0, \pi/2]$. Here we consider ρ_s to be the clockwise angle from tb in plane Π. The other case can be handled symmetrically.

We fix plane Π (i.e., I and Ω) and proceed to characterize ρ_s as a function of θ. Two different cases are examined separately, depending on whether line segment s lies 1) inside C_Π or 2) outside C_Π and inside $C_\Pi{}'$.

Case 1: s lies inside C_Π. For the sake of brevity, the quarter-circular sector associated with a point b (i.e., the maximum possible area swept by segment bc to reach point t), where the angle of tb relative to the x'-axis is θ, is henceforth referred to as the *quarter-circular sector associated with θ*.

Let $\phi_{s,1}$, $\phi_{s,2}$, and $\phi_{s,3}$ be defined as follows (see Figure 4(A)). $\phi_{s,1}$ is the smallest angle θ at which the circular arc of the quarter-circular sector associated with θ intersects with line segment s at one of its endpoints or an interior point. $\phi_{s,2}$ is the smallest angle θ at which segment bt of the quarter-circular sector associated with θ intersects with line segment s at one of its endpoints. $\phi_{s,3}$ is the largest angle θ at which segment bt of the quarter-circular sector associated with θ intersects with line segment s at one of its endpoints. In other words, as θ varies from 0 to 2π, $\phi_{s,1}$ and $\phi_{s,3}$ are the angles θ at which the quarter-circular sector associated with θ first and last intersects with line segment s, respectively.

For a line segment s lying inside C_Π, as shown in Figure 4, we are only concerned with computing ρ_s for $\theta \in [\phi_{s,1}, \phi_{s,2}]$, given that $\theta \in [\phi_{s,2}, \phi_{s,3}]$ is infeasible due to intersection of bt with s, and $\rho_s = \pi/2$ for $\theta \in [0, \phi_{s,1}] \cup [\phi_{s,3}, 2\pi)$.

For $\theta \in [\phi_{s,1}, \phi_{s,2}]$, $\rho_s(\theta)$ can be represented by a piecewise continuous curve, which consists of at most two pieces, corresponding to two intervals $[\phi_{s,1}, \alpha_s]$ and $[\alpha_s, \phi_{s,2}]$, where α_s is the angle θ of the intersection point between C_Π and the supporting line of s. Note that, if $\phi_{s,1} \leq \alpha_s$, then the curve of $\rho_s(\theta)$ has two pieces; otherwise, $\rho_s(\theta)$ is composed of one single piece.

For any $\theta \in [\phi_{s,1}, \alpha_s]$, as depicted in Figure 4(B), $\rho_s(\theta)$ is given by the angle between segments bt and bc', where c' is the intersection point between line segment s and the circle D_Π of radius r centered at b. If no intersection occurs between line segment s and circle D_Π, then $\rho_s(\theta)$ is given by the angle between segments bt and bc', where bc' intersects an endpoint of line segment s at an interior point of bc'.

Similarly, for any $\theta \in [\alpha_s, \phi_{s,2}]$, $\rho_s(\theta)$ is the angle between segments bt and bc', where bc' intersects an endpoint of line segment s at an interior point of bc' (see Figure 4(D)). Observe that $\rho_s(\theta) = 0$ when $\theta = \phi_{s,2}$. A crude plot of function $\rho_s(\theta)$ is shown in Figure 5.

Case 2: s lies outside C_Π and inside $C_\Pi{}'$. The analysis is similar to Case 1 and thus omitted.

Each of the curves $\rho_s(\theta)$ just described (for $\theta \in [\phi_{s,1}, \phi_{s,2}]$) is partially defined, continuous, and monotone over θ. Specifically, $\rho_s(\theta)$ is monotonically decreasing (resp. increasing) with respect to θ over the range of $[\phi_{s,1}, \phi_{s,2}]$ in Case 1 (resp. Case 2). In fact, the curves behave like pseudo-line segments, since any two curves can only intersect at most once.

Observe that $\rho_s(\theta)$ is an inverse trigonometric function. Nonetheless, we can easily define an algebraic function $f_s = \sin(\rho_s/2)$ in terms of variables x_b and y_b (i.e., the x- and y-coordinates of $b \in C_\Pi$). Since ρ_s is partially defined, continuous, and monotone over $\theta \in [\phi_{s,1}, \phi_{s,2}]$, so is function f_s (see Appendix D for

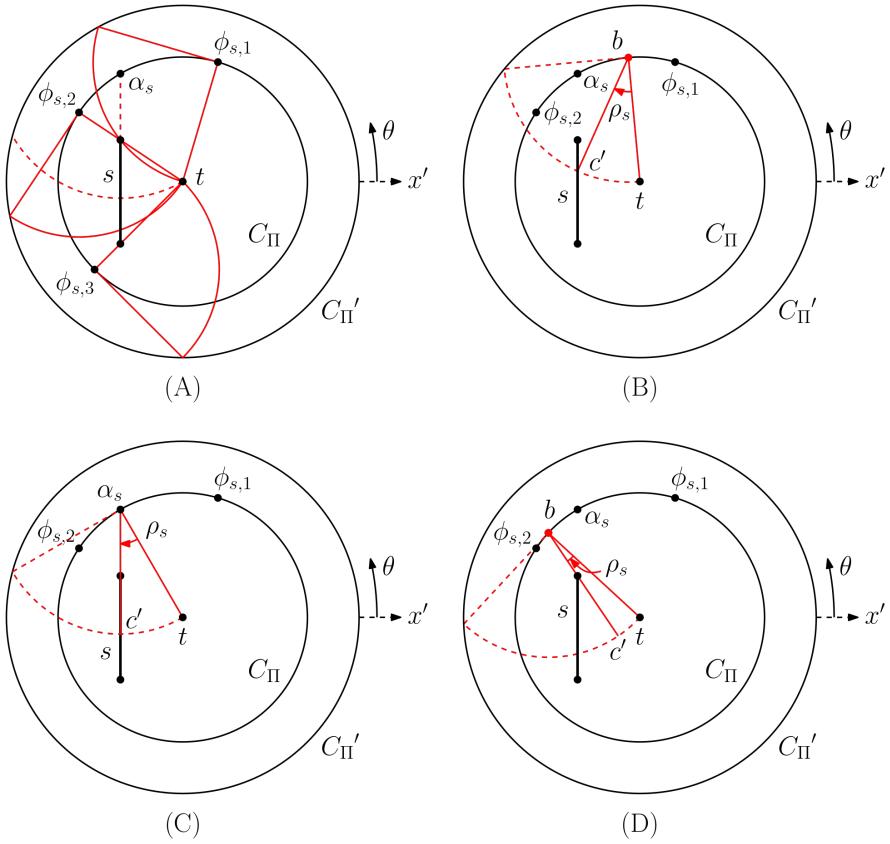


Figure 4: (A) Characterizing $\rho_s(\theta)$ for $\theta \in [\phi_{s,1}, \phi_{s,2}]$ in Case 1. Illustrations of $\rho_s(\theta)$ for (B) $\phi_{s,1} < \theta < \alpha_s$, (C) $\theta = \alpha_s$, and (D) $\alpha_s < \theta < \phi_{s,2}$, respectively.

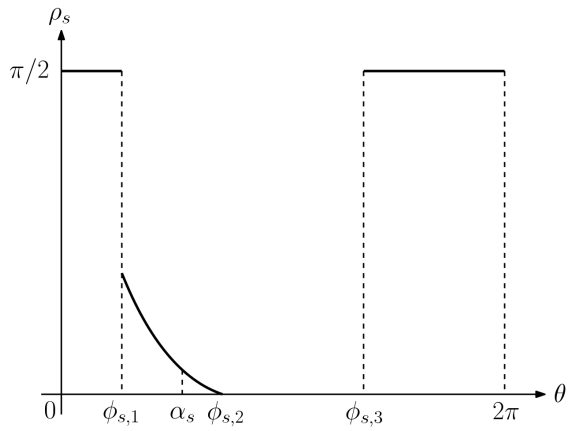


Figure 5: Illustration of function $\rho_s(\theta)$ in Case 1.

details).

At this point, we are in position to claim a new unified data structure for solving the circular sector emptiness query problem in 2D. This $\mathbb{R}^2$ data structure simplifies the two-step approach described in [4] (which consists of a circular arc intersection query and a circular sector emptiness query) while having the same time and space complexity.

Specifically, we construct two lower envelopes V_1 and V_2 of the piecewise algebraic curves f_s for all given line segments $s \in P$ as follows. V_1 is the lower envelope of the curves f_s for $y_b \geq 0$ (i.e., $0 \leq \theta \leq \pi$), whereas V_2 is for $y_b < 0$ (i.e., $\pi < \theta < 2\pi$). Note that V_1 and V_2 are computed as functions of x_b only, given that $y_b = +(r^2 - {x_b}^2)^{1/2}$ for $y_b \geq 0$, and $y_b = -(r^2 - {x_b}^2)^{1/2}$ for $y_b < 0$.

Since each pair of curves f_s intersect in at most one point, the size of the lower envelope is bounded by the third-order Davenport-Schinzel sequence, whose length is at most $O(n\alpha(n))$, where $\alpha(n)$ is the inverse Ackermann function. The lower envelope can be computed in $O(n \log n)$ time [7, 11].

Given a query circular sector σ, let b_σ be the apex of σ. If $y_{b_\sigma} \geq 0$, then x_{b_σ} is looked up in V_1 by using a binary search, which takes $O(\log n)$ time; otherwise, x_{b_σ} is looked up in V_2. Let ρ be the acute angle between the two bounding radii of σ. If $\sin(\rho/2)$ is less than $f_s(x_b)$ for all line segments $s \in P$, then σ does not intersect P. Hence, we have the following result.

Theorem 4 *A set P of n line segments in $\mathbb{R}^2$ can be preprocessed in $O(n \log n)$ time into a data structure of size $O(n\alpha(n))$ so that, for a query circular sector σ with*

a fixed radius r and a fixed arc endpoint t, one can determine if σ intersects P in $O(\log n)$ time.

Remarks. Notice that we do not actually need the function description of the lower envelope; it suffices to have an implicit representation, specifically the vertices of the lower envelope and the segments that define various pieces of the lower envelope. Generally, what is required to compute the lower envelope representation (of a given set of curves) includes i) the number of times each pair of curves intersect, ii) the ability to compute the intersection points between two given curves, and iii) the ability to determine whether one curve lies above or below another to the left and right of their intersection.

Recall that each pair of curves f_s only intersect in at most one point. The intersection between any two curves f_{s_i} and f_{s_j} (induced by segments s_i and s_j, respectively) can be determined algebraically in $O(1)$ time, given that the intersection corresponds to the angle θ at which bc' is supported by both s_i and s_j. We can also decide in constant time which of the two curves f_i and f_j, to the left and right of their intersection, lies above/below the intersection point (see Appendix E). At query time, given a circular sector σ (and its associated x_b), we can easily determine if σ intersects any line segment in P, without having to know the equation for the curve f_s defining the lower envelope at x_b, as follows. We retrieve the segment s that induces the curve f_s defining the lower envelope at x_b, and check if bc of σ intersects s. If bc of σ does not intersect s, then σ does not intersect P.

We now continue to derive our result for circular sector emptiness queries in 3D. For each triangle $\tau \in P$, we define a trivariate function $\rho_\tau(I, \Omega, \theta)$, so that $\rho_s(\theta)$ is characterized with respect to all planes (I, Ω). As with the two-dimensional case, $\rho_\tau(I, \Omega, \theta)$ is an inverse trigonometric function, based on which we can define an algebraic function f_τ (of constant degree) in terms of four variables (see Appendix F). Since ${x_b}^2 + {y_b}^2 + {z_b}^2 = r^2$, we can construct two lower envelopes V_1 and V_2 of the piecewise algebraic functions f_τ for all given triangles $\tau \in P$, such that V_1 is the lower envelope of f_τ for $z_b \geq 0$, and V_2 is for $z_b < 0$. As a result, V_1 and V_2 are functions of only three variables.

For constructing the lower envelope of the trivariate piecewise algebraic functions (of constant description complexity) just described, the best-known performance bounds are given by Koltun [8] and Agarwal et al. [1]. Specifically, according to Koltun [8], we can compute, deterministically, the lower envelope in $O(n^{4+\epsilon})$ time and store it in a data structure of size $O(n^{4+\epsilon})$ and query time $O(\log n)$, for any $\epsilon > 0$. On the other hand, Agarwal et al. [1] showed that the lower envelope can be constructed in randomized expected time $O(n^{3+\epsilon})$ and stored in an $O(n^{3+\epsilon})$-size data structure with a query time of $O(\log^2 n)$. Thus, we obtain the following result.

Theorem 5 *For any constant $\epsilon > 0$, a set P of n triangles in $\mathbb{R}^3$ can be preprocessed in $O(n^{4+\epsilon})$ time into a data structure of size $O(n^{4+\epsilon})$ so that, for a query circular sector σ with a fixed radius r and an endpoint of its arc located at t, one can determine whether σ intersects P in $O(\log n)$ time.*

Given that $O(n^4)$ queries are to be processed in the worst case, the following result is obtained.

Lemma 6 *A feasible articulated probe trajectory, if one exists, can be determined in $O(n^{4+\epsilon})$ time using $O(n^{4+\epsilon})$ space, for any constant $\epsilon > 0$.*

Since the space/time complexity of finding an extremal feasible articulated trajectory (Lemma 6) is dominant over that of the case of unarticulated trajectory (Lemma 3), the final result can be stated as follows.

Theorem 7 *One can determine if a feasible trajectory exists, and if so, report (at least) one such trajectory in $O(n^{4+\epsilon})$ time using $O(n^{4+\epsilon})$ space, for any constant $\epsilon > 0$.*

As an alternative, an $O(n^5)$-time algorithm with linear space usage is achievable by performing a simple $O(n)$-check on each of the $O(n^4)$ extremal trajectories. The proposed enumeration algorithm is easy to implement and could be quite fast in practice, since the enumeration stops once a feasible solution is found.

4 Conclusion

We have presented efficient data structures and algorithms for solving a trajectory planning problem involving a simple articulated probe in 3D space. In particular, we have shown that a feasible probe trajectory, among n triangular obstacles, can be found in $O(n^{4+\epsilon})$ time, for any constant $\epsilon > 0$. In the process, we have solved a special case of the circular sector emptiness query problem in 3D and simplified the corresponding data structure in 2D. We leave open the following questions: 1) Since our approach is enumerative, can we speed up the process of finding one feasible solution? 2) Is it possible to extend the current algorithm to finding feasible probe trajectories of a given (or maximum) clearance?

Acknowledgment. The authors would like to thank Pankaj K. Agarwal for helpful discussions and comments.

References

[1] P. K. Agarwal, B. Aronov, and M. Sharir. Computing envelopes in four dimensions with applications. *SIAM Journal on Computing*, 26(6):1714–1732, 1997.

[2] P. K. Agarwal and J. Matousek. On range searching with semialgebraic sets. *Discrete & Computational Geometry*, 11(4):393–418, 1994.

[3] O. Daescu, K. Fox, and K. Y. Teo. Computing trajectory with clearance for an articulated probe. In *28th Annual Fall Workshop on Computational Geometry*, 2018.

[4] O. Daescu, K. Fox, and K. Y. Teo. Trajectory planning for an articulated probe. In *30th Annual Canadian Conference on Computational Geometry*, pages 296–303, 2018.

[5] O. Daescu and K. Y. Teo. Characterization and computation of the feasible space of an articulated probe. In *34th Annual European Workshop on Computational Geometry*, 2019.

[6] M. de Berg, D. Halperin, M. Overmars, J. Snoeyink, and M. van Kreveld. Efficient ray shooting and hidden surface removal. *Algorithmica*, 12(1):30–53, 1994.

[7] J. Hershberger. Finding the upper envelope of n line segments in $O(n \log n)$ time. *Information Processing Letters*, 33(4):169–174, 1989.

[8] V. Koltun. Almost tight upper bounds for vertical decompositions in four dimensions. *Journal of the ACM*, 51(5):699–730, 2004.

[9] M. McKenna. Worst-case optimal hidden-surface removal. *ACM Transactions on Graphics*, 6(1):19–28, 1987.

[10] M. Pellegrini. Ray shooting on triangles in 3-space. *Algorithmica*, 9(5):471–494, 1993.

[11] M. Sharir and P. K. Agarwal. *Davenport-Schinzel sequences and their geometric applications*. Cambridge University Press, 1995.

[12] M. Sharir and H. Shaul. Semialgebraic range reporting and emptiness searching with applications. *SIAM Journal on Computing*, 40(4):1045–1074, 2011.

[13] N. Simaan, R. M. Yasin, and L. Wang. Medical technologies and challenges of robot-assisted minimally invasive intervention and diagnostics. *Annual Review of Control, Robotics, and Autonomous Systems*, 1:465–490, 2018.

Appendices

A Proof of Lemma 1

Let T be a feasible probe trajectory such that its final configuration is unarticulated. Specifically, in the final configuration of the probe, abc is a straight line segment, and point c coincides with t.

Let Π be an arbitrary plane passing through t and containing line segment abc. Assume that plane Π intersects at least one triangular obstacle. Observe that such an arbitrary plane Π always exists; otherwise, workspace S must be free of obstacles. Let T' be the trajectory resulting from rotating T about t in plane Π until T intersects an obstacle edge at either i) an endpoint or ii) an interior point.

If the intersection occurs at an endpoint v of the obstacle edge, then we have a feasible trajectory T' that is isolated by a support vertex v (since t and v define a unique line).

On the other hand, suppose that T' intersects the obstacle edge at an interior point. Let e_1 denote the intersecting obstacle edge, and Ψ be the unique plane containing e_1 and t. When T' is rotated about t in plane Ψ until T' intersects an endpoint of e_1 or some obstacle edge e_2, a new feasible trajectory T'' is obtained. If T'' intersects e_1 or e_2 at an endpoint, then T'' is isolated with respect to a support vertex. If T'' intersects e_2 at an interior point, then T'' is isolated with respect to two support edges (i.e., e_1 and e_2).

B Proof of Lemma 2

Let T denote a feasible probe trajectory such that its final configuration is articulated. Namely, in the final configuration of the probe, ab and bc are not collinear, and point c coincides with t.

Let Π be the unique plane containing ab and t. Since the whole trajectory of the probe is planar (i.e., in plane Π), without loss of generality, assume that bc of T is rotated clockwise about b to reach t in plane Π. Hereafter, for ease of discussion, bc and bt (of an articulated trajectory) stand for line segment bc of the probe in its intermediate and final configurations, respectively.

A feasible trajectory T' can be obtained by rotating ab of T about b in clockwise direction in plane Π until either ab and bt become collinear or ab intersects an obstacle edge e_1 outside C. In the former case, T' has an unarticulated final configuration, and by directly applying Lemma 1, we obtain an extremal feasible unarticulated trajectory as a result. In the latter case, let T'' be a trajectory resulting from rotating bt of T' about t counter-clockwise in plane Π while keeping ab intersecting e_1 until either 1) bt or 2) ab intersects some obstacle edge e_2. T'' is a feasible trajectory, and the proof is given in the Appendix of [4].

Case 1: bt intersects e_2 (inside C). Let p_1 be the intersection point between ab and e_1. Let $e_2{}'$ be the projection of e_2 onto the surface of sphere C from center t. Notice that $e_2{}'$ is a circular arc on C, and point b of trajectory T'' must lie on $e_2{}'$ as long as bt intersects e_2 (see Figure 6).

Let σ_{bct} denote the minor circular sector (with a central angle $\leq 90^\circ$) bounded by radii bc and bt, and let γ_{ct} be the

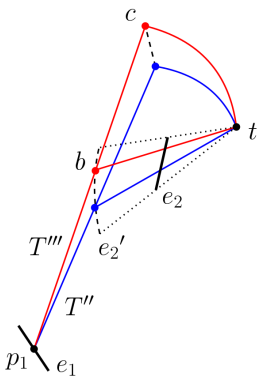


Figure 6: Illustration for Case 1.

circular arc of σ_{bct}. Let point b of T'' be translated along $e_2{}'$ in the direction such that the obtuse angle $\angle abt$ increases, while keeping ab intersecting e_1 at p_1, until a) bt, b) bc, c) ab, d) σ_{bct}, or e) γ_{ct} intersects with an obstacle edge e_3, f) γ_{ct} is tangent to the surface of some obstacle triangle, g) bt intersects with an endpoint of e_2, or h) ab becomes collinear with bt. Let T''' denote the resulting trajectory, as depicted in Figure 6. Since T''' is obtained by simply perturbing T'' until the trajectory comes into contact with an obstacle (or $\angle abt$ reaches 180°), T''' is feasible.

Case 1(a): bt intersects e_3. Note that two generic line segments and a point induce an incident line in 3D space. Since t is fixed and bt intersects two obstacle edges e_2 and e_3, bt is isolated (and thus point b is fixed).

Let Ψ be the unique plane containing e_1 and b. As illustrated in Figure 7 Case 1(a), let T'''' be the trajectory obtained from rotating abc of T''' around b in plane Ψ in the direction such that the obtuse angle $\angle abt$ increases until i) bc, ii) ab, iii) σ_{bct}, or iv) γ_{ct} intersects with some obstacle edge e_4, v) γ_{ct} is tangent to the surface of an obstacle triangle, vi) ab intersects an endpoint of e_1, or vii) ab becomes collinear with bt. Obviously, given that T'''' is derived from T''' through a straightforward perturbation, T'''' is feasible. Note that the detailed analyses for Cases 1(a)(i)–(vii) (as well as similar others in the cases that follow) are omitted due to their similarity to their parent cases.

Case 1(b): bc intersects e_3. Note that ab intersects e_1, and bc intersects e_3. Let $P(e_1, e_3)$ be the set of points q for which there exists a line segment starting at e_1, passing through q, and ending at e_3. Let I be the intersection of $P(e_1, e_3)$, C, and $e_2{}'$ (I is indicated, in Figure 7 Case 1(b), as the portion of the blue dashed line lying in the green shaded area). I is a circular arc on C. An isolated feasible trajectory can be obtained by translating point b of T''' along I in the direction such that the obtuse angle $\angle abt$ increases until i) bt, bc, ab, σ_{bct}, or γ_{ct} intersects with some obstacle edge e_4, ii) γ_{ct} is tangent to the surface of an obstacle triangle, iii) bt, bc, or ab reaches an endpoint of its currently intersecting obstacle edge, or iv) ab becomes collinear with bt.

Case 1(c): ab intersects e_3. Notice that ab intersects two edges e_1 and e_3. Assume, without loss of generality,

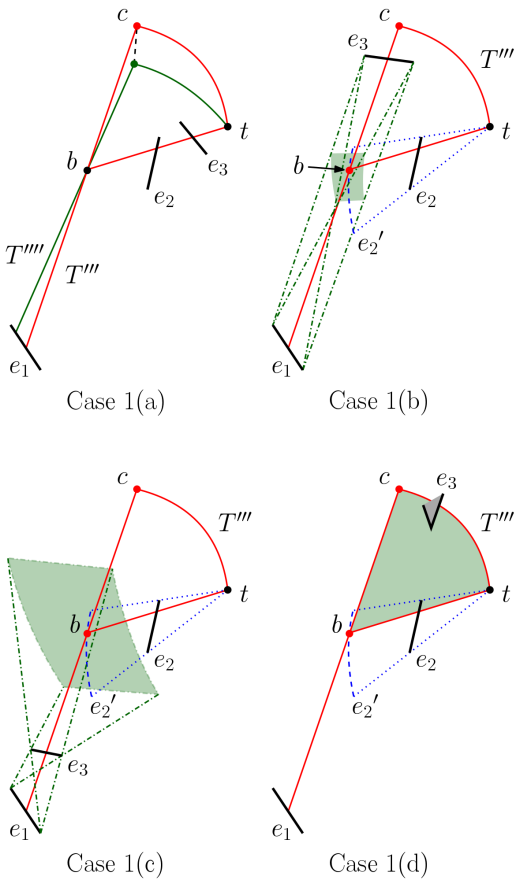


Figure 7: Illustrations for Cases 1(a)–(d).

that e_1 and e_3 are intersected by ab in the way depicted in Figure 7 Case 1(c). Let $P(e_1, e_3)$ be the set of points q for which there exists a ray starting at e_1, passing through e_3, and then through q. Let I be the intersection of $P(e_1, e_3)$, C, and $e_2{}'$. I is a circular arc on C. An isolated feasible trajectory can be obtained by moving point b of T''' along I in the direction that increases the obtuse angle $\angle abt$ until i) bt, bc, ab, σ_{bct}, or γ_{ct} intersects with an obstacle edge e_4, ii) γ_{ct} is tangent to the surface of an obstacle triangle, iii) bt or ab reaches an endpoint of its currently intersecting obstacle edge, or iv) ab becomes collinear with bt.

Case 1(d): σ_{bct} intersects e_3 (at one of its endpoints). Observe, as shown in Figure 7 Case 1(d), that T''' can be made isolated if point b of T''' is further translated along e_2', while keeping ab intersecting e_1 and maintaining the incidence between σ_{bct} and the endpoint of e_3, until i) bt, bc, ab, σ_{bct}, or γ_{ct} intersects with some obstacle edge e_4, ii) γ_{ct} is tangent to the surface of an obstacle triangle, iii) bt or ab reaches an endpoint of its currently intersecting obstacle edge, or iv) ab becomes collinear with bt. The resulting trajectory is feasible.

Case 1(e): γ_{ct} intersects e_3. Let D be a sphere of radius r with center b located on sphere C. Note that D passes through t. Consider the scenario that D is tangent to an

obstacle edge e_3. When D is rotated around t while maintaining its tangency to e_3, the center b of D translates along a curve $e_3{}'$ on C (see Figure 8).

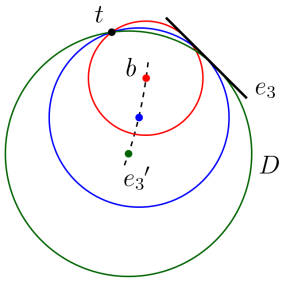


Figure 8: Illustration for Case 1(e). A top sectional view along the plane containing e_3 and t.

A plane can be defined by using two points on D and the center b of D. Consider a sphere D with a given center b on $e_3{}'$. D intersects t and is tangent to e_3 at a point p. Thus, we can define a plane Π_b containing t, p, and b, for any given b on $e_3{}'$.

Notice that γ_{ct} is a circular arc on D originating at t, and γ_{ct} must be on the circle of intersection between D and Π_b, for some b on $e_3{}'$. Since $e_2{}'$ and $e_3{}'$ intersect at a point (i.e., point b on C), point b is fixed. As a result, plane Π_b is set, and so is the circle on which γ_{ct} lies. If Π_b does not contain e_1, then abc is isolated. Otherwise, abc can be rotated about b, while continuing to intersect e_1, until either ab or bc intersects an obstacle edge (or an endpoint of e_1). In the end of either scenario, an extremal feasible trajectory is obtained.

Case 1(f): γ_{ct} is tangent to the surface of an obstacle triangle. The analysis is similar to Case 1(e). Let τ denote the obstacle triangle to whose surface γ_{ct} is tangent. As D is rotated around t while remaining tangent to the surface of τ, the center b of D moves along a curve κ_τ on C. Given that $e_2{}'$ and κ_τ intersect at a point, b is fixed. The same argument as in Case 1(e) immediately follows.

Case 1(g): bt intersects an endpoint of e_2. Given that t is fixed, bt of T''' is isolated by its intersection with an endpoint of e_2. A similar analysis as in Case 1(a) then follows.

Case 1(h): bc becomes collinear with bt. In this case, Lemma 1 is directly applicable, resulting in an extremal feasible unarticulated trajectory.

Case 2: ab intersects e_2 (outside C). Using similar arguments as given in Case 1, we can obtain an isolated feasible articulated trajectory such that i) ab intersects two edges, and ii) any of bt, bc, ab, σ_{bct}, and γ_{ct} are supported by at most two vertices, edges, and/or surfaces.

One may find some overlaps between Cases 1 and 2. On the whole, an extremal feasible articulated trajectory is characterized by its intersections (or tangencies) with at most four supports consisting of obstacle endpoints, edges, surfaces, or a combination of the three.

C Computing extremal articulated trajectories

Table 1 lists all the distinct cases of isolated articulated trajectories, each of which is indicated by the number of obstacle edges (i.e., support edges) intersected by ab, bc, bt, and γ_{ct}, and the number of obstacle endpoints (i.e., support vertices) incident on the interior of σ_{bct}.

Table 1: Extremal articulated trajectories.

Case	ab	bc	bt	σ_{bct}	γ_{ct}
1*	1	1	1	1	
2	1	2	1		
3	1	1	2		
4	1		1	2	
5	1		2	1	
6*	2		1	1	
7*	2	1		1	
8	2	1	1		
9	2	2			
10	2		2		
11	2			2	
12	3	1			
13	3		1		
14*	3			1	
15	4				
16	1		1		1
17	1	1	1		1
18	1		1	1	1
19	1		2		1
20	2	1			1
21	2		1		1
22*	2			1	1

For conciseness, certain scenarios are omitted from Table 1 given their trivial nature, and they include those involving the isolation of a feasible articulated trajectory due to incidence of its segment (i.e., ab, bc, or bt) with obstacle endpoints (i.e., support vertices). In addition, the cases in which γ_{ct} is tangent to the surface of an obstacle triangle are omitted due to their similarity to those where γ_{ct} intersects

an obstacle edge.

Note that the set of extremal articulated trajectories, in all cases besides those denoted by *, can be found by using a sequence of simple geometric operations (e.g., computing the intersection of two line segments or planes). An extremal articulated trajectory in each of the five cases designated by * can be computed by using an algebraic-geometric approach.

All in all, the worst-case running time for computing an extremal articulated trajectory is $O(n^4)$.

D Deriving algebraic function f_s based on $\rho_s(\theta)$

As one shall see, $\rho_s(\theta)$ is indeed an inverse trigonometric function, but we can define an algebraic function f_s as in the following discussion based on $\rho_s(\theta)$.

Recall that, in Case 1, we are only concerned with characterizing $\rho_s(\theta)$ for $\theta \in [\phi_{s,1}, \phi_{s,2}]$, which consists of at most two pieces, corresponding to two sub-intervals $[\phi_{s,1}, \alpha_s]$ and $[\alpha_s, \phi_{s,2}]$. Let us consider the two sub-intervals individually. Note that, in the following analysis, θ is represented by the x- and y-coordinates of $b \in C_\Pi$ (in order to obtain an algebraic expression).

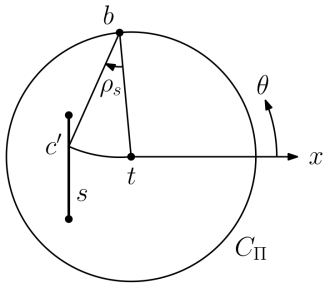


Figure 9: Case for $\theta \in [\phi_{s,1}, \alpha_s]$.

Sub-interval 1: $\theta \in [\phi_{s,1}, \alpha_s]$. For simplicity, let Π be the xy-plane (in which line segment s lies). Without loss of generality, let t be located at the origin. Let $x_{c'}$ and $y_{c'}$ denote the x- and y-coordinates of c', respectively. Observe that

$$\sin\frac{\rho_s}{2} = \frac{\sqrt{{x_{c'}}^2 + {y_{c'}}^2}}{2r} \tag{1}$$

Recall that c' is the intersection point between segment s and the circle D_Π of radius r centered at b (see Figure 9). The coordinates of c' can be expressed in terms of the coordinates of b in the following manner. Let ℓ be the supporting line of s. Line ℓ can be represented by $y = mx + d$, where m is the slope of ℓ, and d is the y-intercept of ℓ. Circle D_Π can be formulated as $(x–x_b)^2 + (y–y_b)^2 = r^2$, where x_b and y_b are the x- and y-coordinates of b, respectively. Then, $x_{c'}$ and $y_{c'}$ can be written as

$$x_{c'} = \frac{x_b + y_b m + dm \pm \sqrt{\delta}}{1+m^2} \tag{2}$$

$$y_{c'} = \frac{d + x_b m + y_b m^2 \pm \sqrt{\delta}}{1+m^2} \tag{3}$$

where $\delta = r^2(1+m^2) - (y_b - mx_b - d)^2$. Let $f_s = \sin(\rho_s/2)$ (i.e., Equation 1). Note that $f_s \in [0, \sqrt{1/2}]$. Clearly, by substituting Equations 2 and 3 into Equation 1, f_s can be expressed algebraically as a function of x_b and y_b, where $b \in C_\Pi$ and ${x_b}^2 + {y_b}^2 = r^2$.

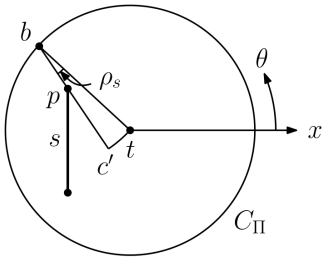


Figure 10: Case for $\theta \in [\alpha_s, \phi_{s,2}]$.

Sub-interval 2: $\theta \in [\alpha_s, \phi_{s,2}]$. Observe that Equation 1 also holds true in this case. Let p be the intersection point between s and bc' (see Figure 10). Note that p is an endpoint of s. Let x_p and y_p denote the x- and y-coordinates of p. Then, $x_{c'}$ and $y_{c'}$ can be expressed as

$$x_{c'} = x_b + \sqrt{\frac{r^2}{1 + (\frac{y_p - y_b}{x_p - x_b})^2}} \tag{4}$$

$$y_{c'} = y_b + \sqrt{\frac{r^2}{1 + (\frac{y_p - x_b}{y_p - y_b})^2}} \tag{5}$$

As before, if we let $f_s = \sin(\rho_s/2)$, then f_s can be expressed algebraically as a function of x_b and y_b by substituting Equations 4 and 5 into Equation 1.

A similar derivation of f_s can also be performed in Case 2. Given that ρ_s is partially defined, continuous, and monotone over $\theta \in [\phi_{s,1}, \phi_{s,2}]$, function $f_s = \sin(\rho_s/2)$ must also be the same.

E Computing implicit lower envelope

Consider the following approach for computing an implicit representation of the lower envelope. We only address the case of computing V_1 ($0 \le \theta \le \pi$), and the other case of V_2 ($\pi < \theta < 2\pi$) can be handled symmetrically.

Recall that each pair of curves f_s intersect at most once. Thus, as with computing the lower envelope of a set of n lines segments using a divide-and-conquer algorithm, we first sort the curves f_s by $\phi_{s,1}$ (note that, for a given line segment s, $\phi_{s,1}$ can be computed without knowing f_s, since $\phi_{s,1}$ corresponds to the smallest angle θ at which the circle of radius r centered at $b \in C_\Pi$ intersects with an endpoint of s). We divide the set of curves f_s into two equal sets by $\phi_{s,1}$, recursively compute the lower envelope of each set, and then merge the two lower envelopes to obtain the final result. The intersection of any two curves f_{s_i} and f_{s_j} (which correspond to lines segments s_i and s_j, respectively) can be determined as follows. There are two cases to be considered, depending on how segment bc' is supported by segments s_i and s_j.

Case A. Segment bc' intersects segments i and j at their endpoints (see Figure 11). Note that the corresponding x_b

can be computed algebraically, given that b is the intersection point between C_Π and bc'. We can also decide in constant time (based on the slope of bc') which of the two curves f_{s_i} and f_{s_j}, to the left and right of their intersection at x_b, lies above/below the intersection point.

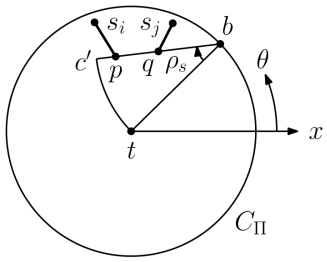


Figure 11: Case A.

Case B. Segment bc' intersects segment s_i such that c' coincides with an interior point of s_i, and intersects segment s_j at its endpoint (see Figure 12).

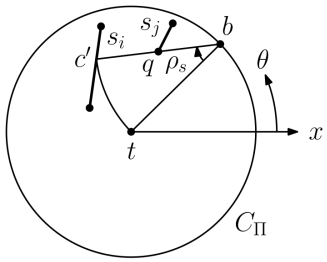


Figure 12: Case B.

Similar to the previous case, we can compute the corresponding x_b algebraically and determine in constant time which of the two curves f_{s_i} and f_{s_j}, to the left and right of their intersection at x_b, lies above/below the intersection point.

Notice that we did not actually compute a full description of the lower envelope. Instead, for each piece of the lower envelope, we store the information about segment s associated with the curve f_s to which the piece of the lower envelope belongs.

F Deriving algebraic function f_τ based on $\rho_\tau(I, \Omega, \theta)$

In the following analysis, Cartesian variables are used in place of I, Ω, and θ, in order to derive an algebraic expression for f_τ.

As with characterizing the univariate function $\rho_s(\theta)$, there are two cases to be considered, depending on whether a given triangle τ is located 1) inside sphere C or 2) outside sphere C and inside sphere C'.

Given the similarity in analysis between the two cases, we only present the arguments for Case 1. Similar as before, we are concerned with computing ρ_τ for two contiguous sub-intervals of (I, Ω, θ). In one of the sub-intervals, segment bc' always intersects the interior of τ at point c'. In the other sub-interval, segment bc' intersects an edge of τ. Let us examine the two sub-intervals individually.

Sub-interval 1: bc' intersects the interior of τ at c'. Without loss of generality, let t be located at the origin. Let $x_{c'}$, $y_{c'}$, and $z_{c'}$ denote the x-, y-, and z-coordinates of c', respectively. Observe that

$$f_\tau = \sin\frac{\rho_\tau}{2} = \frac{\sqrt{{x_{c'}}^2 + {y_{c'}}^2 + {z_{c'}}^2}}{2r} \tag{6}$$

Let G be the plane passing through points t and b. Plane G can be represented as $g_1x + g_2y + g_3z + g_4 = 0$, where $g_4 = 0$ and $g_3 = (-g_1x_b - g_2y_b)/z_b$. If we define $g = g_1/g_2$, then the expression for G becomes $gx + y + [(-g_1x_b - g_2y_b)/z_b]z = 0$. Given that plane G contains c',

$$gx_{c'} + y_{c'} + \left(\frac{-g_1x_b - g_2y_b}{z_b}\right) z_{c'} = 0 \tag{7}$$

Let H be the supporting plane of τ. Plane H can be expressed as $h_1x + h_2y + h_3z + h_4 = 0$, where h_1, h_2, h_3, and h_4 are the known parameters determined based on the three vertices of τ. Since H contains c',

$$h_1x_{c'} + h_2y_{c'} + h_3z_{c'} + h_4 = 0 \tag{8}$$

Notice that $|bc'| = r$. Thus,

$$(x_{c'} - x_b)^2 + (y_{c'} - y_b)^2 + (z_{c'} - z_b)^2 = r^2 \tag{9}$$

Based on the three Equations 7, 8, and 9, we can obtain an algebraic expression for $x_{c'}$, $y_{c'}$, and $z_{c'}$, respectively, in terms of variables g, x_b, y_b, and z_b. By substituting the resulting algebraic expressions for $x_{c'}$, $y_{c'}$, and $z_{c'}$ into Equation 6, f_τ can be expressed algebraically as a function (of degree one) of g, x_b, y_b, and z_b, where $b \in C$ and ${x_b}^2 + {y_b}^2 + {z_b}^2 = r^2$.

Sub-interval 2: bc' intersects an edge of τ. Let s denote the edge of τ intersected by bc'. Segment s can be expressed in parametric form as

$$\begin{aligned} x &= (1 - \lambda_s)x_u + \lambda_s x_v \\ y &= (1 - \lambda_s)y_u + \lambda_s y_v \\ z &= (1 - \lambda_s)z_u + \lambda_s z_v \end{aligned}$$

where u and v are the two endpoints of s, and $0 \le \lambda_s \le 1$. Let p denote the intersection point between bc' and s. The supporting line ℓ of segment bc' can be represented as

$$\begin{aligned} x &= (1 - \lambda_\ell)x_b + \lambda_\ell x_p \\ y &= (1 - \lambda_\ell)y_b + \lambda_\ell y_p \\ z &= (1 - \lambda_\ell)z_b + \lambda_\ell z_p \end{aligned}$$

where $\lambda_\ell \in \mathbb{R}$. Given that p lies in s, and line ℓ contains c',

$$x_{c'} = (1 - \lambda_\ell)x_b + \lambda_\ell[(1 - \lambda_s)x_u + \lambda_s x_v] \tag{10}$$

$$y_{c'} = (1 - \lambda_\ell)y_b + \lambda_\ell[(1 - \lambda_s)y_u + \lambda_s y_v] \tag{11}$$

$$z_{c'} = (1 - \lambda_\ell)z_b + \lambda_\ell[(1 - \lambda_s)z_u + \lambda_s z_v] \tag{12}$$

Note that Equation 9 still applies in this case. Substitute Equations 10, 11, and 12 into Equation 9, and solve for λ_ℓ in

respect of variables λ_s, x_b, y_b, and z_b. Then, by substituting the resulting expression for λ_ℓ into Equations 10, 11, and 12, we obtain an algebraic expression (of degree one) for $x_{c'}$, $y_{c'}$, and $z_{c'}$, respectively, in terms of variables λ_s, x_b, y_b, and z_b.

At last, by substituting the algebraic expressions for $x_{c'}$, $y_{c'}$, and $z_{c'}$ into Equation (6), f_τ can be expressed algebraically as a function of λ_s, x_b, y_b, and z_b, where $b \in C$ and ${x_b}^2 + {y_b}^2 + {z_b}^2 = r^2$.

The Lighthouse Problem*

Navigating by Lighthouses in Geometric Domains

Bengt J. Nilsson† Paweł Żyliński‡

Abstract

We study the computational properties of placing a minimum number of lighthouses in different geometric domains and under different notions of visibility, enabling a vehicle placed anywhere in the domain to navigate to a given specific target. This problem shares common elements with the art gallery problem in that the whole domain must be covered with as few lighthouses as possible. Our main result is an algorithm that places a minimum set of *strip lighthouses* in a simple rectilinear polygon. These correspond to sliding cameras in art gallery vernacular.

1 Introduction

We consider a problem lying in the intersection of *routing amongst obstacles* and *the art gallery problem*. Our problem is that of placing as few landmarks in a domain as possible such that a vehicle (being a ship, an airplane or a drone) can safely navigate the domain to reach a specified target. It is related to the routing problem since the vehicle should be guaranteed to avoid the obstacles while following a simple routing protocol to reach the target. At the same time, the landmarks must "cover" the whole domain to ensure that the vehicle can begin to navigate from any point in the domain, thus connecting our problem to the art gallery problem [18].

Beacon-based direct-visibility routing was used in the early days of aviation to guide (badly equipped) airplanes. The airplane would fly in the direction of the beacon until a beacon closer to the target could be seen from the plane. The plane would then continue in the direction of the new beacon, hopping from beacon to beacon until it reached the target [19]. Minimizing the number of beacons to place in a domain was therefore important to make beacon-based direct-visibility routing practically feasible. Herein, we consider the two-dimensional variant of this problem and to emphasize this, we use the concepts of *lighthouses* and *ships* rather than beacons and airplanes or drones.

*The authors were supported by the grants 2018-04001 (Swedish Research Council) and 2015/17/B/ST6/01887 (National Science Centre, Poland).

†Malmö University, Sweden, email bengt.nilsson.TS@mau.se

‡University of Gdańsk, Poland, email zylinski@inf.ug.edu.pl

Our navigation protocol for the ships is very simple but places certain restrictions on the placement of the lighthouses in the domains, sometimes making our placement problem computationally easier than the corresponding art gallery problem. Each lighthouse has an associated identifying number that is transmitted through the lighthouse signal. Thus, each ship can identify the lighthouse it is moving towards. Our *standard navigation protocol* specifies that the ship should move towards the lighthouse with the smallest identifying number that it has currently seen. The target always has identifying number 0 while the other lighthouses should have successively larger numbers as we move away from the target in the domain.

The Lighthouse Problem (LP)

> Given a domain and a *target* t in the domain, determine the minimum number of lighthouses, together with their locations and identifying numbers, ensuring that a ship starting from any position in the domain can travel to t with the *standard navigation protocol*: move towards the lighthouse with the smallest identifying number that is visible.

We can thus identify models of lighthouse problems by specifying different domain, lighthouse, and visibility types. We consider two general variants in this paper, defined by the domain type: in Section 2, we consider the lighthouse problem in grid domains, while in Section 3 — in rectilinear polygons, in both cases with strip lighthouses that define the lighthouse type and the visibility (the SLP problem for short). A *strip lighthouse* is an axis-aligned line segment l that can *guide/attract/is visible to* a point p in the domain, if the perpendicular projection of p onto l is not exterior to the domain. Finally, we also give some basic results for edge lighthouses in rectilinear polygons and for laser lighthouses in grids. A *laser* is a point that can illuminate in exactly one of the four compass directions.

Background. Our lighthouse problem is a variant of the art gallery problem, originally posed by Klee in 1973 as the question of determining the minimum number of guards sufficient to see every point of the interior of a simple polygon; for more details, see O'Rourke [18], Shermer [20], Urrutia [21] — combined with the concept

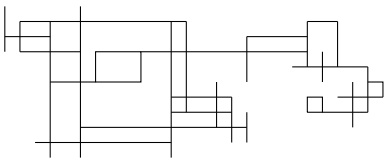


Figure 1: An example of a grid.

of cooperative guards [13, 15, 22] in which visibility-based connectivity between guards is required. We state this crucial property more formally.

Property 1 *If $\mathcal{L}$ is a solution to LP in some domain, then the visibility graph of $\mathcal{L}$ (under the visibility model considered) is connected.*

The visibility model we consider, immediately relates our lighthouse problem to the Minimum Sliding Cameras (MSC) problem introduced by Katz and Morgenstern [14] and then studied in [3, 4, 5, 6, 7, 8, 9, 12]. For that problem, combinatorial lower and upper bounds on the minimum number of sliding cameras are provided in [5, 12] and it is shown that the MSC problem is NP-hard in polygons with holes [5, 8] but admits a PTAS in simple polygons [3] — but so far NP-hardness in simple polygons still remains an open problem, although there exist linear-time exact algorithms with some additional assumptions either on the input polygon or the solution itself [4, 12].

For this visibility model, we may always assume that the strip lighthouses are maximal (within the domain) and furthermore, Property 1 translates to the fact that the union of strip lighthouses that together allow to navigate within the domain is a connected set.

2 Navigating in Grids

A *grid* is an arrangement of distinct vertical and horizontal closed line segments in the plane, where every two collinear line segments are disjoint and their union is connected; the grid can be thought of as a polygon with holes, representing a region of intersecting very thin corridors, see Figure 1 for an example. So a *strip lighthouse* in a grid is a subsegment of a grid segment and visibility is considered to be perpendicular to its direction along the grid segments intersecting that subsegment. Recall that a strip lighthouse is always assumed to be maximal and therefore we may identify it with the corresponding grid segment. Also, we assume that the target is a complete grid segment, to avoid issues with objects on that grid segment but not on the target.

Observe that for any feasible solution to the LP problem in a grid $\mathbf{G}$, any grid segment must be intersected by at least one lighthouse, so strip lighthouses must constitute a complete cover of $\mathbf{G}$. Since the union of strip lighthouses that together allow to navigate within $\mathbf{G}$ is a connected set, it follows from [15] that the LP problem in $\mathbf{G}$ can be solved by reduction to the minimum cooperative mobile guard set problem in the grid obtained from $\mathbf{G}$ by adding a new grid segment intersecting only the target segment (to force that target segment to be included in the optimal solution; identifying number may be then assigned in a greedy DFS-like manner, starting with 0 for the target segment). On the other hand, the minimum cooperative mobile guard set problem in $\mathbf{G}$ can be solved by reduction to the LP problem in that grid (by taking the best solution over those resulting from checking each of the grid segments as a possible candidate for the target segment). Consequently, as the problem of finding a minimum connected mobile guard set is NP-hard [15], we obtain the following result.

Corollary 2 *The SLP problem in grids is NP-hard.*

Furthermore, since there is one-to-one correspondence between a minimum cooperative mobile guard set in a grid $\mathbf{G}$ and the minimum dominating set of the intersection graph of $\mathbf{G}$, and Guha and Khuller [11] proposed an $O(\log \Delta)$-approximation algorithm for computing the minimum connected dominating set of a graph, where Δ is the maximum degree of that graph, and proved a lower bound of $\Omega(\log \Delta)$ even for bipartite graphs, we may conclude the following corollary.

Corollary 3 *The SLP problem in grids can be approximated with an $O(\log \Delta)$ approximation ratio, where Δ is the maximum number of intersections on a grid segment.*

3 Navigating in Rectilinear Polygons

Our main result of this section is a quadratic time algorithm for computing an optimum set of strip lighthouses in a simple rectilinear polygon. The input to the algorithm is a rectilinear polygon $\mathbf{P}$ and a specified target edge t of $\mathbf{P}$.

We first observe that the target t must be considered to be a strip lighthouse and so included in the optimum set of strip lighthouses for $\mathbf{P}$, with the identifying number 0. Following this observation, consider the *histogram partition* $\mathcal{H}$ of $\mathbf{P}$ with t as the base, see Figure 2(a), for which the dual graph T is a tree rooted at the histogram having t as its base, and with the *windows* of a histogram recursively acting as the *bases* for the relevant histograms corresponding to child nodes in T [16]. Recall that a *histogram* is a rectilinear x- or y-monotone polygon with one boundary chain being a line segment (called the *base*). Now, for each histogram $\mathbf{h}$ in $\mathcal{H}$, we associate the *base direction* of $\mathbf{h}$ to be the direction towards its base, denoted $b_{\mathbf{h}}$, and for a set $\mathcal{L}$ of strip lighthouses in $\mathbf{P}$, we define the *canonical set*

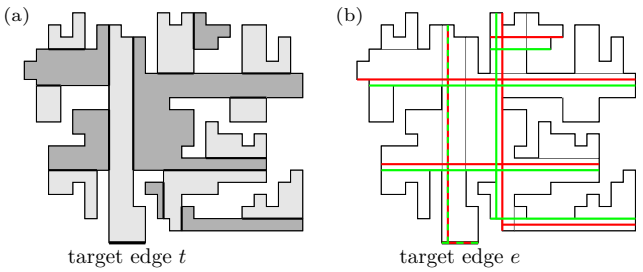


Figure 2: (a) Partition into histograms, starting from the target edge t. (b) An optimal solution (red) to the LP problem and the relevant canonical one (green).

with respect to $\mathcal{L}$ as follows. First, we modify the set $\mathcal{L}$ by recursively considering each histogram $\mathbf{h}$ in $\mathcal{H}$ corresponding to a leaf in T. For every strip lighthouse l intersecting $\mathbf{h}$ and being parallel to $b_{\mathbf{h}}$ (assuming l to be maximal), we move l continuously in the base direction as far as possible while not decreasing the visibility region of l. If $\mathbf{h}$ contains several such lighthouses, the movement is done for each lighthouse in sequence in the order of decreasing distance to the base $b_{\mathbf{h}}$. Once this process has been completed for each lighthouse in $\mathbf{h}$ parallel to $b_{\mathbf{h}}$, we remove any lighthouses that coincide, thus reducing the size of the canonical set. Next, we remove the corresponding node from T and repeat the process until all histograms of $\mathcal{H}$ have been considered; see Figure 2(b) for an example. Observe that any strip lighthouse in a canonical set intersecting a histogram and being parallel to the base, is not completely contained in the histogram.

Our idea is then to associate a canonical feasible set of strip lighthouses in $\mathbf{P}$ with a set $\mathcal{S}_{\mathbf{h}}$ of pairs of 0/1-valued intervals on the base $b_{\mathbf{h}}$ of each histogram $\mathbf{h}$ that a maximal strip lighthouse orthogonal to $b_{\mathbf{h}}$ can intersect. For convenience, we identify the first interval in an interval pair as blue and the second one as red, respectively, using $B_{\mathbf{h}}$ and $R_{\mathbf{h}}$ to denote them when no confusion arise; an interval denoted $I_{\mathbf{h}}$ could be either blue or red.

For the purpose of defining the set $\mathcal{S}_{\mathbf{h}}$, we start with a few definitions. Consider two histograms $\mathbf{h}$ and $\mathbf{h}'$, $\mathbf{h}$ being a child of $\mathbf{h}'$ in T, and assume for the definition that the base direction of $\mathbf{h}$ is down and the base direction of $\mathbf{h}'$ is left; see Figure 3. When we henceforth define objects in histograms, we will always make this assumption, thereby avoiding having to define each object also for the other seven possible cases. Let $I_{\mathbf{h}}$ be an interval lying on the base $b_{\mathbf{h}}$ (which is a window of $\mathbf{h}'$). We first project the endpoint of $I_{\mathbf{h}}$ that is closest to the base $b_{\mathbf{h}'}$ vertically onto the opposite horizontal edge of $\mathbf{h}'$ — this gives us two endpoints of a vertical line segment in $\mathbf{h}'$. Next, we project this line segment horizontally onto the base $b_{\mathbf{h}'}$ — this gives us the interval $I_{\mathbf{h}'}$. Following this sequence of two projections,

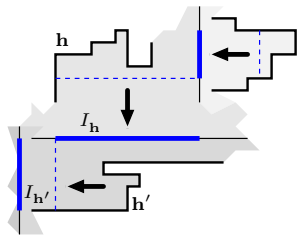


Figure 3: Illustrating the propagation of intervals.

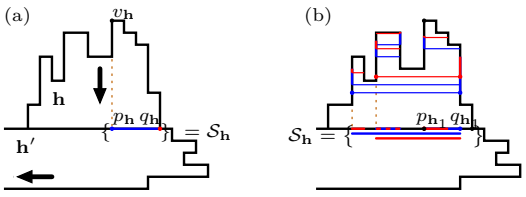


Figure 4: Defining the histogram intervals.

we say that $I_{\mathbf{h}}$ *propagates* from $\mathbf{h}$ to $I_{\mathbf{h}'}$ in $\mathbf{h}'$ in one step, which is denoted by $I_{\mathbf{h}'} = \mathrm{pr}(I_{\mathbf{h}})$. An interval can thus be propagated using a sequence of one-step propagations from a histogram to any ancestral histogram in T; again see Figure 3. Finally, we need to define some special points in a histogram $\mathbf{h}$; see Figure 4(a). First, the point $q_{\mathbf{h}}$ is the rightmost point of $b_{\mathbf{h}}$. Next, we follow the boundary of $\mathbf{h}$ from $q_{\mathbf{h}}$ in counterclockwise order along the xy-monotone staircase until the end of that staircase at vertex $v_{\mathbf{h}}$ is encountered (the vertex $v_{\mathbf{h}}$ is a convex vertex with the adjacent vertical edge below $v_{\mathbf{h}}$ and the adjacent horizontal edge to the right of $v_{\mathbf{h}}$) — projecting a point vertically from $v_{\mathbf{h}}$ onto $b_{\mathbf{h}}$ defines the point $p_{\mathbf{h}}$.

Thereby, the encoding $\mathcal{S}_{\mathbf{h}}$ is defined recursively as follows (see Figure 4(b)):

$$\mathcal{S}_{\mathbf{h}} = \Big\{ \big(\mathrm{pr}(B_{\bar{\mathbf{h}}}), \mathrm{pr}(R_{\bar{\mathbf{h}}})\big) \,|\, (B_{\bar{\mathbf{h}}}, R_{\bar{\mathbf{h}}}) \in \mathcal{S}_{\bar{\mathbf{h}}}, \forall \bar{\mathbf{h}} \in T_{\mathbf{h}} \Big\} \cup P_{\mathbf{h}}, \quad (1)$$

where $T_{\mathbf{h}}$ denotes the set of all child histograms of $\mathbf{h}$ in T, and $P_{\mathbf{h}} = \emptyset$ if there exists a pair $(B_{\bar{\mathbf{h}}}, R_{\bar{\mathbf{h}}}) \in \mathcal{S}_{\bar{\mathbf{h}}}$ for some $\bar{\mathbf{h}} \in T_{\mathbf{h}}$ such that $\mathrm{pr}(R_{\bar{\mathbf{h}}}) \cap [p_{\mathbf{h}}, q_{\mathbf{h}}] \neq \emptyset$, and $P_{\mathbf{h}} = \big\{[p_{\mathbf{h}}, q_{\mathbf{h}}], [q_{\mathbf{h}}, q_{\mathbf{h}}]\big\}$ otherwise. All but $[q_{\mathbf{h}}, q_{\mathbf{h}}]$ intervals in $\mathcal{S}_{\mathbf{h}}$ are then valued 1, while the interval $[q_{\mathbf{h}}, q_{\mathbf{h}}]$, if it belongs to $\mathcal{S}_{\mathbf{h}}$, is valued 0.

The set $\mathcal{S}_{\mathbf{h}}$ can be computed in linear time, given $\mathcal{S}_{\bar{\mathbf{h}}}$ for all child histograms $\bar{\mathbf{h}}$ of $\mathbf{h}$ in T, by a pass over the boundary and using a stack data structure. Thus, the sets $\mathcal{S}_{\mathbf{h}}$ for each $\mathbf{h}$ in T can be computed in quadratic time.

Now, given the encoding $\mathcal{S}_{\mathbf{h}}$, we define a *realization* $\Gamma_{\mathbf{h}}$ of $\mathcal{S}_{\mathbf{h}}$ to be a set of intervals such that $\Gamma_{\mathbf{h}}$ contains at most one interval from each interval pair in $\mathcal{S}_{\mathbf{h}}$. Each such realization $\Gamma_{\mathbf{h}}$ can be associated with the minimum canonical set $\mathcal{L}(\Gamma_{\mathbf{h}})$ of strip lighthouses such that each 1-valued interval in $\Gamma_{\mathbf{h}}$ is intersected by a lighthouse in

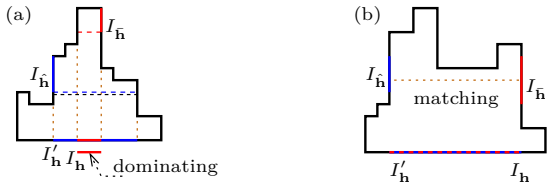


Figure 5: Illustrating domination and matching.

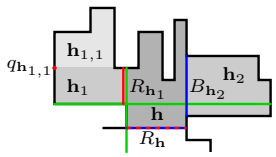


Figure 6: Illustrating the proof of Lemma 4. The green segments are the strip lighthouses in $\mathcal{L}(\Gamma_{\mathbf{h}})$.

$\mathcal{L}(\Gamma_{\mathbf{h}})$. Clearly, the number of possible realizations of $\mathcal{S}_{\mathbf{h}}$ is at most $3^{|\mathcal{S}_{\mathbf{h}}|}$. Note that all realizations do not necessarily correspond to parts of solutions to the SLP problem in $\mathbf{P}$.

Before we present the crucial recursive formula for the size of $\mathcal{L}(\Gamma_{\mathbf{h}})$, we need to introduce additional definitions. Let $I_{\mathbf{h}}$ and $I'_{\mathbf{h}}$ be two intervals on $b_{\mathbf{h}}$ such that $I_{\mathbf{h}} = \mathrm{pr}(I_{\bar{\mathbf{h}}})$ and $I'_{\mathbf{h}} = \mathrm{pr}(I_{\hat{\mathbf{h}}})$ for some $\bar{\mathbf{h}}, \hat{\mathbf{h}} \in T_{\mathbf{h}}$; see Figure 5(a). We say that $I_{\mathbf{h}}$ *dominates* $I'_{\mathbf{h}}$ if the subhistogram of $\mathbf{h}$ formed by cutting along the line segment with one endpoint at the lower endpoint of $I_{\hat{\mathbf{h}}}$ and the other endpoint by its horizontal projection in $\mathbf{h}$ contains $I_{\bar{\mathbf{h}}}$. If $I_{\mathbf{h}}$ is not dominated by any interval in a realization $\Gamma_{\mathbf{h}}$, we call $I_{\mathbf{h}}$ a *master* for $\Gamma_{\mathbf{h}}$. Next, we say that $I_{\mathbf{h}}$ and $I'_{\mathbf{h}}$ *match* if there is a point on $I_{\bar{\mathbf{h}}}$ whose horizontal projection in $\mathbf{h}$ lies on $I_{\hat{\mathbf{h}}}$; see Figure 5(b).

Now, let $V_{\Gamma_{\mathbf{h}}}$ denote the number of master intervals among the intervals in $\Gamma_{\mathbf{h}}$ and let $M_{\Gamma_{\mathbf{h}}}$ denote the number of distinctly matched intervals in $\Gamma_{\mathbf{h}}$, i.e., any interval is matched to at most one other interval. It is clear that $V_{\Gamma_{\mathbf{h}}}$ corresponds to the number of vertical strip lighthouses required in the realization $\Gamma_{\mathbf{h}}$, while $M_{\Gamma_{\mathbf{h}}}$ corresponds to the number of horizontal strip lighthouses that traverse $\mathbf{h}$ going from one child histogram of $\mathbf{h}$ to another. We thereby set

$$s(\Gamma_{\mathbf{h}}) = \begin{cases} 0 & \text{if } \mathbf{h} \text{ is a leaf and } \Gamma_{\mathbf{h}} = \{[q_{\mathbf{h}}, q_{\mathbf{h}}]\} \\ 1 & \text{if } \mathbf{h} \text{ is a leaf and } \Gamma_{\mathbf{h}} = \{[p_{\mathbf{h}}, q_{\mathbf{h}}]\} \\ V_{\Gamma_{\mathbf{h}}} - M_{\Gamma_{\mathbf{h}}} + \sum_{\bar{\mathbf{h}} \in T_{\mathbf{h}}} s(\Gamma_{\bar{\mathbf{h}}}) & \text{otherwise,} \end{cases}$$

which allows us to state the following lemma.

Lemma 4 *For every histogram* $\mathbf{h}$*, there is a realization* $\Gamma_{\mathbf{h}}$ *from* $\mathcal{S}_{\mathbf{h}}$ *such that* $\mathcal{L}(\Gamma_{\mathbf{h}}) \cup \{b_{\mathbf{h}}\}$ *is a solution to the SLP problem in* $\mathbf{P}_{\mathbf{h}}$ *and* $s(\Gamma_{\mathbf{h}}) = |\mathcal{L}(\Gamma_{\mathbf{h}})|$*, where* $\mathbf{P}_{\mathbf{h}}$ *is the subpolygon of* $\mathbf{P}$ *consisting of the histograms in* T *rooted at* $\mathbf{h}$*.*

Proof. Consider an optimal canonical solution $\mathcal{L}^*_{\mathbf{h}}$ to the LP problem in $\mathbf{P}_{\mathbf{h}}$. We prove that $\mathcal{L}^*_{\mathbf{h}}$ has a corresponding realization $\Gamma^*_{\mathbf{h}}$ of $\mathcal{S}_{\mathbf{h}}$ inductively in a bottom up fashion. We consider each subhistogram $\mathbf{h}_i$ in $T_{\mathbf{h}}$ separately.

If $\mathbf{h}_i$ is a leaf histogram in $T_{\mathbf{h}}$, then $\mathcal{L}^*_{\mathbf{h}}$ either has a lighthouse that intersects $[p_{\mathbf{h}_i}, q_{\mathbf{h}_i}]$ or there is a lighthouse in $\mathcal{L}^*_{\mathbf{h}}$ in the ancestor histogram of $\mathbf{h}_i$ for which the projection onto $b_{\mathbf{h}_i}$ intersects $q_{\mathbf{h}_i}$, otherwise not all of $\mathbf{h}_i$ is seen by $\mathcal{L}^*_{\mathbf{h}}$; see Figure 6 illustrating both these cases. Both these intervals $B_{\mathbf{h}_i} = [p_{\mathbf{h}_i}, q_{\mathbf{h}_i}]$ and $R_{\mathbf{h}_i} = [q_{\mathbf{h}_i}, q_{\mathbf{h}_i}]$ are paired in $\mathcal{S}_{\mathbf{h}}$ by construction and only one of them is used by $\mathcal{L}^*_{\mathbf{h}}$.

If $\mathbf{h}_i$ is an internal histogram in $T_{\mathbf{h}}$, then consider those histograms $\mathbf{h}_j$ that are children of $\mathbf{h}_i$ in $T_{\mathbf{h}_i}$. By induction, any (maximal) lighthouse in $\mathcal{L}^*_{\mathbf{h}}$ intersect the bases of $\mathbf{h}_j$ in at most one interval per pair lying in $\mathcal{S}_{\mathbf{h}_j}$ and furthermore these lighthouses intersect $\mathbf{h}_i$ between two (vertical) boundary edges (assuming $\mathbf{h}_i$ has our standard orientation). These two boundary edges define a (horizontal) interval and since $\mathcal{L}^*_{\mathbf{h}}$ is a solution to the SLP problem in $\mathbf{P}_{\mathbf{h}}$, it must have a lighthouse intersecting this interval. The interval is propagated from an interval in $\mathbf{h}_j$ and is therefore in an pair in $\mathcal{S}_{\mathbf{h}}$. The other interval from that pair is not propagated by induction, concluding the proof. □

The above lemma has the following immediate corollary.

Corollary 5 *There is a realization* $\Gamma_{\mathbf{h}_t}$ *from* $\mathcal{S}_{\mathbf{h}_t}$ *such that* $\mathcal{L}(\Gamma_{\mathbf{h}_t}) \cup \{t\}$ *is a solution to the SLP problem in* $\mathbf{P}$ *and* $s(\Gamma_{\mathbf{h}_t}) = |\mathcal{L}(\Gamma_{\mathbf{h}_t})|$*, where* $\mathbf{h}_t$ *is the root histogram in* T*.*

Clearly, an brute-force implementation of the above approach results in an exponential time algorithm for computing a solution to the SLP problem in a rectilinear polygon. However, in the following, we show that an "optimal" representation — and so an optimal set of strip lighthouses as well — can be (re-)constructed recursively, in a more efficient way, using some additional data when constructing the relevant sets $\mathcal{S}_{\mathbf{h}}$ of interval pairs.

For a given histogram $\mathbf{h}$, let $\Gamma^*_{\mathbf{h}}$ be an *optimal feasible* realization of $\mathcal{S}_{\mathbf{h}}$, i.e., a realization such that:

- $\mathcal{L}(\Gamma^*_{\mathbf{h}})$ has the smallest number of strip lighthouses among the realizations obtainable from $\mathcal{S}_{\mathbf{h}}$,
- the union of these lighthouses is a connected set,
- the whole of $\mathbf{P}_{\mathbf{h}}$ is covered.

It follows from Lemma 4 that $\mathcal{L}(\Gamma^*_{\mathbf{h}}) \cup \{b_{\mathbf{h}}\}$ constitutes an optimal solution to the SLP problem in $\mathbf{P}_{\mathbf{h}}$. Note that $\mathcal{L}(\Gamma^*_{\mathbf{h}})$ is canonical in the sense defined above.

Lemma 6 *For every histogram* $\mathbf{h}$*, there is an optimal feasible realization* $\Gamma^*_{\mathbf{h}}$ *using only red intervals matched if possible, except when*

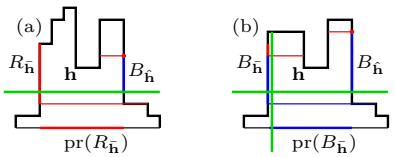


Figure 7: Illustrating the proof of Lemma 6. The green segments are the strip lighthouses.

- *a red and a blue interval* (*but not the corresponding red intervals*) *match, or*
- *two blue intervals match* (*but not the corresponding red or red/blue intervals do*)*, at least one of which is a master,*

in which cases these matched intervals are used.

Proof. We sketch an inductive proof maintaining the following invariant: *the realization* $\Gamma^*_{\mathbf{h}}$ *induces as few strip lighthouses as possible and additionally the used intervals are as large as possible.*

If $\mathbf{h}$ is a leaf, the invariant holds trivially. On the other hand, if $\mathbf{h}$ is not a leaf, we assume without loss of generality, that considering intervals from two child histograms $\bar{\mathbf{h}}$ and $\hat{\mathbf{h}}$ in $T_{\mathbf{h}}$, then $\mathrm{pr}(R_{\bar{\mathbf{h}}}) \subset \mathrm{pr}(B_{\bar{\mathbf{h}}})$ and $\mathrm{pr}(R_{\hat{\mathbf{h}}}) \subset \mathrm{pr}(B_{\hat{\mathbf{h}}})$. We have four main cases.

1. The used interval $I_{\mathbf{h}}$ comes from the pair $P_{\mathbf{h}}$, see Equality (1).

   The argument holds trivially by the same consideration as for leaf histograms.

2. The used intervals $\mathrm{pr}(R_{\bar{\mathbf{h}}})$ and $\mathrm{pr}(R_{\hat{\mathbf{h}}})$ match with $R_{\bar{\mathbf{h}}}$ in $\Gamma^*_{\bar{\mathbf{h}}}$ and $R_{\hat{\mathbf{h}}}$ in $\Gamma^*_{\hat{\mathbf{h}}}$.

   Assume $\Gamma^*_{\mathbf{h}}$ does not contain both $\mathrm{pr}(R_{\bar{\mathbf{h}}})$ and $\mathrm{pr}(R_{\hat{\mathbf{h}}})$, then at least one of their corresponding blue intervals is used and by replacing it with the red interval, the size of the solution can be reduced by one contradicting optimality.

3. The used intervals $\mathrm{pr}(R_{\bar{\mathbf{h}}})$ and $\mathrm{pr}(B_{\hat{\mathbf{h}}})$ match with $R_{\bar{\mathbf{h}}}$ in $\Gamma^*_{\bar{\mathbf{h}}}$ and $R_{\hat{\mathbf{h}}}$ in $\Gamma^*_{\hat{\mathbf{h}}}$.

   We can assume that the red interval $\mathrm{pr}(R_{\hat{\mathbf{h}}})$ in the same pair as $\mathrm{pr}(B_{\hat{\mathbf{h}}})$ does not match $\mathrm{pr}(R_{\bar{\mathbf{h}}})$ (otherwise an optimal realization would have picked those two intervals, see Case 2). Assume $\Gamma^*_{\mathbf{h}}$ does not contain both of $\mathrm{pr}(R_{\bar{\mathbf{h}}})$ and $\mathrm{pr}(B_{\hat{\mathbf{h}}})$. If $\Gamma^*_{\mathbf{h}}$ contains $\mathrm{pr}(R_{\bar{\mathbf{h}}})$ and $\mathrm{pr}(R_{\hat{\mathbf{h}}})$ (not matching), it will use as many strip lighthouses but since $\mathrm{pr}(R_{\hat{\mathbf{h}}})$ is a subinterval of $\mathrm{pr}(B_{\hat{\mathbf{h}}})$, it violates the invariant; see Figure 7(a).

4. The used intervals $\mathrm{pr}(B_{\bar{\mathbf{h}}})$ and $\mathrm{pr}(B_{\hat{\mathbf{h}}})$ match and at least one of $\mathrm{pr}(R_{\bar{\mathbf{h}}})$ and $\mathrm{pr}(R_{\hat{\mathbf{h}}})$ is a master with $R_{\bar{\mathbf{h}}}$ in $\Gamma^*_{\bar{\mathbf{h}}}$ and $R_{\hat{\mathbf{h}}}$ in $\Gamma^*_{\hat{\mathbf{h}}}$.

   We can assume that the interval $\mathrm{pr}(R_{\bar{\mathbf{h}}})$ does not match $\mathrm{pr}(B_{\hat{\mathbf{h}}})$ and that $\mathrm{pr}(B_{\bar{\mathbf{h}}})$ does not match $\mathrm{pr}(R_{\hat{\mathbf{h}}})$ (otherwise an optimal realization would have picked one of those two pairs of matching intervals, see Case 3). Assume $\Gamma^*_{\mathbf{h}}$ contains both $\mathrm{pr}(R_{\bar{\mathbf{h}}})$ and $\mathrm{pr}(R_{\hat{\mathbf{h}}})$, then $\mathbf{h}$ must contain two horizontal strip lighthouses (assuming the standard orientation of $\mathbf{h}$), one intersecting each of $R_{\bar{\mathbf{h}}}$ and $R_{\hat{\mathbf{h}}}$, and two vertical strip lighthouses, one intersecting each of the horizontal ones. However, this solution violates the invariant since it can be replaced by one horizontal strip lighthouse intersecting both $B_{\bar{\mathbf{h}}}$ and $B_{\hat{\mathbf{h}}}$, since $\mathrm{pr}(B_{\bar{\mathbf{h}}})$ and $\mathrm{pr}(B_{\hat{\mathbf{h}}})$ match, and one vertical strip lighthouse in $\mathbf{h}$ intersecting the horizontal one. The size of the solution remains the same but the interval $\mathrm{pr}(B_{\bar{\mathbf{h}}}) = \mathrm{pr}(B_{\hat{\mathbf{h}}})$ contains each of $\mathrm{pr}(R_{\bar{\mathbf{h}}})$ and $\mathrm{pr}(R_{\hat{\mathbf{h}}})$; see Figure 7(b).

Enumerating the remaining possibilities, it follows that in these cases the invariant holds by always selecting and propagating the red intervals. □

The algorithm makes passes over the boundary of the histogram and computes, given $\mathcal{S}_{\bar{\mathbf{h}}}$ for each child histogram $\bar{\mathbf{h}}$, a maximal set of unique red/red matching intervals, given these, a maximal set of unique red/blue matching intervals, and given these, a maximal set of unique blue/blue matching intervals where at least one is a master. After the matching intervals have been established the algorithm selects the remaining red ones for an optimal solution.

It is easy to verify that the running time of the above algorithm is quadratic (with respect to the number of vertices of the input polygon $\mathbf{P}$).

Theorem 7 *A solution to the SLP problem for a simple rectilinear polygon* $\mathbf{P}$ *and a target edge* t *can be computed in quadratic time.*

For rectilinear polygons with holes, a simple modification of the NP-hardness proof for optimally guarding a rectilinear polygon with holes [8], adding a target notch in a corner of the construction shows the result.

Theorem 8 *The SLP problem in rectilinear polygons with holes is NP-hard.*

3.1 Edge Lighthouses

Another natural model is to assume that an edge lighthouse sees all points of the histogram with this edge as its base. Unfortunately, our standard navigation protocol can get stuck in this model and therefore another navigation protocol is needed: *If stuck at a lighthouse with the identifying number* l*, then move towards the lighthouse with the maximum number you see until you see a lighthouse with an identifying number smaller than* l (*and then continue with the standard protocol until you*

either reach the target or get stuck again). This modification is sufficient (for the existence of a solution), since by taking all edges as lighthouses and labelling them consecutively along the boundary (with 0 at the target, and then increasingly in counterclockwise manner), we obtain a feasible solution.

4 Laser Lighthouses in Grids

In grids, a natural counterpart of (directed) edge lighthouses in rectilinear polygons, are laser lighthouses. Specifically, a *laser lighthouse* is a point which can illuminate only towards one of the forth directions: North, East, South or West. We begin with two simple but crucial observations.

Observation 1 *For any grid, n is a lower bound.*

Observation 2 *Putting a laser not at the endpoint of a segment results in two lasers associated with this segment, so in order to have only n lasers, each segment must have at least one endpoint being an intersection point and all of the lasers must be located at some of these intersection end-points.*

Taking into account the above two remarks, one can easily observe that $2n$ is an upper bound. The idea is to place two lasers on each grid segment, starting from the (at most) two segments that the target is located at (and so, at most four lasers in total), in opposite direction, and continue with each new, so far uncovered segment intersecting some covered one, by placing another two lasers, at the intersection point, in the directions alternative to the intersected already covered segment. Therefore, we have a simple linear-time 2-approximation algorithm.

In general, the problem is polynomially tractable; recall that by Observation 1, each segment must be assigned at least one laser. The idea of our algorithm is as follows. For a grid $\mathbf{G} = V_{\mathbf{G}} \cup H_{\mathbf{G}}$, where $V_{\mathbf{G}}$ ($H_{\mathbf{G}}$, resp.) is the set of all vertical (horizontal, resp.) line segments of $\mathbf{G}$, we first construct the weighted directed bipartite intersection graph $G_{\mathbf{G}}$, with the bipartition $(V_{\mathbf{G}}, H_{\mathbf{G}})$ [17], where the weight $w(a)$ of an arc $a = (x, y)$, corresponding to the intersection of line segments x and y, is set to 1 if x has an endpoint on y, and 2 otherwise. (The graph $G_{\mathbf{G}}$ can be constructed in $O(n \log n + m)$ time, where $m = O(n^2)$ is the number of intersection points of grid segments of $\mathbf{G}$ [2].) Then we modify the graph $G_{\mathbf{G}}$ by adding the new vertex t that corresponds to the target point t, and by adding at most two arcs (z, t), depending on the location of t on a line segment z, with the weight of (z, t) equal to 1 if t is located at the endpoint of z, and 2 otherwise.

Let $D_{\mathbf{G}}$ be the resulting digraph from the above modification of $G_{\mathbf{G}}$. One can that observe that any arborescence T_t of $D_{\mathbf{G}}$ with the root t corresponds to a feasible laser assignment in $\mathbf{G}$ with the number of used lasers equal to the cost of T_t, and *vice versa*. Therefore, there is one-to-one correspondence between any arborescence of $D_{\mathbf{G}}$ with the root at t and an optimal solution to the Lighthouse Problem in $\mathbf{G}$. Consequently, since the problem of computing a minimum spanning tree of a weighted digraph can be solved in $O(n \log n + m)$ time [10], where n and m are the number of vertices and edges of the input graph, respectively, we immediately obtain the following result.

Theorem 9 *The Laser Lighthouse Problem in grids can be solved in $O(n \log n + k)$ time, where n is the number of grid segments of the input grid while k is the number of their intersection points.*

References

[1] Avis, D., Toussaint, G.T.: An optimal algorithm for determining the visibility of a polygon from an edge. IEEE Transactions on Computers **30**(12), 910–914 (1981)

[2] Balaban, I.J.: An optimal algorithm for finding segments intersections. In: Proceedings of the 11th Annual ACM Symposium on Computational Geometry, 211–219, ACM New York (1995)

[3] Bandyapadhyay, S., Roy, A.B.: Effectiveness of local search for Art Gallery Problems. In: Ellen, F., Kolokolova, A., Sack, J.-R. (eds.) WADS 2017, LNCS, vol. 10389, pp. 49–60. Springer, Cham (2017). DOI 10.1007/978-3-319-62127-2.

[4] de Berg, M., Durocher, S., Mehrabic, S.: Guarding monotone art galleries with sliding cameras in linear time. Journal of Discrete Algorithms **44**, 39–47 (2017)

[5] Biedl, T., Chan, T.M., Lee, S., Mehrabi, S., Montecchiani, F., Vosoughpour, H.: On guarding orthogonal polygons with sliding cameras. In: Poon, H.-H., Rahman, S.M., Yen, H.-Ch. (eds.) WALCOM 2017, LNCS, vol. 10167, pp. 54-65. Springer, Cham (2017). DOI 10.1007/978-3-319-53925-6.

[6] Biedl, T., Chan, T.M., Lee, S., Mehrabi, S., Montecchiani, F., Vosoughpour, H., Yu, Z.: Guarding orthogonal art galleries with sliding k-transmitters: hardness and approximation. Algorithmica **81**(1), 69–97 (2019)

[7] Durocher, S., Filtser, O., Fraser, R., Mehrabi, A.D., Mehrabi, S.: A (7/2)-approximation algorithm for guarding orthogonal art galleries with sliding cameras. In: Pardo A., Viola A. (eds.) LATIN 2014, LNCS, vol. 8392, pp. 294–305. Springer, Heidelberg (2014). DOI 10.1007/978-3-642-54423-1.

[8] Durocher, S., Mehrabi, S.: Guarding orthogonal art galleries using sliding cameras: algorithmic and hardness results. In: Chatterjee, K., Sgall, J. (eds.) MFCS 2013, LNCS, vol. 8087, pp. 314–324. Springer, Heidelberg (2013). DOI 10.1007/978-3-642-40313-2.

[9] Durocher, S., Mehrabi, S.: A 3-approximation algorithm for guarding orthogonal Art Galleries with sliding cameras. In: Chatterjee, K., Sgall, J. (eds.) IWOCA 2014, LNCS, vol. 8986, pp. 140–152. Springer, Cham (2014). DOI 10.1007/978-3-319-19315-1.

[10] Gabow, H.N., Galil, Z., Spencer, T., Tarjan, R.E.: Efficient algorithms for finding minimum spanning trees in undirected and directed graphs. Combinatorica **6**(2), 109–122 (1986)

[11] Guha, S., Khuller, S.: Approximation algorithms for connected dominating sets. Algorithmica **20**(4), 374–387 (1998)

[12] Györi, E., Mezei, T.R.: Mobile versus point guards. Discrete & Computational Geometry **61**(2), 421–451 (2019)

[13] Hernández-Peñalver, G.: Controlling guards. In: Proceedings of the 6th Canadian Conference on Computational Geometry, 387–392, University of Saskatchewan (1994)

[14] Katz, M.J., Morgenstern, G.: Guarding orthogonal art galleries with sliding cameras. International Journal of Computational Geometry & Applications **21**(2), 241–250 (2011)

[15] Kosowski, A., Małafiejski, M., Żyliński, P.: Cooperative mobile guards in grids. Computational Geometry **37**(2), 59–71 (2007)

[16] Levcopoulos, Ch.: Heuristics for Minimum Decompositions of Polygons. Ph. D. Thesis, Linköping University (1987)

[17] Ntafos, S.: On gallery watchman in grids. Information Processing Letters **23**(2), 99–102 (1986)

[18] O'Rourke, J.: Art Gallery Theorems and Algorithms. Oxford University Press, New York (1987)

[19] Scaramuzza, D., et al.: Vision-controlled micro flying robots: from system design to autonomous navigation and mapping in GPS-denied environments. IEEE Robotics & Automation Magazine. **21**(3), 26–40 (2014).

[20] Shermer, T.: Recent results in Art Galleries. Proceedings of the IEEE **80**(9), 1384–1399 (1992)

[21] Urrutia, J.: Art gallery and illumination problems. In: Sack J.-R., Urrutia J. (eds), Handbook of Computational Geometry, pp. 973–1027. Elsevier, North Holland (2000)

[22] Żyliński, P.: Cooperative guards in art galleries. Dissertationes Mathematicae **450**, 132 p., Polish Academy of Sciences (2008)

An Optimal Algorithm for Maintaining Connectivity of Wireless Network on a Line

Shimin Li* Zhongjiang Yan† Jingru Zhang‡

Abstract

A problem of generating a connected graph by moving the points on a line is studied in this paper. Given n points on a line L and a positive value r, if the distance of two points is not larger than r, these two points are connected by an edge in the generated graph G. The goal of this problem is to move n points along the line such that the graph G is connected and the maximum moving distance of the points is minimized. We propose a linear-time algorithm to solve this problem based on the greedy technique in this paper.

1 Introduction

We consider the problem of maintaining the connectivity of the network by moving points in it along a line in this paper. In this problem, we are given a set of n points that represent wireless sensors or other devices with the same communication radius r on a line L. If the distance between any pair of points is equal to or smaller than r, then they can communicate with each other through wireless connection directly. The connection between any pair of points can be represented by a graph G, in which each point is a vertex and there exists an edge connecting two vertices if the distance of those two points is no more than r. The goal of our problem is to move all the n points along L such that the graph G of n points is connected and the maximum movement of these points is minimized.

One application of the algorithm of this problem is the connectivity maintenance of the 1-D vehicular ad hoc network [13]. In a vehicular ad hoc network, each vehicle is a node in the ad hoc network constructed by all the vehicles on the road. If the distance of two vehicles is equal to or less than the wireless communication radius r, then these two vehicles can communicate with each other through wireless connection directly. To guarantee the transmission of information between vehicles on accidents or road conditions, it is necessary to keep the ad hoc network connected so that all the vehicles can send and receive the information.

The algorithm can also be used to maintain the connectivity of a wireless sensor network. In a wireless sensor network, a sensor can communicate with another one directly if the distance between these two sensors is at most r. Through the wireless ad hoc network built upon the connecting of sensors, the data collected by all the sensors can be transmitted to the server. In this situation, the connectivity is critical to gather the data from all the sensors. Further, our algorithm can also be used to generate a continuous coverage on a line by moving the wireless sensors, where the r is the sensing radius of sensor. Because of the generality of this problem, it must have a large number of applications arising from a variety of fields.

1.1 Previous Work

This problem arises from the mobile wireless network connectivity problem [5] where the vertices(sensors) move in random directions, there is an edge connecting two vertices if their distance is no more than a given value, the target is to maintain the graph connectivity. Díaz et al. provide an analytical model for the connectivity of dynamic random geometric graphs which is used for mobile wireless networks [5]. In the model, the vehicles move in random directions, and if the distance between a pair of mobiles is not greater than a given value, there exists an edge connecting them. A similar connectivity model for wireless ad hoc networks with communication constraints is also presented in [1, 7]. The relationship between the node degree and k-connectivity of wireless multihop network is also researched by Bettstetter [2].

Cheng et al. solve the mobile ad hoc network connectivity problem by deploying relay nodes to the network [3]. Biconnectivity of the wireless network is also considered with the application of connected mobile robots [4]. In general, k-connectivity problem with unreliable network links is considered and solved by Zhao [14]. The coverage problems of wireless sensor network attracted quite a lot of attention from researchers and in some situations it is necessary to maintain the connectivity of the wireless network [6, 8, 9, 10, 11]. Specifically, the probability of a static wireless ad hoc

*Department of Computer Science, Winona State University, shimin.li@winona.edu

†School of Electronics and Information, Northwestern Polytechnical University, zhjyan@nwpu.edu.cn

‡Department of Computer Science, University of Texas-Rio Grande Valley, jingru.zhang@utrgv.edu

network being biconnected is presented by Tian et al. [12].

1.2 Our Approaches

We solve this problem by using a greedy technique which adds the points to the connected graph G one by one from left to right. During the process of this greedy algorithm, the positions of the added points are computed and updated without moving any point explicitly. At last, we just need to move each point to its final position found by our greedy algorithm to obtain an optimal solution to the problem. In the following, we say moving the point instead of moving the position of the point for simplicity. Without loss of generality, let the line L be the x-axis with the origin at the position of the leftmost point. Suppose all the n points $p_1, p_2, \ldots, p_n$ are in the increasing order of their x-coordinates.

Initially, the graph G consists of only p_1 and the maximum moving distance of points $d_{\max} = 0$ since it is not necessary to move p_1. Then the following $n-1$ points are added to G one by one from left to right. During this process, the connectivity of G is maintained by moving the points close enough along L when it is necessary. Consider the general case where the connected G consists of points $p_1, p_2, \ldots, p_i$. To add the next point p_{i+1} into G and keep its connectivity, we have to move p_{i+1} leftwards if the distance between p_i and p_{i+1} is larger than r. Further, to minimize the maximum moving distance, it is necessary to move points of G and p_{i+1} towards each other if p_{i+1} is very far away from the current location of p_i. It seems like that we have to move each point of G rightwards or leftwards in the worst case and update the maximum moving distance each time, which leads us to an $O(n^2)$ algorithm. To obtain a linear-time algorithm, we maintain several sequences of points of G each satisfying several properties such that the points in a sequence can be shifted together. By maintaining the sequences of points of G, we are able to achieve a linear-time algorithm for this problem. Further, in each iteration, we obtain an optimal solution to the problem with the considered points so far. Therefore, if the input points are already sorted, our algorithm is an online algorithm for this problem.

The rest of the paper is organized as follows. Section 2 shows the description of our algorithm. In detail, we present the general case of adding the next point on L to the connected graph G in Section 2.2 after presenting the initial case in Section 2.1. Then the process of merging two sequences is introduced in Section 2.3. In Section 3, we prove the correctness of our algorithm and analyze the running time at last.

2 The Algorithm Description

Let $P = \{p_1, p_2, \ldots, p_n\}$ be the set of n points on the x-axis. Suppose all the n points on the x-axis are already sorted (break the tie arbitrarily) and the first point is at the origin. Let x_i be the initial position of p_i, so we have $x_1 = 0 \leq x_2 \leq \ldots \leq x_n$. Before the introduction to our algorithm, we first present a key observation on the *order preserving property*.

Observation 1 *There must exist an optimal solution where all the points keep the same order as that in the input.*

Proof. Suppose we have an optimal solution OPT with a different order of points. In OPT, there exists at least one pair of points p_i and p_j satisfy $x_i \leq x_j$ and $x'_j \leq x'_i$. Let us prove that we can obtain another solution OPT' by swapping p_i and p_j without increasing the maximum movement of all the points in OPT. Based on the order of the four positions, we have four cases to consider.

1. Case $x'_j \leq x'_i \leq x_i \leq x_j$ or $x'_j \leq x_i \leq x'_i \leq x_j$. In this case, we must have $|x'_j x_i| \leq |x'_j x_j|$ and $|x'_i x_j| \leq |x'_j x_j|$.
2. Case $x_i \leq x'_j \leq x'_i \leq x_j$. In this case, we have $|x_i x'_j| \leq |x_i x'_i|$ and $|x'_i x_j| \leq |x'_j x_j|$.
3. Case $x'_j \leq x_i \leq x_j \leq x'_i$. This case is symmetric to the above case, so we have $|x'_j x_i| \leq |x'_j x_j|$ and $|x_j x'_i| \leq |x_i x'_i|$.
4. Case $x_i \leq x_j \leq x'_j \leq x'_i$ or $x_i \leq x'_j \leq x_j \leq x'_i$. This case is symmetric to the first case, so we have $|x_i x'_j| \leq |x_i x'_i|$ and $|x_j x'_i| \leq |x_i x'_i|$.

To sum up, among all the above four cases, we can always swap the positions of p_i and p_j in the OPT and obtain another optimal solution OPT'. All the pairs of points with different orders can be eliminated by this swapping process and the maximum movement does not increase. By swapping a pair of points with different order without increasing $d_{\max}$, we can obtain another optimal solution with the order of points in the input. Therefore, this observation holds. □

By Observation 1, the *order preserving property* holds in this problem, which means that we can find an optimal solution to the problem without changing the order of the points in the input. Thus, suppose all the points are indexed from left to right on L, then each point p_i has a unique index i and holds that index in the optimal solution. In the following sections, we always use the index i to denote the i-th point on L for $1 \leq i \leq n$.

To avoid moving the points one by one along the line L, we maintain a group of sequences of points with some properties. Denote by S_j the j-th sequence from left to

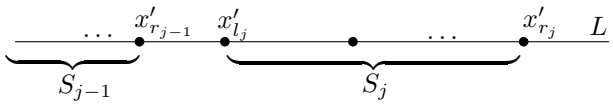


Figure 1: Illustrating the sequence S_j of points.

right. Let $l_j(resp., r_j)$ be the index of the leftmost(resp., rightmost) point of sequence S_j, respectively. See Fig. 1 for an example of a sequence. Specially, if p_i is the only point in S_j, then we have $l_j = r_j = i$. Denote by C_j the number of points in sequence S_j. Denote by $d(l, j)$(resp., $d(r, j)$) the index of the point which moved the maximum distance towards left (resp., right) along L in S_j. Let D_j be the maximum moving distance of points in sequence S_j. All the sequences are stored in a stack T with all the above parameters. Let m be the number of sequences in stack T. The properties of the sequences are in the following.

1. In sequence S_j, the distance between any pair of adjacent points is exactly r.
2. There exist at least a pair of points in S_j that move D_j towards each other.
3. For any pair of adjacent sequences in stack T, say S_j and S_{j+1}, it is true that $|r_j l_{j+1}| < r$ for $1 \leq j < m$.

During the process of our algorithm, we add the points to the connected graph G one by one. After processing each newly added point, the properties mentioned above still hold. Once all the n points were processed, the n positions of these points are found. Then we move each point to its position computed by our algorithm to obtain an optimal solution to the problem. Now let us present the initial case of adding the first point into G.

2.1 The Initial Case

Initially, let G be a graph with only one vertex p_1 and set the position of it at its original position, i.e., let $x'_1 = x_1 = 0$. Thus, the moving distance of p_1 satisfies $d_1 = |x_1 x'_1| = 0$ at the beginning. Initialize the maximum moving distance of points by $d_{\max} = d_1 = 0$, because p_1 is the only point processed by our algorithm so far. There is only one sequence S_1 with only one point p_1 in it after the process of adding the first point to G. Thus, we have $l_1 = r_1 = 1$ and $C_1 = 1$ hold at the beginning of our algorithm. The p_1 in S_1 is not moved at all, so we have $D_1 = 0$. Push the sequence S_1 to the stack T. Obviously, the graph G is a connected graph, since it consists of only one vertex currently. Now we finish the process of the first point in the input. In the following, we will present how the general case is handled.

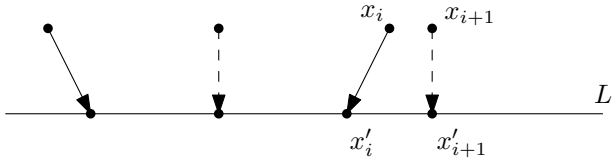


Figure 2: Illustrating the first subcase of the general case where $|x'_i x_{i+1}| < r$.

2.2 The General Case

Suppose we just processed the point p_i on the line L in the input, now we are considering the next point p_{i+1}. Before the processing of the point p_{i+1}, we assume the following loop invariants hold. It is apparent that the loop invariants hold in the initial case.

1. The stack T contains $m \geq 1$ sequences, $\{S_1, S_2, \ldots, S_m\}$.
2. The maximum moving distances of points in these sequences are D_1, D_2, ..., and D_m, respectively.
3. All the sequences hold all the three properties we mentioned above.
4. The graph G consisting of the points in $\{p_1, p_2, \ldots, p_i\}$ is connected.

All invariants are true for the processed i points, so we are done if there is no more points on L in the input.

Now suppose there is another point p_{i+1} in the input to the right side of p_i. We will show how to move points in a greedy way such that the maximum moving distance increases as small as possible and the graph G is still connected. Let us present the process of adding the next point p_{i+1} to the connected graph G.

Denote by x'_i the new position of p_i after the movement, then based on the distance between x'_i and x_{i+1}, we have three different cases to consider.

The Case $|x'_i x_{i+1}| < r$: In this case, we do not need to move any point because the distance between x'_i and x_{i+1} is small enough to keep the connectivity of the graph G. See Fig. 2 for an example of this case. Add the point p_{i+1} to the graph G at its original position, i.e., $x'_{i+1} = x_{i+1}$. Then create a new sequence S_{m+1} containing the point p_{i+1} only, because we have $|x'_i x'_{i+1}| < r$. There is only one point p_{i+1} in sequence S_{m+1}, so we have $C_{m+1} = 1$ and $l_{m+1} = r_{m+1} = i+1$. Because we do not move the position of p_{i+1}, initialize the other variables of the sequence S_{m+1} by $d(l, m+1) = d(r, m+1) = i + 1$ and $D_{m+1} = 0$. At last, push S_{m+1} into the stack T and increase m by 1.

The Case $r \leq |x'_i x_{i+1}| \leq r + D_m$: We move the point p_{i+1} to $x'_i + r$ in this case to maintain the connectivity of the graph G without increasing $d_{\max}$, the maximum moving distance of points. See Fig. 3 for an example of this case. The moving distance of p_{i+1} is $d_{i+1} =$

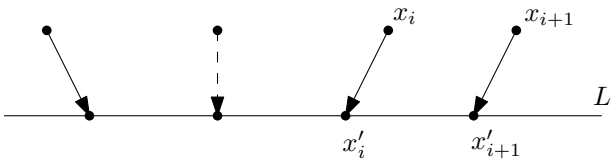


Figure 3: Illustrating the second subcase of the general case where $r \leq |x'_i x_{i+1}| \leq r + D_m$.

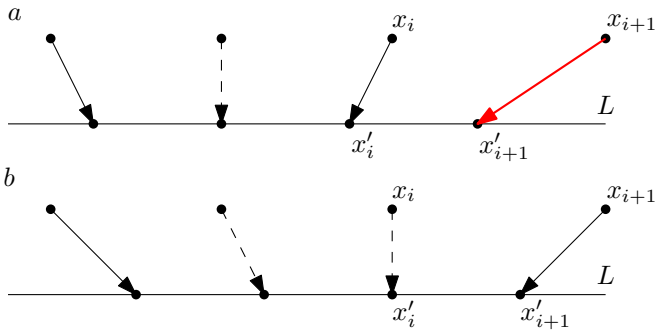


Figure 4: Illustrating the third subcase of the general case where $|x'_i x_{i+1}| > r + D_m$.

$|x'_i x_{i+1}| - r$ and it holds that $d_{i+1} \leq D_m$, because we have $d_{i+1} = |x'_i x_{i+1}| - r \leq D_m$ in this case. Add p_{i+1} to the last sequence S_m by setting $r_m = i+1$ and increasing C_m by 1. Now we have done the process of this case.

The Case $|x'_i x_{i+1}| > r + D_m$: It is not enough to move p_{i+1} leftwards only to restore the connectivity of G without increasing $d_{\max}$ in this case, because the moving distance of p_{i+1} would be larger than D_m. Hence, we have to move points in sequence S_m and p_{i+1} towards each other such that $|x'_i x'_{i+1}| = r$ and the maximum moving distance of the points in $\{p_{i+1}\} \cup S_m$ is minimized. See Fig. 4 for an example of this case. In the following, we will show how to handle this more complicated case.

Since the distance between any adjacent pair of points is exactly r in S_m, we can shift the points in S_m altogether. Note that once the sequence S_m is shifted rightwards, the distance between the leftmost point in S_m and the rightmost point of S_{m-1} might be larger than r. For this reason, we need to check the value of $|x'_{r_{m-1}} x'_{l_m}|$ and shift sequences S_{m-1} and S_m if $|x'_{r_{m-1}} x'_{l_m}| > r$. Further, once we move S_{m-1} and S_m towards each other such that $|x'_{r_{m-1}} x'_{l_m}| = r$, then these two sequences would be merged into one sequence. Thus, we continue to check the distance between the last two sequences in T, i.e., $|x'_{r_{m-1}} x'_{l_m}|$, until we find that the distance is not larger than r or there is only one sequence in T. During this process, the maximum moving distance $d_{\max}$ might increase.

Let us consider the process of moving S_m and p_{i+1} first. In this case, we move p_{i+1} from x_{i+1} to $x_{i+1} - D_m - \frac{|x'_i x_{i+1}| - r - D_m}{2}$ and then shift all the points in S_m rightwards by $\frac{|x'_i x_{i+1}| - r - D_m}{2}$. Now the distance between x'_i and x'_{i+1} is r, so add the point p_{i+1} to S_m by updating the values of r_m as $r_m = i+1$ and increasing C_m by 1. Increase D_m, the maximum moving distance of the points in S_m, to $D_m + \frac{|x'_i x_{i+1}| - r - D_m}{2}$, since the maximum moving distance of $p_{d(r,m)}$ was increased after the shifting of S_m. If we have $D_m > d_{\max}$ after the increasing, then update the value of $d_{\max}$ to D_m. Now we added the point p_{i+1} to the last sequence S_m.

Because the sequence S_m was shifted rightwards during the above process, this might cause the distance between $p_{r_{m-1}}$ and p_{l_m} larger than r. If $|x'_{r_{m-1}} x'_{l_m}| \leq r$ still holds after the rightward shifting of S_m, then we have finished the process of the point p_{i+1}. In the other case, we have to shift the last two sequences towards each other and merge them into one sequence until $|x'_{r_{m-1}} x'_{l_m}| \leq r$ or $m = 1$ is true. If neither of the above two conditions is true, we keep merging the last two sequences in T. The process of merging two sequences will be shown in Section 2.3. Let us present the processing result of this case first. If we have $|x'_{r_{m-1}} x'_{l_m}| \leq r$, then the last two sequences are connected, so the graph G is also connected. If we have $m = 1$, then there is only one sequence in the stack T. By the properties of the sequence, the distance between any adjacent points in that sequence is exactly r, so G is obviously connected. Now we are done with the point p_{i+1} and the connectivity of G is maintained.

2.3 Merging Two Sequences

If $m > 1$ and $|x'_{r_{m-1}} x'_{l_m}| > r$, then we merge the last two sequences, S_{m-1} and S_m, by shifting them towards each other until $|x'_{r_{m-1}} x'_{l_m}| = r$. Note that the maximum moving distances of points in S_{m-1} and S_m are D_{m-1} and D_m, respectively. To minimize the maximum moving distance of points in the merged sequence, we have to move these two sequences based on the values of D_{m-1} and D_m. Intuitively, to minimize the maximum moving distance in the merged sequence, we always move the sequence with the smaller maximum moving distance by a relatively longer distance, so the maximum moving distances to the opposite directions are always the same in the merged sequence. The moving distances of S_{m-1} and S_m can be computed in constant time by the following formulas. Based on the values of D_{m-1}, D_m, and $|x'_{r_{m-1}} x'_{l_m}|$, we have three different cases to consider.

1. $|x'_{r_{m-1}} x'_{l_m}| - r \leq D_{m-1} - D_m$

   In this case, we just move S_m leftwards such that $|x'_{r_{m-1}} x'_{l_m}| = r$, then the sequences S_{m-1} and S_m are merged into one sequence, S_{m-1}. Because S_{m-1} is not moved at all during this process, the merging process stops here. Once S_{m-1} and S_m are merged, we update r_{m-1} by $r_{m-1} = r_m$ and increase C_{m-1} by C_m. At last, decrease m by 1,

because of the decrement of the count of sequences in stack T.

2. $|D_{m-1} - D_m| < |x'_{r_{m-1}}x'_{l_m}| - r$

   In this case, we move S_{m-1} and S_m towards each other such that $|x'_{r_{m-1}}x'_{l_m}| = r$. The points in these two sequences will be shifted together as we are moving two points. The moving distances of these two sequences can be computed by the following formulas,

$$\begin{cases} \Delta_{m-1} &= \frac{|x'_{r_{m-1}}x'_{l_m}|-r-D_{m-1}+D_m}{2}, \\ \Delta_m &= \frac{|x'_{r_{m-1}}x'_{l_m}|-r+D_{m-1}-D_m}{2}. \end{cases}$$

   Where Δ_{m-1} and Δ_m are the shifting distances of S_{m-1} and S_m, respectively.

   Note that after the shifting of S_{m-1} and S_m by these two distances towards each other, the maximum moving distances of the points in these two sequences are the same but to opposite directions. Therefore, the maximum moving distance of points is minimized after the merging process. At last, we update the values of some variables to maintain the properties and status of all the sequences. Recall that D_{m-1} is the maximum moving distance of points in sequence S_{m-1}. Update D_{m-1} by $D_{m-1} = D_{m-1} + \Delta_{m-1}$, since we increased the maximum moving distance of points in S_{m-1}. The current maximum moving distance of points is also updated by $d_{\max} = \max\{D_{m-1}, d_{\max}\}$. Recall that $d(l, m-1)$ is the index of the point which moved D_{m-1} leftward in S_{m-1}. Let $d(l, m-1)$ be $d(l, m)$, because $p_{d(l,m)}$ moves the maximum distance leftwards in the new sequence S_{m-1} obtained by merging S_{m-1} and S_m. Once the two sequences are merged into one, we increase the count of points in S_{m-1} by $C_{m-1} = C_{m-1} + C_m$.

   Notice that, in this case, we shift S_{m-1} rightwards, then it is necessary to check the distance between $x'_{r_{m-2}}$ and $x'_{l_{m-1}}$. If $|x'_{r_{m-2}}x'_{l_{m-1}}| < r$, then we stop the merging process and continue to process the next point if there exists. In case $|x'_{r_{m-2}}x'_{l_{m-1}}| \geq r$, we need to keep merging the last two sequences to maintain the connectivity of G. In the same way, the merge process will continue until the last two sequences are connected or there exists only one sequence in the stack T. At last, update m to the number of sequences in stack T.

3. $|x'_{r_{m-1}}x'_{l_m}| - r \leq D_m - D_{m-1}$

   In this case, we move S_{m-1} towards S_m such that $|x'_{r_{m-1}}x'_{l_m}| = r$. The process is the same as that for the above case, except that we do not move S_m in this round of merging process, so the detail of the process for this case is omitted here.

3 Correctness and Time Analysis

The properties mentioned in Section 2 guarantee the connectivity of the graph G and the minimized maximum moving distance after processing a newly added point. To show the correctness of our algorithm, it is sufficient to present that those properties mentioned in Section 2 hold both in the initial case and in the general case.

For the initial case, we have only one point in the graph G, and it is obviously that the properties hold in this case. It is obvious to verify that the loop invariants hold at the beginning of our algorithm when there exists only one point in G and its position does not change during the process of adding it into G. Hence, we have the following result.

Lemma 1 *All the loop invariants and the properties of sequences hold in the initial case.*

Then we consider the general case of adding the next point p_{i+1} by following the rules of our algorithms in Sections 2.2 and 2.3, and based on the processing result, we have the following Lemma 2.

Lemma 2 *The newly added point is either in a new sequence S_{m+1} or in the last sequence S_m. All the properties and loop invariants of the sequences hold after the merging process of all the sequences in T.*

By Lemma 2, we know that the result obtained by our algorithm to the problem is a connected graph G. Denote by $l_{\max}$ (resp., $r_{\max}$) the index of the point which moves the maximum distance $d_{\max}$ towards right (resp., left) along L in the result obtained by our algorithm. Let us present the following Lemma 3, which is helpful to prove the correctness of our algorithm.

Lemma 3 $d_{\max} = \frac{|x_{l_{\max}}x_{r_{\max}}|-(r_{\max}-l_{\max})\cdot r}{2}$

Proof. During the process of our algorithm, the maximum moving distance $d_{\max}$ increases when we join the next point p_{i+1} into S_m and when we merge two sequences. In either case, the pair of points that move the maximum distance towards each other must be in the joined or merged sequence. Thus, $p_{l_{\max}}$ and $p_{r_{\max}}$ must be in one sequence in the result. By the Property 1 of the sequences maintained by our algorithm, we must have that the distance between any adjacent pair of points is r from $p_{l_{\max}}$ to $p_{r_{\max}}$. Further, by the Property 2, these two points move the same distance. Then the lemma follows. □

Based on Lemma 3, we claim that the output of our algorithm is optimal.

Lemma 4 *The result obtained by our algorithm is an optimal solution to the problem.*

Proof. By Observation 1, we know that there must exist an optimal solution in which the order of points is the same as that in the input. Thus, consider the lower bound of the moving distance of any pair of points, p_i and p_j, to obtain a connected graph of points between p_i and p_j without changing their order. To keep two points p_i and p_j connected, the largest distance d_{ij} between p_i and p_j must satisfy $d_{ij} \leq (j-i) \cdot r$, for $1 \leq i < j \leq n$, because there are at most $j-i-1$ points between them and the maximum distance between any pair of adjacent points is at most r. Hence, $\max_{1 \leq i < j \leq n} \left\{ \frac{|x_i x_j| - (j-i) \cdot r}{2} \right\}$ is the lower bound of the maximum moving distance of all the n points to keep their connectivity. Let d_{opt} be the maximum moving distance in an optimal solution, we must have the following Equation 1.

$$d_{opt} \geq \max_{1 \leq i < j \leq n} \left\{ \frac{|x_i x_j| - (j-i) \cdot r}{2} \right\} \tag{1}$$

To prove the lemma, we just need to prove that the maximum moving distance $d_{\max}$ in the result of our algorithm satisfies the following Equation 2.

$$d_{\max} = \max_{1 \leq i < j \leq n} \left\{ \frac{|x_i x_j| - (j-i) \cdot r}{2} \right\} \tag{2}$$

By Lemma 3, we must have the following Equation 3.

$$d_{\max} \leq \max_{1 \leq i < j \leq n} \left\{ \frac{|x_i x_j| - (j-i) \cdot r}{2} \right\} \tag{3}$$

Further, by the Equation 1, we also have the following Equation 4 to keep the connectivity of the graph.

$$d_{\max} \geq \max_{1 \leq i < j \leq n} \left\{ \frac{|x_i x_j| - (j-i) \cdot r}{2} \right\} \tag{4}$$

By comparing the Equation 3 and Equation 4, the Equation 2 follows. By Equations 1 and 2, it is true that $d_{\max} = d_{opt}$. Therefore, the maximum moving distance of all the points of G is minimized, i.e., our algorithm finds an optimal solution to the problem. □

For the running time of our algorithm, we have the following Lemma 5.

Lemma 5 *The running time of our greedy algorithm is $O(n)$.*

Proof. All the points on line L are already sorted in the input and they are processed from left to right. When we are processing a new point, we move that point separately at most once, then it is added into a sequence (either S_m or S_{m+1}). Obviously, a point is added into a sequence at most once during the process of our algorithm, so the moving and adding operations of all the points take $O(n)$ time totally.

The merging of two sequences takes $O(1)$ time, because we compute their shifting distance and shift all the points in the sequence together. The length of sequence S_j can also be calculated in $O(1)$ time by $(C_m - 1) \cdot r$. Thus, the merging of two sequences takes $O(1)$ time. Further, once we merged two sequences, the number of sequences decreases by 1. There exist at most n sequences in T, so the total running time of the merging process is $O(n)$.

At last, it takes $O(n)$ time to move all the points to the computed positions by our algorithm. Therefore, the running time of our algorithm is $O(n)$. □

Based on the above Observations and Lemmas, the following theorem follows.

Theorem 6 *Our algorithm can find the optimal solution to the connectivity problem in $O(n)$ time.*

References

[1] P. Balister, A. Sarkar, and B. Bollobás. *Percolation, Connectivity, Coverage and Colouring of Random Geometric Graphs*, volume 18, pages 117–142. 05 2010.

[2] C. Bettstetter. On the minimum node degree and connectivity of a wireless multihop network. In *Proceedings of the 3rd ACM International Symposium on Mobile Ad Hoc Networking &Amp; Computing*, MobiHoc '02, pages 80–91, New York, NY, USA, 2002. ACM.

[3] M. X. Cheng, Y. Ling, and B. M. Sadler. Wireless ad hoc networks connectivity assessment and relay node deployment. In *2014 IEEE Global Communications Conference*, pages 399–404, Dec 2014.

[4] S. Das, H. Liu, A. Nayak, and I. Stojmenović. A localized algorithm for bi-connectivity of connected mobile robots. *Telecommunication Systems*, 40(3-4):129–140, 2009.

[5] J. Díaz, D. Mitsche, and X. Pérez-Giménez. On the connectivity of dynamic random geometric graphs. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '08, pages 601–610, Philadelphia, PA, USA, 2008. Society for Industrial and Applied Mathematics.

[6] A. Ghosh and S. K. Das. Coverage and connectivity issues in wireless sensor networks: A survey. *Pervasive and Mobile Computing*, 4(3):303 – 334, 2008.

[7] K. Krzywdziński and K. Rybarczyk. Geometric graphs with randomly deleted edges - connectivity and routing protocols. In *Mathematical Foundations of Computer Science 2011*, pages 544–555, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[8] J. Li, L. Andrew, C. Foh, M. Zukerman, and H.-H. Chen. Connectivity, coverage and placement in wireless sensor networks. *Sensors (Basel, Switzerland)*, 9:7664–93, 10 2009.

[9] J. Liang, M. Liu, and X. Kui. A survey of coverage problems in wireless sensor networks. *Sensors and Transducers*, 163:240–246, 01 2014.

[10] D. Miorandi and E. Altman. Coverage and connectivity of ad hoc networks presence of channel randomness. In *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, volume 1, pages 491–502 vol. 1, March 2005.

[11] J. Norman. Connectivity and coverage in hybrid wireless sensor networks using dynamic random geometric graph model. *International Journal on Applications of Graph Theory In wireless Ad Hoc Networks And sensor Networks*, 3, 09 2011.

[12] Y. Tian, M. Sheng, J. Li, Y. Zhang, and J. Yao. Critical transmitting range for biconnectivity of one-dimensional wireless ad hoc networks. In *Vehicular Technology Conference, 2008. VTC Spring 2008. IEEE*, pages 108–112. IEEE, 2008.

[13] Z. Yan, H. Jiang, Z. Shen, Y. Chang, and L. Huang. k-connectivity analysis of one-dimensional linear vanets. *IEEE Transactions on Vehicular Technology*, 61(1):426–433, Jan 2012.

[14] J. Zhao. Minimum node degree and k-connectivity in wireless networks with unreliable links. pages 246–250, 06 2014.

Unfolding Polyhedra

Joseph O'Rourke*

Abstract

Starting with the unsolved "Dürer's problem" of edge-unfolding a convex polyhedron to a net, we specialize and generalize (a) the types of cuts permitted, and (b) the polyhedra shapes, to highlight both advances established and which problems remain open.

1 Introduction

Dürer's problem asks whether every convex polyhedron may be cut along edges and unfolded to a single non-overlapping simple polygon in the plane, a *net* [DO07] [O'R13]. This is attributed to Dürer because he drew many such unfoldings ca. 1500, although the question was not formulated mathematically until 1975 [She75]. It remains open, although there has been recent (minor) progress [O'R18] [O'R17]. Here we survey several generalizations and specializations of this central problem, emphasizing what is settled and what remains unresolved.

Unfolding the surface of a polyhedron to a single, flat piece in the plane requires that the cuts form a spanning tree of the vertices. We classify cuts in four types $\mathcal{C}$:

1. *edge-unfold*: All cuts are polyhedron edges, as in Dürer's problem.
2. *anycut-unfold*: The cuts may be generalized to any curve on the surface that form a spanning tree of the vertices.[1]
3. *edge-unzip*: The cut edges form a Hamiltonian path of the 1-skeleton. This natural specialization was introduced by Shephard [She75] and explored as "unzipping" in [DDL$^+$10].[2]
4. *anycut-unzip*: The cuts form a simple curve on the surface that includes every vertex. So a generalization (anycut) of a specialization (unzipping).

The second classification we explore varies the shapes $\mathcal{P}$ of the polyhedra:

1. *convex polyhedra*: All faces convex, all dihedral angles $\leq\pi$, as in Dürer's problem.
2. *spherical polyhedra*: Specializing that all vertices lie on a sphere [O'R15].[3]
3. *nonconvex polyhedra*. A broad generalization, and where most applications lie.
4. *orthogonal polyhedra* form an important subclass of nonconvex polyhedra [BDD$^+$98] [O'R08] [DFO07] [DDFO17]. All faces meet at right angles.
5. *polycubes*: Polyhedra built by gluing unit cubes whole-face to whole-face. Here all cube edges, even those with dihedral angle π, are available for cutting. So these are potentially easier to edge-unfold than are orthogonal polyhedra [RALSZ19].

For each class of polyhedra $\mathcal{P}$, and each type of cuts $\mathcal{C}$, we can ask:

Can every polyhedron in $\mathcal{P}$ be $\mathcal{C}$-unfolded to a net?

The status of these $4 \times 5 = 20$ questions is summarized in Table 1: 7 are unresolved.

References

[BDD$^+$98] Therese Biedl, Erik D. Demaine, Martin L. Demaine, Anna Lubiw, Joseph O'Rourke, Mark Overmars, Steve Robbins, and Sue Whitesides. Unfolding some classes of orthogonal polyhedra. In *Proc. 10th Canad. Conf. Comput. Geom.*, pages 70–71, 1998. Full version in *Elec. Proc.*: `http://cgm.cs.mcgill.ca/cccg98/proceedings/cccg98-biedl-unfolding.ps.gz`.

[DDFO17] Mirela Damian, Erik Demaine, Robin Flatland, and Joseph O'Rourke. Unfolding genus-2 orthogonal polyhedra with linear refinement. *Graphs and Combinatorics*, 33(5):1357–1379, 2017.

[DDL$^+$10] Erik Demaine, Martin Demaine, Anna Lubiw, Arlo Shallit, and Jonah Shallit. Zipper unfoldings of polyhedral complexes. In *Proc. 22nd Canad. Conf. Comput. Geom.*, pages 219–222, August 2010.

[DFO07] Mirela Damian, Robin Flatland, and Joseph O'Rourke. Epsilon-unfolding orthogonal polyhedra. *Graphs and Combinatorics*, 23(1):179–194, 2007.

*Smith College, `jorourke@smith.edu`

[1] "Anycut" is new terminology, intended to replace the "general unfoldings" in [DO07].

[2] "Unzipping" is my slight variation on their "zipper unfoldings."

[3] "Spherical" is often called *inscribed* in the literature, and "spherical polyhedra" are sometimes tilings of the sphere by geodesic arcs.

Shapes	*Edge-Unf*	(*Specialize*) *Edge-Unzip*	(*Generalize*) *Anycut-Unf*	(*Gen/Spec*) *Anycut-Unzip*
Convex Polyh	?	✗	✓	?
Spherical	?	✗	✓	?
Nonconvex Polyh	✗	✗	?	✗
Orthogonal	✗	✗	✓	✓
Polycubes	?	?	✓	✓

Table 1: Open Problems: ?=open, ✓=proven true, ✗=counterexamples.

[DO07] Erik D. Demaine and Joseph O'Rourke. *Geometric Folding Algorithms: Linkages, Origami, Polyhedra*. Cambridge University Press, 2007. `http://www.gfalop.org`.

[O'R08] Joseph O'Rourke. Unfolding orthogonal polyhedra. In J.E. Goodman, J. Pach, and R. Pollack, editors, *Proc. Snowbird Conf. Discrete Comput. Geom.: Twenty Years Later*, pages 307–317. American Mathematical Society, 2008.

[O'R13] Joseph O'Rourke. Dürer's problem. In Marjorie Senechal, editor, *Shaping Space: Exploring Polyhedra in Nature, Art, and the Geometrical Imagination*, pages 77–86. Springer, 2013.

[O'R15] Joseph O'Rourke. Spiral unfoldings of convex polyhedra. *arXiv:1509.00321*, 2015. `https://arxiv.org/abs/1509.00321`.

[O'R17] Joseph O'Rourke. Addendum to: Edge-unfolding nearly flat convex caps. *arXiv:1709.02433*, 2017. `https://arxiv.org/abs/1709.02433`.

[O'R18] Joseph O'Rourke. Edge-unfolding nearly flat convex caps. In *Prod. Symp. Comput. Geom.*, volume 99, pages 64:1–64:14. Leibniz Internat. Proc. Informatics, June 2018. Full version: `http://arxiv.org/abs/1707.01006`.

[RALSZ19] Lydie Richaume, Eric Andres, Gaëlle Largeteau-Skapin, and Rita Zrour. Unfolding level-1 Menger polycubes of arbitrary size with help of outer faces. In *Proc. 21st IAPR Internat. Conf., DGCI 2019*, March 2019.

[She75] Geoffrey C. Shephard. Convex polytopes with convex nets. *Math. Proc. Camb. Phil. Soc.*, 78:389–403, 1975.

Guarantees on Nearest-Neighbor Condensation heuristics*

Alejandro Flores-Velazco† David Mount‡

Abstract The problem of nearest-neighbor (NN) condensation aims to reduce the size of a training set of a nearest-neighbor classifier while maintaining its classification accuracy. Although many condensation techniques have been proposed, few bounds have been proved on the amount of reduction achieved. In this paper, we present one of the first theoretical results for practical NN condensation algorithms. We propose two condensation algorithms, called RSS and VSS, along with provable upper-bounds on the size of their selected subsets. Additionally, we shed light on the selection size of two other state-of-the-art algorithms, called MSS and FCNN, and compare them to the new algorithms.

1 Introduction

In machine learning, *classification* involves a training set $P \subset \mathbb{R}^d$ of n *labeled* points in Euclidean space. The label $l(p)$ of each point $p \in P$ indicates the *class* to which the point belongs to, partitioning of P into a finite set of *classes*. Given an *unlabeled* query point $q \in \mathbb{R}^d$ the goal of a *classifier* is to predict q's label using P.

The *nearest-neighbor* (NN) *rule* is among the best-known classification techniques [6]. It classifies a query point q with the label of its closest point in P, according to some metric. Throughout, we will assume the Euclidean ℓ_2 metric. Despite its simplicity, the NN rule exhibits good classification accuracy both experimentally and theoretically [13, 4, 5]. However, it's often criticized due to its high space and time complexities. This raises the question of whether it is possible to replace P with a significantly smaller subset without affecting the classification accuracy under the NN rule. This problem is called *nearest-neighbor condensation*. In this paper we propose two new NN condensation algorithms and analyze their worst-case performance.

1.1 Preliminaries

For any point $p \in P$, define an *enemy* of p to be any point in P of different class than p. The *nearest enemy* (NE) of p, denoted $\mathrm{ne}(p)$, is the closest such point, and its distance from p, called the NE *distance*, is denoted as $\mathsf{d}_{\mathrm{ne}}(p) = \mathsf{d}(p, \mathrm{ne}(p))$. Similarly, denote the NN distance as $\mathsf{d}_{\mathrm{nn}}(p) = \mathsf{d}(p, \mathrm{nn}(p))$. Define the NE *ball* of p to be the ball centered at p with radius $\mathsf{d}_{\mathrm{ne}}(p)$. Let κ denote the number of distinct NE points of P.

A point $p \in P$ is called a *border point* if it is incident to an edge of the Delaunay triangulation of P whose opposite endpoint is an enemy of p. Otherwise, p is called an *internal point*. By definition, the border points of P completely characterize the portion of the Voronoi diagram that separates Voronoi cells of different classes. Let k denote the number of border points of P.

1.2 Related work

A subset $R \subseteq P$ is said to be *consistent* if for all $p \in P$ its nearest neighbor in R is of the same class as p. Intuitively, R is consistent if and only if every point of P is correctly classified using the NN rule over R. Formally, *nearest-neighbor condensation* involves finding an (ideally small) consistent subset of P [9].

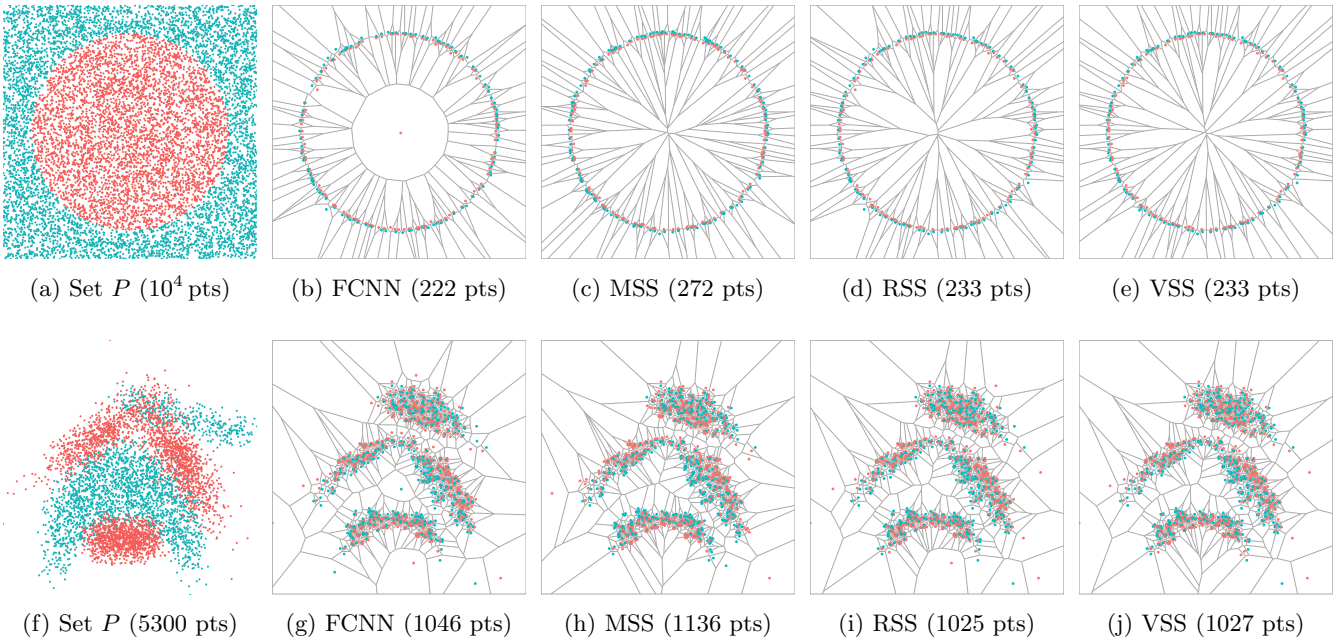
Other criteria for condensation have been studied in the literature. One such criterion is known as *selectivity* [12]. A subset $R \subseteq P$ is said to be *selective* if and only if for all $p \in P$, its nearest neighbor in R is closer to p than its nearest enemy in P. Clearly selectivity implies consistency, as the NE distance in R of any point of P is at least its NE distance in P. Note that neither consistency nor selectivity imply that every query point of $\mathbb{R}^d$ is correctly classified, just those in P.

The strongest criteria, known as *Voronoi condensation*, consists of selecting all border points of P [16]. This guarantees the correct classification of any query point in $\mathbb{R}^d$. In contrast, a consistent subset only guarantees correct classification of P. For the case when $P \subset \mathbb{R}^2$, an output-sensitive algorithm was proposed [3] for finding all border points of P in $\mathcal{O}(n \log k)$ worst-case time. Unfortunately, it is not known how to generalize this algorithm to higher dimensions, and a straightforward approach suffers from the super-linear worst-case size of the Delaunay triangulation.

In general, it has been shown that the problems of computing consistent and selective subsets of minimum cardinality are both NP-complete [17, 18, 11]. Thus, most research has focused on practical heuristics. For comprehensive surveys, see [14, 15, 10]. CNN (Condensed Nearest-Neighbor) [9] was the first algorithm proposed for computing consistent subsets. Even though it has been widely cited in the literature, CNN suffers from several drawbacks: its running time is cubic in the worst-case, and the resulting subset is

*Research supported by NSF grant CCF–1618866.

†University of Maryland, College Park, afloresv@cs.umd.edu
‡University of Maryland, College Park, mount@umd.edu



(a) Set P (10^4 pts) (b) FCNN (222 pts) (c) MSS (272 pts) (d) RSS (233 pts) (e) VSS (233 pts)

(f) Set P (5300 pts) (g) FCNN (1046 pts) (h) MSS (1136 pts) (i) RSS (1025 pts) (j) VSS (1027 pts)

Figure 1: Examples of the subsets selected by FCNN, MSS, RSS, and VSS, on two different training sets. Training set (a) is a set of uniformly distributed points in $\mathbb{R}^2$ of two classes: *red* points lying inside a disk, and blue points lying outside. Training set (f) is a well-known benchmark from the *UCI Machine Learning repository*, called *Banana*, consisting of points in $\mathbb{R}^2$ of two classes, *red* and *blue*.

order-dependent, meaning that the result is determined by the order in which points are considered by the algorithm. Alternatives include FCNN (Fast CNN) [1] and MSS (Modified Selective Subset) [2], which produce consistent and selective subsets respectively. Both algorithms run in $\mathcal{O}(n^2)$ worst-case time, and are order-independent. These algorithms are considered the state-of-the-art for the NN condensation problem, subject to achieving these properties. While such heuristics have been extensively studied experimentally [10, 7], theoretical results are scarce. Unfortunately, to the best of our knowledge, no bounds are known for the size of the subsets generated by any of these heuristics.

More recently, an approximation algorithm called NET [8] was proposed, along with almost matching hardness lower bounds for the problem. The idea is to compute a γ-net of P, with γ equal to the minimum NE distance in P, implying that the resulting subset is consistent. Unfortunately, while NET has provable worst-case performance, this approach allows little room for condensation, and in practice, the resulting subset can be too large. For example, on the training set in Figure 1a, NET selects a subset of over 90% of the points, while other algorithms select only 3% of the points.

1.3 Our contributions

In this paper, we propose two new NN condensation algorithms, called RSS and VSS. We will establish asymptotically tight upper-bounds on the sizes of their selected subsets. Moreover, we prove that these algorithms have similar complexity to the popular state-of-the-art algorithms FCNN and MSS. Additionally, we also analyze the selection size of FCNN and MSS. To the best of our knowledge, these represent the first theoretical results on *practical* NN condensation algorithms. The following is a summary of our contributions.

Algorithm	Selection size
RSS	$\mathcal{O}(\kappa\ c^{d-1})$
VSS	$\leq k$
MSS [2]	$\Omega(1/\varepsilon)$ w.r.t. κ and k
FCNN [1]	$\Omega(k)$

2 Results on condensation size

One of the most significant shortcomings in research on practical NN condensation algorithms is the lack of theoretical results on the sizes of the selected subsets. Typically, the performance of these heuristics has been established experimentally.

We establish bounds with respect to the size of two well-known and structured subsets of points: (a) the set of all NE points of P of size κ, and (b) the set of border points of P of size k. Throughout the paper, we refer equally to the algorithms and their selected subsets.

2.1 The state-of-the-art

Let's begin our analysis with a state-of-the-art algorithm for the problem: MSS or *Modified Selective Subset* (see Algorithm 1). The selection process of the algorithm can be simply described as follows: for every $p \in P$, MSS selects the point with smaller NE distance contained inside the NE ball of p.

Clearly, this approach computes a selective subset of P, which by definition, is order-independent. MSS can be implemented in $\mathcal{O}(n^2)$ worst-case time. Unfortunately, the selection criteria of MSS can be too strict, requiring one particular point to be added for each point $p \in P$. Note that any point inside the NE ball of p suffices for achieving selectiveness. In practice, this can lead to much larger subsets than needed.

Algorithm 1: Modified Selective Subset

Input: Initial training set P
Output: Condensed training set MSS $\subseteq P$

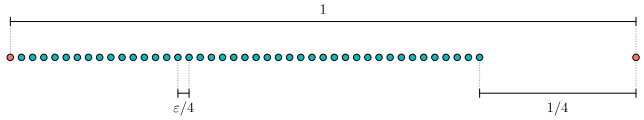
```
Let {p_i}_{i=1}^n be the points of P sorted in increasing order of NE distance d_ne(p_i)
MSS ← ∅
S ← P
foreach p_i ∈ P, where i = 1...n do
    add ← false
    foreach p_j ∈ P, where j = i...n do
        if p_j ∈ S ∧ d(p_j, p_i) < d_ne(p_j) then
            S ← S \ {p_j}
            add ← true
    if add then
        MSS ← MSS ∪ {p_i}
return MSS
```

This intuition is formalized in the following theorem. Here we show that the subset selected by MSS can select a subset of unbounded size as a function of κ or k.

Theorem 1 *Given $0 < \varepsilon < 1$, there exists a training set $P \subset \mathbb{R}^d$ with a constant number of* NE *and border points such that* MSS *selects $\Omega(1/\varepsilon)$ points.*

Proof. Recall that for each point in P, the MSS algorithm selects the point inside its NE ball with minimum NE distance. Given a parameter $0 < \varepsilon < 1$, we construct a training set in d-dimensional Euclidean space, as illustrated in Figure 2a.

Create two points r_1 and r_2, and assign them to the class of *red* points. Without lost of generality, the distance between these two points is 1. Let $\vec{u}$ be the unit vector from r_1 to r_2, create additional points $b_i = r_1 + \frac{i\varepsilon}{4}\vec{u}$ for $i = \{1, 2, \ldots, 3/\varepsilon\}$. Assign all b_i points to the class of *blue* points. The set of all these points constitute the training set P. It is easy to prove that P has only 4 NE points and 4 border points, corresponding to r_1, r_2, b_1 and $b_{3/\varepsilon}$.



(a) Initial training set of collinear points, where both the number of NE points and the number of border points equal to 4. That is, $\kappa = k = 4$.

(b) Subset of points computed by MSS from the original training set (fully colored points belong to the subset, while faded points do not). The size of the subset is $\Omega(1/\varepsilon)$.

Figure 2: Unbounded example for MSS w.r.t. κ and k.

Let's discuss which points are added by MSS for each point in P (see Figure 2b). For points r_1 and r_2, the only points inside their NE balls are themselves, so both r_1 and r_2 belong to the subset selected by MSS. For points b_i with $i \leq 2/\varepsilon$, the point with minimum NE distance contained inside their NE ball is b_1, which is also added to the subset. Now, consider the points b_i with $2/\varepsilon < i < 5/2\varepsilon$. Let $j = i - 2/\varepsilon$, it is easy to prove that the point with minimum NE distance inside the NE ball of b_i is b_{2j+1} (see Figure 2b). Therefore, this implies that the number of points selected by MSS equals $5/2\varepsilon - 2/\varepsilon = 1/2\varepsilon = \Omega(1/\varepsilon)$. □

2.2 A better approach

We propose a new algorithm, called RSS or *Relaxed Selective Subset*, with the idea relaxing the selection process of MSS, while still computing a selective subset. For a given point $p \in P$, while MSS requires to add the point with smallest NE distance inside the NE ball of p, in RSS any point inside the NE ball p suffices.

Algorithm 2: Relaxed Selective Subset

Input: Initial training set P
Output: Condensed training set RSS $\subseteq P$

```
RSS ← ∅
Let {p_i}_{i=1}^n be the points of P sorted in increasing order of NE distance d_ne(p_i)
foreach p_i ∈ P, where i = 1...n do
    if d_nn(p_i, RSS) ≥ d_ne(p_i) then
        RSS ← RSS ∪ {p_i}
return RSS
```

The idea is rather simple (see Algorithm 2). Points of P are examined in increasing order with respect to their NE distance, and we add any point whose NE ball contains no point previously added by the algorithm. This tends to select points close to the decision boundary of P (see Figure 1d), as points far from the boundary are examined later in the selection process, and are more likely to already contain points inside their NE ball.

Theorem 2 RSS *is order-independent and computes a selective subset of P in $\mathcal{O}(n^2)$ worst-case time.*

Proof. By construction, every point in P was either added into RSS, or has a point in RSS inside its NE ball. Therefore, RSS is selective. The order-independence follows from the initial sorting step.

Let's analyze the time complexity of RSS. The initial step requires $\mathcal{O}(n^2)$ time for computing the NE distances of each point in P, plus additional $\mathcal{O}(n \log n)$ time for sorting the points according to such distances. The main loop iterates through each point in P, and searches their nearest neighbor in the current subset, incurring into additional $\mathcal{O}(n^2)$ time. Finally, the worst-case time complexity of the algorithm is $\mathcal{O}(n^2)$. □

Theorem 3 RSS *selects at most $\mathcal{O}(\kappa\ (3/\pi)^{d-1})$ points.*

Proof. The proof follows by a charging argument on each NE point of P. Consider a NE point $p \in P$, and let RSS_p be the set of points selected by RSS such that p is their NE. Let $p_i, p_j \in \text{RSS}_p$ be two such points, and without loss of generality say that $\mathsf{d}_{\text{ne}}(p_i) \leq \mathsf{d}_{\text{ne}}(p_j)$. By construction of the algorithm, we also know that $\mathsf{d}(p_i, p_j) \geq \mathsf{d}_{\text{ne}}(p_j)$. Now, consider the triangle $\triangle p p_i p_j$. Clearly, the side pp_i is the larger of such triangle, and therefore, the angle $\angle p_i p p_j \geq \pi/3$. Meaning that the angle between any two points in RSS_p with respect to p is at least $\pi/3$.

By a standard packing argument, this implies that $|\text{RSS}_p| = \mathcal{O}((3/\pi)^{d-1})$. Finally, we obtain that $|\text{RSS}| = \sum_p |\text{RSS}_p| = \kappa\ \mathcal{O}((3/\pi)^{d-1})$. □

For constant dimension d, the size of RSS is $\mathcal{O}(\kappa)$. Therefore, the following result implies that the upper-bound on RSS is tight up to constant factors. Furthermore, it implies that this is the best upper-bound we can hope to achieve in terms of κ.

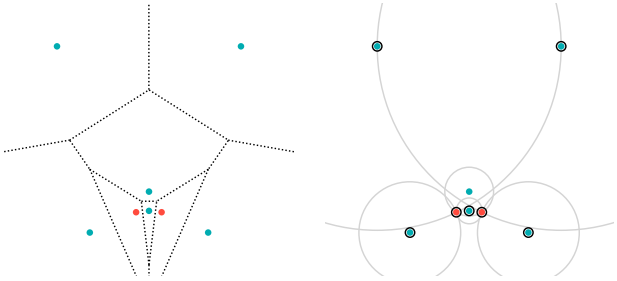
Theorem 4 (Lower-bound) *There exists a training set $P \subset \mathbb{R}^d$ with κ NE points, for which any consistent subset contains $\Omega(\kappa c^{d-1})$ points, for some constant c.*

Proof. We construct a training set P in d-dimensional Euclidean space, which contains points of two classes: *red* and *blue*. Consider the following arrangement of points: create a red point p, and take *every* point at distance 1 from p as a blue point. Simply, the points on the surface of a unit ball centered at p.

Take any consistent subset of this training set, and consider some point p' in such subset, and the bisector between p and p'. The intersection between this bisector and the unit ball centered at p describes a cap of such ball of height $1/2$. Any point located inside this cap is closer to p' than p, and hence, correctly classified. Clearly, by definition of consistency, all points in the ball must be covered by at least one cap. By a simple packing argument, we know such covering needs $\Omega(c^{d-1})$ points, for some constant c.

So far the training set constructed has only two nearest enemy points; the red point p, and one blue point closest to p (assuming general position). Then, we can repeat this arrangement $\kappa/2$ times, using sufficiently separated center points. This generates a training set P with a number of NE points equal to κ, for which any consistent subset has size $\Omega(\kappa c^{d-1})$. □

Different parameters from κ can be used to bound the selection size of condensation algorithms. Let's consider k, the number of border points in the training set P. From the example illustrated in Figure 3, we know that RSS can select more points than k (see Figure 3b). Repeating such arrangement forces RSS to select $\Omega(k)$ points. Yet, the question remains, at most, how many more points than k can this algorithm select?



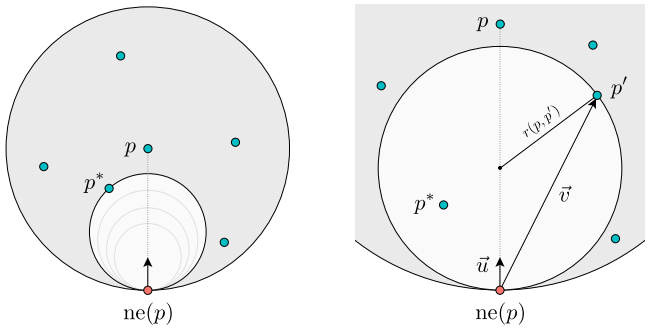
(a) Point arrangement. (b) RSS selection outlined.

Figure 3: Example where RSS selects $k+1$ points.

Lemma 5 *The nearest enemy point of any point in P is a also a border point of P.*

Proof. Take any point $p \in P$. Consider the *empty* ball of maximum radius, tangent to point $\text{ne}(p)$, and with center in the line segment between p and $\text{ne}(p)$. Being maximal, this ball is tangent to another point $p^* \in P$ (see Figure 4a). Clearly, p^* is inside the NE ball of p, which implies that p and p^* belong to the same class, making p^* and $\text{ne}(p)$ enemies. By the empty ball property, this means that both p^* and $\text{ne}(p)$ are border points of P. □

From Lemma 5, we know that in Euclidean space, the number of NE points of P is at most the number of border points of P. That is, $\kappa \leq k$. While this implies



(a) (b)

Figure 4: (a) The largest empty ball tangent to ne(p) and center in $\overline{p\ \mathrm{ne}(p)}$, is also tangent to some point p^*, making p^* and ne(p) border points. (b) Computing the radius of a ball with center in the line segment between p and ne(p), and tangent to both ne(p) and p'.

an easy extension of the bound for RSS, now in terms of k, it is unclear if the other factors can be improved.

Alternatively, this opens another idea for condensation. In order to prove Lemma 5, we show that there exist at least one border point inside the NE ball of any point $p \in P$. Therefore, any algorithm that only selects such border points, can guarantee to compute a selective subset of size at most k. Consider then a modification of RSS, where for each point $p_i \in P$, if no other point lying inside the NE ball of p has been added yet, instead of adding p_i as RSS does, we add a border point inside NE ball of p. We call this new algorithm VSS or *Voronoi Selective Subset* (see Algorithm 3).

Algorithm 3: Voronoi Selective Subset

Input: Initial training set P
Output: Condensed training set VSS $\subseteq P$

```
VSS ← ∅
Let {p_i}_{i=1}^n be the points of P sorted in increasing
  order of NE distance d_ne(p_i)
foreach p_i ∈ P, where i = 1...n do
    if d_nn(p_i, RSS) ≥ d_ne(p_i) then
        Find a border point that lies inside the NE
          ball of p_i and add it to VSS
return VSS
```

Theorem 6 VSS *computes a selective subset of P of size at most k in $\mathcal{O}(n^2)$ worst-case time.*

Proof. By construction, for any point in $p \in P \setminus$ VSS the algorithm selected one border point inside the NE ball of p. This implies that the resulting subset is selective, and contains no more than k points.

Now, we describe an efficient implementation of VSS. Recall that for every point $p \in P \setminus$ VSS, the algorithm finds a border point inside its NE ball. Without loss of generality implement VSS to compute the point p^* that minimizes the radius of an empty ball tangent to both ne(p) and p^*, and center in the line segment between p and ne(p). For any given point p' inside the NE ball of p, denote $r(p, p')$ to be the radius of the ball tangent to p' and ne(p) and center in the line segment between p and ne(p). As illustrated in Figure 4b, let vectors $\vec{u} = \frac{p-\mathrm{ne}(p)}{\|p-\mathrm{ne}(p)\|}$ and $\vec{v} = p' - \mathrm{ne}(p)$, the radius of this ball can be derived from the formula $r(p, p') = \|\vec{v}+r(p, p')\vec{u}\|$ as $r(p, p') = \frac{\vec{v}\cdot\vec{v}}{2\vec{u}\cdot\vec{v}}$.

As p^* is defined as the point that minimizes such radius, a simple scan over the points of P suffices to identify the corresponding p^* for any point $p \in P$. Therefore, this implies that VSS can be computed in $\mathcal{O}(n^2)$ worst-case time. □

2.3 What about FCNN?

FCNN or *Fast Condensed Nearest-Neighbor* is yet another popular state-of-the-art algorithm for the NN condensation problem. In contrast with MSS, which finds selective subsets, FCNN selects consistent subsets of P.

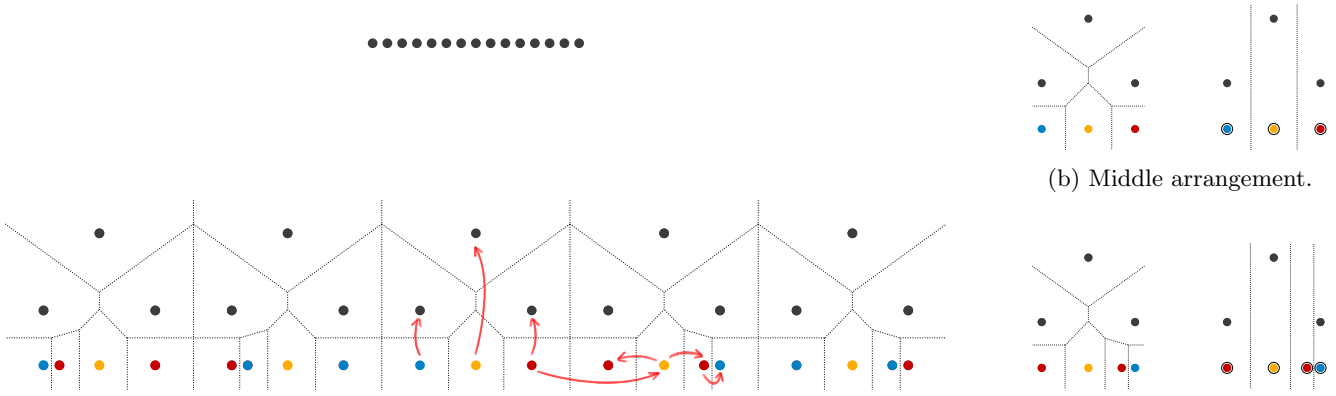
Let's now describe the selection process of FCNN (see Algorithm 4). Essentially, FCNN maintains a subset of P, which is updated in each iteration, by adding points that are incorrectly classified using the current subset. The iterations stop when all points of P are correctly classified by the current subset, that is, when FCNN is consistent. Starting with the centroids of each class, set S contains some misclassified points from $P \setminus$ FCNN that will be added in the next iterartion. How does the algorithm decide which points to include in S? Define voren(p, FCNN, P) as the set of enemy points of p in P, whose NN in FCNN is p, that is, the set $\{q \in P \mid l(q) \neq l(p) \wedge \mathrm{ne}(q, \mathrm{FCNN}) = p\}$. Then, for each point $p \in$ FCNN, the algorithm selects one representative out of its corresponding voren(p, FCNN, P), which is usually defined as the NN to p.

Algorithm 4: Fast Condensed Nearest-Neighbor

Input: Initial training set P
Output: Condensed training set FCNN $\subseteq P$

```
FCNN ← ∅
S ← centroids(P)
while S ≠ ∅ do
    FCNN ← FCNN ∪ S
    S ← ∅
    foreach p ∈ FCNN do
        S ← S ∪ {rep(p, voren(p, FCNN, P))}
return FCNN
```

Theorem 7 *There exists a training set $P \subset \mathbb{R}^d$ with k*



(a) Entire arrangement of points.

(b) Middle arrangement.

(c) Side arrangement.

Figure 5: Example of a training set $P \subset \mathbb{R}^2$ for which FCNN selects $\Omega(k)$ points.

border points, for which FCNN *selects* $\Omega(k)$ *points.*

Proof. Consider the arrangement in Figure 5b (left), consisting of points of 4 classes. The centroids of the blue, yellow, and red classes are the only points labeled as such. By placing a sufficient number of black points far at the top of this arrangement, we avoid their centroid to be any of the three black points in the arrangement. Beginning with the centroids, the first iteration of FCNN would have added the points outlined in Figure 5b (right). Now each of these points have one black point inside their Voronoi cells, and therefore, these black points will be the representatives added in the second iteration. This small example, with $k = 5$, shows how to force FCNN to add all the border points plus two internal points. Out of these two internal black points, one is the centroid added in the initial step. The remaining internal black point, however, was added by the algorithm during the iterative process. This scheme can be extended to larger values of k, without increasing the number of classes.

The previous is the first building block of the entire training set, shown in Figure 5a. To this "middle" arrangement, we append "side" arrangements of points, as the one illustrated in Figure 5c, which will have similar behavior to the middle arrangement. This particular side arrangement will be appended to the right of the middle one, such that the distance between the red points is greater than the distance from the yellow to the red point. Every time we append a new side arrangement, its blue and red labels are swapped. The arrangements appended to the left side are simply a horizontal flip of the right arrangement. Now, the behavior of FCNN in such a setup is illustrated with the arrows in Figure 5a. The extreme point of the previous arrangement adds the yellow point at the center of the current arrangement, which then adds the red point next to the blue point, as is closer than the other red point. Next, this red point adds the blue point, and the yellow point adds the remaining red point. Finally, the Voronoi cells of these points will look as shown in Figure 5c (right), and in the next iteration, the tree black points will be added.

After adding side arrangements as needed (same number of the left and right), it is easy to show that the centroids are still the tree points in the middle arrangement and the black point at the top (by adding a sufficient number of black points in the top cluster). This implies than FCNN will be forced to select $\Omega(k)$ points. □

While this example sheds light on the selection behavior of FCNN, an upper-bound is still missing. Based on the following lemma, we conjecture that FCNN selects at most $\mathcal{O}(\kappa\ c^{d-1})$ points, for some constant c.

Lemma 8 *Consider a point p selected by* FCNN. *Then, the number of representatives of p selected throughout the algorithm does not exceed $\mathcal{O}((3/\pi)^{d-1})$ points.*

Proof. This proof follows from similar arguments to the ones described in the proof of Theorem 3. Consider $p_1, p_2 \in$ FCNN to be two points added to by the algorithm as representatives of p. Without loss of generality, p_1 was added before p_2, implying that $\mathsf{d}(p, p_1) \leq \mathsf{d}(p, p_2)$. By construction, if p_2 was added as a representative of p, and not of p_1, we also know that $\mathsf{d}(p, p_2) \leq \mathsf{d}(p_1, p_2)$. From this, consider the triangle $\triangle pp_1p_2$ and the angle $\angle p_1pp_2$. This is the largest angle of the triangle, meaning that $\angle p_1pp_2 \geq \pi/3$. Finally, by a standard packing argument, there are at most $\mathcal{O}((3/\pi)^{d-1})$ such points. □

3 Open problems

A few key questions remain unsolved:

- In terms of k, our best upper-bound on the selection size of RSS is not tight. Can it be improved?
- Is it possible to prove an upper-bound on the selection size of FCNN in terms of either κ or k?

References

[1] F. Angiulli. Fast nearest neighbor condensation for large data sets classification. *IEEE Transactions on Knowledge and Data Engineering*, 19(11):1450–1464, 2007.

[2] R. Barandela, F. J. Ferri, and J. S. Sánchez. Decision boundary preserving prototype selection for nearest neighbor classification. *International Journal of Pattern Recognition and Artificial Intelligence*, 19(06):787–806, 2005.

[3] D. Bremner, E. Demaine, J. Erickson, J. Iacono, S. Langerman, P. Morin, and G. Toussaint. Output-sensitive algorithms for computing nearest-neighbour decision boundaries. In F. Dehne, J.-R. Sack, and M. Smid, editors, *Algorithms and Data Structures: 8th International Workshop, WADS 2003, Ottawa, Ontario, Canada, July 30 - August 1, 2003. Proceedings*, pages 451–461, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[4] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Trans. Inf. Theor.*, 13(1):21–27, Jan. 1967.

[5] L. Devroye. On the inequality of cover and hart in nearest neighbor discrimination. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (1):75–78, 1981.

[6] E. Fix and J. L. Hodges. Discriminatory analysis, nonparametric discrimination: Consistency properties. *US Air Force School of Aviation Medicine*, Technical Report 4(3):477+, Jan. 1951.

[7] S. Garcia, J. Derrac, J. Cano, and F. Herrera. Prototype selection for nearest neighbor classification: Taxonomy and empirical study. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34(3):417–435, Mar. 2012.

[8] L.-A. Gottlieb, A. Kontorovich, and P. Nisnevitch. Near-optimal sample compression for nearest neighbors. In *Advances in Neural Information Processing Systems*, pages 370–378, 2014.

[9] P. Hart. The condensed nearest neighbor rule (corresp.). *IEEE Trans. Inf. Theor.*, 14(3):515–516, Sept. 1968.

[10] N. Jankowski and M. Grochowski. Comparison of instances selection algorithms I. Algorithms survey. In *Artificial Intelligence and Soft Computing-ICAISC 2004*, pages 598–603. Springer, 2004.

[11] K. Khodamoradi, R. Krishnamurti, and B. Roy. Consistent subset problem with two labels. In *Conference on Algorithms and Discrete Applied Mathematics*, pages 131–142. Springer, 2018.

[12] G. Ritter, H. Woodruff, S. Lowry, and T. Isenhour. An algorithm for a selective nearest neighbor decision rule. *IEEE Transactions on Information Theory*, 21(6):665–669, 1975.

[13] C. J. Stone. Consistent nonparametric regression. *The annals of statistics*, pages 595–620, 1977.

[14] G. Toussaint. Open problems in geometric methods for instance-based learning. In J. Akiyama and M. Kano, editors, *JCDCG*, volume 2866 of *Lecture Notes in Computer Science*, pages 273–283. Springer, 2002.

[15] G. Toussaint. Proximity graphs for nearest neighbor decision rules: Recent progress. In *Progress, Proceedings of the 34th Symposium on the INTERFACE*, pages 17–20, 2002.

[16] G. T. Toussaint, B. K. Bhattacharya, and R. S. Poulsen. The application of Voronoi diagrams to non-parametric decision rules. *Proc. 16th Symposium on Computer Science and Statistics: The Interface*, pages 97–108, 1984.

[17] G. Wilfong. Nearest neighbor problems. In *Proceedings of the Seventh Annual Symposium on Computational Geometry*, SCG '91, pages 224–233, New York, NY, USA, 1991. ACM.

[18] A. V. Zukhba. NP-completeness of the problem of prototype selection in the nearest neighbor method. *Pattern Recog. Image Anal.*, 20(4):484–494, Dec. 2010.

A Simple Randomized Algorithm for All Nearest Neighbors

Soroush Ebadian* Hamid Zarrabi-Zadeh*

Abstract

Given a set P of n points in the plane, the all nearest neighbors problem asks for finding the closest point in P for each point in the set. The following folklore algorithm is used for the problem in practice: pick a line in a random direction, project all points onto the line, and then search for the nearest neighbor of each point in a small vicinity of that point on the line. It is widely believed that the expected number of points needed to be checked by the algorithm in the vicinity of each point is $O(\sqrt{n})$ on average. We confirm this conjecture in affirmative by providing a careful analysis on the expected number of comparisons made by the algorithm. We also present a matching lower bound, showing that our analysis is essentially tight.

1 Introduction

The *all nearest neighbors* problem considers finding, for a set P of n points in the plane, the nearest neighbor of each point in P. This is a fundamental and well-studied problem in computational geometry, with various applications, e.g., in statistics, similarity search, and image processing.

Several $O(n \log n)$ time algorithms are available for the problem. In particular, it is well-known that the Delaunay triangulation of P contains all edges connecting nearest neighbors. (See Figure 1.) Therefore, one can solve the all nearest neighbors problem in the plane in $O(n \log n)$ time using any of the optimal algorithms available for the Delaunay triangulation [2, 4]. In higher fixed dimensions, one can solve the problem in $O(n \log n)$ time using the algorithms of Clarkson [1] and Vaidya [6]. Both algorithms make use of spatial data partitioning trees, such as compressed quad-trees [5] and R-trees [3].

In this paper, we study an extremely simple randomized algorithm for the all nearest neighbors problem that uses no geometric data structure, and can be implemented in a few lines of code. It basically projects all the points onto a random line and searches for the nearest neighbor of each point in a small vicinity of that point on the line.

*Department of Computer Engineering, Sharif University of Technology, Tehran, Iran. Email: ebadian@ce.sharif.edu, zarrabi@sharif.edu.

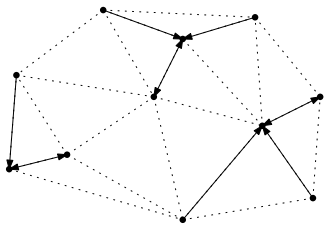


Figure 1: An example of the problem. Each point is connected to its nearest neighbor by an arrow. Dotted segments show the Delaunay triangulation edges.

The main contribution of this paper is a careful and tight analysis of the expected runtime of this randomized algorithm. More precisely, we show that the expected number of comparisons made by the algorithm is $O(\alpha n\sqrt{n})$ in total, where $\alpha = \sqrt{\log \delta + 1}$, and δ is the ratio of the largest to smallest pairwise distance between the points and their nearest neighbors. In practice, α is upper bounded by a constant. For example, when input coordinates are represented by rational numbers with 64-bit integers, we have $\delta \leq 2^{128}$, and hence, $\sqrt{\log_2 \delta + 1}$ is at most 12.

The utter simplicity of the algorithm has made it a popular choice in cases where a fast implementation is preferable at the cost of slightly relaxing the optimal runtime. Due to its simplicity and removing the overhead of geometric data structures, the algorithm is even faster in practice compared to the other standard algorithms for the problem, such as Delaunay triangulations, when input data has only a few thousand points. Moreover, the algorithm finds the nearest neighbor of each point independently, after an initial step, which makes it highly flexible for parallel implementation.

2 Preliminaries

Let p and q be two points in the plane. We denote the Euclidean distance of p and q by $\|pq\|$. For a unit vector u in the plane, we denote by $\|pq\|_u$ the *projected distance* between p and q along direction u. In other words, $\|pq\|_u = (p - q) \cdot u = \|pq\| \cos\theta$, where θ is the angle between $\overrightarrow{pq}$ and u. Since $\cos\theta \leq 1$, we always have $\|pq\|_u \leq \|pq\|$.

3 The Algorithm

In this section, we present the folklore randomized algorithm for the all nearest neighbors problem, and prove its correctness. The algorithm in its entirety is given in Algorithm 1. It takes as input a set $P = \{p_1, \ldots, p_n\}$ of n points in the plane, and returns for each point p_i its nearest neighbor q_i in P.

Algorithm 1 ALL NEAREST NEIGHBORS$(p_1, \ldots, p_n)$

```
1: pick a random unit vector u
2: for i from 1 to n do
3:     d_i ← ∞
4:     for p_j in increasing order of ‖p_i p_j‖_u ≤ d_i do
5:         if ‖p_i p_j‖ < d_i then
6:             d_i ← ‖p_i p_j‖, q_i ← p_j
7: return q_1, ..., q_n
```

The algorithm works as follows. After picking a random unit vector, the algorithm processes each point p_i by checking the points in $P \setminus \{p_i\}$ in their increasing projected distance to p_i, while keeping the minimum Euclidean distance found so far in d_i. The search for the nearest neighbor of p_i is terminated whenever we reach a point whose projected distance to p_i is more than d_i.

An example of the execution of Algorithm 1 for a point p is illustrated in Figure 2. In this example, points are numbered in their increasing projected distance to p. The algorithm stops when it reaches point p_5, whose projected distance to p is more than the best distance found so far, i.e., $\|pp_2\|$.

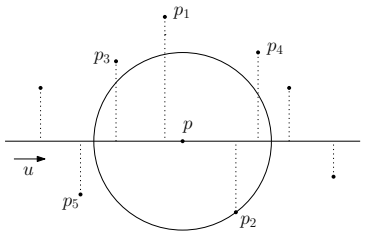


Figure 2: An example of the execution of Algorithm 1.

To quickly iterate over the points in their increasing projected distance from a point p, we can perform a simple preprocess step as follows. We select a line ℓ in direction u, project each point $p_i \in P$ to a point p_i' on ℓ, and sort the projected points along ℓ. Then, in the main loop for each point p_i, we keep two pointers on ℓ initially set to the points right before and after p_i' on ℓ, walking in opposite directions. At each step, we compare the distance of p_i' to the two projected points specified by the pointers, select whichever is smaller, and advance the corresponding pointer to the next one. This way, iterating over each point p_j takes $O(1)$ time in the algorithm.

The correctness of the algorithm is proved in the following lemma.

Lemma 1 *For each point $p_i \in P$, the algorithm correctly finds the nearest neighbor of p_i.*

Proof. Fix a point p_i, and let q be the nearest point of p_i in P. Suppose by way of contradiction that the inner loop of the algorithm terminates on a point p_j, before reaching q. Thus, $\|p_ip_j\|_u < \|p_iq\|_u$. The inner loop terminates if $\|p_ip_j\|_u > d_i$, where d_i is the distance between p_i and a previously-visited point p_k. Therefore, we have $\|p_iq\| \geq \|p_iq\|_u > \|p_ip_j\|_u > d_i = \|p_ip_k\|$, which contradicts the fact that q is the closest point to p_i. □

4 The Analysis

Let $P = \{p_1, \ldots, p_n\}$ be the set of input points. For $1 \leq i \leq n$, we denote by d_i the distance of p_i to its nearest neighbor in P. Let $P_i = \{p_j \in P - \{p_i\} : \|p_ip_j\|_u \leq d_i\}$ be the set of points compared by the algorithm during the search for the nearest neighbor of p_i.

Let X be a random variable indicating the total number of comparisons made by Algorithm 1. We can decompose X into n^2 indicator variables

$$X_{i,j} = \begin{cases} 1 & \text{if } p_j \in P_i, \\ 0 & \text{otherwise.} \end{cases}$$

Note that $X_{i,i} = 0$ for all i, and $X = \sum_{1 \leq i,j \leq n} X_{i,j}$.

Lemma 2 *For all $1 \leq i, j \leq n$, $i \neq j$,*

$$\Pr\{X_{i,j} = 1\} \leq \frac{d_i}{\|p_ip_j\|}.$$

Proof. Fix two points p_i and p_j in P. For all $r \geq d_i$, let C_r be a circle of radius r centered at p_i. Consider a strip S of width $2d_i$ enclosing C_{d_i} orthogonal to direction u

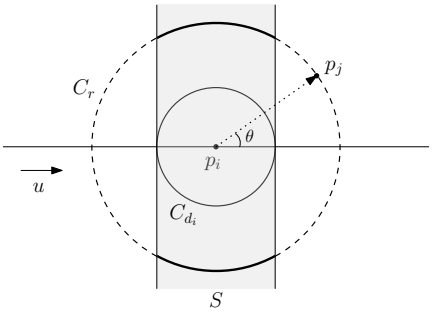


Figure 3: An illustration of Lemma 2. The strip S is shown in gray, and $C_r \cap S$ is shown by thick arcs.

(see Figure 3). Note that for all points $p \in P$, we have $p \in P_i$ if and only if p lies in S.

Let $A(r)$ denote the length of $C_r \cap S$, for all $r \geq d_i$. The curvature of the arcs in $C_r \cap S$ decreases as r is increased, and hence, $A(r)$ is a decreasing function on $[d_i, \infty)$. Therefore, for all $r \geq d_i$, $A(r) \leq A(d_i) = 2\pi d_i$.

Since direction u is chosen uniformly at random, the angle θ between $\overrightarrow{p_i p_j}$ and u is uniformly chosen from the range $[0, 2\pi)$. In other words, p_j lies uniformly at random on a circle C_r with $r = \|p_i p_j\|$. Therefore, the event p_j lies in S corresponds to the fraction $A(r)/2\pi r$ of the points on C_r. Hence,

$$\Pr\{X_{i,j} = 1\} = \frac{A(r)}{2\pi r} \leq \frac{2\pi d_i}{2\pi r} = \frac{d_i}{r},$$

which completes the proof. □

Let $\mathbb{E}[X_{\cdot,j}] = \sum_{i=1}^{n} \mathbb{E}[X_{i,j}]$. An upper bound B on $\mathbb{E}[X_{\cdot,j}]$ yields an upper bound nB on $\mathbb{E}[X]$, because $\mathbb{E}[X] = \sum_{j=1}^{n} \mathbb{E}[X_{\cdot,j}]$. The rest of this section focuses on finding such an upper bound on $\mathbb{E}[X_{\cdot,j}]$.

Lemma 3 *For each $1 \leq j \leq n$, there is a permutation σ of $\{1, \ldots, n\}$ such that*

$$\mathbb{E}[X_{\cdot,j}] \leq 3 \sum_{i=1}^{n} \frac{d_{\sigma_i}}{\sqrt{\sum_{k=1}^{i} {d_{\sigma_k}}^2}}.$$

Proof. Let $p_{\sigma_1}, \ldots, p_{\sigma_n}$ be the points of P ordered in their increasing distance from p_j. Note that $p_{\sigma_1} = p_j$. For $1 \leq i \leq n$, let C_i be a circle of radius $d_{\sigma_i}/2$ centered at p_{σ_i}. For any pair of points p_{σ_i} and p_{σ_j}, $\|p_{\sigma_i} p_{\sigma_j}\| \geq \max\{d_{\sigma_i}, d_{\sigma_j}\} \geq (d_{\sigma_i} + d_{\sigma_j})/2$. Therefore, all C_i's are non-overlapping.

Fix an index $2 \leq i \leq n$. Let $\ell = \|p_{\sigma_i} p_j\|$, and $B_i = \{C_1, \ldots, C_i\}$. Every circle in B_i has radius at most $\ell/2$, and its center lies within distance ℓ to p_j. (See Figure 4.) Therefore, all circles in B_i fit in a disk C of radius $\frac{3}{2}\ell$ centered at p_j. As the circles are non-overlapping, the area of C must be at least as large as the total area of the circles in B_i. Therefore, $(\frac{3\ell}{2})^2\pi \geq \sum_{k=1}^{i} (\frac{d_{\sigma_k}}{2})^2\pi$, and thus, $\ell = \|p_{\sigma_i} p_j\| \geq \frac{1}{3}\sqrt{\sum_{k=1}^{i} {d_{\sigma_k}}^2}$. Now,

$$\begin{aligned}\mathbb{E}[X_{\cdot,j}] = \sum_{i=1}^{n} \mathbb{E}[X_{i,j}] &\leq \sum_{i \in [n]-\{j\}} \frac{d_i}{\|p_i p_j\|} \quad \text{(by Lemma 2)} \\ &= \sum_{i=2}^{n} \frac{d_{\sigma_i}}{\|p_{\sigma_i} p_j\|} \leq \sum_{i=2}^{n} \frac{3 \cdot d_{\sigma_i}}{\sqrt{\sum_{k=1}^{i} {d_{\sigma_k}}^2}},\end{aligned}$$

which implies the lemma's statement. □

Based on the upper bound proved in Lemma 3, we define the following function:

$$f(a_1, \ldots, a_n) = \sum_{i=1}^{n} \frac{a_i}{\sqrt{\sum_{j=1}^{i} {a_j}^2}},$$

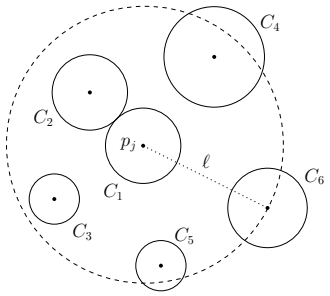


Figure 4: A set of non-overlapping circles $\{C_1, \ldots, C_6\}$.

where $a_1, \ldots, a_n$ is a sequence of real numbers. We prove some useful properties of f in the next lemmas.

Lemma 4 *Let $A = \{a_1, \ldots, a_n\}$ be a set of positive real numbers, and σ be a permutation of A. Then $f(\sigma)$ is maximized if σ is a non-decreasing sequence.*

Proof. Suppose by contradiction that f is maximized by a permutation $\sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_n\}$ which is not non-decreasing. Then, there exists an index i such that $x = \sigma_i > \sigma_{i+1} = y$. Let $\pi = \{\sigma_1, \ldots, \sigma_{i+1}, \sigma_i, \ldots, \sigma_n\}$ be the ordering achieved by swapping σ_i and σ_{i+1}. As σ is an ordering that maximizes f, we have $f(\sigma) \geq f(\pi)$. Since the two permutations only differ in the i-th and $(i+1)$-th term, by the definition of f, and by setting $s = x^2 + y^2 + \sum_{j=1}^{i-1} {\sigma_j}^2$, we have

$$\frac{x}{\sqrt{s - y^2}} + \frac{y}{\sqrt{s}} \;\geq\; \frac{x}{\sqrt{s}} + \frac{y}{\sqrt{s - x^2}}$$

which yields

$$x \cdot \left[\frac{1}{\sqrt{s - x^2}} - \frac{1}{\sqrt{s}}\right]^{-1} \geq y \cdot \left[\frac{1}{\sqrt{s - y^2}} - \frac{1}{\sqrt{s}}\right]^{-1}.$$

Since function $z \cdot \left[\frac{1}{\sqrt{s - z^2}} - \frac{1}{\sqrt{s}}\right]^{-1}$ is decreasing in the range $(0, s)$, the last inequality implies $x \leq y$. But this contradicts the fact that $x > y$. □

Lemma 5 *For any integer $n \geq 1$, and any real number $a \geq 0$,*

$$\sum_{i=1}^{n} \frac{1}{\sqrt{a + i}} \leq 2\sqrt{n}.$$

Proof. Since $\frac{1}{\sqrt{x}}$ is a decreasing function on $(0, +\infty)$, for any real number $b > 1$, we have

$$\int_{x=b-1}^{b} \frac{1}{\sqrt{x}} \;>\; \int_{x=b-1}^{b} \frac{1}{\sqrt{b}} \;=\; \frac{1}{\sqrt{b}}.$$

Therefore,

$$\sum_{i=1}^{n} \frac{1}{\sqrt{a + i}} \leq \int_{x=a}^{a+n} \frac{1}{\sqrt{x}} dx = 2(\sqrt{a + n} - \sqrt{a}) \leq 2\sqrt{n},$$

where the last inequality follows from the fact that $\sqrt{x+y} \leq \sqrt{x} + \sqrt{y}$, for all $x, y \geq 0$. □

Lemma 6 *Given real numbers $a_1, \ldots, a_n$ with $1 \leq a_i \leq c$, for some constant $c \geq 1$,*

$$f(a_1, \ldots, a_n) \leq 2b\sqrt{n}\sqrt{\log_b c + 1}.$$

for all $b > 1$.

Proof. Let $\hat{a}_i = b^{\lfloor \log_b a_i \rfloor}$. Since $b > 1$ and $a_i \geq 1$, we have $\hat{a}_i \leq a_i < b \cdot \hat{a}_i$. Therefore,

$$\begin{aligned} f(a_1, \ldots, a_n) &= \sum_{i=1}^{n} \frac{a_i}{\sqrt{\sum_{j=1}^{i} {a_j}^2}} \\ &\leq \sum_{i=1}^{n} \frac{b \cdot \hat{a}_i}{\sqrt{\sum_{j=1}^{i} \hat{a}_j^2}} = b \cdot f(\hat{a}_1, \ldots, \hat{a}_n). \end{aligned}$$

By Lemma 4, $f(\hat{a}_1, \ldots, \hat{a}_n)$ is maximized when $\hat{a}$'s are sorted non-decreasingly. Let $s_i = |\{j : \lfloor \log_b a_j \rfloor = i\}|$, for all $i \in \{0, 1, \ldots, \lfloor \log_b c \rfloor\}$. Then

$$\begin{aligned} f(\hat{a}_1, \ldots, \hat{a}_n) &\leq \sum_{i=0}^{\lfloor \log_b c \rfloor} \sum_{k=1}^{s_i} \frac{b^i}{\sqrt{k \cdot b^{2i} + \sum_{j=0}^{i-1} s_j \cdot b^{2j}}} \\ &= \sum_{i=0}^{\lfloor \log_b c \rfloor} \sum_{k=1}^{s_i} \frac{1}{\sqrt{k + \sum_{j=0}^{i-1} s_j \cdot b^{2(j-i)}}} \end{aligned}$$

which by Lemma 5 is at most $2\sum_{i=0}^{\lfloor \log_b c \rfloor} \sqrt{s_i}$. This sum is maximized at equality. Hence, as $\sum s_i = n$, we have

$$\sum_{i=0}^{\lfloor \log_b c \rfloor} \sqrt{s_i} \leq \sum_{i=0}^{\lfloor \log_b c \rfloor} \sqrt{\frac{n}{\lfloor \log_b c \rfloor + 1}} \leq \sqrt{n} \cdot \sqrt{\log_b c + 1}.$$

Therefore, $f(a_1, \ldots, a_n) \leq 2b\sqrt{n} \cdot \sqrt{\log_b c + 1}$. □

Now, we have all the ingredients needed to prove the main theorem of this section.

Theorem 7 *The expected runtime of Algorithm 1 on a set P of n points is $O(n\sqrt{n} \cdot \sqrt{\log \delta + 1})$, where $\delta = \max_i \{d_i\} / \min_i \{d_i\}$ and $d_i = \min_{q \in P \setminus \{p_i\}} \|p_i q\|$.*

Proof. By Lemma 3, $\mathbb{E}[X_{.,j}]$ is upper bounded by $3f(\sigma_1, \ldots, \sigma_n)$ for some permutation σ of $\{d_1, \ldots, d_n\}$. Scaling all variables by a constant does not change $f(\sigma_1, \ldots, \sigma_n)$. Therefore, we can assume w.l.o.g. that $1 \leq \sigma_i \leq \delta$ for all i. By setting $b = 2$ and $c = \delta$ in Lemma 6, we get

$$\mathbb{E}[X_{.,j}] \leq 12\sqrt{n}\sqrt{\log_2 \delta + 1}.$$

Therefore, $\mathbb{E}[X] = \sum_{j=1}^{n} \mathbb{E}[X_{.,j}]$ is upper bounded by $12n\sqrt{n}\sqrt{\log_2 \delta + 1}$, which completes the proof. □

4.1 Lower Bound

In this section, we show that the analysis presented in Section 4 is essentially tight by providing a lower bound example on which Algorithm 1 has a matching expected runtime. Our example is simply formed by the points of a $\sqrt{n} \times \sqrt{n}$ square lattice. The nearest neighbor to each point in this lattice has distance exactly one, and hence, $\delta = 1$ in this case. The following theorem proves a lower bound of $\Omega(n\sqrt{n})$ on the expected runtime of the algorithm on this example, which matches the upper bound of $O(n\sqrt{n})$ proved in the previous section.

Theorem 8 *The expected runtime of Algorithm 1 on a $\sqrt{n} \times \sqrt{n}$ square lattice is $\Omega(n\sqrt{n})$.*

Proof. Let $P = \{p_1, \ldots, p_n\}$ be the set of points on the lattice, and let u be the random unit vector chosen by the algorithm. The nearest neighbor to each point in P has distance one. Therefore, if X is a random variable indicating the size of the set $\{(p_i, p_j) : \|p_i p_j\|_u \leq 1\}$, then $\mathbb{E}[X]$ is a lower bound on the expected runtime of the algorithm.

We first claim that $\Pr\{\|p_i p_j\|_u \leq 1\} \geq \frac{1}{\pi \cdot \|p_i p_j\|}$, for all $1 \leq i, j \leq n$. Fix two points p_i and p_j. Let ℓ be a line in direction u passing through p_i, and let p'_j be the projection of p_j onto ℓ. Therefore, p_i, p_j, and p'_j form a right triangle. (See Figure 5.) Now, $\|p_i p_j\|_u \leq 1$ holds if and only if

$$\angle p_i p_j p'_j = \arcsin\left(\frac{\|p_i p'_j\|}{\|p_i p_j\|}\right) \leq \arcsin\left(\frac{1}{\|p_i p_j\|}\right).$$

As $\angle p_i p_j p'_j$ is chosen randomly, and $\arcsin(x) > x$ for all $0 < x \leq 1$, we have

$$\Pr\{\|p_i p_j\|_u \leq 1\} \geq \frac{1}{\pi} \arcsin\left(\frac{1}{\|p_i p_j\|}\right) > \frac{1}{\pi \cdot \|p_i p_j\|}.$$

Every two points in the lattice have distance at most $2\sqrt{n}$. Therefore, $\Pr\{\|p_i p_j\|_u \leq 1\} > \frac{1}{2\pi\sqrt{n}}$. Thus,

$$\mathbb{E}[X] > \frac{n(n-1)}{2\pi\sqrt{n}},$$

and hence, the expected runtime of the algorithm is $\Omega(n\sqrt{n})$. □

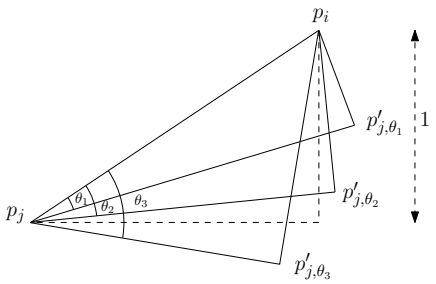


Figure 5: Projection of p_j on different lines specified by the random vector u.

5 Conclusions

In this paper, we analyzed an extremely simple randomized algorithm for the all nearest neighbors problem. We proved that the algorithm has $O(\alpha n\sqrt{n})$ expected runtime, where α is a parameter of the input point set, usually bounded by a constant in practice.

Our analysis can be extended in a natural way to the case of general L_p metric, yielding the same expected runtime. For higher d-dimensional space, we conjecture that the expected runtime of the algorithm is $O(n^{2-\frac{1}{d}}\,\mathrm{poly}(\alpha))$. We can also extend the algorithm to report k nearest neighbors of each point. While our analysis immediately implies an upper bound of $O(k\alpha \cdot n\sqrt{n})$ on the expected number of comparisons made by the algorithm, it is intriguing to obtain a tighter analysis for this variant of the problem.

Acknowledgments The authors would like to thank Michael Tikhomirov for suggesting the analysis of the algorithm studied in this paper as an open problem.

References

[1] K. L. Clarkson. Fast algorithms for the all nearest neighbors problem. In *Proc. 24th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 226–232, 1983.

[2] S. Fortune. Voronoi diagrams and delaunay triangulations. In *Computing in Euclidean geometry*, pages 225–265. 1995.

[3] A. Guttman. R-trees: A dynamic index structure for spatial searching. In *Proc. ACM SIGMOD Conf.*, pages 47–57, 1984.

[4] D.-T. Lee and B. J. Schachter. Two algorithms for constructing a Delaunay triangulation. *Int. J. of Comput. & Inform. Sci.*, 9(3):219–242, 1980.

[5] H. Samet. The quadtree and related hierarchical data structures. *ACM Computing Surveys*, 16(2):187–260, 1984.

[6] P. M. Vaidya. An $O(n \log n)$ algorithm for the all-nearest-neighbors problem. *Discrete Comput. Geom.*, 4(2):101–115, 1989.

Hardness results on Voronoi, Laguerre and Apollonius diagrams

Kevin Buchin* Pedro Machado Manhães de Castro† Olivier Devillers‡ Menelaos Karavelas§

Abstract

We show that converting Apollonius and Laguerre diagrams from an already built Delaunay triangulation of a set of n points in 2D requires at least $\Omega(n \log n)$ computation time. We also show that converting an Apollonius diagram of a set of n weighted points in 2D from a Laguerre diagram and vice-versa requires at least $\Omega(n \log n)$ computation time as well. Furthermore, we present a very simple randomized incremental construction algorithm that takes expected $O(n \log n)$ computation time to build an Apollonius diagram of non-overlapping circles in 2D.

1 Introduction

Voronoi diagrams in 2D are one of the most classical objects of computational geometry. Given a set of n points $S = \{p_1, p_2, \ldots, p_n\}$ in the plane, consider n regions $\mathcal{R}_i$ such that $\mathcal{R}_i$ contains all the points *closer* to p_i than any other point $p_j \in S$ with $p_i \neq p_j$. The word *closer* here is crucial. If the distance used is the Euclidean distance in the plane (i.e., $\|p - p'\|$ for points p and p'), each region is a convex (possibly unbounded) polygon and their union is the Voronoi diagram of S; see Figure 1a. The dual of the Voronoi diagram of S is the *Delaunay triangulation* of S.

Now, consider a set of n circles (or weighted points) $\Sigma = \{c_1, c_2, \ldots, c_n\}$ in the plane, with $c_i = (p_i, r_i)$ for $i = 1, \ldots, n$ and the regions $\mathcal{R}_i$, for $i = 1, \ldots, n$, defined as above but with the concept of Euclidean distance replaced by the Power distance between a point and a circle (i.e., $\|p-p'\|^2-r^2$ for point p' and circle $c = (p, r)$). Then, again, the regions are convex, but their union is the *Laguerre diagram* (or Power diagram) of Σ. Here, input circles may not have a region associated with;[1] we call such circles as *hidden circles*. The dual of the Laguerre diagram of Σ is the *regular triangulation* of Σ; see Figure 1b. The dual, as its name suggest, must be a triangulation. Furthermore, it might not include all centers of the input circles as vertices, since the final construction might have hidden circles.

Finally, consider the same set of circles Σ above, but the distance now is the signed Euclidean distance between a point and a circle (i.e., $\|p - p'\| - r$ for point p' and circle $c = (p, r)$). Then, the regions are no longer convex and their union is the *Apollonius diagram* of Σ; see Figure 1c. Actually, these regions are bounded by segments of lines or hyperbola. As in the Laguerre diagram, some input circles may not have a region associated with, which we call analogously a *hidden point*. The dual of the Apollonius diagram of Σ is the *Apollonius graph* of Σ. Conversely to both structures mentioned above, the Apollonius graph may not be a triangulation.

A lower bound of $\Omega(n \log n)$ in the algebraic computation tree model of computation [1] is known for building any of these diagrams for an input set of size n; this can be proved by a reduction to the problem of sorting n numbers. Also, optimal algorithms achieving a computational complexity of $O(n \log n)$ for building any of these three diagrams (or their duals) are well known [4]. Randomized incremental constructions obtaining an expected cost of $O(n \log n)$ for Voronoi and Power diagrams are also computational geometry classics [5, 3, 2] with very efficient implementations. However, the situation is not the same for the construction of an Apollonius diagram.

There is an optimal algorithm for Apollonius diagram construction: it is a sweep-line algorithm that has been proposed in the early days of computational geometry [4]. However, this algorithm is complicated, requires high degree predicates, and is not used in practice. A provable expected $O(n \log n)$ Apollonius diagram randomized incremental construction can be made available by using an abstract Voronoi diagrams construction framework [9], but it remains theoretical with no efficient implementation available to the best of our knowledge. The implementation in CGAL [6, 7, 8] is based on randomized incremental construction and more precisely on a generalization of the Delaunay hierarchy [2]. The Delaunay hierarchy allows a logarithmic time point location in a Delaunay triangulation (or a Voronoi diagram). Unfortunately, while generalizing the algorithm to Laguerre or Apollonius diagrams is straightforward, the proof of complexity requires some special proper-

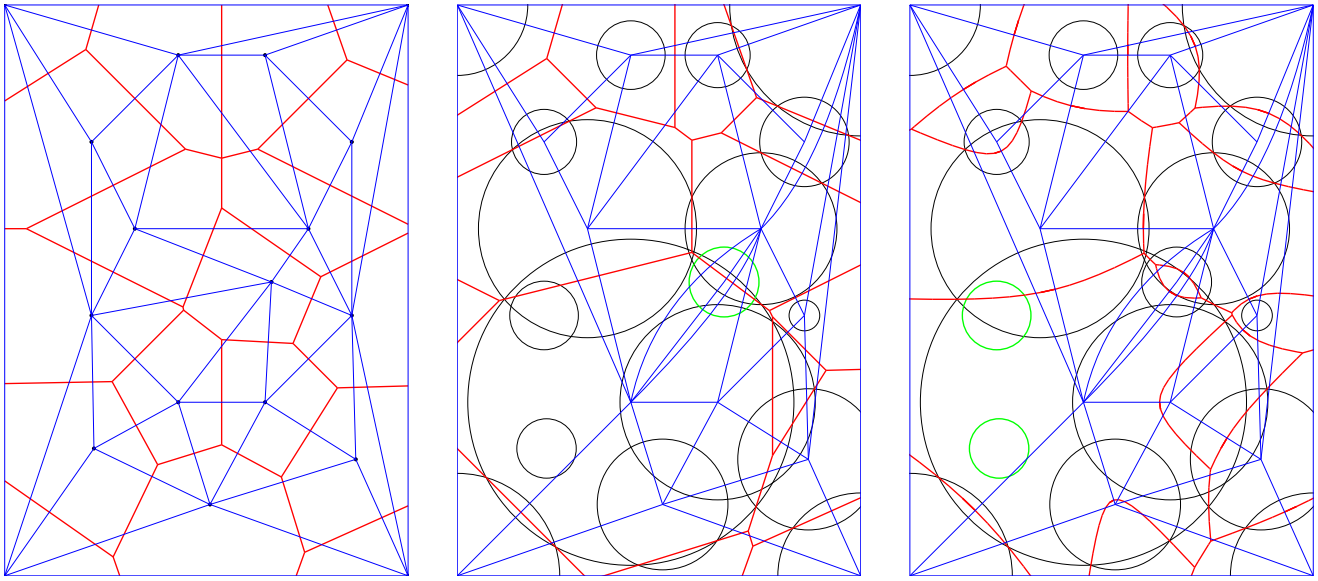
*Technical University Eindhoven, The Netherlands www.win.tue.nl/∼kbuchin

†Universidade Federal de Pernambuco, Brasil cin.ufpe.br/∼pmmc/

‡Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France. members.loria.fr/Olivier.Devillers

§Department of Applied Mathematics, University of Crete & Institute of Applied and Computational Mathematics, Greece tem.uoc.gr/∼mkaravel/

[1] The term *distance* albeit classically used in that case is actually not the most appropriate, since it can be negative when points are inside the circle.



(a) Voronoi (b) Laguerre (c) Apollonius

Figure 1: **Diagrams and their dual.** *In red, we have respectively: the Voronoi diagram, the Laguerre diagram and the Apollonius diagram. In blue we have their dual. Green circles are hidden. For the three figures above, input points in (a) are the same as the centers of input circles in (b) and (c). Moreover, the input circles are the same for (b) and (c). As we can see above these diagrams may have close combinatorial structure, and hence we may ask whether it is cheap to convert from one to another.*

ties of Delaunay triangulation and does not generalize so easily. Karavelas and Yvinec [7] propose to go from a site to the next one using a dichotomic search in the neighbors of the site. This approach yields a provable expected time complexity of $O(n \log^2 n)$ to construct the Apollonius diagram.

With this in mind, we propose the following randomized incremental contruction: computing the Apollonius diagram of a set of circles with zero radius (i.e., the Voronoi diagram of the centers), then increasing the radii of all circles in a random order maintaining the diagram. In this paper, we prove that such approach is in expected $O(n \log n)$ computational cost for n non-overlapping circles.

The idea above is appealing because we already have very efficient algorithms and software for the computation of Voronoi diagram. Also, as shown by the similarity between the different diagrams in Figure 1, one might hope that converting one diagram to another could be done quickly (i.e., linear in the size of the input set). Then, the following question arises:

> *" Is the knowledge of any of the Voronoi, Laguerre (power), or Apollonius diagrams of any help to compute any of the two others? "*

In this paper, we answer negatively any of the six instances of that question.

2 Lower bounds

In this section, we present the hardness results on any conversion between the diagrams mentioned above. More precisely, we show that such a conversion has a $\Omega(n \log n)$ computational cost in the algebraic computation tree model of computation [1]. When more convenient, we consider the dual of these structures, respectively: Delaunay triangulation, regular triangulation and Apollonius graph (converting primals to their duals and vice-versa is of course in $\Theta(n)$).

2.1 Knowing the Voronoi diagram does not help to compute the Laguerre diagram

Theorem 1 *Computing the regular triangulation of a set of n weighted points knowing the Delaunay triangulation of the unweighted points has $\Omega(n \log n)$ complexity lower bound.*

Proof. Consider a set of points $p_i = \{(x_i, y_i)\}_{0 \leq i < n}$ with $y_i > 0$. We first remark that the Laguerre diagram allows to sort numbers, actually assigning weights $w_i = y_i^2$ to points p_i (i.e. radius y_i) ensure that points with consecutive x coordinates are neighbors in the regular triangulation. Actually for a point $(x, 0)$ its weighted distance to p_i is $(x - x_i)^2$ and a moving point on the x-axis gets as closest site all the sites in the order of their x-coordinates (see Figure 2). If the Laguerre di-

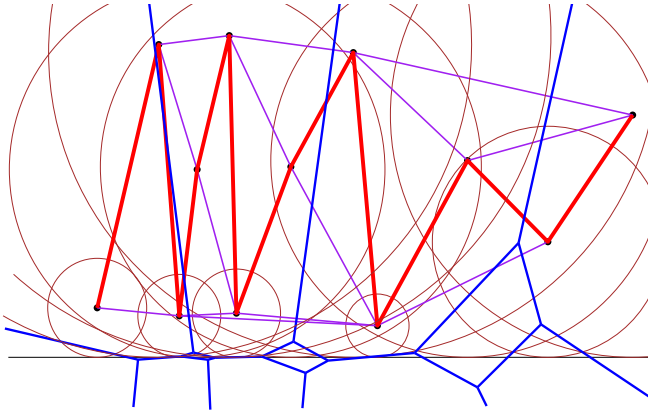


Figure 2: **Laguerre diagram allows to sort the centers by x-coordinate.** *The red x monotone curve is a subset of the dual: the regular triangulation.*

agram is known, the x-order can be retrieved looking for the x-successor of a site in its neighbors in the regular triangulation. Since the sum of the degrees of all sites is less than $6n$ this operation can be done in linear time.

It is known that having the Delaunay triangulation already built does not help to sort points by x coordinates (in the sense that it is still $\Omega(n \log n)$). More precisely, Seidel [10] proposed a construction that, given n numbers, presents a set of points having these numbers as abscissa and their Delaunay triangulation in linear time. If Delaunay would help to sort vertices by x coordinates it would contradict the sorting lower bound.

Combining the two constructions, it is possible to sort n numbers by first building Seidel's Delaunay triangulation in linear time, then building the regular triangulation from the Delaunay triangulation with the above weights and finally extracting the x-order of the sites. If the Delaunay to regular transform used $o(n \log n)$ computation time, we would get a contradiction on the lower bound result for sorting. □

2.2 Knowing the Voronoi diagram does not help to compute the Apollonius diagram

Theorem 2 *Computing the Apollonius diagram of a set of n circles knowing the Delaunay triangulation of the centers has $\Omega(n \log n)$ complexity lower bound.*

Proof. The construction is almost the same as the one in the proof of Theorem 1. We use the same circles and the same moving point on the line $x = 0$, the distance from $(x, 0)$ to the weighted points p_i is $\sqrt{(x - x_i)^2 + y_i(y_i - 1)}$. The distance to the closest site is always positive, being zero for all points in turn according to their x-order. The rest of the proof is identical. □

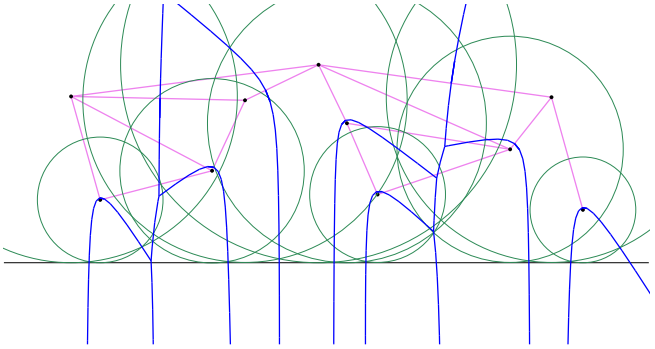


Figure 3: **Apollonius diagram allows to sort the centers by x-coordinate.** *The lower part of the dual (in pink) enumerates all points in x-order.*

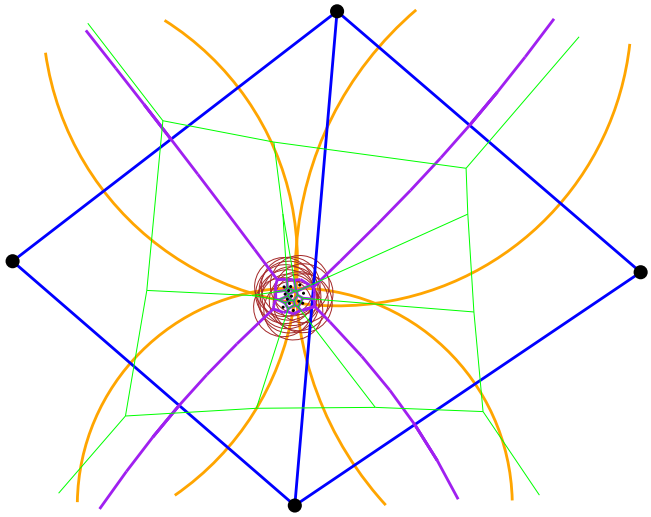


Figure 4: **Regular triangulation does not help to compute Apollonius diagram nor the Voronoi diagram.** *Voronoi diagram in green, Apollonius diagram in purple, regular triangulation in blue.*

2.3 Other lower bounds

Computing Apollonius or Laguerre diagram is not so much helpful when many points are hidden. For instance, the conversion of either regular or Apollonius to Delaunay triangulation is hopeless. This is because, for any set of centers, by adjusting the radii, essentially all points but one can be hidden, thus the Delaunay would need to be built from scratch. The hardness results for converting Apollonius to Regular and vice-versa are presented in the sequel.

Theorem 3 *Computing the Apollonius diagram of a set of n circles or the Voronoi diagram of their centers knowing the regular triangulation has $\Omega(n \log n)$ complexity lower bound.*

Proof. As described in Figure 4, consider four big circles that pass close to the origin, and a set of small cir-

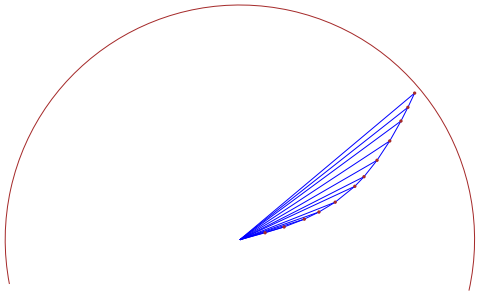


Figure 5: **Apollonius diagram does not help to compute the regular triangulation nor the Delaunay triangulation.** *Delaunay triangulation in blue. Note the big red circle hidden every point inside it.*

cles of same radii centered close to the origin. Radii can be tuned so that all centers of small circles are hidden in the regular triangulation while their Voronoi diagram, up to the convex hull, is present in the Apollonius or Voronoi diagram. □

Theorem 4 *Computing the regular triangulation of a set of n circles or the Delaunay triangulation of their centers knowing the Apollonius diagram has $\Omega(n \log n)$ complexity lower bound.*

Proof. Consider a set of circles with centers $p_i = (x_i, x_i^2)$ for $0 \leq i < n$ with $x_i > 0$ and zero radius plus one circle center at the origin with radius big enough to contain all p_i's. Then the Delaunay and the regular triangulation are equal and allow to sort the points by x-coordinate while the Apollonius diagram does not give any hint since the big circle is the only one with non empty region (see Figure 5). □

3 Building the Apollonius diagram of non-overlapping circles quickly

3.1 Static randomized incremental construction

Let $AG(\Sigma)$ be the Apollonius graph of $\Sigma = \{c_1, c_2, \ldots, c_i\}$ and $c_i = (p_i, w_i)$, $AG_i(\Sigma)$ be the Apollonius graph of $\{c_1, c_2, \ldots, c_i, (p_{i+1}, 0), \ldots, (p_n, 0)\}$ and $DT(S)$ be the Delaunay triangulation of $S = \{p_1, p_2, \ldots, p_n\}$. The Apollonius graph can be computed as described in Algorithm 1.

First, the Delaunay triangulation of the centers of the circles is computed, which is equivalent to the Apollonius graph of circles with radius zero. Then the circles with their true radii are incrementally added in a random order updating the Apollonius graph. Notice that the insertion of a circle hides its center, thus the final result is just the Apollonius graph of the circles. We obtain the Apollonius diagram by extracting the dual from the primal.

Data: A set Σ of n circles (or weighted points) $c_i = (p_i, w_i), i = 1 \ldots n$.
Result: $AG(\Sigma)$, which is the Apollonius graph of Σ.
Let $S = \{p_i, i = 1 \ldots n\}$;
Build $DT(S)$ the Delaunay Triangulation of S;
Let $AG_0(\Sigma)$ be $DT(S)$;
drop $DT(S)$;
Shuffle indices of circles in Σ;
for $i = 1 \ldots n$ **do**
  get $AG_i(\Sigma)$ by inserting c_i into $AG_{i-1}(\Sigma)$ using p_i as hint;
end
return $AG_n(\Sigma)$;

Algorithm 1: Algorithm for building the Apollonius graph of non-overlapping circles.

Theorem 5 *Algorithm 1 constructs the Apollonius diagram of n disjoint circles in $O(n \log n)$ expected time.*

Proof. Let $d^o_{AG_i(\Sigma)}(c)$ the degree of c in $AG_i(\Sigma)$. Consider the diagram at step i, its total complexity is linear, thus the expected complexity of the cell of the last (the ith) circle $\mathcal{E}_i$ is bounded as follows:

$$\begin{aligned}\mathcal{E}_i &= \frac{1}{i} \sum_{c \in \{c_1, \ldots, c_i\}} d^o_{AG_i(\Sigma)}(c) \\ &\leq \frac{1}{i} \sum_{c \in \{c_1, \ldots, c_i\} \cup \{p_{i+1}, \ldots, p_n\}} d^o_{AG_i(\Sigma)}(c) \\ &\leq \frac{6n}{i}.\end{aligned}$$

When summing $\mathcal{E}_i$, for $i = 1, \ldots, n$, the total structural change is $O(n \log n)$. □

3.2 Lower bound on the number of structural changes overall

For usual randomized incremental construction in the context of Voronoi diagrams of points, the total complexity of the structural change has $O(n)$ size and the usual expected $O(n \log n)$ computation time arises because of point locations, which is actually the algorithm's bottleneck. In the second part of our algorithm (when converting from Voronoi to Apollonius), the point location is avoided, but the size of structural change becomes $\Omega(n \log n)$ since the total size of the diagram is linear from the beginning of that second part. Figure 6 shows an example where the structural change has actually $\Theta(n \log n)$ size.

3.3 Issue with overlapping circles

When a circle is inserted, if its center is not hidden in the diagram just before the insertion, this center is a

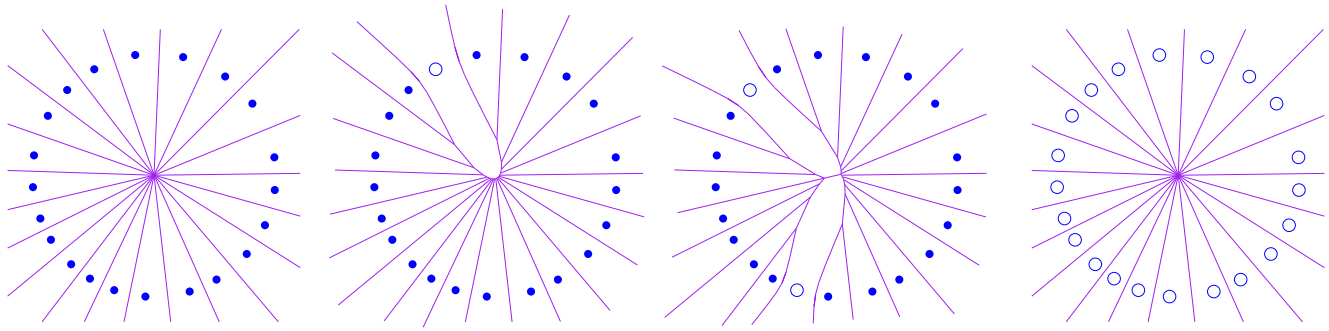


Figure 6: **The number of structural changes is not linear.** *Going from Voronoi to Apollonius incrementally may requires $\Omega(n \log n)$ expected structural changes: the first insertion requires $\Omega(n)$, the second $\Omega(\frac{n}{2})$ expected, and so on.*

perfect hint to locate the circle. The point location part is completely avoided since a conflict with the new circle is known, which is the case for disjoint circles. However, if the center is already hidden, which can happen when allowing overlapping circles as input, point location is still needed.

Consider one big circle, and n small disjoint circles intersecting the big circle whose centers are inside the big circle. A typical increase of radius for a small circle after the insertion of the big circle is problematic. To avoid point location, we need a good hint to insert the circle, but its center is hidden and no longer present in the Apollonius graph and its nearest neighbor is the big circle and has a high degree in the Apollonius graph. Thus defining an hint allowing fast point location seems difficult in such a case.

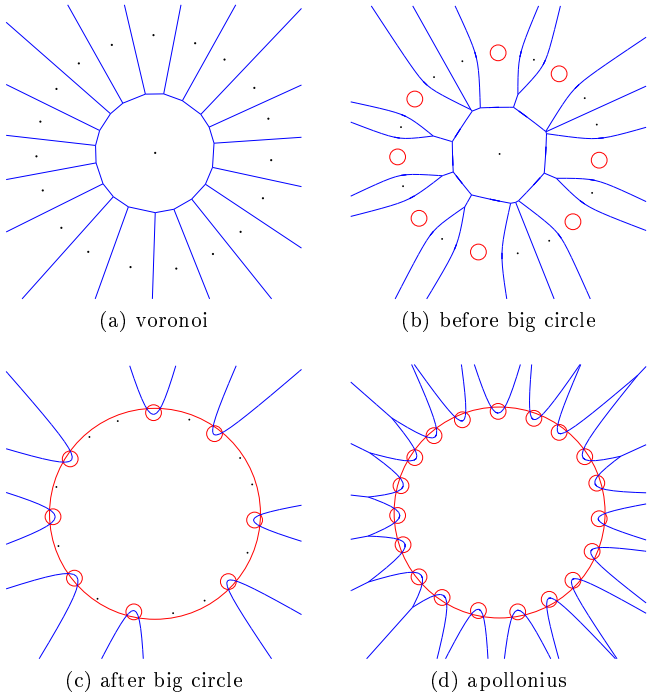


Figure 7: **Handling hidden circles may be not trivial.** *Black points are circles yet to be grown whereas red circles are already inserted. In blue, we have the corresponding Apollonius diagram of the set of points and circles. Note that after inserting the big circle, each remaining circle to be grown has as its nearest neighbor the big circle, which has a high degree.*

4 Acknowledgement

The authors would like to thank the reviewers for pinpointing reference [9].

References

[1] M. Ben-Or. Lower Bounds for Algebraic Computation Trees *Proc. 15th ACM Symposium on Theory of Computing*, pages 80–86, 1983.

[2] O. Devillers. The Delaunay hierarchy *Internat. J. Found. Comput. Sci.*, 13:163–180, 2002.

[3] H. Edelsbrunner and N. R. Shah. Incremental topological flipping works for regular triangulations *Algorithmica*, 15(3):223–241, 1996.

[4] S. J. Fortune. A sweepline algorithm for Voronoi diagrams *Algorithmica*, 2:153–174, 1987.

[5] L. J. Guibas, D. E. Knuth, and M. Sharir. Randomized incremental construction of Delaunay and Voronoi diagrams *Algorithmica*, 7(1):381–413, 1992.

[6] M. I. Karavelas and M. Yvinec. 2D Apollonius graphs (Delaunay graphs of disks) In *CGAL User and Reference Manual*. CGAL Editorial Board, 4.2 edition, 2013.

[7] M. I. Karavelas and M. Yvinec. Dynamic additively weighted Voronoi diagrams in 2D In *Proceedings of the 10th European Symposium on Algorithms*, volume 2461 of *LNCS*, pages 568–598, 2002.

[8] M. I. Karavelas and M. Yvinec. The Voronoi diagram of planar convex objects In *Proceedings of the 11th European Symposium on Algorithms*, volume 2832 of *LNCS*, pages 337–348, 2003.

[9] K. Mehlhorn, St. Meiser, and C. Ó'Dúnlaing. On the construction of abstract voronoi diagrams *Discrete & Computational Geometry*, 6(2):211–224, 1991.

[10] R. Seidel. A method for proving lower bounds for certain geometric problems In *Computational Geometry*, G. T. Toussaint, North-Holland, pages 319–334, 1985.

A Note on Guarding Staircase Polygons*

Matt Gibson† Erik Krohn‡ Bengt J. Nilsson§ Matthew Rayford¶ Paweł Żyliński‖

Abstract

We exhibit two linear time approximation algorithms for guarding rectilinear staircase polygons both having approximation factor 2. The first algorithm benefits from its simplicity, whereas the second provides more insight to the problem.

1 Introduction

The *art gallery problem* is one of the best known problems in computational geometry. An instance of the art gallery problem takes as input a polygon $\mathbf{P}$ and seeks to find a set of points $G \subseteq \mathbf{P}$ such that every point $p \in \mathbf{P}$ is seen by a point in G. We call this set G a *guarding set*. In the point guarding problem, guards can be placed anywhere inside of $\mathbf{P}$. The optimization problem is thus defined as finding the smallest such G.

The art gallery problem has been shown to be NP-hard and APX-hard for simple polygons [1, 7, 13]. A constant factor approximation for vertex guarding a simple polygon has been claimed in [3]. Bonnet and Miltzow [4] present a polynomial time algorithm for finding a point guard cover with approximation factor $O(\log OPT)$ under certain mild restrictions. Achieving constant ratio approximation for the point guarding problem in a simple polygon remains elusive and precious little is known about it [5].

Due to the inherent difficulty in fully understanding the art gallery problem for simple polygons, there has been some work done on finding constant factor, polynomial time approximation algorithms for guarding polygons with some additional structure. This has been successful for monotone polygons [8, 11] and for guarding orthogonal polygons with sliding cameras [2, 6]

Our objective is to establish for what polygon classes the guarding problem lies on the borderline between tractable and intractable. Given the intractability for monotone polygons [10], we look at an even simpler class of polygons.

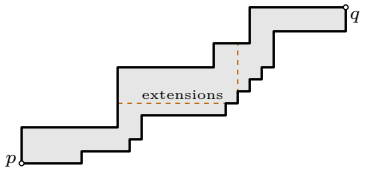


Figure 1: Illustrating staircase polygons.

A *staircase polygon* is a restricted version of a rectilinear polygon. A rectilinear polygon is a polygon where all interior angles are either 90° or 270°. We define a staircase polygon to be a rectilinear and xy-monotone polygon where the "staircase" is only allowed to go "up" and to the "right." More formally, let us call the lowest-leftmost point of the polygon p and the highest-rightmost point of the polygon q. The edges of the polygon, as one traverses the boundary clockwise (resp., counterclockwise) from p to q, are called the *ceiling* (resp., *floor*). The leftmost ceiling edge is a vertical edge that goes up out of p. The ceiling will then alternate between 90° turns going right, then left. The lowest floor edge is a horizontal edge that goes right out of p. The floor will then alternate between 90° turns going left, then right. The floor and ceiling boundary will not cross until they meet at q, see Figure 1.

The maximal horizontal and vertical segments inside $\mathbf{P}$, collinear to the edges of $\mathbf{P}$ are called the *extensions* in $\mathbf{P}$; see Figure 1. For a convex vertex u of $\mathbf{P}$, the extension of the horizontal (resp., vertical) edge incident to u is denoted by $\text{he}(u)$ (resp., $\text{ve}(u)$). We sometimes use $\text{he}(p)$ (resp., by $\text{ve}(p)$) when p is a point on a horizontal (resp. vertical) edge.

1.1 Our Results

We consider point guarding staircase polygons. We will use G^* to denote an optimal point guard set for the staircase polygon $\mathbf{P}$. In Section 2.1, we give a simple greedy linear time algorithm that guarantees a guard set of size at most $2|G^*|-1$. In Section 2.2, we give another greedy linear time algorithm that guarantees a guard set of size at most $2|G^*|$ but also provides more insight into the difficulty in providing better approximations for the problem.

*The authors were supported by grant 2015/17/B/ST6/01887 (National Science Centre, Poland), by grant 2018-04001 from the Swedish Research Council, and by a faculty development grant from UW-Oshkosh.

†University of Texas at San Antonio, TX, USA..... email gibson@cs.utsa.edu

‡University of Wisconsin - Oshkosh, Oshkosh, USA..... email krohne@uwosh.edu

§Malmö University, SE-205 06 Malmö, Sweden..... email bengt.nilsson.TS@mau.se

¶University of Wisconsin - Oshkosh, Oshkosh, USA..... email mattrayford8686@gmail.com

‖University of Gdańsk, 80-952 Gdańsk, Poland..... email zylinski@inf.ug.edu.pl

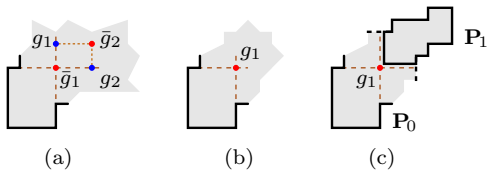


Figure 2: Constructing the CCCG set.

2 The Algorithms

2.1 The CCCG Algorithm

Let C^* be a smallest set of guards for the convex corners of the staircase polygon $\mathbf{P}$, i.e., the smallest set of points in $\mathbf{P}$ that see all the convex vertices of $\mathbf{P}$ but are not required to see anything else in $\mathbf{P}$. It is clear that $|C^*| \leq |G^*|$, since G^* also sees all of the convex vertices of $\mathbf{P}$. We will show that we can greedily and in linear time compute a *canonical convex corner guard set*, a CCCG set for short, of size $|C^*|$ that sees all the convex corners of $\mathbf{P}$. We then also show how to extend the CCCG set to a complete guard cover of $\mathbf{P}$.

Observe first that, given C^*, we can move any guard g to the nearest vertical extension of the floor to the right of g, and the nearest horizontal extension of the ceiling above g. More specifically, let $C_0^* \leftarrow C^*$, $C_0 \leftarrow \emptyset$, and $\mathbf{P}_0 \leftarrow \mathbf{P}$. Consider the first convex vertex u_1 on the floor and the first convex vertex $\bar{u}_1$ on the ceiling (we let the lowest leftmost convex vertex be the 0-th convex vertex on each boundary chain). If these two vertices are not seen by any single guard in C_0^*, then pick the lowest guard g_1 in C_0^* on the vertical extension $\mathrm{ve}(u_1)$ and the leftmost guard g_2 in C_0^* on the horizontal extension $\mathrm{he}(\bar{u}_1)$. Consider the axis-parallel rectangle inside $\mathbf{P}_0$ whose diagonal is $[g_1, g_2]$. We can then exchange g_1 and g_2 by two other guards $\bar{g}_1$ and $\bar{g}_2$ placed at the other two corners of that rectangle without changing the visibility of the convex vertices of $\mathbf{P}_0$; see Figure 2(a). Therefore, we let $C_1^* \leftarrow C_0^* \cup \{\bar{g}_2\} \setminus \{g_1, g_2\}$ and let $C_1 \leftarrow C_0 \cup \{\bar{g}_1\}$.

If the two first convex vertices are seen by a single guard g_1 in C_0^*, then g_1 lies on the intersection point of the vertical and horizontal extensions $\mathrm{ve}(u_1)$ and $\mathrm{he}(\bar{u}_1)$, respectively; see Figure 2(b). In this case, we let $C_1^* \leftarrow C_0^* \setminus \{g_1\}$ and let $C_1 \leftarrow C_0 \cup \{g_1\}$.

In both cases, we cut $\mathbf{P}_0$ vertically and horizontally along the two extensions $\mathrm{ve}(u_1)$ and $\mathrm{he}(\bar{u}_1)$, and retain the upper right portion as $\mathbf{P}_1$; see Figure 2(c). Notice that C_1^* is a guard set for the convex vertices in $\mathbf{P}_1$.

In general, given $\mathbf{P}_i$, C_i^*, and C_i, we repeat the steps described above to obtain $\mathbf{P}_{i+1}$, C_{i+1}^*, and C_{i+1}, respectively. After $|C^*|$ iterations in total, we finish with the set $\hat{C} = C_{|C^*|}$ (with $|\hat{C}| = |C^*|$). Notice that we have an ordering of the guards in $\hat{C}$ according to the iteration i where the new guard is added to C_i as $\hat{C}$ is being constructed.

Observation 1 *We can compute the CCCG set without having C^* to begin with. It suffices to realize that given C_i, we can obtain C_{i+1} by adding a guard at the intersection point of the horizontal and vertical extensions collinear with the first two convex corners not yet seen on the ceiling and the floor.*

The resulting CCCG set $\hat{C}$ covers the convex vertices of $\mathbf{P}$, but may have hidden parts in $\mathbf{P}$ that are unseen by the guards in $\hat{C}$; see Figure 3(a). It is clear that at most $|\hat{C}| - 1$ such unguarded parts of $\mathbf{P}$ can be introduced by $\hat{C}$, since any part must lie between two guards g_i and g_{i+1}, $1 \leq i < |C^*|$ in the ordering of $\hat{C}$. We prove that each unseen part can be guarded by a single additional guard.

Lemma 1 *If the region in $\mathbf{P}$ between g_i and g_{i+1} is not seen by the guards in $\hat{C}$, then this region is contained in a convex quadrilateral and can thus be guarded by a single guard.*

Proof. Consider the point p on the ceiling directly above g_i and the point q on the floor directly to the right of g_i; see Figures 3(b)–(d). The point p lies on the horizontal ceiling edge with the rightmost reflex vertex $\bar{v}_p$ and q lies on the vertical floor edge with the topmost reflex vertex v_q. The two convex vertices $\bar{u}_p$ and u_q directly above $\bar{v}_p$ and v_q on the ceiling and floor, respectively, are not seen by g_i, and so they must be seen by g_{i+1} by construction. Now, each of g_i and g_{i+1} sees one or two of the reflex vertices $\bar{v}_p$ and v_q, and if they see only one each, they must see different ones since at most one reflex vertex can block the other. This gives rise to the following three cases.

1. g_i and g_{i+1} both see $\bar{v}_p$ and v_q.

   This means that the triangles $\triangle g_i p \bar{v}_p$, $\triangle g_i \bar{v}_p v_q$, and $\triangle g_i v_q q$ are seen from g_i, and symmetrically, the triangles $\triangle g_{i+1} \bar{u}_p \bar{v}_p$, $\triangle g_{i+1} \bar{v}_p v_q$, and $\triangle g_{i+1} v_q u_q$ are seen from g_{i+1}. Thus, there is no unseen part between g_i and g_{i+1}; see Figure 3(b).

2. g_i sees $\bar{v}_p$.

   Hence, g_{i+1} sees v_q and the triangles $\triangle g_i p \bar{v}_p$, $\triangle g_i \bar{v}_p q$, $\triangle g_{i+1} \bar{u}_p v_q$, and $\triangle g_{i+1} v_q u_q$ are seen by g_i and g_{i+1}. Thus, only the convex quadrilateral $\square \bar{v}_p q v_q \bar{u}_p$ may contain unseen parts between g_i and g_{i+1}; see Figure 3(c).

3. g_i sees v_q.

   Symmetrically to the previous case, g_{i+1} sees $\bar{v}_p$ and the triangles $\triangle g_i p v_q$, $\triangle g_i v_q q$, $\triangle g_{i+1} \bar{u}_p \bar{v}_p$, and $\triangle g_{i+1} \bar{v}_p u_q$ are seen by g_i and g_{i+1}. Again, only the convex quadrilateral $\square \bar{v}_p p v_q u_q$ may contain unseen parts between g_i and g_{i+1}; see Figure 3(d).

This concludes the proof. □

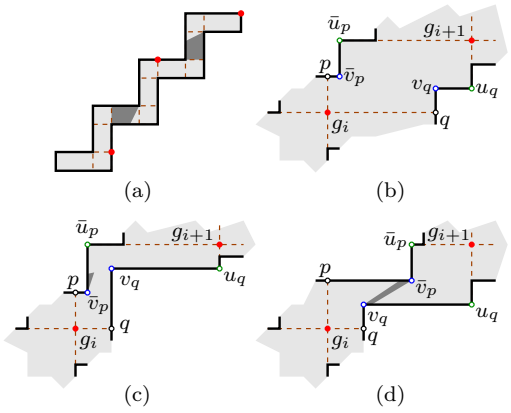


Figure 3: Illustrating the proof of Lemma 1. Unseen parts are shown in dark grey.

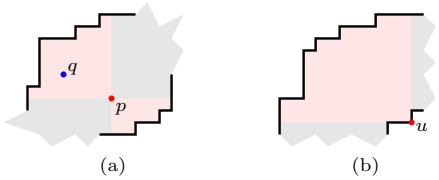


Figure 4: The fan visibility property and the fan visibility polygons (in pink) of a point p and of a convex vertex u.

Now, given the CCCG set $\hat{C}$, we consider a set G_X of extra guards in the convex quadrilaterals that contain unseen parts of $\mathbf{P}$. Let $G^{(1)} = \hat{C} \cup G_X$ be the set of *extended canonical convex corner guards*.

Theorem 2 *The set $G^{(1)}$ of extended canonical convex corner guards covers all of $\mathbf{P}$ and*

$$|G^{(1)}| = |\hat{C}| + |G_X| \leq 2|C^*| - 1 \leq 2|G^*| - 1.$$

We note that we can apply the algorithm for constructing a partial CCCG set as described above, starting with any pair of convex corners on the floor and ceiling that see each other in $\mathbf{P}$. We will use this property in the next section.

2.2 A Second Algorithm

Any point p can see any point q in a staircase polygon $\mathbf{P}$, if $x(q) \leq x(p)$ and $y(q) \geq y(p)$ or if $x(q) \geq x(p)$ and $y(q) \leq y(p)$, where $x(\cdot)$ and $y(\cdot)$ respectively denote the x- and y-coordinates of the corresponding points. We call this the *fan visibility property*; see Figure 4(a). The symmetry of visibility implies that for any convex vertex u (except the lowest leftmost vertex or the top rightmost vertex), the visibility polygon of u must contain a guard in any guard set of $\mathbf{P}$. The visibility polygon of u resembles a *fan* in $\mathbf{P}$; see Figure 4(b).

Order the convex vertices of the floor $u_0, \ldots, u_m$, where u_0 is the lowest leftmost vertex and u_m is the top rightmost vertex of $\mathbf{P}$. Similarly, order the convex vertices of the ceiling $\bar{u}_0, \ldots, \bar{u}_{\bar{m}}$, where $\bar{u}_0 = u_0$ and

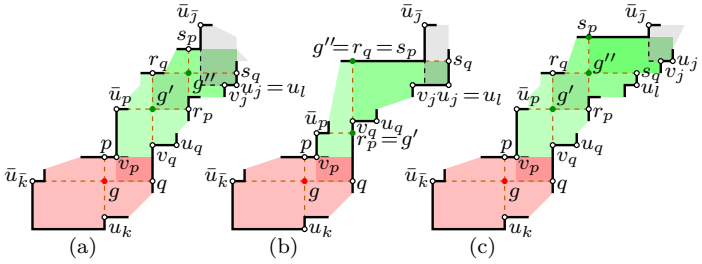


Figure 5: Illustrating the second algorithm.

$\bar{u}_{\bar{m}} = u_m$. Our next algorithm iteratively constructs a first guard using the CCCG algorithm as described in the previous section. It then places two or three additional guards ensuring that the region guarded using the fan visibility property contains at least two convex vertices such that their visibility polygons (fans) do not intersect. The polygon $\mathbf{P}$ can then be cut into one part visible by the (at most four) guards placed so far and the remaining polygon. In this way, we achieve a 2-approximation, since any guard set requires two guards in the fans. We present the algorithm in further detail below.

We initialize the algorithm with $\mathbf{P}_0 \leftarrow \mathbf{P}$, $G_0^{(2)} \leftarrow \emptyset$, $i \leftarrow 0$, $k \leftarrow 1$, and $\bar{k} \leftarrow 1$. Let g be the first guard obtained by running the CCCG algorithm from the previous section on $\mathbf{P}_i$. Let p be the upper endpoint of the vertical extension $\mathrm{ve}(u_k)$, and let q be the right endpoint of the horizontal extension $\mathrm{he}(\bar{u}_{\bar{k}})$. The point p lies on a horizontal ceiling edge with rightmost (reflex) vertex $\bar{v}_p$. Let $\bar{u}_p$ be the convex vertex directly above $\bar{v}_p$ on the ceiling, if it exists. Similarly, q lies on a vertical floor edge with topmost (reflex) vertex v_q. Let u_q be the convex vertex on the floor directly to the right of v_q on the floor, if it exists. (In the last iteration of the algorithm, either $\bar{v}_p$, v_q, or both will be the top rightmost convex vertex.)

If $y(v_q) > y(\bar{v}_p)$, then let r_p be the right endpoint of the horizontal extension $\mathrm{he}(\bar{u}_p)$ and let r_q be the top endpoint of the vertical extension $\mathrm{ve}(q)$; see Figure 5. Let g' be the intersection point between $\mathrm{he}(\bar{u}_p)$ and $\mathrm{ve}(q)$ (which could lie on the vertical floor edge containing q and v_q; see Figure 5(b)) and let g'' be the intersection point between $\mathrm{he}(r_q)$ and $\mathrm{ve}(r_p)$. (If g' lies on a vertical floor edge, then g'' will lie on a horizontal ceiling edge at r_q.)

Let s_q be the intersection of $\mathrm{he}(r_q)$ with the floor, and let u_l be the last convex vertex on the floor before s_q. (Note that u_l could potentially coincide with u_q.) Let s_p be the intersection of $\mathrm{ve}(r_p)$ with the ceiling and let $\bar{u}_{\bar{\jmath}}$ be the first convex vertex on the ceiling after s_p; see Figure 5. If $x(u_l) < x(\bar{u}_{\bar{\jmath}})$, we let $G_i^{(2)} \leftarrow G_{i-1}^{(2)} \cup \{g, g', g'', s_p\}$ and let u_j be the topmost convex vertex on the floor seen by s_p; see Figure 5(c). Otherwise, $x(u_l) \geq x(\bar{u}_{\bar{\jmath}})$, we let $G_i^{(2)} \leftarrow G_{i-1}^{(2)} \cup \{g, g', g''\}$ and $j \leftarrow l$. By construction, $x(u_j) \geq x(\bar{u}_{\bar{\jmath}})$ always holds.

Let v_j be the reflex vertex before u_j on the floor. If $x(v_j) > x(\bar{u}_{\bar{j}})$, we cut $\mathbf{P}_i$ along $\text{ve}(\bar{u}_{\bar{j}})$ and $\text{he}(u_j)$ until the two extensions intersect, as is shown at the top of Figures 5(a) and (c). If $x(v_j) \leq x(\bar{u}_{\bar{j}})$, we cut $\mathbf{P}_i$ along $\text{ve}(\bar{u}_{\bar{j}})$ until it hits the floor; see the top of Figure 5(b). In each case, $\mathbf{P}_i$ is partitioned into two pieces, $\mathbf{Q}_i$ containing the guards and $\mathbf{P}_{i+1}$ lying above and to the right of r_q and r_p, respectively; see Figures 5(a)–(c).

If $x(v_q) < x(\bar{v}_p)$, then the situation is completely symmetrical to the one above, with respect to a flip along the line $y = x$, and we construct the guards and partition $\mathbf{P}_i$ into $\mathbf{Q}_i$ and $\mathbf{P}_{i+1}$ in exactly the same way as before.

If $y(v_q) \leq y(\bar{v}_p)$ and $x(v_q) \geq x(\bar{v}_p)$, then we proceed slightly differently. We let r_p be the right endpoint of the horizontal extension $\text{he}(\bar{u}_p)$, as before, but let r_q be the top endpoint of the vertical extension $\text{ve}(u_q)$. We let g be the intersection point between $\text{he}(\bar{u}_p)$ and $\text{ve}(u_q)$ and then proceed as before to obtain the guards and the partition of $\mathbf{P}_i$ into $\mathbf{Q}_i$ and $\mathbf{P}_{i+1}$.

Finally, we increase i by one, set $k \leftarrow j$, $\bar{k} \leftarrow \bar{j}$, and repeat the steps described above.

Let M be the index of the last iteration of our algorithm. It may be that $\mathbf{P}_M$ is starshaped and thus guardable with a single guard, in which case our algorithm also guards $\mathbf{P}_M$ with one guard [9, 12]. Otherwise, $\mathbf{P}_M$ requires two guards and we will then guard it with at most four guards as described above. We claim that the set $G^{(2)} \leftarrow G^{(2)}_M$ is a guard set for $\mathbf{P}$.

Lemma 3 *The guards in $G^{(2)}$ see all of $\mathbf{P}$.*

Proof. It suffices to prove that for each $1 \leq i \leq M$, the subpolygon $\mathbf{Q}_i$ is seen by the relevant guards placed by the algorithm within $\mathbf{Q}_i$.

By construction, g sees the rectangles spanned by u_k and $\bar{u}_{\bar{k}}$ and by p and q, since these rectangles have g at one corner. By the fan visibility property, g also sees the regions spanned by g, $\bar{u}_{\bar{k}}$, p and by g, q, u_k. The guard g' sees the rectangle spanned by $\bar{u}_p$ and q, for the first case of the algorithm, the rectangle spanned by p and u_q, for the second case of the algorithm, or the rectangle spanned by $\bar{u}_p$ and u_q, for the third case of the algorithm. In each case, the rectangle intersects the rectangle spanned by p and q (seen by g) so everything between g and g' is seen by the guards.

The rectangle spanned by g' and g'' is seen by both these guards and for each of the three cases, the regions spanned by g', $\bar{u}_p$, s_p, g'' and by g'', s_q, q, g', in the first case, the regions spanned by g', p, s_p, g'' and by g'', s_q, u_q, g', in the second case, and the regions spanned by g', $\bar{u}_p$, s_p, g'' and by g'', s_q, u_q, g', in the third case, are seen by g' and g'', by the fan visibility property. Some of these regions intersect with $\mathbf{P}_{i+1}$.

The rectangle spanned by s_p and s_q is seen by g'' since the rectangle has g'' at one corner. This rectangle may also intersect $\mathbf{P}_{i+1}$.

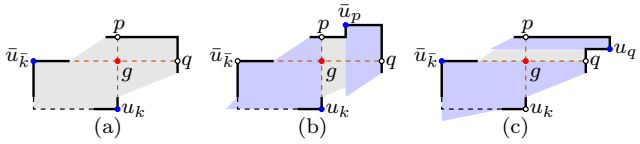


Figure 6: Illustrating the base case of the proof of Lemma 4.

Lastly, for the case that the algorithm adds s_p as a guard, this guard sees the region spanned by s_p, u_j, s_q, g''. Thus, the placed guards see all of $\mathbf{Q}_i$. □

A *witness set* W is a set of convex vertices (excluding $u_0 = \bar{u}_0$ and $u_m = \bar{u}_{\bar{m}}$) such that no pair of vertices in W have visibility polygons that intersect. Evidently, $|W| \leq |G^*|$ and we have the following lemma.

Lemma 4 *There is a witness set W in $\mathbf{P}$ such that each $\mathbf{Q}_i$, for $1 \leq i \leq M-1$, contains at least two vertices in W and $\mathbf{Q}_M$ contains at least one vertex in W.*

Proof. We make a proof by induction on decreasing index i and show that given the witness vertices in $\mathbf{P}_{i+1}$ how to choose the witness vertices in $\mathbf{Q}_i$, thus giving us the witness vertices in $\mathbf{P}_i$.

Consider the base case when $\mathbf{Q}_M = \mathbf{P}_M$. If $\mathbf{Q}_M$ is a starshaped staircase polygon, then g sees all of $\mathbf{Q}_M$ and we can choose one of u_k and $\bar{u}_{\bar{k}}$ as the witness for $\mathbf{Q}_M$; see Figure 6(a) for an example.

If $\mathbf{Q}_M$ is not starshaped, then the region above and to the right of g contains a convex vertex not seen from g. Let $\bar{u}_p$ be such a vertex, if it lies on the ceiling, then we choose $\bar{u}_p$ and u_k as the witnesses. It is clear that no point in $\mathbf{P}$ can see both $\bar{u}_p$ and u_k. Otherwise, u_q is a convex vertex on the floor and we choose u_q and $\bar{u}_{\bar{k}}$ as the two witnesses for $\mathbf{Q}_M$ in this case. Again, no point in $\mathbf{P}$ can see both u_q and $\bar{u}_{\bar{k}}$; see Figures 6(b) and (c) for two examples.

Assume inductively that we have a witness set obeying our criteria in $\mathbf{P}_{i+1}$. Assume further that the first convex vertices on the floor and the ceiling in $\mathbf{P}_{i+1}$ are u_j and $\bar{u}_{\bar{j}}$ respectively. Now consider the subpolygon $\mathbf{Q}_i$. We have to consider three cases, depending on which of the three cases of the algorithm was used to construct $\mathbf{Q}_i$. If the first case was used, then $y(v_q) > y(\bar{v}_p)$ and no point in $\mathbf{P}$ can see both $\bar{u}_p$ and either of u_j or $\bar{u}_{\bar{j}}$. Furthermore, any point that sees u_k cannot see any of $\bar{u}_p$, u_j, and $\bar{u}_{\bar{j}}$, hence we can use $\bar{u}_p$ and u_k as witnesses for $\mathbf{Q}_i$ in this case; see Figures 7(a) and (b).

If the second case was applied, then $x(v_q) < x(\bar{v}_p)$ and no point in $\mathbf{P}$ can see both u_q and either of u_j or $\bar{u}_{\bar{j}}$. Furthermore, any point that sees $\bar{u}_{\bar{k}}$ cannot see any of u_q, u_j, and $\bar{u}_{\bar{j}}$, hence we can use u_q and $\bar{u}_{\bar{k}}$ as witnesses for $\mathbf{Q}_i$ in this case; see Figures 7(c) and (d).

If the third case was applied, then $x(v_q) \geq x(\bar{v}_p)$ and $y(v_q) \leq y(\bar{v}_p)$, then we can in fact use either of the two cases above and either choose $\bar{u}_p$ and u_k, or u_q and $\bar{u}_{\bar{k}}$ as witnesses for $\mathbf{Q}_i$.

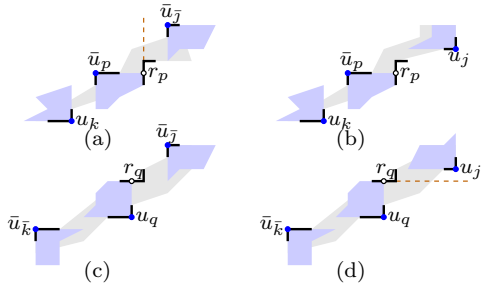


Figure 7: Illustrating the inductive case of the proof of Lemma 4.

Furthermore, in each of the three cases, u_j cannot be seen by any point with y-coordinate smaller than $y(u_j)$, and $\bar{u}_{\bar{j}}$ cannot be seen by any point with x-coordinate smaller than $x(\bar{u}_{\bar{j}})$. Thus, specifically no point below or to the left of g can see any witness in $\mathbf{P}_{i+1}$ concluding our proof. □

The set $G^{(2)}$ was constructed by greedily iterating the same construction M times. Lemma 4 shows that, if the witness set contains $2(M-1)+1$ vertices, then $G^{(2)}$ has at most $4(M-1)+1$ guards, otherwise the witness set contains $2M$ vertices and $G^{(2)}$ has at most $4M$ guards. In either case, $|G^{(2)}| \leq 2|W|$.

The discussion above and Lemma 3 prove our main result.

Theorem 5 *The guards in the set $G^{(2)}$ see all of* $\mathbf{P}$ *and*

$$|G^{(2)}| \leq 2|W| \leq 2|G^*|.$$

3 Conclusions

We have given two 2-approximation algorithms for guarding staircase polygons. Unfortunately, this does not answer the complexity status for guarding these polygons, since constant factor approximations were already known for monotone polygons [11].

It would be of immense interest to settle the complexity question for guarding staircase polygons, either by providing a polynomial time algorithm that optimally solves it or a hardness proof. Short of this, improving the approximation factor or even giving a PTAS would help to understand the problem. Staircase polygons, although having very simple structure, exhibit a surprising amount of intricacy and we feel that fully answering the complexity status for these polygons will be challenging.

References

[1] Alok Aggarwal. *The art gallery theorem: its variations, applications and algorithmic aspects*. PhD thesis, The Johns Hopkins University, 1984.

[2] Mark de Berg, Stephane Durocher, and Saeed Mehrabi. Guarding monotone art galleries with sliding cameras in linear time. *J. Discrete Algorithms*, 44:39–47, 2017. URL: https://doi.org/10.1016/j.jda.2017.04.005, doi:10.1016/j.jda.2017.04.005.

[3] Pritam Bhattacharya, Subir Kumar Ghosh, and Sudebkumar Prasant Pal. Constant approximation algorithms for guarding simple polygons using vertex guards. *CoRR*, abs/1712.05492, 2017. URL: http://arxiv.org/abs/1712.05492, arXiv:1712.05492.

[4] Édouard Bonnet and Tillmann Miltzow. An approximation algorithm for the art gallery problem. In *33rd International Symposium on Computational Geometry, SoCG 2017, July 4-7, 2017, Brisbane, Australia*, pages 20:1–20:15, 2017. URL: https://doi.org/10.4230/LIPIcs.SoCG.2017.20, doi:10.4230/LIPIcs.SoCG.2017.20.

[5] Ajay Deshpande, Taejung Kim, Erik D. Demaine, and Sanjay E. Sarma. A pseudopolynomial time O (log n)-approximation algorithm for art gallery problems. In *Algorithms and Data Structures, 10th International Workshop, WADS 2007, Halifax, Canada, August 15-17, 2007, Proceedings*, pages 163–174. Springer, 2007. URL: https://doi.org/10.1007/978-3-540-73951-7_15, doi:10.1007/978-3-540-73951-7_15.

[6] Stephane Durocher, Omrit Filtser, Robert Fraser, Ali D. Mehrabi, and Saeed Mehrabi. Guarding orthogonal art galleries with sliding cameras. *Comput. Geom.*, 65:12–26, 2017. URL: https://doi.org/10.1016/j.comgeo.2017.04.001, doi:10.1016/j.comgeo.2017.04.001.

[7] Stephan Eidenbenz. Inapproximability results for guarding polygons without holes. In *ISAAC '98*, pages 427–436, 1998.

[8] Matt Gibson, Erik Krohn, and Matthew Rayford. Guarding monotone polygons with half-guards. In *Proceedings of the 29th Canadian Conference on Computational Geometry, July 26-28, 2017, Carleton University, Ottawa, Ontario, Canada*, pages 168–173, 2017.

[9] Barry Joe and R. Bruce Simpson. Corrections to Lee's visibility polygon algorithm. *BIT*, 27(4):458–473, 1987.

[10] Erik Krohn and Bengt J. Nilsson. The complexity of guarding monotone polygons. In *Proc. 24th Canadian Conference on Computational Geometry, CCCG'2012*, pages 167–172, 2012.

[11] Erik Krohn and Bengt J. Nilsson. Approximate guarding of monotone and rectilinear polygons. *Algorithmica*, 66(3):564–594, 2013. URL: https://doi.org/10.1007/s00453-012-9653-3, doi:10.1007/s00453-012-9653-3.

[12] Der-Tsai Lee. Visibility of a simple polygon. *Computer Vision, Graphics, and Image Processing*, 22(2):207–221, 1983. URL: https://doi.org/10.1016/0734-189X(83)90065-8, doi:10.1016/0734-189X(83)90065-8.

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

Attraction-convexity and Normal Visibility

Prosenjit Bose* Thomas C. Shermer†

Abstract

Beacon attraction, or simply *attraction*, is a movement system whereby a point moves in a free space so as to always locally minimize its Euclidean distance to an activated beacon (also a point). This results in the point moving directly towards the beacon when it can, and otherwise sliding along the edge of an obstacle or being stuck (unable to move). When the point can reach the activated beacon by this method, we say that the beacon *attracts* the point. In this paper, we study *attraction-convex* polygons, which are those where every point in the polygon attracts every other point. We find that these polygons are a subclass of *weakly externally visible* polygons, which are those where every point on the boundary is visible from some point arbitrarily distant (or at infinity on the projective plane). We propose a new class of polygons called *normally visible*, and show that this is exactly the class of attraction-convex polygons. This alternative characterization of attraction-convex polygons leads to a simple linear-time attraction-convex polygon recognition algorithm. We also give a Helly-type characterization of inverse-attraction star-shaped polygons.

1 Background

Beacon attraction has appeared in the literature as a model of greedy geographical routing in dense sensor networks [2, 3, 4, 5, 8, 9, 10, 11, 12, 14]. In this application, each node of the network has a location, and each communication packet knows the location of its destination. Nodes having a packet to deliver will forward the packet to their neighbor that is the closest (using Euclidean distance) to the packet's destination [6, 7].

Another application, closer to the abstract form, is for a robot that heads towards a "homing signal." The robot will follow walls it encounters if, in doing so, it gets closer to the source of the signal.

Here we study beacon attraction in the abstract setting, considering only simple singly-connected polygons with interior as our free space. In this setting, the destination point or signal source is called a beacon, and the message or robot is considered to be a point that greedily moves towards the beacon. This results in the point moving directly towards the beacon when it can, and otherwise sliding along an edge or being stuck (unable to move). The point, under this motion, may or may not reach the beacon—if it does reach the beacon, we say that the beacon *attracts* the robot's starting point (see Figure 1).

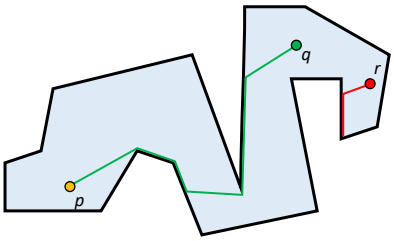


Figure 1: A beacon at p attracts the point at q but not the point at r. The paths of the attraction are shown. Note that q does not attract p.

Given a polygon P, the *attraction region* $A(p)$ of point p is the set of all points in P that p attracts, and the *inverse attraction region* $A^{-1}(p)$ is the set of all points in P that p is attracted to. See Figure 2 for an illustration of these definitions.

The attraction relation between points has the flavor of a visibility-type relation, with the interesting property of asymmetry: if point p attracts point q, then it does not follow that point q attracts p (as is the case in Figure 1). In a series of publications, Biro, Gao, Iwerks, Kostitsyna, and Mitchell have studied various visibility-type questions for beacon attraction, such as computing attraction (and inverse-attraction) regions for points, computing attraction kernels, guarding, and routing [5, 4, 3].

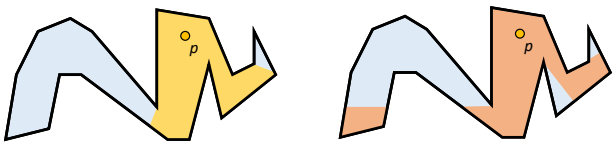


Figure 2: (a) The attraction region $A(p)$. (b) The inverse attraction region $A^{-1}(p)$. Note that, in this instance, $A^{-1}(p)$ is not connected.

*Research supported in part by NSERC. School of Computer Science, Carleton University, jit@scs.carleton.ca

†School of Computing Science, Simon Fraser University , shermer@sfu.ca

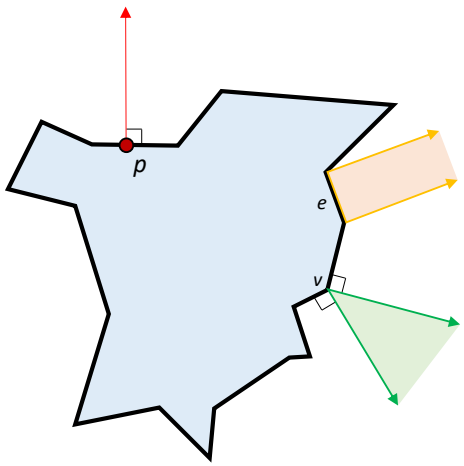


Figure 3: A normally visible polygon, with the normal from p, the beam from e, and the cone of v.

From a geometric standpoint, visibility, convexity, and star-shapedness are fundamental concepts. Two points x and y in a shape (compact set of points) are called *visible* if the line segment $\overline{xy}$ is contained in the shape. A shape S is called *convex* if $\forall x, y \in S$ x and y are visible. A shape S is called *star-shaped* if $\exists x \in S$, $\forall y \in S$ x and y are visible. The *kernel* of S is the set of all points from which it is star-shaped: $kerS = \{x \in S \mid \forall y \in S$ x and y are visible$\}$.

These notions are easily extended to other "visibility-type" relations R: a shape S is called *R-convex* if $\forall x, y \in S$ xRy, and it is called *R-star* if $\exists x \in S$ $\forall y \in S$ xRy. The *R-kernel* of S is $kerS = \{x \in S \mid \forall y \in S$ $xRy\}$.

Although Biro *et al.*[5] have studied attraction kernels and their computation, they did not produce characterizations of polygons that are attraction-star-shaped or attraction-convex. In this paper, we address these issues.

To this end, we introduce a new notion of exterior visibility of a polygon. We call a polygon *normally visible* if the normal (exterior perpendicular) to each point on the boundary hits no point of the interior of the polygon. This is a specialization of weak exterior visibility: weak exterior visibility only requires a ray from each boundary point that does not intersect the interior, whereas normal visibility constrains those rays to be perpendicular to the polygon boundary. Figure 3 shows a normally-visible polygon and the nonintersecting perpendicular ray for point p.

Another way to conceptualize normally visible polygons is to imagine an ant crawling counterclockwise around the boundary of the polygon. This ant has an attached laser that points directly to its right. If the ant can crawl all the way around the polygon with the laser never hitting the polygon interior, then the polygon is normally visible.

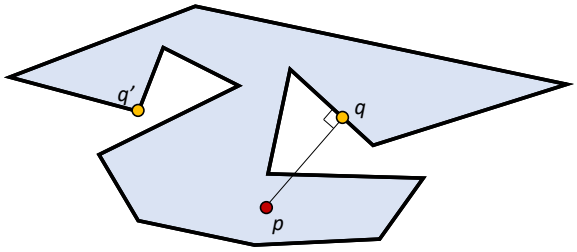


Figure 4: The dead points q and q' of a beacon at point p.

To simplify thinking about normal visibility, it is useful to consider the *beam* $B(e)$ of each edge e, and the *cone* $C(v)$ of each vertex v. The beam of e is the union of all normals from points on e, and the cone of v are the normals that are swept out as the crawling ant continuously changes heading from one edge to the other at v. A polygon is normally visible if no beam or cone hits its interior.

We may dispense with cones by a chain of observations. First, for the polygon interior to intersect a cone, the polygon must *cross* (not simply intersect) one of the cone's two bounding rays, as the polygon is a Jordan curve. Next, the bounding rays of the cone are also bounding rays of the beams of the two adjacent edges. Thus the polygon crosses a ray in a beam; that ray will hit an interior point of the polygon. So we conclude that a polygon is normally visible if no beam hits its interior.

2 Convex Characterization

Given a polygon P, a point q is called a *dead point* of point p if a beacon at p does not move q. A dead point of p is either a convex vertex or a point q in the interior of an edge such that pq is perpendicular to the edge (see Figure 4). In either case, pq is exterior to the polygon in the neighborhood of q.

When a beacon is activated, all points end up either at the beacon or at a dead point.

Figure 3 depicts an attraction-convex polygon.

Lemma 1 *If a polygon is attraction-convex, then it is normally visible.*

Proof. Suppose this is not the case. Assume P is an attraction-convex polygon, with q being a point on edge e of P that is not normally visible. The outside ray r starting at q perpendicular to e must encounter some other point q' on the boundary of P.

Now we note that, in P, q' does not attract q. (q is a dead point of q'.) This contradicts the assumption that

P is attraction-convex. □

Lemma 2 *If a polygon is normally visible, then it is attraction-convex.*

Proof. We prove the contrapositive: If a polygon P is not attraction-convex, then it is not normally visible.

Since P is not attraction-convex, there are points p and q in P such that p does not attract q. Acting under attraction from p, the point q will arrive at a dead point q' of q, and become stuck there.

If the dead point is on the interior of an edge e of P, then the outside ray r from q' perpendicular to e hits p. The beam $B(e)$ extends along e to both sides of q', which implies that for ε small enough, $B(e)$ contains an ε- neighborhood of p. Since any ε-neighborhood of p contains interior points of P, the beam $B(e)$ contains interior points of P, implying that P is not normally visible.

If the dead point q' is a convex vertex of P, then let e_1 and e_2 be the edges of P incident on q'. Let r_1 be the outside ray from q' perpendicular to e_1, and r_2 be the outside ray from q' perpendicular to e_2. In order for q' to be a dead point, p must lie in the closed convex cone from q' bounded by r_1 and r_2.

However, p must be connected to q' by a path interior to P except possibly at p and q'. This path cannot reach q' directly from the cone, as the polygon is simple. Since the path cannot go to infinity, as P does not, it must cross either r_1 or r_2. So a beam of P contains a point of the interior p, and P is not normally visible.

As these are the only two possibilities for the dead point q', the lemma is proved. □

We thus have:

Theorem 2.1 *A polygon is attraction-convex iff it is normally visible.*

Corollary 3 *In an attraction-convex polygon, every interior angle is at most* $3\pi/2$.

3 Algorithm for recognizing attraction-convex polygons

Before outlining the algorithm to recognize an attraction-convex polygon, we first introduce some terminology. Let $P = v_0, v_1, \ldots, v_{n-1}$ be the vertices of a simple polygon ordered in counterclockwise order. All indicies are manipulated modulo n. Let $CH(P)$ be the convex hull of polygon P. Let v_pv_q be an edge of the convex hull. If q is not $p+1$, i.e. if v_q is not the next counterclockwise vertex on the boundary of P, we call edge v_pv_q a *pocket lid* and the polygonal chain from $v_p, \ldots, v_q$ on the boundary of P as the *pocket chain*. Together, a pocket lid and the corresponding pocket chain form the boundary of a simple polygon called a *pocket polygon*.

Recall that a polygon is normally visible if for every edge e, $B(e)$ does not intersect the polygon boundary. This means that for every edge e on a pocket chain, $B(e)$ intersects only the pocket lid. In fact, if pocket chain edge $e = ab$, then $B(e) \cap CH(P)$ results in a segment $a'b'$ that is a subsegment of the pocket lid (where a' is the intersection of the normal ray originating from a with the pocket lid and b' is the intersection of the normal ray originating from b with the pocket lid.)

A *terrain polygon* is a monotone polygon Q that has a distinguished edge e such that (1) Q is monotone with respect to the direction of e, and (2) e is one of the two monotone chains of Q in this direction. We call a pocket chain $\Pi = v_p, \ldots, v_q$ *a terrain with respect to its lid* v_pv_q if the corresponding pocket polygon is a terrain polygon with distinguished edge v_pv_q. Equivalently, Π is a terrain with respect to v_pv_q provided that for every vertex $v_k \in \Pi$, the line perpendicular to v_pv_q going through v_k, denoted by $\ell(v_k)$, intersects Π only at v_k and intersects the lid v_pv_q at a point v'_k. Note that v'_k is the orthongonal projection of v_k onto the pocket lid. Moreover, the sequence $v'_p, v'_{p+1}, \ldots, v'_{q-1}, v'_q$ is a sorted sequence.

Lemma 4 *If a simple polygon P is normally visible, then every pocket chain of P is a terrain with respect to its pocket lid.*

Proof. For sake of a contradiction, suppose that P is normally visible and that there is a pocket chain $\Pi = v_p, \ldots, v_q$ that is not a terrain with respect to its pocket lid v_pv_q. Let Q be the corresponding pocket polygon. If $e \in \Pi$, then the interior of the beam $B(e)$, in a neighborhood of e, is inside Q. Since the beam is infinite, it must leave Q somewhere. Since P is normally visible, the beam cannot leave Q via any edge in Π. Therefore the beam must leave Q via v_pv_q. This implies that the angle between the normal of the convex hull edge v_pv_q and the normal of (any) edge e of Π is strictly less than $\pi/2$.

Without loss of generality, assume that v_pv_q is on the X-axis with $v_p = v'_p$ at the origin, and the polygon above. If Π were a terrain with respect to v_pv_q, then $v'_p, \ldots, v'_q$ would be an increasing sequence. (Here we use v' as a stand-in for the x-coordinate of v.) Since Π is not a terrain with respect to v_pv_q, there must exist a vertex v_k, with $k \in p+1, \ldots, q-1\}$, such that v'_{k-1} and v'_{k+1} are both greater than v'_k or both are less than v'_k in the sequence.

Without loss of generality, assume that both v'_{k-1} and v'_{k+1} are greater than v'_k. This implies that either the angle between the normal of $v_{k-1}v_k$ and the normal of v_pv_q is at least $\pi/2$ or the angle between the normal of v_kv_{k+1} and the normal of v_pv_q is at least $\pi/2$, which

```
    // Maintain a stack S of vertices. Initially, S is empty.
    // Let k = |S| and let c_k refer to the top element of S.
    // c_{k-1} is the element below c_k in the stack
    ccwScan()
1       push(v_0)
2       for i ← 1 to n
3           while k ≠ 1 and c_{k-1}c_k v_i is a right turn do
4               pop()
5
6           // Now k = 1 or c_{k-1}c_k v_i is not a right turn
7           push(v_i)
8           if c_{k-1}c_k v_{i+1} is a right turn and
                ∠c_{k-1}c_k v_{i+1} < π/2 then
9               return false
10          if c_{k-1}c_k v_{i+1} is a left turn and
                v_{i-1}v_i v_{i+1} is a right turn then
11              return false
12      return true
```

Figure 5: The counterclockwise scan of the algorithm

is a contradiction since these two angles must both be strictly less than $\pi/2$. □

Our algorithm for recognizing whether a polygon P is attraction-convex consists of three major steps. The first step is to compute the convex hull of P. The second step is to verify that every pocket chain is a terrain with respect to its lid. The third step consists of running a specialized scan twice: once counterclockwise and once clockwise. If either scan returns **false** then the polygon is not attraction-convex; if both return **true** then the polygon is attraction-convex.

The algorithm for the counterclockwise scan is shown in Figure 5; the clockwise scan is symmetric. Here, v_0 is the leftmost top vertex of the polygon and vertices are numbered sequentially in counterclockwise order.

At the end of each **for** loop iteration, the stack can be viewed as a left-turning spiral chain starting at v_0 and ending at v_i; this chain does not cross the polygonal chain seen so far $(v_0, v_1, \ldots v_i)$. The noncrossing property implies that each edge $c_{j-1}c_j$ of the spiral chain is an edge of a (possibly degenerate) closed polygon Q_j whose boundary consists of that edge and the portion of the boundary of the input polygon from c_j to c_{j+1} (see Figure 6).

Theorem 3.1 *The algorithm described above correctly determines if an input polygon is attraction-convex in O(n) time.*

Proof. The first step can be computed in linear time with Melkman's linear time convex hull algorithm [13]. The second step can also be computed in linear time by simply verifying that for every pocket chain $v_p, \ldots, v_q$,

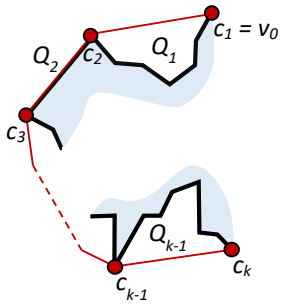


Figure 6: The spiral chain $c_1, c_2, \ldots c_k$ maintained in the stack and the polygons $Q_1, Q_2, \ldots Q_{k-1}$. Note that Q_2 here is degenerate.

the sequence $v'_p, \ldots, v'_q$ appears in the same order. If some pocket chain is not a terrain with respect to its lid, then we stop, and Lemma 4 guarantees us that **false** is the correct answer.

So suppose the algorithm proceeds to the third step and then reports **false**. Without loss of generality, assume that it is the counterclockwise scan that reports **false**. If this was reported from line 9 of the algorithm, then some angle $c_{k-1}c_k v_{i+1}$ is right-turning and acute. Since we have just pushed v_i in line 7, $c_k = v_i$. Similar to the proof of Theorem 3, we take a point p sufficiently close to $c_k = v_i$ on $v_i v_{i+1}$, and the normal ray r from p will intersect $c_{k-1}c_k$. If Q_{k-1} is degenerate, then this is an intersection q with the polygon boundary. If it is not degenerate, r enters Q_{k-1} and must exit Q_{k-1} at some point q not on $c_{k-1}c_k$. The point q is an intersection with the polygon boundary. In either case, the ray r enters the interior of the polygon directly after it hits the point q, so this polygon is not normally visible. (Or attraction-convex, by Theorem 2.1.)

If instead the **false** was reported in line 11, we will have a similar situation. If $c_{k-1}c_k v_{i+1}$ is a left turn and $v_{i-1}v_i v_{i+1}$ is a right turn, then the edge $v_i v_{i+1}$ lies in Q_{k-1} with polygon interior on the side closest to $c_{k-1}c_k$ (see Figure 7). If we let p be a point on the relative interior of $v_i v_{i+1}$, and r be the normal ray from p, then the ray r will leave Q_{k-1} at some point other than those on $c_{k-1}c_k$. Immediately after exiting Q_{k-1}, the ray r encounters a point in the interior of P. Thus the polygon is not normally visible.

In either case, then, the algorithm is correct when it reports **false**.

Now suppose the algorithm reports **true** from both scans. Consider an arbitrary edge $v_j v_{j+1}$ of the polygon. If $v_j v_{j+1}$ is a convex hull edge, then its beam does not intersect the polygon interior. Otherwise, $v_j v_{j+1}$ is an edge in a pocket chain with some pocket lid $v_p v_q$. During the counterclockwise scan, after v_j was pushed

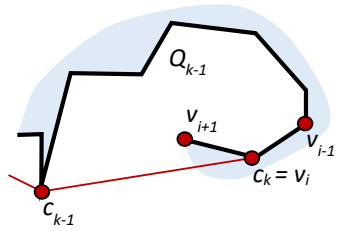


Figure 7: When $c_{k-1}c_kv_{i+1}$ is a left turn and $v_{i-1}v_iv_{i+1}$ is a right turn, v_iv_{i+1} lies in Q_{k-1}.

on the stack, the stack was a left-spiral chain from v_0 to v_j, and the angle $c_{k-1}c_kv_{j+1}$ was at least $\pi/2$. The convex hull vertex v_p is somewhere on the left-spiral chain. Because the pocket chain is a terrain with respect to v_pv_q, the vertices of the sub chain $v_p, v_{p+1}, \ldots v_{j-1}, v_j$ all lie to the right of v_jv_j' where v_j' is the projection of v_j onto v_pv_q (see Figure 8). In particular, this means that this chain, and thus the left-spiral chain from v_0, cannot wind around v_jv_{j+1}. A similar argument using the clockwise scan establishes that the right-spiral chain from v_0 reaches v_{j+1} without winding around v_jv_{j+1}.

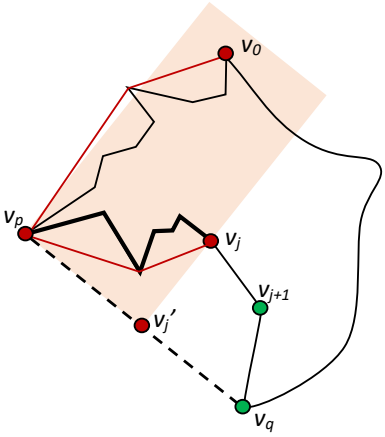


Figure 8: The subchain $v_p, v_{p+1}, \ldots, v_j$ must lie in the shaded region.

Refer now to Figure 9. Since the angle $c_{k-1}c_kv_{j+1}$ was at least $\pi/2$, the left-spiral chain from the counterclockwise scan may graze the boundary but not the interior of the beam $B(v_jv_{j+1})$. Furthermore, as the beam gets farther from v_jv_{j+1}, the left-spiral chain diverges from the beam, and as it doesn't wind, it cannot later intersect the beam. Similarly, the right-spiral chain from the clockwise scan may graze but not intersect the interior of the beam. The polygon P is contained entirely by the polygon consisting of the left-spiral chain from v_0 to v_j, the edge v_jv_{j+1}, and the right-spiral chain from v_0 to v_{j+1} (backwards). Thus the polygon P may intersect the boundary but not the interior of $B(v_jv_{j+1})$. Finally, since both P and the beam are closed, this means that the beam (including its boundary) does not intersect the interior of P.

Since we started with an arbitrary edge v_jv_{j+1}, and (given that both scans return **true**) established that the beam of that edge does not intersect the polygon interior, this means that all beams are free from such intersections and the polygon is normally visible.

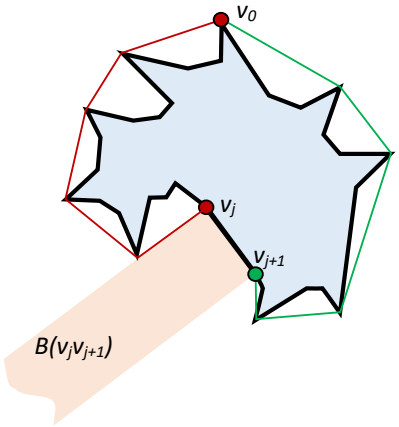


Figure 9: The left-spiralling chain from v_0 to v_j, the edge v_jv_{j+1}, and the right-spiralling chain from v_0 to v_{j+1} enclose P and can touch only the boundary of $B(v_jv_{j+1})$.

Thus the algorithm is correct when it returns **true**, and correct overall. During each of the scans of step three, each vertex of the polygon is pushed onto the stack once, and popped at most once, giving linear stack manipulation. The condition of line 3 is checked once for each pop (linear time) and once for each time through the **for** loop (also linear time). The other conditions are checked once per iteration of the **for** loop, giving linear time for these, and linear time overall. □

4 Geodesic Convexity and Inverse-attraction-star Shapes

In this section, we highlight a relationship between geodesic convexity and inverse-attraction. This relationship allows us to give a Helly-type characterization of Inverse-attraction star-shaped polygons.

4.1 Geodesic convexity

A set $A \subseteq P$ is called geodesically convex relative to P if, for all points p, q in A, the shortest path (geodesic) $\Pi(p, q)$ between p and q in P is contained in A. We

will drop the phrase "relative to P" when the set P is understood.

Geodesic convexity is transitive.

Lemma 5 *If R is geodesically convex relative to Q, and Q is geodesically convex relative to P, then R is geodesically convex relative to P.*

Proof. Consider two points p, q in R. These points are also in Q and P. Since Q is geodesically convex relative to P, the shortest path $\Pi(p, q)$ from p to q in P is also the shortest path from p to q in Q. Since R is geodesically convex relative to Q, $\Pi(p, q)$ is also the shortest path from p to q in R. □

Lemma 6 *Let P_1 and P_2 be the two (closed) subpolygons of P formed when P is cut by a chord C. Then P_1 and P_2 are geodesically convex with respect to P.*

Proof. We prove this for P_1. Let x, y be two points in P_1. We claim the shortest path $\Pi(p, q)$ from p to q in P stays in P_1. If not, then there is a section of $\Pi(p, q)$ in P_2 that enters P_2 from some point r_1 of C, and exits P_2 to some point r_2 of C. Replacing this section of $\Pi(p, q)$ with the straight line segment from r_1 to r_2 results in a shorter path, which is a contradiction. □

Lemma 7 *Let P' be a subpolygon of P produced by repeating the operation of cutting off a section of P with a chord. Then P' is geodesically convex with respect to P.*

Proof. This follows directly from Lemmas 6 and 5. □

An *et al.* [1] established that geodesically-convex sets in $\mathbf{R}^2$ have Helly number 3:

Theorem 4.1 (Geodesic-convexity Helly Theorem) *Let $\mathcal{C}$ be a collection of sets all geodesically convex with respect to some compact base set S in $\mathbf{R}^2$. If every triple of sets in $\mathcal{C}$ has an intersection point, then all sets in $\mathcal{C}$ do.*

4.2 Inverse-attraction Star-shaped Polygons

Lemma 8 *Attraction polygons are geodesically convex.*

Proof. As shown by Biro [3, Thm. 3.2.11], the attraction polygon of a point p in polygon P can be constructed by cutting several regions off of P, each region bounded by a chord (a "split edge"). The result follows from Lemma 7. □

Theorem 4.2 *A polygon P is inverse-attraction star-shaped iff for every triple of points p, q, and r in P, $A(p) \cap A(q) \cap A(r) \neq \emptyset$.*

Proof. P is inverse-attraction star-shaped if there exists a point k, called a kernel point, that inverse-attracts all points in P. This is the same as saying that k is in $A(p)$ for every p in P.

If P is inverse-attraction star-shaped with kernel point k, then for every triple of points p, q, r in P, k is in $A(p) \cap A(q) \cap A(r)$.

On the other hand, if for every triple of points p, q, r in P, $A(p) \cap A(q) \cap A(r) \neq \emptyset$, then the attraction polygons triple-wise intersect. Since attraction polygons are geodesically convex, we may then invoke the Geodesic-convexity Helly Theorem to obtain a point k that is in the intersection of all of the attraction polygons. □

We can simplify the conditions on the previous theorem to consider only triples of vertices rather than triples of points, but first we require an additional type of convexity, and a result.

A set $Q \subseteq P$ is said to be convex with respect to P if, for every pair of points p, q in Q, the line segment pq is in P iff pq is in Q [3]. Biro showed that inverse-attraction polygons are convex with respect to P.

Theorem 4.3 *A polygon P is inverse-attraction star-shaped iff for every triple of vertices u, v, and w in P, $A(u) \cap A(v) \cap A(w) \neq \emptyset$.*

Proof. If P is inverse attraction star-shaped with kernel point k, then for every triple of vertices u, v, w in P, k is in $A(u) \cap A(v) \cap A(w)$.

Suppose now that for every triple of vertices u, v, w in P, $A(u) \cap A(v) \cap A(w) \neq \emptyset$. By the geodesic-convexity Helly theorem, there is a point k in the intersection of all attraction polygons of vertices. We show that k is also in the attraction polygon of every point in P.

Consider an arbitrary point p of P. The point p resides in some triangle uvw of a triangulation of P. $A^{-1}(k)$ includes u, v, and w. Since u, v, and w are pairwise visible, the entire triangle u, v, w is in $A^{-1}(k)$ by Biro's convexity relative to P. □

It is an open problem to determine if there are equivalents of Theorems 4.2 and 4.3 for attraction as opposed to inverse attraction.

References

[1] P. T. An, D. T. Giang, and N. N. N. Hai. Some computational aspects of geodesic convex sets in a simple polygon. *Numerical Functional Analysis and Optimization*, 31:3:221–231, 2010.

[2] S. W. Bae, C.-S. Shin, and A. Vigneron. Improved bounds for beacon-based coverage and routing in simple rectilinear polygons. *arXiv preprint arXiv:1505.05106*, 2015.

[3] M. Biro. *Beacon-based routing and guarding.* PhD thesis, State University of New York at Stony Brook, 2013.

[4] M. Biro, J. Gao, J. Iwerks, I. Kostitsyna, and J. S. B. Mitchell. Combinatorics of beacon routing and coverage. In *CCCG*. Carleton University, Ottawa, Canada, 2013.

[5] M. Biro, J. Iwerks, I. Kostitsyna, and J. S. Mitchell. Beacon-based algorithms for geometric routing. In *WADS*, pages 158–169. 2013.

[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] B. Karp and H.-T. Kung. Gpsr: Greedy perimeter stateless routing for wireless networks. In *Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 243–254. ACM, 2000.

[8] I. Kostitsyna, B. Kouhestani, S. Langerman, and D. Rappaport. An optimal algorithm to compute the inverse beacon attraction region. In *SoCG*, 2018.

[9] B. Kouhestani, D. Rappaport, and K. Salomaa. On the inverse beacon attraction region of a point. In *CCCG*, 2015.

[10] B. Kouhestani, D. Rappaport, and K. Salomaa. The length of the beacon attraction trajectory. In *CCCG*, pages 69–74, 2016.

[11] B. Kouhestani, D. Rappaport, and K. Salomaa. Routing in a polygonal terrain with the shortest beacon watchtower. *Comput. Geom.*, 68:34–47, 2018.

[12] B. Kouhestani, D. Rappaport, and K. Salomaa. Routing in a polygonal terrain with the shortest beacon watchtower. *Comput. Geom.*, 68:34–47, 2018.

[13] A. A. Melkman. On-line construction of the convex hull of a simple polyline. *ipl*, 25:11–12, 1987.

[14] T. C. Shermer. A combinatorial bound for beacon-based routing in orthogonal polygons. *arXiv preprint arXiv:1507.03509*, 2015.

A MapReduce Algorithm for Metric Anonymity Problems

Sepideh Aghamolaei* Mohammad Ghodsi* SeyyedHamid Miri*

Abstract

We focus on two metric clusterings namely r-gather and (r, ϵ)-gather. The objective of r-gather is to minimize the radius of clustering, such that each cluster has at least r points. (r, ϵ)-gather is a version of r-gather with the extra condition that at most $n\epsilon$ points can be left unclustered (outliers).

MapReduce is a model used for processing big data. In each round, it distributes data to multiple servers, then simultaneously processes each server's data.

We prove a lower bound 2 on the approximation factor of metric r-gather in the MapReduce model, even if an optimal algorithm for r-gather exists. Then, we give a $(4+\delta)$-approximation algorithm for r-gather in MapReduce which runs in $O(\frac{1}{\delta})$ rounds. Also, for (r, ϵ)-gather, we give a $(7 + \delta)$-approximation algorithm which runs in $O(\frac{1}{\delta})$ MapReduce rounds, for any constant $\delta > 0$.

1 Introduction

Privacy is a fundamental concern in publishing data [7] or providing input to untrusted programs [20]. One of the privacy-preservation methods is **k-anonymity** [21, 10], where given a table of records, the goal is to change some attributes of some records such that each record appears at least k times and the maximum distance between the modified and the original records is minimized. Approximation algorithms with factors $O(1)$ for $k = 2, 3$ [1] and $O(\log k)$ for all k [19] exist.

When records are points in a metric space, the problem is called **r-gather** [2]. Aggarwal et al [2] introduced the problem and gave a 2-approximation algorithm for this problem and a matching lower bound, assuming $P \neq NP$. Ene et al [13] proved the lower bound 1.8 for Euclidean r-gather. If we allow $n\epsilon$ points to be left unclustered (outliers), the problem is called **(r, ϵ)-gather** and has a 4-approximation algorithm [2].

r-gather is formally defined in Definition 1.1. In the rest of the paper, we denote $\{1, 2, \cdots, n\}$ with $[n]$.

Definition 1.1 *r-Gather problem clusters n points $p_1, \cdots, p_n$ in a metric space into a set of clusters centered at $C \subset \{p_1, \cdots, p_n\}$, such that each cluster has at least r points. The objective is to minimize the maximum radius among the clusters (R).*

- $\forall i \in [n], \exists c \in C, d(p_i, c) \leq R$,
- $\forall c \in C, |\{p_i : i \in [n], d(p_i, c) \leq R\}| \geq r$.

Some related problems are k-center with lower bound (LB k-center) with 3-approximation [4] and Euclidean [13] algorithms, or both lower and upper bounds [12] (6-approximation) on the number of points in each cluster. In these problems, the number of clusters (k) and the minimum number of points in each cluster (r) are given and the goal is to minimize the radius (R). So, r-gather with $r = 1, k = n$ is k-center.

General frameworks for extending α-approximation sequential algorithms to MapReduce via coresets with approximation factors 32α [5] and $\alpha\beta$ using a β-approximation at the last step [15] with k-center as subroutine exist. Aghamolaei and Ghodsi [3] proved a covering algorithm can be used as the coreset for k-center. Any covering with radius less than k-center has a lower cost than r-gather, so it is a coreset for r-gather. Computing a k-center on a MapReduce coreset for r-gather extends our results to $\theta(\frac{n}{k})$-balanced k-center (by setting $r = \frac{n}{k}$) and LB k-center. Using (r, ϵ)-gather instead, solves k-center with outliers.

We give $O(1)$-approximation algorithms for r-gather and (r, ϵ)-gather in MapReduce (see Table 1).

Table 1: Summary of results ($k \leq \frac{n}{p}$, for p servers).

Conditions	Rounds	App.	Refs
r-gather:			
lower bound	$\geq k$	2	Thm. 1
-	2	16	Alg. 5
-	$O(\frac{1}{\delta})$	$4 + \delta$	Alg. 4
(r, ϵ)-gather	$O(\frac{1}{\delta})$	$7 + \delta$	Alg. 7
LB k-center:			
$\theta(\frac{n}{k})$-balanced	$O(1)$	96	[5, 4], $p = \frac{n}{k}$
-	$O(\frac{1}{\delta})$	$16 + \delta$	Alg. 4, [17]
-	$O(\frac{1}{\delta})$	$8 + \delta$	Alg. 4, [15, 14]
$\theta(\frac{n}{k})$-balanced	$O(\frac{1}{\delta})$	$8 + \delta$	", $r = \frac{n}{k}$
lower bound	$\geq k$	2	Thm. 1
" (outliers):			
-	$O(\frac{1}{\delta})$	$21 + \delta$	Alg. 7, [15, 9]
-	2	52	[17, 2]
doubling dim.	2	$12 + \delta$	[8, 2]

*Department of Computer Engineering, Sharif University of Technology, Tehran, Iran aghamolaei@ce.sharif.edu, ghodsi@sharif.edu

Computing an r-gather algorithm on a k-center with outliers, or running a k-center with outliers [9] on (r, ϵ)-gather both solve LB k-center with outliers.

2 Preliminaries

2.1 MapReduce

In the MapReduce model [11], data is distributed among a set of independent servers. A MapReduce algorithm runs in several rounds. In each round, servers process their data locally and simultaneously. At the end of each round, they communicate with each other by sending a subset of their data to other servers. Two main theoretical models for MapReduce are MapReduce Class (MRC) [16] and Massively Parallel Communication (MPC) [6].

In the MPC model, there are p servers, each with memory $m = O(\frac{N}{p^{1-\gamma}})$ bits of data, where N is the input size and $\gamma \in [0, 1]$ is a parameter of the model. The complexity of MPC algorithms is measured in the number of rounds and the communications between servers.

2.2 k-Center

Given a set of points, k-Center chooses k points as centers, minimizing the maximum distance from each point to its nearest center. Greedy Min-Max (GMM) [14] computes a 2-approximation of metric k-center (algorithm 1). Algorithm 2 is a modified GMM that adds centers until all points are covered with radius R.

Algorithm 1 GMM-k

Input: a set of points S, the number of clusters k
Output: a set of centers T, the cluster sizes $\{n_c\}_{c\in T}$

$T =$ an arbitrary point $p \in S$
for $i = 2, \cdots, k$ **do**
  find a point $p \in S \setminus T$ maximizing $\min_{t\in T} d(p, t)$
  $T \leftarrow T \cup \{p\}$
end for
assign each point to its nearest center from set T
$n_c =$ the number of points assigned to c
return $T, \{n_c\}_{c\in T}$

Algorithm 2 GMM-R

Input: a set of points S, the radius of clustering R
Output: a set of centers T, the cluster sizes $\{n_c\}_{c\in T}$

$dist = \infty, T =$ an arbitrary point $p \in S$
while $dist \geq R$ **do**
  find a point $p \in S \setminus T$ maximizing $\min_{t\in T} d(p, t)$
  $T \leftarrow T \cup \{p\}$, $dist \leftarrow \min_{t\in T} d(p, t)$
end while
assign each point to its nearest center from set T
$n_c =$ the number of points assigned to c
return $T, \{n_c\}_{c\in T}$

Algorithm 3 is a modification of the 4-approximation MapReduce k-center algorithm of [18] that keeps the order of finding centers.

Algorithm 3 Preprocess

Input: point-sets P_i for $i \in [p]$, an integer k
Output: k centers

▷ parallel in all servers, do:
$T_i =$ the centers returned by GMM-k (P_i) for $i \in [p]$
▷ sequentially in the first server, do:
$(c_1, \cdots, c_k) =$ the order of finding centers in GMM-k $(\cup_{i=1}^{p} T_i)$
return $(c_1, \cdots, c_k)$

3 A Lower Bound For r-Gather In MapReduce

Theorem 1 *There are no α-approximation r-gather clusterings in MPC for $\alpha < 2$ with less than k rounds, where k is the number of clusters.*

Proof. Let G be a graph whose vertices are grouped into $2k$ subsets $A_i \in A$ for $i \in [k]$ and $B_i \in B$ for $i \in [k]$. Each subset in A has $l \in [\frac{r}{2}, r)$ points and each subset $B_i \in B$ has at least $\frac{r}{2}$ points. Points inside each $A_i \in A$ have distance 1 from each other and points in $B_i \in B$ have distance 2. For each pair $(A_i, B_i), i \in [k]$, there is an optimal center in set A_i called o_i with distance 1 from all points of B_i. All other edges have weight 2. So the radius of r-gather is 1 if and only if A_i and B_i, for $i \in [k]$, form clusters $A_i \cup B_i$ with o_i as the center, see Figure 3. Also, assume all subsets $A_i \in A$ are in the same server and each subset $B_i \in B$ is in a different server, excluding the one containing subsets of A.

Let ALG be an r-gather algorithm. Since for each $i \in [k]$ points of A_i and B_i are not in the same server, ALG cannot find o_i knowing only A. To find $o_i \in A_i$, ALG needs all points of A_i and at least one point of B_i to be in the same server. Since ALG cannot differentiate between the points of $B_j \in B$ without knowledge of A, in the worst case, ALG has to send all points of A_i to all servers to find o_i, which takes one MPC round. This will only give one point o_i, if $|A_i| = \theta(m)$. So finding the k optimal points requires at least k rounds. □

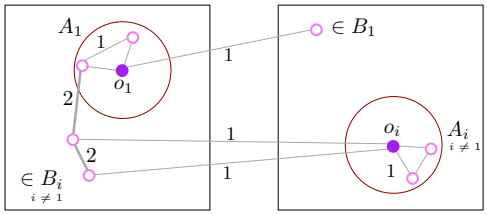


Figure 1: A lower-bound for r-gather (Theorem 1)

4 r-Gather in MapReduce

We propose an algorithm (Algorithm 4) for r-gather. It first finds a lower bound on the optimal radius R using **Bounds(P)** subroutine, which computes a 16-approximation algorithm for r-gather (algorithm 5).

Then it multiplies the lower bound by factor $1+\delta$ and tests the resulting radius with an r-gather decider algorithm **Decider(R,P)** (algorithm 6) that finds an r-gather of radius $4R$ if an r-gather of radius R exists, otherwise it returns *FAIL*.

Algorithm 4 r-Gather 4-approximation

Input: P a set of points distributed in servers
Output: clusters
1: $UB = Bounds(P), R = \frac{UB}{16}$
2: **while** $R \leq UB$ **do**
3: **if** $Decider(R, P)$ returns *FAIL* **then**
4: $R \leftarrow (1+\delta)R$
5: **else**
6: **return** the output of $Decider(R, P)$
7: **end if**
8: **end while**

4.1 r-Gather Bounds

Algorithm 5 finds a $O(1)$-approximation for r-gather. Assume k is chosen such that $pk^2 \leq m$. **Preprocess**() is algorithm 3, which keeps the order of centers to guarantee $|T_i| \leq k, \forall i \in [p]$

Algorithm 5 Bounds

Input: point-sets P_i for $i \in [p]$, a constant $k \in \mathbb{N}$
Output: r-gather clusters
1: $T = \textbf{Preprocess}(\{P_i\}_{i\in[p]}, k+1)$
2: send T to all servers
▷ parallel in all servers:
3: $C = \emptyset, j = 1$
4: **for** $j \in [k+1]$ **do**
5: $C \leftarrow C \cup \{c_j\}$
6: assign points to their nearest center in C
7: $n(i, c, j)$ = the sizes of clusters centered at $c \in C$
8: **end for**
9: send $n(i, c, j)$ to the first server for $i \in [p], j = [k+1], c \in T$
▷ sequentially in the first server, do:
10: find the minimum $k' \in [k+1]$ such that there exists a center $c \in T$, $\sum_{i=1}^{p} n(i, c, k') < r$
11: **return** $\{c_1, \cdots, c_{k'-1}\}$

Lemma 2 *In algorithm 5, each cluster has at least r points.*

Proof. According to line 10 of the algorithm, k' is the first step that the number of points assigned to a center becomes less than r. So, in step $k'-1$ each center in $\{c_1, \cdots, c_{k'-1}\}$ has at least r points. □

Theorem 3 *Algorithm 5 is a 16-approximation for r-gather.*

Proof. Based on Lemma 2, each cluster has at least r points.

Consider a point s that was reassigned at step k'. Let c_i be the center of the cluster that contained s at step $k'-1$. Since points are assigned to their nearest centers, we have $d(c_j, s) \leq d(c_i, s)$, and using triangle inequality $d(c_i, c_j) \leq d(c_i, s) + d(c_j, s)$. So, $d(c_i, c_j) \leq 2d(c_i, s)$.

Each point $c_i \in C$ covers all points of the optimal cluster containing c_i, with radius $2R^*$, where R^* is the radius of the optimal r-gather of S, i.e. $d(c_i, s) \leq 2R^*$. Putting the two previous inequalities together gives the bound $d(c_i, c_j) \leq 4R^*$.

GMM-k adds the farthest point as a center each time, so the cluster containing c_j has the maximum radius R' among the clusters. Let c_k be the nearest center to c_j, i.e. $d(c_k, c_j) \leq d(c_i, c_j)$ for all $i \neq k$. Since c_j is assigned to its nearest center, $R' = d(c_k, c_j)$. So, we have $R' \leq 4R^*$.

Algorithm 3 is a 4-approximation for k-Center, so $R \leq 4R'$, where R is the radius of k-Center from first round. The last two inequalities yield $R \leq 4R' \leq 16R^*$. □

In algorithm 5, for each server we send k^2 numbers to the first server, so there are $pk^2 \leq m$ numbers in total. Assuming $pk^2 \leq m$, algorithm 5 is in MPC.

4.2 r-Gather Decider

r-Gather decider takes R as input and gives an r-gather clustering with radius $3R$ if an r-gather with radius R exists, otherwise it returns *FAIL*. Assume the r-gather bounds algorithm uses the order of adding centers in GMM-k algorithm.

4.2.1 Cluster Re-assignments via Max. Flow

Two sets of clusters D and S with the number of points assigned to them, and $\mathcal{R}$ an upper bound on the radius of clustering are given. For each $p \in D \cup S$, we denote the number of points assigned to p with n_p. The goal is to re-assign the points of set S to the centers of clusters in D using a flow network.

Definition 4.1 *Consider the graph $G = (V, E)$ with $V = \{s, t\} \cup D \cup S$ and $E = E_s \cup E_m \cup E_t$, where E_s, E_m, E_t are defined as follows:*

- $E_s = \{(s, v, r - n_v) \mid v \in D\}$
- $E_m = \{(u, v, n_v) \mid u \in D, v \in S, d(u, v) \leq \mathcal{R}\}$

- $E_t = \{(u, t, n_u) \mid u \in S\}$

and where (u, v, c) denotes an edge from vertex u to vertex v with capacity c.

Let $MaxFlow(D, S, \{n_c\}_{c\in D\cup S}, \mathcal{R})$ be the function that if the max. flow of G in the flow network of Definition 4.1 is less than $\sum_{(u,v,c)\in E_s} c$, returns (*FAIL*, $\emptyset$) and otherwise returns (*SUCCESS*, $\{(u, v, c) \in E_m\}$).

Figure 2 shows the original clustering, the flow network and the clusters after re-assignment.

Algorithm 6 Decider

Input: point-sets P_i for $i \in [p]$, radius R
Output: r-gather clustering or *FAIL*

▷ parallel in all servers, do:
1: $T_i, \{n_c\}_{c\in T_i}$ = GMM-R (P_i, R) for $i \in [p]$
2: send T_i for $i \in [p]$ to the first server
▷ sequentially in the first server, do:
3: D = centers with $\geq r$ points assigned to it
4: S = centers with $< r$ points assigned to it
5: **for** $c_s \in S$ **do**
6: $N(c_s, R)$ = centers in radius of R of c_s
7: **if** all centers $c \in N(c_s, 3R)$ are in S **then**
8: $D \leftarrow D \cup \{c_s\}$
9: **end if**
10: **end for**
11: $(flag, E) = MaxFlow(D, S, \{n_c\}_{c\in D\cup S}, 3R)$
12: **if** $flag = FAIL$ **then**
13: **return** *FAIL*
14: **end if**
15: send E to all servers
16: according to E, assign points to centers in D
▷ parallel in all servers, do:
17: **return** centers in set D and their assigned points

4.2.2 Analysis

Lemma 4 *In algorithm 6, each cluster has at least r points.*

Proof. In the flow network of definition 4.1, there is an edge with capacity $r - n_{c_d}$ from node s to every center $c_d \in D$. From the definition of $MaxFlow(.,.,.,.)$ we know the max. flow is $r|D| - \sum_{c\in D} n_c$, which means c_d is assigned $r - n_{c_d}$ new points. Since c_d had n_{c_d} existing assigned points, in total it has r assigned points. □

Theorem 5 *Algorithm 6 is a 4-approximation decider.*

Proof. Let $C_1, \cdots, C_k$ be an r-gather of radius R for $\cup_i P_i$. GMM-R keeps at least one point from each cluster C_i. For each point p assigned to c_s, $d(c_s, p) \leq R$.

Each point $p \in C_i$ covers all the points in C_i with radius $2R$, so either there is a point in D which covers C_i or there are a set of points in S that cover C_i.

Now, we prove it is possible to reassign points using radius $3R$ such that each cluster has at least r points. Let $c_d \in C_i$ and $c_s \in C_j, i \neq j$ be two points in the output of GMM-R and $d(o_i, o_j) \leq 2R$, where o_i, o_j are the centers of C_i, C_j. Assume c_s covers points $X \subset C_j$ with radius R, such that the number of points assigned to c_d becomes less than r. Then for all $x \in X$, $d(c_d, c_s) \leq d(c_d, x) + d(x, c_s) \leq 3R$.

Line 8 guarantees there is a center in D for each C_i, by adding a point of S to D if there are no points of D with distance $3R$ from it.

By triangle inequality and the bounds on distances, $d(c_d, p) \leq d(c_d, c_s) + d(c_s, p) \leq 4R$. □

Corollary 6 *In theorem 5, for $r = 1$, radius $3R$ can be replaced by $2R$ which gives a 3-approximation decider.*

In algorithm 6, we send k points per server to the first server, which is pk points in total. Since we assumed $pk^2 \leq m$, algorithm 6 runs in the MPC model.

4.3 Analysis

Theorem 7 *Algorithm 4 takes $O(\frac{1}{\delta})$ rounds in MapReduce and gives a $(4 + \delta)$-approximation for r-gather.*

Proof. According to algorithm 4, r-gather 4-approximation algorithm runs r-gather bounds once at line 1. By theorem 3, we know r-gather bounds takes 2 rounds in MapReduce model. In algorithm 4 line 2, each while iteration runs algorithm 6 once. By theorem 5, we know r-gather decision takes 2 rounds in the MapReduce model. In line 2 the **while** statement iterates at most $O(\frac{1}{\delta})$ times, so in total the algorithm takes $O(\frac{1}{\delta})$ rounds in the MapReduce model.

Let R be the greatest radius for which r-gather decider returns *FAIL* and R^* be the optimal r-gather radius. According to theorem 5 and the fact that algorithm 4 checks factors of $1 + \delta$, $R \leq R^*(1 + \delta)$ and the algorithm returns a clustering of radius $4R \leq 4(1+\delta)R^*$.

By theorems 3 and 5, algorithms 5 and 6 are in the MPC model. So, algorithm 4 runs in MPC, too. □

5 (r, ϵ)-Gather in MapReduce

In this section, we provide an algorithm that takes the optimal radius R as input and finds a (r, ϵ)-gather clustering of radius $7R$ if one exists, or returns *FAIL* otherwise. Also, we give an algorithm for finding a lower bound and an upper bound on the radius of clustering (algorithm 8 denoted by (r, ϵ)-GLB(.)), which we then use with a 7-approximation algorithm as its decision subroutine to solve the problem.

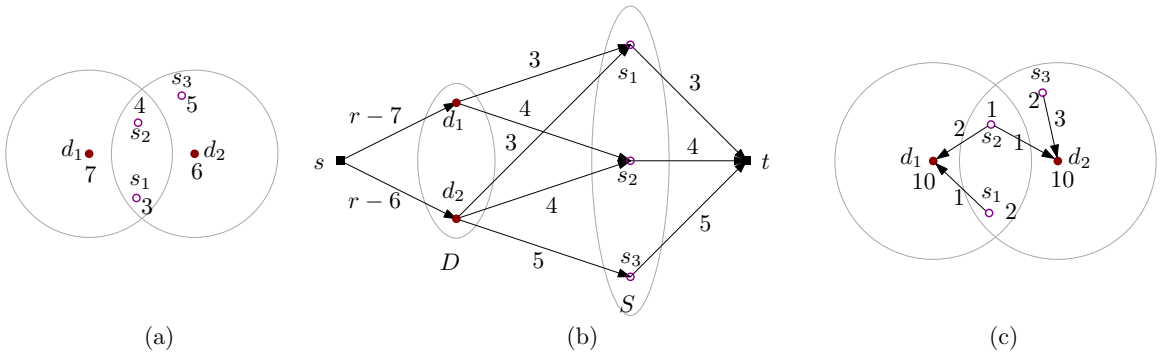


Figure 2: (a) a clustering, (b) the network flow of (a), and (c) a reassignment of points to centers for $r = 10$

Algorithm 7 (r, ϵ)-Gather

Input: A set of sets of points P, constants $\epsilon > 0, \delta > 0$
Output: clusters

1: $R = (r, \epsilon) - GLB(P)$
2: **while** (r, ϵ)-Gather Decision$(R, S) = FAIL$ **do**
3: $\quad R \leftarrow (1 + \delta)R$
4: **end while**
5: **return** the output of algorithm 9

5.1 (r, ϵ)-Gather Lower-Bound ((r, ϵ)-GLB)

The inputs are a set of points distributed among a set of servers and a parameter ϵ that means at most $n\epsilon$ points can be left unclustered. Assume we choose k such that $pk^2 \leq m$, and k is greater than the number of clusters in the optimal (r, ϵ)-gather. In the first round of algorithm 9, we use algorithm 3.

Lemma 8 *The value returned by algorithm 8 is a lower bound within a constant factor of the optimal radius of (r, ϵ)-gather.*

Proof. In line 18 of algorithm 8 the algorithm terminates if the points assigned to the centers in L become more than $n\epsilon$. Let outlier candidates (set OC) be the set of centers in L and their assigned points. Since $|OC| > n\epsilon$, there is a point $p \in OC$ that must be clustered. Suppose the nearest center to p is a center called c, then we have $d(c, p) \leq \frac{R_c}{2}$. Since $c \in L$, we know c does not have r points within radius of R_c. For each point q outside the radius R_c around c, by triangle inequality, we have $d(c, q) \leq d(c, p) + d(p, q)$. So we have $R_c \leq \frac{R_c}{2} + d(p, q)$ and $\frac{R_c}{2} \leq d(p, q)$. So p does not have more than r points within radius $\frac{R_c}{2}$. We know in any (r, ϵ)-gather clustering with radius R' each point can cover all points of its cluster with radius $2R'$. Since p must be clustered in the optimal (r, ϵ)-gather clustering, the cluster contains p cannot have radius less than $\frac{R_c}{4}$. So we have $R^* \geq \frac{R_c}{4}$ where R^* is the optimal radius. □

Algorithm 8 (r, ϵ)-GLB

Input: point-sets P_i for $i \in [p]$, constants $\epsilon > 0, k \in \mathbb{N}$
Output: lower bound for (r, ϵ)-gather clustering

1: $T = \textbf{Preprocess}(\{P_i\}_{i\in[p]}, k)$
2: send centers in T to all servers
▷ parallel in all servers, do:
3: $C = \emptyset$
4: **for** $j \in [k]$ **do**
5: $\quad C \leftarrow C \cup \{c_j\}$
6: $\quad$ assign each point to its nearest center $c \in C$
7: $\quad R_c =$ the radius of the cluster centered at c
8: $\quad n(i, c, R, j) =$ the number of input points within distance R of c in server i
9: **end for**
10: send set C with $\{n(i, c, R_c, j)\}$ and $\{n(i, c, \frac{R_c}{2}, j)\}$ for $i \in [p], j \in [k], c \in C$ to the first server
▷ sequentially in the first server, do:
11: $C_T = \emptyset$, $L = \emptyset$
12: **for** $j \in [k]$ **do**
13: $\quad n(c, R, j) = \sum_{i=1}^{p} n(i, c, R, j)$ for $c \in C_T$
14: $\quad C_T \leftarrow C_T \cup \{c_j\}$
15: $\quad$ **if** $n(c_j, R_c, j) < r$ **then**
16: $\quad\quad L \leftarrow L \cup \{c_j\}$
17: $\quad$ **end if**
18: $\quad$ **if** $\sum_{c\in L} n(c, \frac{R_c}{2}, j) > n\epsilon$ **then**
19: $\quad\quad$ break
20: $\quad$ **end if**
21: **end for**
22: $R = min_{c\in C_T} R_c$
23: **return** $\frac{R}{4}$

5.2 (r, ϵ)-Gather Decision

The inputs are R, the radius of (r, ϵ)-gather clustering, and ϵ, a parameter which indicates at most $n\epsilon$ points can be left unclustered. (r, ϵ)-Gather decision algorithm gives an (r, ϵ)-gather with radius $7R$, if there exists a (r, ϵ)-gather clustering with radius R, and returns $FAIL$ otherwise.

Algorithm 9 (r, ϵ)-Gather Decision

Input: point-sets P_i for $i \in [p]$, radius R, a constant ϵ
Output: (r, ϵ)-Gather clustering or *FAIL*

▷ parallel in all servers, do:
1: $T_i, \{n_c\}_{c \in T_i} =$ GMM-R (P_i, R)
2: assign points in P_i to their nearest center in T_i
3: send sets T_i along with $\{n_c\}_{c \in T_i}$ to the first server
▷ sequentially in the first server, do:
4: **for** $c \in \cup_{i=1}^{p} T_i$ **do**
5: $N(c, R) =$ centers within radius R of c
6: **if** $\sum_{q \in N(c,3R)} n_q < r$ **then** remove c and the points assigned to c as outliers
7: **if** the number of outliers exceeds ϵn **then return** *FAIL*
8: **end for**
9: $D = \emptyset$, $S = \emptyset$, $U = \cup_{i=1}^{p} C_i$
10: **for** $c_u \in U$ **do**
11: $N_U(c, R) = U \cap N(c, R)$
12: $n(N_U(c, R)) = \sum_{t \in N_U(c,R)} n_t$
13: **if** $n(N_U(c_u, 3R)) \geq r$ **then**
14: $U \leftarrow U \setminus N_U(c_u, 3R)$
15: $D \leftarrow D \cup \{c_u\}$
16: $S \leftarrow S \cup (N_U(c_u, 3R) \setminus \{c_u\})$
17: **end if**
18: **end for**
19: $(flag, E) = MaxFlow(D, S, \{n_c\}_{c \in D \cup S}, 3R)$
20: **if** $flag =$ *FAIL* **then return** *FAIL*
21: send E to all servers
▷ parallel in all servers, do:
22: reassign the points of S to D based on E
23: reassign the points assigned to centers in U to their nearest center in D
24: **return** centers in set D and their assigned points

Lemma 9 *All the clustered points in the optimal (r, ϵ)-gather will also be clustered by algorithm 9, for $R \geq R^*$, where R^* is the optimal radius of (r, ϵ)-gather.*

Proof. Based on line 6 of algorithm 9, centers with less than r points within radius $3R$ of themselves are outliers. Let c be a center that was removed as an outlier, and p be a point assigned to c, so p has also been removed as an outlier.

For any point q with $d(c, q) \geq 3R$, the triangle inequality gives $d(c, q) \leq d(c, p) + d(p, q)$, and line 1 guarantees $d(c, p) \leq R$. So, $d(p, q) \geq 2R$. This means in any (r, ϵ)-gather, points p and q cannot belong to the same cluster. Since c has less than r points within distance $3R$ of itself, no point with distance R from c can be clustered in the optimal (r, ϵ)-gather. So, any point that is clustered by the optimal solution is also clustered by the algorithm. □

Theorem 10 *The approximation factor of (r, ϵ)-gather is 7.*

Proof. In the flow network of line 19, there are only edges between *dense* centers $c_d \in D$ and *sparse* centers $c_s \in S$, if $d(c_d, c_s) \leq 3R$. So, c_d can only get new points from points $c_s \in S$ of distance at most $3R$ from it. Let p be a point assigned to c_s, with $d(c_s, p) \leq R$. By triangle inequality, $d(c_d, p) \leq d(c_d, c_s) + d(c_s, p) \leq 4R$. Therefore, all centers $c_s \in S$ and their assigned points can be clustered with radius $4R$ and with c_d as center.

Consider an unclustered center $c_u \in U$. If c_u has less than r points with distance at most $3R$, it gets removed at line 6. After lines 13-16, some of the points with distance $3R$ of c_u must have been assigned to other centers to make c_u an *unclustered* center. Since the points assigned to a dense center are marked as *sparse*, then there are only *sparse* centers within distance $3R$ of c_u. Based on line 23, c_u is assigned to its closest *sparse* center c_s. From line 16, there is a *dense* center with distance at most $3R$ from c_s. By triangle inequality, $d(c_d, c_u) \leq d(c_d, c_s) + d(c_s, c_u) \leq 6R$. Let p be a point assigned to c_u, so using triangle inequality, $d(c_d, p) \leq d(c_d, c_u) + d(c_u, p) \leq 7R$. □

We send at most k points per server to the first server, which is at most pk points in total. So, for $pk^2 \leq m$, algorithm 9 follows the memory constraints of MPC.

5.3 Analysis

Theorem 11 *Algorithm 7 takes $O(\frac{1}{\delta})$ MapReduce rounds and gives a $(7 + \delta)$-approximation (r, ϵ)-gather.*

Proof. According to line 1, (r, ϵ)-gather algorithm runs (r, ϵ)-GLB once. Algorithms 8 and 9 each take 2 rounds in the MapReduce model. In line 2, the **while** loop iterates at most $O(\frac{1}{\delta} \log \frac{OPT}{LB})$ times, for $LB = (r, \epsilon)$-$GLB(P)$. So, the total round complexity of the algorithm is $O(\frac{1}{\delta})$.

Let R be the maximum radius for which (r, ϵ)-gather decision algorithm returns *FAIL*, and R^* be the optimal radius of (r, ϵ)-gather. By theorem 10, and the fact that we multiply R by a factor $1 + \delta$ each time, the approximation factor of the algorithm is $7(1 + \delta)$. □

6 Conclusion and Open Problems

We gave $O(1)$-approximation algorithms for two minimum radius covering problems with lower bounds on the number of members per cluster in MapReduce. Unlike most MapReduce capacitated clustering algorithms that are based on linear programming, our algorithm uses maximum flow.

Improving the round complexity or the approximation factor of our algorithm and removing the assumption $k \leq m$ to improve scalability remain open. Proving similar results for other clusterings with ℓ_p-based costs such as k-means and k-median is also interesting.

References

[1] G. Aggarwal, T. Feder, K. Kenthapadi, R. Motwani, R. Panigrahy, D. Thomas, and A. Zhu. Approximation algorithms for k-anonymity. *Journal of Privacy Technology (JOPT)*, 2005.

[2] G. Aggarwal, R. Panigrahy, T. Feder, D. Thomas, K. Kenthapadi, S. Khuller, and A. Zhu. Achieving anonymity via clustering. *ACM Transactions on Algorithms (TALG)*, 6(3):49, 2010.

[3] S. Aghamolaei and M. Ghodsi. A composable coreset for k-center in doubling metrics. In *30th Canadian Conference on Computational Geometry (CCCG)*, 2018.

[4] S. Ahmadian and C. Swamy. Approximation algorithms for clustering problems with lower bounds and outliers. In *43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.

[5] M. Bateni, A. Bhaskara, S. Lattanzi, and V. Mirrokni. Distributed balanced clustering via mapping coresets. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2591–2599, 2014.

[6] P. Beame, P. Koutris, and D. Suciu. Communication steps for parallel query processing. In *Proceedings of the 32nd ACM SIGMOD-SIGACT-SIGAI symposium on Principles of database systems*, pages 273–284. ACM, 2013.

[7] C. Benjamin, M. Fung, K. Wang, R. Chen, and P. Yu. Privacy-preserving data publishing: A survey of recent developments. *ACM Computing Surveys*, 42(4):1–53, 2010.

[8] M. Ceccarello, A. Pietracaprina, and G. Pucci. Solving k-center clustering (with outliers) in mapreduce and streaming, almost as accurately as sequentially. *Proceedings of the VLDB Endowment*, 12(7):766–778, 2019.

[9] M. Charikar, S. Khuller, D. M. Mount, and G. Narasimhan. Algorithms for facility location problems with outliers. In *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, pages 642–651. Society for Industrial and Applied Mathematics, 2001.

[10] D. J. De Witt, R. Ramakrishnan, et al. Mondrian multidimensional k-anonymity. In *Proc. of the 22nd IEEE International Conference on Data Engineering (ICDE)*, 2006.

[11] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[12] H. Ding, L. Hu, L. Huang, and J. Li. Capacitated center problems with two-sided bounds and outliers. In *Workshop on Algorithms and Data Structures*, pages 325–336. Springer, 2017.

[13] A. Ene, S. Har-Peled, and B. Raichel. Fast clustering with lower bounds: No customer too far, no shop too small. *arXiv preprint arXiv:1304.7318*, 2013.

[14] T. F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38:293–306, 1985.

[15] P. Indyk, S. Mahabadi, M. Mahdian, and V. S. Mirrokni. Composable core-sets for diversity and coverage maximization. In *Proceedings of the 33rd ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 100–108. ACM, 2014.

[16] H. Karloff, S. Suri, and S. Vassilvitskii. A model of computation for mapreduce. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*, pages 938–948. SIAM, 2010.

[17] G. Malkomes, M. J. Kusner, W. Chen, K. Q. Weinberger, and B. Moseley. Fast distributed k-center clustering with outliers on massive data. In *Advances in Neural Information Processing Systems*, pages 1063–1071, 2015.

[18] J. McClintock. Parallel algorithms for k–center clustering. In *4TH ANNUAL DOCTORAL COLLOQUIUM*, page 33, 2016.

[19] H. Park and K. Shim. Approximate algorithms for k-anonymity. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 67–78. ACM, 2007.

[20] I. Roy, S. T. Setty, A. Kilzer, V. Shmatikov, and E. Witchel. Airavat: Security and privacy for mapreduce. In *NSDI*, volume 10, pages 297–312, 2010.

[21] P. Samarati and L. Sweeney. Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression. Technical report, technical report, SRI International, 1998.

Peeling Digital Potatoes

Loïc Crombez Guilherme D. da Fonseca Yan Gérard *

Abstract

The potato-peeling problem (also known as convex skull) is a fundamental computational geometry problem that consist in finding the largest convex shape inside a given polygon. The fastest algorithm to date runs in $O(n^8)$ time for a polygon with n vertices that may have holes. In this paper, we consider a digital version of the problem. A set $K \subset \mathbb{Z}^2$ is *digital convex* if $\text{conv}(K) \cap \mathbb{Z}^2 = K$, where $\text{conv}(K)$ denotes the convex hull of K. Given a set S of n lattice points, we present polynomial time algorithms for the problems of finding the largest digital convex subset K of S (*digital potato-peeling problem*) and the largest union of two digital convex subsets of S. The two algorithms take roughly $O(n^3)$ and $O(n^9)$ time, respectively. We also show that those algorithms provide an approximation to the continuous versions.

1 Introduction

The *potato-peeling problem* [24] (also known as *convex skull* [35]) consists of finding the convex polygon of maximum area that is contained inside a given polygon (possibly with holes) with n vertices. The fastest exact algorithm known takes $O(n^7)$ time without holes and $O(n^8)$ if there are holes [12]. The problem is arguably the simplest geometric problem for which the fastest exact algorithm known is a polynomial of high degree and this high complexity motivated the study of approximation algorithms [11, 26]. Multiple variations of the problem have been considered, including triangle-mesh [1] and orthogonal [19, 36] versions. In this paper, we consider a digital geometry version of the problem.

Digital geometry is the field of mathematics that studies the geometry of points with integer coordinates, also known as *lattice points* [28]. Different definitions of convexity in $\mathbb{Z}^2$ have been investigated, such as digital line, triangle, line [27], HV (for Horizontal and Vertical [4]), and Q (for Quadrant [17]) convexities. These definitions guarantee that a digital convex set is connected (in terms of the induced grid subgraph), which simplifies several algorithmic problems.

Throughout this paper, however, we use the main and original definition of digital convexity from the geometry of numbers [25]. A set of lattice points $K \subset \mathbb{Z}^d$ is *digital convex* if $\text{conv}(K) \cap \mathbb{Z}^d = K$, where $\text{conv}(K)$ denotes the convex hull of K. This definition does not guarantee connectivity of the grid subgraph, but provides several other important mathematical properties, such as being preserved under certain affine transformations. The authors recently showed how to efficiently test digital convexity in the plane [15]. A natural question is to determine the largest digital convex subset.

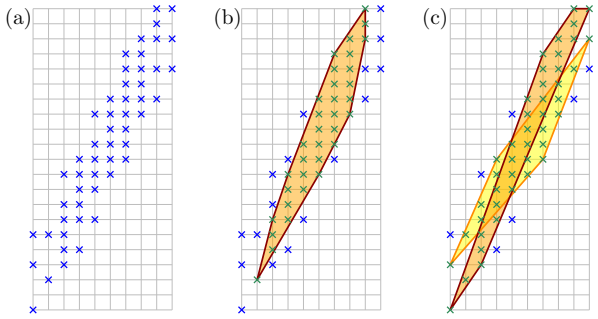


Figure 1: (a) Input lattice set S. (b) Largest digital convex subset of S (Problem 1). (c) Largest union of two digital convex subsets of S (Problem 2).

The *digital potato-peeling problem* is defined as follows and is illustrated in Figure 1(a,b).

Problem 1 (Digital potato-peeling) *Given a set $S \subset \mathbb{Z}^2$ of n lattice points described by their coordinates, determine the largest set $K \subseteq S$ that is digital convex (i.e., $\text{conv}(K) \cap \mathbb{Z}^2 = K$), where largest refers to the area of $\text{conv}(K)$.*

Our algorithms can easily be modified to maximize the number of points in K instead of the area of $\text{conv}(K)$. Compared to the continuous version, the digital geometry setting allows us to explicitly represent the whole set of input points, instead of limiting ourselves to polygonal shapes with polygonal holes. Note that the input of the continuous and digital problems is intrinsically different, hence we cannot compare the complexity of the two problems. Related continuous problems have been studied, such as the maximum volume of an empty convex body amidst n points [18], or the *optimal island problem* [6, 22], in which we are given two sets $S_p, S_n \subset \mathbb{R}^2$, and the goal is to determine that largest subset $K \subseteq S_p$ such that $\text{conv}(K) \cap S_n = \emptyset$.

Heuristics for the digital potato-peeling problem have

*Université Clermont Auvergne and LIMOS, Clermont-Ferrand, France

been presented in [10, 13], but no exact algorithm was known. We solve this open problem by providing the first polynomial-time exact algorithm.

We also solve the question of covering the largest area with two digital convex subsets. The problem is defined as follows and is illustrated in Figure 1(a,c).

Problem 2 (Digital 2-potato peeling) *Given a set $S \subset \mathbb{Z}^2$ of n lattice points described by their coordinates, determine the largest set $K = K_1 \cup K_2 \subseteq S$ such that K_1 and K_2 are both digital convex, where largest refers to the area of* $\operatorname{conv}(K_1) \cup \operatorname{conv}(K_2)$.

A related continuous problem consists of completely covering a polygon by a small number of convex polygons inside of it. O'Rourke showed that covering a polygon with the minimum number of convex polygons is decidable [29, 30], but the problem has been shown to be NP-hard with or without holes [16, 31]. Shermer [34] presents a linear time algorithm for the case of two convex polygons and Belleville [8] provides a linear time algorithm for three. We are not aware of any previous results on finding a fixed (non-unit) number of convex polygons inside a given polygon and maximizing the area covered.

Our results

We present polynomial time algorithms to solve each of these two problems. In Section 2, we show how to solve the digital potato-peeling problem in $O(n^3 + n^2 \log r)$ time, where r is the diameter of the input S. We adapt an algorithm designed to solve the optimal island problem [6, 22]. This algorithm builds the convex polygon $\operatorname{conv}(K)$ through its triangulation. We use Pick's theorem [32] to test digital convexity for each triangle and the $O(\log r)$ factor in the running time comes from the gcd computation required to apply Pick's theorem. The algorithm makes use of the following two properties: (i) it is possible to triangulate K using only triangles that share a common bottom-most vertex v and (ii) if the polygons lying on both sides of one such triangle (including the triangle itself) are convex, then the whole polygon is convex.

These two properties are no longer valid for Problem 2, in which the solution $\operatorname{conv}(K_1) \cup \operatorname{conv}(K_2)$ is the union of two convex polygons. Also, since convex shapes are not pseudo-disks (the boundaries may cross an arbitrarily large number of times), separating the input with a constant number of lines is not an option. Instead of property (i), our approach uses the fact that the union of two (intersecting) convex polygons can be triangulated with triangles that share a common vertex ρ (that may not be a vertex of either convex polygon). Since ρ may not have integer coordinates, we can no longer use Pick's theorem, and resort to the formulas from Beck and Robins [7] or the algorithm from Barvinok [5] to count the lattice points inside each triangle in $O(\operatorname{polylog} r)$ time.

Furthermore, to circumvent the fact that the solution no longer obeys property (ii), we use a directed acyclic graph (DAG) that encapsulates the orientation of the edges of both convex polygons. For those reasons, the running time of our algorithm for Problem 2 increases to $O(n^9 + n^6 \operatorname{polylog} r)$. The corresponding algorithm is described in Section 3.

In Section 4, we show that a solution to the digital version of the problems provides an approximation to the continuous versions, establishing a formal connection between the continuous and digital versions.

Reducing the complexity of our algorithms or extending the result to higher numbers of convex polygons remain intriguing open questions, which are discussed in Section 5. Throughout, we assume the RAM model of computation, in which elementary operations on the input coordinates take constant time.

2 Digital Potato Peeling

In this section, we present an algorithm to solve the digital potato-peeling problem in $O(n^3 + n^2 \log r)$ time, where n is the number of input points and r is the diameter of the point set.

Fischer [22] and Bautista et al. [6] showed how to solve the following related problem in $O(n^3)$ time, where n is the total number of points.

Problem 3 (Optimal Island) *Given two sets $S_p, S_n \subset \mathbb{R}^2$, determine the largest subset $K \subseteq S_p$ such that* $\operatorname{conv}(K) \cap S_n = \emptyset$.

The potato peeling problem 1 for an input $S \subset \mathbb{Z}^2$ is the optimal island problem with $S_p = S$ and $S_n = \mathbb{Z}^2 \setminus S_p$. Restricting the problem to the bounding box of S_p, makes S_n finite as $|S_n| = O(r^2)$. The resulting $O(r^6)$ complexity being very large relative to r, we do not use this direct approach. Nevertheless, the algorithm provides some key insights.

The algorithm consists of two phases. First, a list $\mathcal{T}$ of all *valid* triangles is computed. A triangle $\triangle$ is said to be valid if its vertices are a subset of S_p and if $\triangle \cap S_n = \emptyset$. Second, using $\mathcal{T}$ and the fact that every convex polygon has a fan triangulation in which all the triangles share a common bottom vertex, the solution is computed by appending valid triangles using dynamic programming. In order to adapt this algorithm to solve the digital potato peeling, it suffices to compute the list of valid triangles $\mathcal{T}$.

2.1 Valid Triangles

For any triangle whose vertices are lattice points $\triangle$, and any digital set S: $|\triangle \cap S| = |\triangle \cap \mathbb{Z}^2|$ implies that $\triangle$ is

valid. As in [6], we use the following result of Eppstein et al. [20] to compute $|\triangle \cap S|$.

Theorem 1 *Let S be a set of n points in the plane. The set S can be preprocessed in $O(n^2)$ time and space in order to, for any query triangle $\triangle$ with vertices in S, compute the number of points $|\triangle \cap S|$ in constant time.*

In order to compute $|\triangle \cap \mathbb{Z}^2|$, first, for all pairs of points $p_1, p_2 \in S$, we compute the number of lattice points lying on the edge p_1p_2 using a gcd computation. This takes $O(n^2 \log r)$ time, where r is the diameter of S. Now, using Pick's formula [32] which requires to compute both $area(\triangle)$ and the number of lattice points lying on the edges of $\triangle$, we determine in $O(1)$ time the validity of a triangle. Since there are $O(n^3)$ triangles with vertices in S, the list $\mathcal{T}$ of all valid triangles is computed in $O(n^3 + n^2 \log r)$ time. Using $\mathcal{T}$, the algorithm of Bautista et al. [6] determines the largest convex polygon formed by triangles in $\mathcal{T}$ in $O(n^3)$ time. Hence, we have the following theorem.

Theorem 2 *There exists an algorithm to solve Problem 1 (digital potato peeling) in $O(n^3 + n^2 \log r)$ time, where n is the number of input points and r is the diameter of the input.*

3 Digital 2-Potato Peeling

In this section, we show how to find two digital convex sets K_1, K_2, maximizing the area of $\text{conv}(K_1) \cup \text{conv}(K_2)$. We note that the solution described in this section can easily be adapted to solve the optimal 2-islands problem:

Problem 4 (Optimal 2-Islands) *Given two sets $S_p, S_n \subset \mathbb{R}^2$, determine the largest union of subsets $K_1 \cup K_2$ such that $K_1 \cup K_2 \subseteq S_p$, $\text{conv}(K_1) \cap S_n = \emptyset$ and $\text{conv}(K_2) \cap S_n = \emptyset$.*

Consider a solution of the digital 2-potato peeling problem. Either the two convex hulls intersect or they do not (Figure 2). We treat those two cases separately and the solution to Problem 2 is the largest among both. Hence, we consider the two following variations of the 2-potato-peeling problem.

Problem 5 (Disjoint 2-potato peeling) *Given a set $S \subset \mathbb{Z}^2$ of n lattice points given by their coordinates, determine the largest two digital convex sets $K_1 \cup K_2 \subseteq S$ such that $\text{conv}(K_1) \cap \text{conv}(K_2) = \emptyset$.*

Problem 6 (Intersecting 2-potato peeling) *Given a set $S \subset \mathbb{Z}^2$ of n lattice points given by their coordinates, determine the largest union of two digital convex sets $K_1 \cup K_2 \subseteq S$ such that $\text{conv}(K_1) \cap \text{conv}(K_2) \neq \emptyset$. In this case, largest means the maximum area of $\text{conv}(K_1) \cup \text{conv}(K_2)$.*

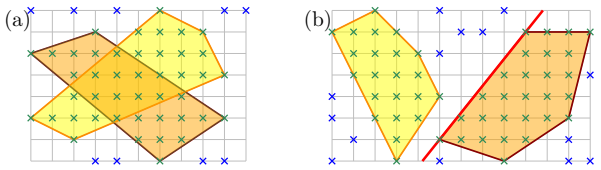


Figure 2: (a) The two optimal sets intersect. (b) The two optimal sets are disjoint and there is a supporting separating line.

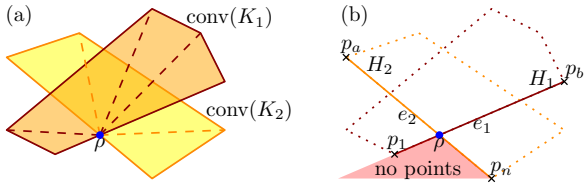


Figure 3: (a) A fan triangulation of two intersecting convex polygons from a point ρ. (b) Definitions used to solve Problem 7.

3.1 Disjoint Convex Polygons

Any two disjoint convex shapes can be separated by a straight line. Moreover two convex polygons can be separated by a supporting line of an edge of one of the convex polygons (Figure 2(b)).

For each ordered pair of distinct points $p_1, p_2 \in S$, we define two subsets S_1, S_2. The set S_1 contains the points on the line p_1, p_2 or to the left of it (according to the direction $p_2 - p_1$). The set S_2 contains the remaining points.

For each pair of sets S_1, S_2, we independently solve Problem 1 for each of S_1 and S_2. Since there are $O(n^2)$ pairs and each pair takes $O(n^3 + n^2 \log r)$ time, we solve Problem 5 in $O(n^5 + n^4 \log r)$ time.

3.2 Intersecting Convex Polygons

The more interesting case is when the two convex polygons intersect (Problem 6). Note that it is possible to triangulate the union of two convex polygons that share a common boundary point ρ using a fan triangulation around ρ (Figure 3). Hence we consider the following rooted version of the problem.

Problem 7 (Rooted 2-potato peeling) *Given a set $S \subset \mathbb{Z}^2$ of n lattice points represented by their coordinates and two edges $e_1, e_2 \in S^2$ that cross at a point ρ, determine the largest union of two digital convex sets $K_1, K_2 \subseteq S$ such that e_1 is an edge of $\text{conv}(K_1)$ and e_2 is an edge of $\text{conv}(K_2)$.*

Let ρ be the intersection point of e_1, e_2. The strategy of the algorithm to solve Problem 7 is to encode the problem into a DAG (V, E) whose longest directed path corresponds to the desired solution. To avoid confusion,

we use the terms *node* and *arc* for the DAG and keep the terms *vertex* and *edge* for the polygons. It is well known that the longest directed path in a DAG (V, E) can be calculated in $O(|V| + |E|)$ time [33].

Let $\mathcal{T}$ be the set of valid triangles with two vertices from S and ρ as the remaining vertex. The nodes $V = \mathcal{T}^2 \cup \{v_0\}$ are ordered pairs of valid triangles and a starting node v_0. The number of nodes is $|V| = O(n^4)$. Before we define the arcs, we give an intuitive idea of our objective.

Each node $(\triangle_1, \triangle_2) \in V$ is such that $\triangle_1$ (resp. $\triangle_2$) is used to build the fan triangulation of $\text{conv}(K_1)$ (resp. $\text{conv}(K_2)$). The arcs will be defined in a way that, at each step as we walk through a path of the DAG, we add one triangle to either $\text{conv}(K_1)$ or to $\text{conv}(K_2)$. The arcs enforce the convexity of both $\text{conv}(K_1)$ and $\text{conv}(K_2)$. Furthermore, we enforce that we always append a triangle to the triangulation that is the least advanced of the two (in clockwise order), unless we have already reached the last triangle of $\text{conv}(K_1)$. This last condition is important to allow us to define the arc lengths in a way that corresponds to the area of the union of the two convex polygons. Figure 4 illustrates the result of following a path on the DAG.

The edge e_1 (respectively, e_2) from the problem input defines two halfplanes, one on each side. Let H_1 (resp. H_2) be the halfplane that contains K_1 (resp. K_2). We have not yet determined K_1 or K_2, but all four possibilities of halfplanes may be tried independently. From now on, we only consider the $O(n)$ points of S lying in the region $H_1 \cup H_2$. Let $p_1, \ldots, p_n$ be the points of S sorted clockwise around ρ, breaking ties arbitrarily. The edge e_1 (resp. e_2) has p_1 (resp. p_n) as a vertex. We define the indices $a < b$ such that $e_1 = (p_1, p_b), e_2 = (p_a, p_n)$ (Figure 3).

We are now ready to define the set E of arcs of the DAG. There are three types of arcs. The *type-0* arcs start from the initial node v_0 to $(\triangle_1, \triangle_2)$ if p_1 is a vertex of $\triangle_1$ and p_a a vertex of $\triangle_2$. These two triangles of vertices ρ, p_1, p_j with $j > 1$ and ρ, p_a, p_v with $v > a$ are respectively bounded by the edges e_1 and e_2. They initialize the triangulations of our two polygons $\text{conv}(K_1)$ and $\text{conv}(K_2)$. There are $O(n^2)$ type-0 arcs.

A *type-1* arc corresponds to advancing the triangulation of $\text{conv}(K_1)$, while a *type-2* arc corresponds to advancing the triangulation of $\text{conv}(K_2)$. There are $O(n)$ type-1, 2 arcs coming out of each node. A *type-1* arc goes from $(\triangle_1, \triangle_2)$ to $(\triangle_3, \triangle_2)$ if:

- the quadrilateral $\triangle_1 \cup \triangle_3$ is convex,
- $\triangle_1$ has vertices ρ, p_i, p_j with $i < j < b$,
- $\triangle_2$ has vertices ρ, p_u, p_v with $a \leq u < v$,
- $\triangle_3$ has vertices ρ, p_j, p_k with $j < k \leq b$,
- and $j \leq v$.

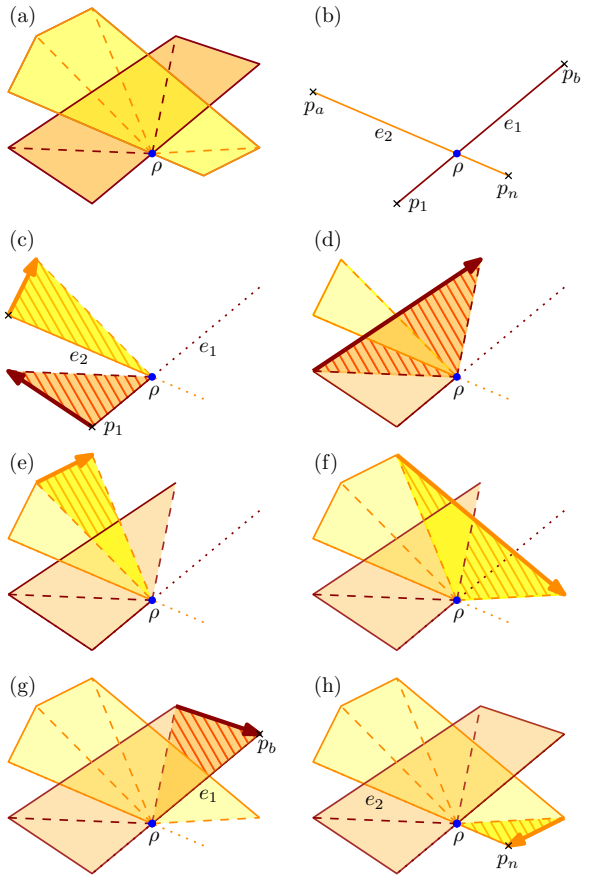


Figure 4: Steps of the algorithm from Section 3.2. Figure (a) represents the solution, while Figures (b) to (h) represent the triangulation obtained at each node of a path. The newly covered area that is assigned as the length of the corresponding arc is marked. In (b), we have the initial pair of edges e_1, e_2 which corresponds to the starting vertex v_0. After following a type-0 arc, a first pair of triangles with vertices p_1 and p_a is obtained in (c). The triangle $\triangle_1$ is brown and triangle $\triangle_2$ yellow. From (c) to (d), we follow a type-1 arc. The triangle $\triangle_1$ (less advanced than triangle $\triangle_2$) advances. From (d) to (e), we follow a type-2 arc, since triangle $\triangle_2$ is less advanced. From (e) to (f) we have again a type-2 arc, and from (f) to (g) we have a type-1 arc. In (g), the triangle $\triangle_1$ has reached the final node p_b and cannot advance anymore. We have only type-2 arcs to follow until $\triangle_2$ reaches p_n, at a node in V_1.

Similarly, there is a type-2 arc from $(\triangle_1, \triangle_2)$ to $(\triangle_1, \triangle_4)$ if:

- the quadrilateral $\triangle_2 \cup \triangle_4$ is convex,
- $\triangle_1$ has vertices ρ, p_i, p_j with $i < j \leq b$,
- $\triangle_2$ has vertices ρ, p_u, p_v with $a \leq u < v$,
- $\triangle_4$ has vertices ρ, p_v, p_w with $v < w$,

- and either $v \leq j$ or $j = b$.

The length of each arc corresponds to the area of the new region covered by appending a new triangle by following the arc. Therefore, the length of a type-0 arc from v_0 to $(\triangle_1, \triangle_2)$ is the area of $\triangle_1 \cup \triangle_2$. The length of a type-1 arc from $(\triangle_1, \triangle_2)$ to $(\triangle_3, \triangle_2)$ is defined as the area of $\triangle_3 \setminus \triangle_2$. Similarly, the length of a type-2 arc from $(\triangle_1, \triangle_2)$ to $(\triangle_1, \triangle_4)$ is defined as the area of $\triangle_4 \setminus \triangle_1$.

We define a set of *end* nodes V_1 as follows. A node $(\triangle_1, \triangle_2)$ is an end node if p_b is a vertex of $\triangle_1$ and p_n is a vertex of $\triangle_2$. The construction of the DAG allows us to prove the following lemma.

Lemma 3 *There is a bijection between the directed paths of the DAG (V, E) (starting from v_0 and ending in V_1) and the digital convex sets $K_1, K_2 \subset S$ such that e_1 is an edge of $\mathrm{conv}(K_1)$ and e_2 is an edge of $\mathrm{conv}(K_2)$. Furthermore, the length of each path is equal to the corresponding area of $\mathrm{conv}(K_1) \cup \mathrm{conv}(K_2)$. (We assume that K_1 (resp. K_2) lie above the supporting line of e_1 (resp. e_2).)*

Proof. First we show that the existence of two digital convex sets $K_1, K_2 \subset S$ as in the lemma statement implies the existence of a directed path in the DAG as in the lemma statement. Let K_1 (resp. K_2) be two convex sets lying above the supporting line of e_1 (resp. e_2). Both $\mathrm{conv}(K_1)$ and $\mathrm{conv}(K_2)$ contain ρ as a boundary point and hence can be triangulated from ρ. It is easy to see that there is a path corresponding to this triangulation. Next, we show that the converse also holds.

The definition of the arcs is such that advancing through one of them adds a triangle to one of the two polygons while preserving convexity, which ensures that all paths correspond to convex polygons. Furthermore, the starting node ensures that the two convex polygons respectively start from p_1 and p_a, while the set of ending nodes ensure that the two convex polygons respectively end at p_b and p_n. Hence all paths from v_0 to V_1 correspond to two convex polygons that fit the lemma statement, one from edge $e_1 = p_1, p_b$ and one from edge $e_2 = p_a, p_n$. The validity test on each triangle ensures that the paths describes digital convex sets.

The definition of the arcs enforces that we only move forward the least advanced triangle, that is the triangle that has the minimum maximum index among its vertices. The only exception is when $\mathrm{conv}(K_1)$ is completed, that is the triangle with vertex p_b has been added to its triangulation. This ensures that the new area covered by a type-1, 2 arc is simply the set theoretic difference of two triangles (instead of a triangle and an arbitrary convex object). As the length of the arcs is defined as the area of the difference of the two triangles, the total length of the path is equal to the area of the union of the two convex polygons. Hence each path from v_0 to V_1 describe two digital convex sets $K_1, K_2 \in S$ such that e_1 is an edge of $\mathrm{conv}(K_1)$ and e_2 is an edge of $\mathrm{conv}(K_2)$, and the length of each path is equal to the corresponding area of $\mathrm{conv}(K_1) \cup \mathrm{conv}(K_2)$. □

Theorem 4 *There exists an algorithm to solve Problem 2 (digital 2-potato peeling) in $O(n^9 + n^6 \operatorname{polylog} r)$ time, where n is the number of input points and r is the diameter of the input.*

Proof. As explained in Section 3.1, solving the disjoint case (Problem 5) takes $O(n^5 + n^4 \log r)$ time. Next, we show how to solve the rooted intersecting case (Problem 7) in $O(n^5 + n^2 \operatorname{polylog} r)$ time, proving the theorem.

Assume without loss of generality that K_1, K_2 are respectively above the supporting lines of e_1, e_2 (all four possibilities may be tried independently). Our algorithm starts by computing the DAG (V, E) with $O(n^4)$ nodes, each representing a pair of triangles. Since each node has at most $O(n)$ incoming arcs, the number of arcs is $O(n^5)$. Hence the longest path can be found in $O(n^5)$ time.

To build the set of nodes V, we need to test the validity of $O(n^2)$ triangles. Since ρ may not be a lattice point, Pick's theorem [32] cannot be used. Still, ρ is a rational point with denominators bounded by $O(r^2)$. Hence, we can use either the formulas from Beck and Robins [7] or the algorithm from Barvinok [5] to calculate the number of lattice points $|T \cap \mathbb{Z}^2|$ inside each triangle T in $O(\operatorname{polylog} r)$ time. As in Section 2, we compute $|T \cap S|$ using a triangle range counting query, which takes $O(\log n)$ time after preprocessing S in $O(n^2)$ time [14]. The triangle is valid if and only if $|T \cap \mathbb{Z}^2| = |T \cap S|$. The two steps to test the validity of a triangle take $O(\operatorname{polylog} r)$ and $O(\log n)$ time. Since the diameter r of n lattice points is $\Omega(\sqrt{n})$, the dominating term is $O(\operatorname{polylog} r)$. Hence, we test the validity of each triangle in $O(\operatorname{polylog} r)$ time, which gives a total time of $O(n^2 \operatorname{polylog} r)$ to build the list of valid triangles required to build V.

Consequently, we solve Problem 7 in $O(n^5 + n^2 \operatorname{polylog} r)$ time. To obtain a solution to Problem 2, we note that there are $O(n^2)$ candidates for the edge e_1, as well as for the edge e_2. Testing all $O(n^4)$ possible edges e_1, e_2, we achieve the claimed running time of $O(n^9 + n^6 \operatorname{polylog} r)$ time. □

4 From Digital to Continuous

In this section, we show that the exact algorithms for the digital potato-peeling problem and the digital 2-potato-peeling problem can be used to compute an approximation of the respective continuous problems with an arbitrarily small approximation error. For simplicity, we

focus on the potato-peeling problem, but the 2-potato-peeling case is analogous. We note that the reduction presented here does not lead to efficient approximation algorithms and is presented only to formally connect the continuous and digital versions of the problem.

Problem 8 (Continuous potato-peeling) *Given a polygon P (that may have holes) of n vertices, determine the largest convex polygon $K \subseteq P$, where largest refers to the area of K.*

We start with some definitions. Let K_C be the polygon of the optimal solution to the continuous problem above and A_C be the area of K_C. Given an approximation parameter $\varepsilon > 0$, we show how to obtain a set of lattice points $S \subseteq P$ such that the area A_D of the convex hull of the solution K_D of Problem 1 with input S satisfies $|A_C - A_D| = O(r\varepsilon)$. In this section, we use lattice points that are not integers, but points with coordinates that are multiples of ε. Let Λ_ε denote the set of all points with coordinates that are multiple of ε. Of course, a uniform scaling maps Λ_ε to the integer lattice used in the remainder of the paper, and hence the integer lattice algorithms also apply to Λ_ε.

For a polygon P, the *erosion* of P, denoted P^- is the subset of P formed by points within L_∞ distance at least 2ε of all points outside P (Figure 5(a)). Let A_C^- be the area of the optimal solution to the continuous potato-peeling problem with input P^-.

We only give here the main directions of the proof. A more detailed version of the proof can be found in the appendix 5. First, by bounding the number of lattice cells that a convex curve of a given length can cross, we bound by $O(r\varepsilon)$ the area difference between any convex polygon and the convex hull of the lattice points inside it. Then, we use an erosion of 2ε in order to smooth the input and avoid difficulties related to comb like input polygons. We bound the area difference by $O(r\varepsilon)$ between the solution of problem 1 for any polygon and the solution for the erosion of this polygon. Finally, despite the digital solution being potentially outside the input polygon, it can be shown that the area lying outside the input polygon is bounded by $O(r\varepsilon)$ which gives us the following theorem:

Theorem 5 *Let A_C be the area of the solution K_C of Problem 8 with input polygon P of diameter r. Let $\varepsilon > 0$ be a parameter and $S = \Lambda_\varepsilon \cap P^-$, where P^- is the erosion of P by 2ε and Λ_ε is the lattice of size ε. The area A_D of the convex hull of the solution K_D of Problem 1 with input S satisfies $|A_C - A_D| = O(r\varepsilon)$.*

The polygon $\mathrm{conv}(K_D)$ in the previous theorem may partially extend outside P. Nevertheless, the solution K_D of Problem 1 can be used to obtain a convex polygon $K \subseteq P$ which has an area A satisfying $|A_C - A_D| = O(r\varepsilon)$.

5 Conclusion and Open Problems

The (continuous) potato peeling problem is a very peculiar problem in computational geometry. The fastest algorithms known have running times that are polynomials of substantially high degree. Also, we are not aware of any algorithms (or difficulty results) for the natural extensions to higher dimensions (even 3d) or to a fixed number of convex bodies.

In this paper, we focused on a digital version of the problem. Many problems in the intersection of digital, convex, and computational geometry remain open. Our study falls in the following framework of problems, all of which receive as input a set of n lattice points $S \subset \mathbb{Z}^d$ for constant d and are based on a fixed parameter $k \geq 1$.

1. Is S the union of at most k digital convex sets?
2. What is the smallest superset of S that is the union of at most k digital convex sets?
3. What is the largest subset of S that is the union of at most k digital convex sets?

In [15], the authors considered the first problem for $k = 1$, presenting polynomial time solutions (which may still leave room for major improvements for $d > 3$). We are not aware of any previous solutions for $k > 1$. In contrast, the continuous version of the problem is well studied. The case of $k = 1$ can be solved easily by a convex hull computation or by linear programming. Polynomial algorithms are known for $d = 2$ and $k \leq 3$ [8, 34], as well as for $d = 3$ and $k \leq 2$ [9]. The problem is already NP-complete for $d = k = 3$ [9]. Hence, the continuous version remains open only for $d = 2$ and fixed $k > 3$.

It is easy to obtain polynomial time algorithms for the second problem when $k = 1$, since the solution consists of all points in the convex hull of S. The continuous version for $d = k = 2$ can be solved in $O(n^4 \log n)$ time [3]. Also, the orthogonal version of the problem is well studied (see for example [21]). We know of no results for the digital version.

In this paper, we considered the digital version of the third problem for $d = 2$ and $k = 1, 2$, presenting algorithms with respective running times of $O(n^3 + n^2 \log r)$ and $O(n^9 + n^6 \operatorname{polylog} r)$, where r is the diameter of S. Since the first problem trivially reduces to the third problem, we also solved the first problem for $k = d = 2$ in $O(n^9 + n^6 \operatorname{polylog} r)$ time. It is surprising that we are not aware of any faster algorithm for the first problem in this particular case.

The third problem for $d > 2$ or $k > 2$ remains open. The DAG approach that we used for $d = 2$ is unlikely to generalize to higher dimensions, since there is no longer a single order by which to transverse the boundary of a convex polytope. Surprisingly, even the continuous version seems to be unresolved for $d > 2$ or $k \geq 2$.

References

[1] Boris Aronov, Marc Van Kreveld, Maarten Löffler, and Rodrigo I. Silveira. Peeling meshed potatoes. *Algorithmica*, 60(2):349–367, 2011.

[2] Charles Audet, Pierre Hansen, and Frédéric Messine. Extremal problems for convex polygons. *Journal of Global Optimization*, 38(2):163–179, 2007.

[3] Sang Won Bae, Hwan-Gue Cho, William Evans, Noushin Saeedi, and Chan-Su Shin. Covering points with convex sets of minimum size. *Theoretical Computer Science*, 718:14–23, 2018.

[4] Elena Barcucci, Alberto Del Lungo, Maurice Nivat, and Renzo Pinzani. Reconstructing convex polyominoes from horizontal and vertical projections. *Theoretical Computer Science*, 155(2):321–347, 1996.

[5] Alexander I. Barvinok. A polynomial time algorithm for counting integral points in polyhedra when the dimension is fixed. *Mathematics of Operations Research*, 19(4):769–779, 1994.

[6] Crevel Bautista-Santiago, José Miguel Díaz-Báñez, Dolores Lara, Pablo Pérez-Lantero, Jorge Urrutia, and Inmaculada Ventura. Computing optimal islands. *Operations Research Letters*, 39(4):246–251, 2011.

[7] Matthias Beck and Sinai Robins. Explicit and efficient formulas for the lattice point count in rational polygons using Dedekind-Rademacher sums. *Discrete & Computational Geometry*, 27(4):443–459, Jan 2002.

[8] Patrice Belleville. On restricted boundary covers and convex three-covers. In *5th Canadian Conference on Computational Geometry (CCCG)*, pages 467–472, 1993.

[9] Patrice Belleville. Convex covers in higher dimensions. In *7th Canadian Conference on Computional Geometry (CCCG)*, pages 145–150, 1995.

[10] Gunilla Borgefors and Robin Strand. An approximation of the maximal inscribed convex set of a digital object. In *13th International Conference on Image Analysis and Processing (ICIAP)*, pages 438–445, 2005.

[11] Sergio Cabello, Josef Cibulka, Jan Kyncl, Maria Saumell, and Pavel Valtr. Peeling potatoes near-optimally in near-linear time. *SIAM Journal on Computing*, 46(5):1574–1602, 2017.

[12] Jyun S. Chang and Chee K. Yap. A polynomial solution for the potato-peeling problem. *Discrete & Computational Geometry*, 1(2):155–182, 1986.

[13] Jean-Marc Chassery and David Coeurjolly. Optimal shape and inclusion: open problems. In *Mathematical Morphology: 40 Years On, International Symposium on Mathematical Morphology*, Computational Imaging and Vision. Springer Verlag, 2005.

[14] Bernard Chazelle, Micha Sharir, and Emo Welzl. Quasi-optimal upper bounds for simplex range searching and new zone theorems. *Algorithmica*, 8(1-6):407–429, 1992.

[15] Loïc Crombez, Guilherme D. da Fonseca, and Yan Gérard. Efficient algorithms to test digital convexity. In *21st International Conference on Discrete Geometry for Computer Imagery (DGCI)*, 2019. URL: `http://fc.isima.fr/~fonseca/digitalconvexity.pdf`.

[16] Joseph Culberson and Robert A. Reckhow. Covering polygons is hard. *Journal of Algorithms*, pages 17:2–44, 1994.

[17] Alain Daurat. Salient points of q-convex sets. *International Journal of Pattern Recognition and Artificial Intelligence*, 15(7):1023–1030, 2001.

[18] Adrian Dumitrescu, Sariel Har-Peled, and Csaba D. Tóth. Minimum convex partitions and maximum empty polytopes. In Fedor V. Fomin and Petteri Kaski, editors, *Algorithm Theory – SWAT 2012*, pages 213–224, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[19] Mousumi Dutt, Arindam Biswas, Partha Bhowmick, and Bhargab B. Bhattacharya. On finding an orthogonal convex skull of a digital object. *International Journal of Imaging Systems and Technology*, 21(1):14–27, 2011.

[20] David Eppstein, Mark Overmars, Günter Rote, and Gerhard Woeginger. Finding minimum area k-gons. *Discrete & Computational Geometry*, 7(1):45–58, 1992.

[21] Cem Evrendilek, Burkay Genç, and Brahim Hnich. Covering points with minimum/maximum area orthogonally convex polygons. *Computational Geometry*, 54:32–44, 2016.

[22] Paul Fischer. Sequential and parallel algorithms for finding a maximum convex polygon. *Computational Geometry*, 7:187–200, 02 1997.

[23] Yan Gérard, Antoine Vacavant, and Jean-Marie Favreau. Tight bounds in the quadtree complexity

theorem and the maximal number of pixels crossed by a curve of given length. *Theoretical Computer Science*, 624:41–55, 2016.

[24] Jacob E. Goodman. On the largest convex polygon contained in a non-convex n-gon, or how to peel a potato. *Geometriae Dedicata*, 11(1):99–106, 1981.

[25] Peter M. Gruber. Geometry of numbers. In *Handbook of Convex Geometry, Part B*, pages 739–763. Elsevier, 1993.

[26] Olaf Hall-Holt, Matthew J. Katz, Piyush Kumar, Joseph S.B. Mitchell, and Arik Sityon. Finding large sticks and potatoes in polygons. In *17th annual ACM-SIAM symposium on Discrete algorithm (SODA)*, pages 474–483, 2006.

[27] Chul E. Kim and Azriel Rosenfeld. Digital straight lines and convexity of digital regions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 4(2):149–153, 1982.

[28] Reinhard Klette and Azriel Rosenfeld. *Digital geometry: Geometric methods for digital picture analysis*. Elsevier, 2004.

[29] Joseph O'Rourke. The complexity of computing minimum convex covers for polygons. In *20th Allerton Conference on Communication, Control, and Computing*, pages 75–84, 1982.

[30] Joseph O'Rourke. The decidability of covering by convex polygons. Technical Report JHU-EECS 82-4, Johns Hopking University, 1982.

[31] Joseph O'Rourke. Some NP-hard polygon decomposition problems. *IEEE Transactions on Information Theory, IT-30*, pages 181–190, 1983.

[32] Georg Pick. Geometrisches zur zahlenlehre. *Sitzungsberichte des Deutschen Naturwissenschaftlich-Medicinischen Vereines für Böhmen "Lotos" in Prag.*, v.47-48 1899-1900, 1899.

[33] Robert Sedgewick and Kevin Wayne. *Algorithms*. Addison-Wesley Professional, 2011.

[34] Thomas C. Shermer. On recognizing unions of two convex polygons and related problems. *Pattern Recognition Letters , 14(9)*, pages 737–745, 1993.

[35] Tony C. Woo. The convex skull problem. Technical report, Department of Industrial and Operations Engineering, University of Michigan, 1986.

[36] Derick Wood and Chee K. Yap. The orthogonal convex skull problem. *Discrete & Computational Geometry*, 3(4):349–365, 1988.

Appendix

6 From Digital to Continuous

The *width* of P is the minimum distance between two parallel lines ℓ_1, ℓ_2 such that P is between ℓ_1 and ℓ_2.

The following lemma that bounds the area difference between a convex polygon and the convex hull of its intersection with a lattice set will be useful to our proof.

Lemma 6 *Let C be a convex polygon of diameter r. The convex hull $H = \mathrm{conv}(C \cap \Lambda_\varepsilon)$ satisfies*

$$\mathrm{area}(C) \leq \mathrm{area}(H) + 6\sqrt{2}\pi r\varepsilon + 16\varepsilon^2.$$

Proof. The lattice Λ_ε induces a grid with vertex set Λ_ε and square cells of side length ε. Let X^- be the set of grid cells that are completely contained in C and X^∂ be the set of cells that are partially contained in C. All cells in X^∂ intersect the boundary ∂C of C.

Since the perimeter of a convex shape is at most π times its diameter [2], the perimeter of ∂C is at most πr. Since a curve of perimeter p intersects at most $3p/\varepsilon\sqrt{2} + 4$ grid cells of side length ε [23], we have $|X^\partial| \leq 3\pi r/\varepsilon\sqrt{2} + 4$.

All cells in X^- are contained in H and C is covered by $X^- \cup X^\partial$. Therefore, the area of $C \setminus H$ is at most the area in X^∂, which is

$$\varepsilon^2|X^\partial| \leq 4\varepsilon^2 \cdot \left(\frac{3}{\varepsilon\sqrt{2}}\pi r + 4\right) = 6\sqrt{2}\pi r\varepsilon + 16\varepsilon^2,$$

proving the lemma. □

The following lemma bounds the area difference between the optimal solutions of the continuous potato peeling problem with inputs P and P^-.

Lemma 7 *Let P be a polygon of diameter r and P^- be the erosion of P. Let C (resp. C') denote the largest convex polygon inside P (resp. P^-). We have the following inequality:*

$$\mathrm{area}(C) \leq \mathrm{area}(C') + 2\sqrt{2}\pi r\varepsilon.$$

Proof. The erosion C^- of C is a convex polygon that lies inside P^-. Hence the area of C' is at least as large as the area of C^-.

As C is a convex polygon of diameter at most r, the perimeter of C is at most πr. As every eroded points from C in order to obtain C^- are inside C and at a maximum distance of $2\sqrt{2}\varepsilon$ of the boundary of C, they are all included inside a set of rectangles that lie inside C with the edges of C as sides and width $2\sqrt{2}\varepsilon$. Hence, the area difference between C and its erosion is at most $2\sqrt{2}\varepsilon\pi r$, which proves the lemma. □

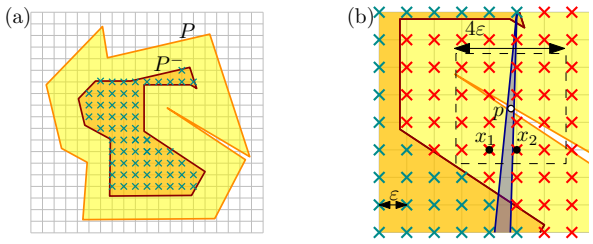


Figure 5: (a) A polygon P, its erosion P^-, and the set $\Lambda_\varepsilon \cap P^-$. (b) To include the point p that is outside P, $\mathrm{conv}(K_D)$ has to go between the lattice points within L_∞ distance 2ε of p.

The digital solution may have portions that lie outside the input polygon P of the continuous version. However, this portion cannot be too big, as shown in the following lemma.

Lemma 8 *Let P be a polygon of diameter r and P^- be the erosion of P. Let $S = P^- \cap \Lambda_\varepsilon$, and K_D be the largest digital convex subset of S. The following inequality holds:*

$$\mathrm{area}(\mathrm{conv}(K_D) \setminus P) \leq 2r\varepsilon.$$

Proof. Let p be a point in $\mathrm{conv}(K_D) \setminus P$. As S is included inside P^-, all the lattice points within L_∞ distance ε of p are not in S (see Figure 5). All 16 lattice points at a L_∞ distance less than 2ε of p are not in S. Hence, in order to include p, $\mathrm{conv}(K_D)$ has to lie between two vertically (or horizontally) consecutive lattice points x_1 and x_2, which are separated by distance ε. Furthermore p is at a horizontal (or vertical) distance strictly greater than ε from x_1 and x_2. The widest angle the incoming and outgoing edges of C can form is hence $2\arctan(1/2)$, effectively forming a turning angle of at least $\pi - 2\arctan(1/2)$. As the sum of turning angles inside a convex polygon is equal to 2π and can never decrease, and as $\pi - 2\arctan(1/2) > 2\pi/3$ such a turning angle can only happen twice. Also, as in order to include any point p outside of P, $\mathrm{conv}(K_D)$ has to go in between x_1 and x_2, the width of this (possible non-contiguous region) including p is at most ε and the diameter at most r, hence, the area is bounded by $r\varepsilon$. Therefore, there can be no more than two such regions in $\mathrm{conv}(K_D)$ (even though each of them can enter and leave P multiple times), which proves the lemma. □

Using lemma 7, it follows that $A_C - A_C^- \leq 2\sqrt{2}\pi r\varepsilon$. Lemma 6 gives us that $A_C^- - A_D \leq 6\sqrt{2}\pi r\varepsilon + 16\varepsilon^2$. Lemma 8 gives us that $A_D - 2r\varepsilon \leq A_C$. Hence,

$$A_C - 8\sqrt{2}\pi r\varepsilon - 16\varepsilon^2 \leq A_D \leq A_C + 2r\varepsilon,$$

proving the following theorem.

Theorem 9 *Let A_C be the area of the solution K_C of Problem 8 with input polygon P of diameter r. Let $\varepsilon > 0$ be a parameter and $S = \Lambda_\varepsilon \cap P^-$, where P^- is the erosion of P by 2ε and Λ_ε is the lattice of size ε. The area A_D of the convex hull of the solution K_D of Problem 1 with input S satisfies $|A_C - A_D| = O(r\varepsilon)$.*

The polygon $\mathrm{conv}(K_D)$ in the previous theorem may partially extend outside P. Nevertheless, the solution K_D of Problem 1 can be used to obtain a convex polygon $K \subseteq P$ which has an area A satisfying $|A_C - A_D| = O(r\varepsilon)$.

The same proof strategy can be applied to obtain an approximation to the continuous version of the 2-potato-peeling problem using the digital version of the problem.

Largest Triangle inside a Terrain

Arun Kumar Das* Sandip Das† Joydeep Mukherjee ‡

Abstract

In this paper, we present an $O(n^2)$ time algorithm to find a largest area triangle contained inside a terrain with n vertices. We also present an $O(n \log n)$ time $\frac{1}{2}$-factor approximation for the same.

1 Introduction

Many well-known algorithms for geometric optimization problems on arbitrary sets often involve finding some other set with some nice properties such that it also approximates the original set very closely. Designing algorithms for the problems on these subsets become easy which lead to good approximate solution for the original problems. These arbitrary sets often come in the form of polygons. A well explored direction in this regard is to come up with polygons with certain desired properties such as convexity, bounded number of sides, etc. which is supposed to approximate the given polygon sufficiently closely. The closeness of approximation is often guided by optimization of some parameters like area, perimeter, etc. associated with the computed polygon.

A rich body of work has been devoted towards computing maximum area enclosed convex polygons in a given polygon. Finding a largest area convex polygon contained in a simple polygon was posed by Jacob E. Goodman [9]. Chang et al. [6] came up with a polynomial time solution for the same. In fact finding largest area triangle within a given convex polygon has been quite well studied by Boyce et al. [2], Dobkin and Snyder [8]. Later Chandran and Mount [5] has shown a parallel algorithm for enclosing and enclosed triangles in a given convex polygon. Melissaratos and Souvaine [15] came up with an $O(n^4)$ time exact algorithm for computing a largest area triangle inside a simple polygon. They also proposed in the same paper an $O(n^3)$ time exact algorithm for computing a largest triangle *inscribed* in a simple polygon. Note that an *inscribed* triangle in a simple polygon is a triangle, which is contained in the polygon and whose vertices lie on the boundary of the polygon. Hall-Holt et al. [11] studied the approximate version of these problems providing an $O(n \log^2 n)$ polynomial time approximation scheme (PTAS) for the biggest stick problem and an $O(n)$ PTAS for the largest area *fat* triangle problem contained in a simple polygon. In the same paper they also provide an $O(1)$ approximation for computing a largest area triangle or convex polygon in a simple polygon. Later on Cabello et al. [4] provided an $O(n(\log^2 n + \frac{1}{\varepsilon^3} \log n + \frac{1}{\varepsilon^4}))$ time PTAS for largest area convex polygon contained in a simple polygon. Cabello et al. [3] provided an $O(n^3)$ exact algorithm for computing a maximum area and maximum perimeter rectangle contained in a convex polygon respectively. They also provided $(1-\varepsilon)$ approximation algorithm for the same which takes $O(\frac{1}{\varepsilon^{\frac{3}{2}}} + \frac{1}{\varepsilon^{\frac{1}{2}}} \log(n))$.

These polygon inclusion problems are motivated by applications in computer graphics for occlusion culling [11]. The enclosure problems find application in stock cutting, collision avoidance [6]. Bose and De Carufel [1] studied the problem of minimum area enclosing triangle with a fixed angle.

In this paper, we find a largest area triangle contained in a terrain. Terrain is a geometric object bounded by a base and a polyline, called the upper boundary, such that any line perpendicular to the base cuts the upper boundary exactly once. We consider here the base lying on the positive x-axis, so our upper boundary is an x-monotone curve in $\mathcal{R}^2$. Figure 1 shows a terrain. Note that there may be more than one triangle with the same largest area, any of them suffices as our output. The vertices defining a largest area triangle contained in a convex polygon are subset of the vertices of the given convex polygon as proved in Keikha et al. [13]. Note that this observation does not hold in a terrain. We also assume that no three vertices of the input terrain are collinear.

2 Preliminaries

Definition 1 *A vertex of a terrain is called a convex vertex if the internal angle between the sides adjacent to this vertex is less than* $180°$*. If this angle is greater than* $180°$ *then it is called a reflex vertex. The endpoints of the base of the terrain are called base vertices. The vertex at the left end of the base is called the left base vertex and denoted by* B_l *and the other base vertex is called the right base vertex and denoted by* B_r*.*

*Advanced Computing and Microelectronics Unit, Indian Statistical Institute, Calcutta, India, arund426@gmail.com

†Advanced Computing and Microelectronics Unit, Indian Statistical Institute, Calcutta, India, sandipdas@isical.ac.in

‡Advanced Computing and Microelectronics Unit, Indian Statistical Institute, Calcutta, India, joydeep.m1981@gmail.com

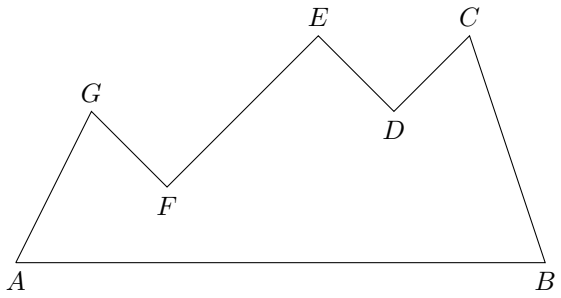


Figure 1: A Terrain

Note that the base vertices are convex vertices.

Observation 1 *For a convex vertex P of a terrain, if a line segment $\overline{PR}$ from P to base of the terrain completely lies inside the terrain, then $\overline{PR}$ does not pass through any convex vertex of the terrain other than the base vertices.*

Now we recall a result from basic Euclidean geometry.

Result 1 *Let $\triangle PQR$ be a triangle and M be the mid-point of $\overline{PQ}$. If a line parallel to $\overline{QR}$ passing through M cuts $\overline{PR}$ at N, then N is mid point of $\overline{PR}$ and length of $\overline{MN}$ is half of the length of $\overline{QR}$.*

3 Properties of maximum area triangle inside terrain

In this section we present some interesting observations about maximum area triangle contained in terrain.

Lemma 1 *One side of a largest area triangle contained in a terrain must coincide with the base of the terrain.*

Proof. We prove this Lemma by showing that for any given triangle inside a terrain there exists another triangle with a side coinciding with the terrain base having area greater or equal to the given triangle.

Let $\triangle PQR$ be any triangle lying inside the terrain and AB be the terrain base. The nearest vertex among P, Q, R with respect to the terrain base $\overline{AB}$ is considered as a lowest vertex of $\triangle PQR$. The orientation of $\triangle PQR$ can happen in four possible ways:

1. The base of $\triangle PQR$ is parallel to the terrain base, i.e. it has two lowest vertices
2. $\triangle PQR$ has only one lowest vertex and the edges adjacent to it have slopes of different sign
3. $\triangle PQR$ has only one lowest vertex and the edges adjacent to it have slopes of same sign
4. $\triangle PQR$ has an edge perpendicular to the terrain base

Case 1: Suppose P and Q are two lowest vertices of $\triangle PQR$, hence $\overline{PQ}$ is parallel to the base of the terrain

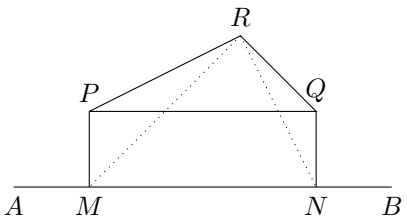


Figure 2: Case 1

$\overline{AB}$. We draw perpendiculars from P and Q, namely $\overline{PM}$ and $\overline{QN}$ on $\overline{AB}$, then $\triangle MNR$ is inside the terrain and $area(\triangle MNR) > area(\triangle PQR)$ (Figure 2).

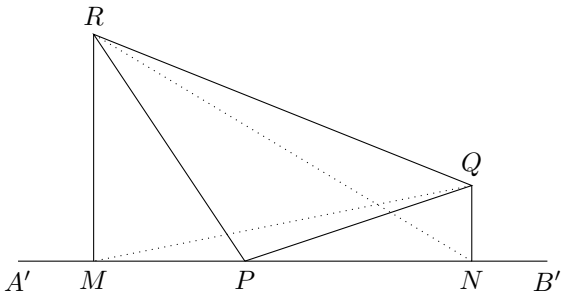


Figure 3: Case 2

Case 2: If $\triangle PQR$ has a unique lowest vertex say P and $\overline{PQ}$ and $\overline{PR}$ have slopes of opposite signs (Figure 3). Then we draw a line $\overline{A'B'}$ parallel to the terrain base and passing through P inside the terrain. Now the perpendiculars from Q, i.e. $\overline{QN}$ and R, i.e. $\overline{RM}$ also reach $\overline{A'B'}$ without crossing the boundary of the terrain. Without loss of generality let $\overline{RM} > \overline{QN}$, then $\overline{RQ}$ is a line of negative slope. As the height of points on a line of negative slope is a decreasing function with respect to x-coordinate, we have, $area(\triangle MNR) = area(\triangle RMQ) > area(\triangle PQR)$. If $\overline{RM} = \overline{QN}$, then $area(\triangle MNR) = area(\triangle RMQ) = area(\triangle PQR)$. Now we can treat $\triangle MNR$ or $\triangle MNQ$ by Case 1.

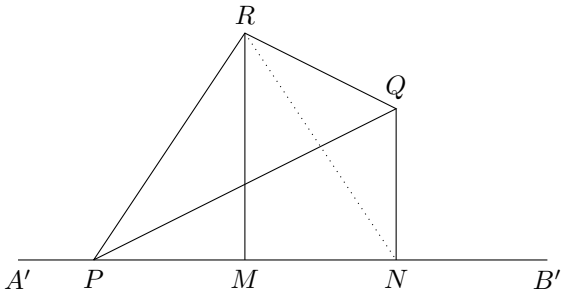


Figure 4: Case 3

Case 3: In this case, $\triangle PQR$ has a unique lowest vertex P and $\overline{PQ}$ and $\overline{PR}$ have slopes of same sign.

Without loss of generality, we assume that $\overline{PQ}$ and $\overline{PR}$ both have positive slopes (Figure 4) and the slope of $\overline{PR}$ is greater than slope of $\overline{PQ}$. We draw $\overline{A'B'}$, $\overline{RM}$ and $\overline{RN}$ as defined in Case 2. Then $\overline{RM}$ must intersect the line segment $\overline{PQ}$, otherwise $\triangle PQR$ lies inside $\triangle PMR$. Now, in this case also $area(\triangle PNR) > area(\triangle PQR)$. As $\overline{PR}$ has positive slope, the perpendicular from Q on $\overline{PR}$ (extended) is shorter than the perpendicular from N on $\overline{PR}$ (extended). Then the result is true using the arguments of Case 1.

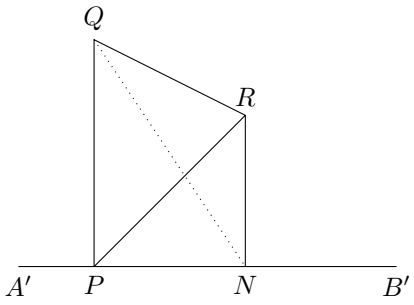


Figure 5: Case 4

Case 4: $\triangle PQR$ has the unique lowest vertex P and either $\overline{PR}$ or $\overline{PQ}$ is perpendicular to $\overline{AB}$. Let us assume PQ is perpendicular to $\overline{AB}$ (Figure 5). Then we draw $\overline{RN}$, perpendicular from R on $\overline{A'B'}$(as defined earlier). Then $area(\triangle PQN) = area(\triangle PQR)$.

Note that these arguments hold even if the lowest vertex P touches the base of the terrain. In that case $\overline{A'B'}$ coincides with the base. Hence the result follows. □

Observation 2 *If the base of an optimal triangle coincides with the terrain base and α and β be the internal angles adjacent to the base of the triangle, then $0° < \alpha \leq 90°$ and $0° < \beta \leq 90°$.*

Proof. Without loss of generality if $\alpha > 90°$, we can draw a perpendicular on the terrain base from the top most vertex of a triangle to obtain a larger triangle, which is a contradiction. □

In an optimal triangle one side is coinciding with the base of the terrain and that side is termed as the *base* of the optimal triangle. The side which has a positive slope with the base of the terrain is expressed as the *left side* of the optimal triangle. Similarly the side which has negative slope with the base of the terrain is expressed as the *right side* of the optimal triangle.

Lemma 2 *Each of the left and right sides of an optimal triangle is a segment of a line which passes through two vertices of the terrain.*

Proof. We prove the Lemma by showing that any line passing through only one vertex of the terrain cannot contribute a segment as a side of an optimal triangle, which has its base lying on the terrain base. We will show that in such cases we can rotate this line and obtain a larger triangle. The base of an optimal triangle lies on the base of the terrain (by Lemma 1). Now if a side of the triangle coincides with a line that passes through only one vertex, which is a convex one, then this vertex must be one of the vertices of this triangle. In such a case we simply rotate the side passing through that convex vertex to increase the area of the considered triangle.

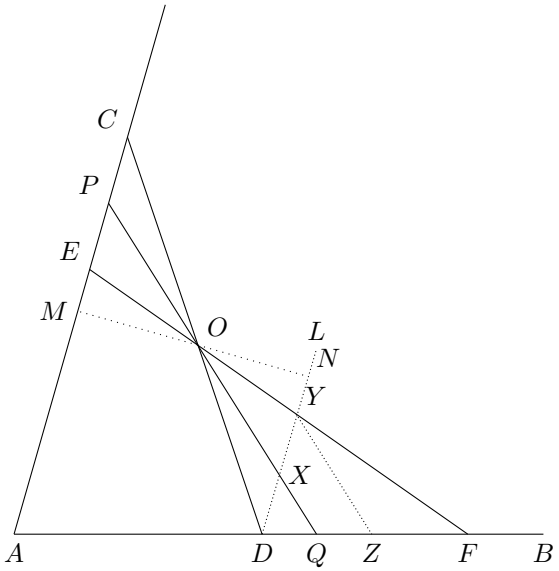


Figure 6: Lines passing through two vertices give optimal solution

Now we consider when the side $\overline{PQ}$ of $\triangle AQP$ passes through only one reflex vertex O of the terrain (Figure 6). We show that $\triangle AQP$ cannot be an optimal triangle. We rotate $\overline{PQ}$ around O in both sides to get $\triangle ADC$ and $\triangle AFE$, such that $\overline{PC} = \overline{EP}$. As O is a reflex vertex, we can always rotate the line in this manner, such that $\overline{EF}$ and $\overline{CD}$ lies inside the terrain. Note that we can always rotate until $\overline{PQ}$ hits another vertex of the terrain. Now we show that either $area(\triangle AFE) \geq area(\triangle AQP)$ or $area(\triangle ADC) \geq area(\triangle AQP)$ or both.

Let $area(\triangle ADC) < area(\triangle AQP)$. Now we draw a line $\overline{DL}$ parallel to $\overline{AC}$ from D which intersects $\overline{PQ}$ at X and EF at Y. Here $\triangle ODX$ is similar to $\triangle POC$ and $\triangle PEO$ is similar to $\triangle OXY$. So we have $\frac{\overline{PC}}{\overline{DX}} = \frac{\overline{OM}}{\overline{ON}} = \frac{\overline{EP}}{\overline{XY}}$, where OM and ON is the perpendiculars from O on $\overline{AC}$ and $\overline{DL}$ respectively. Therefore $\frac{\overline{PC}}{\overline{EP}} = \frac{\overline{DX}}{\overline{XY}}$ which implies $\overline{DX} = \overline{XY}$, as $\overline{PC} = \overline{EP}$. Now we draw a line parallel to $\overline{OQ}$ from Y, which intersects $\overline{AB}$ at Z. Observe that Z lies between Q and F. Hence from the Result 1, $\overline{DQ} = \overline{QZ}$ and hence $area(\triangle DQX) < area(\square QZYX)$. This implies

$area(\triangle DQX) < area(\square QFYX)$.

We assumed that $area(\triangle ADC) < area(\triangle AQP)$ which implies $area(\triangle POC) < area(\triangle QOD)$
$\implies area(\triangle POC) < area(\triangle DXO) + area(\triangle DQX)$
$\implies area(\triangle EOP) < area(\triangle XOY) + area(\triangle DQX)$
$\implies area(\triangle EOP) < area(\triangle XOY) + area(\square QFYX)$
$\implies area(\triangle EOP) < area(\triangle QFO)$
$\implies area(\triangle AFE) \geq area(\triangle AQP)$.
Similarly we can show that the result holds if $area(\triangle AFE) < area(\triangle AQP)$, by drawing a line parallel to the base. Thus the result follows. □

Now it is easy to see that there can be $O(n^2)$ many lines passing through n many vertices, so the candidate solution space is bounded by $O(n^4)$. But not all such lines participate to produce a candidate solution, such as the lines joining the convex vertices of the terrain except the base vertices. So we now count the number of such lines those actually participate to produce a candidate solution.

Lemma 3 *For each vertex V of the terrain there exists at most one vertex W of the terrain on the left and below of V such that the line passing through V and W intersects the base of the terrain at P(say) and the segment $\overline{VP}$ lies completely inside the terrain.*

Proof. Consider the vertices on the left of V (i.e. having lesser x-coordinate than V), which are visible from V and choose the one with least y-coordinate, say W'. So W' should either be a reflex vertex or the left base vertex of the terrain. We will show that the vertex W' serves as W. Let the line joining V and W' intersects the terrain base at P'. Observe that, the segment $\overline{VP'}$ is lying completely inside the terrain. For contradiction, suppose the point W is not the point W'. Note that W must be a reflex vertex here. As we assumed no three vertices of the terrain are collinear, W cannot lie on the line $\overline{VP'}$.

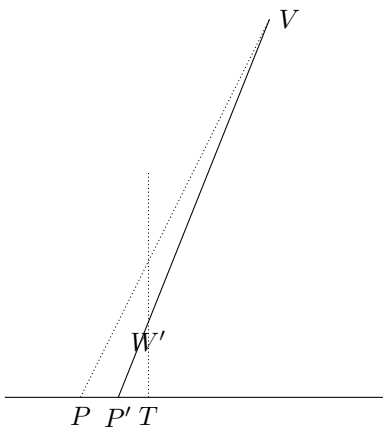


Figure 7: Counting the candidate lines

Here $\overline{VW'}$ and $\overline{VW}$ cuts the base at P' and P respectively (Figure 7). Without loss of generality P lies on the left of P', i.e. $\overline{VP}$ has smaller positive slope than $\overline{VP'}$. Then the extension of the perpendicular $\overline{W'T}$ from W' on the base, cuts $\overline{VP}$ above W'. Since W' is a reflex vertex of a terrain, $\overline{W'T}$ is the only part of the perpendicular that lies inside the terrain. This implies some portion of $\overline{VP}$ is lying outside the terrain, which leads to a contradiction. If P lies on the right of P' then by similar argument some portion of $\overline{VP'}$ lies outside terrain, which is also a contradiction.

Now if W' is a convex vertex, then it has to be the left endpoint of the terrain base. Then P' coincides with W'. By using similar arguments as above we establish the claim. □

Similarly, we conclude the following Lemma for the vertex, which lies on the right side of a vertex V of the terrain.

Lemma 4 *For each vertex V of the terrain there exists at most one vertex W of the terrain on the right and below of V such that the line passing through V and W intersects the base of the terrain at Q(say) and the segment $\overline{VQ}$ lies completely inside the terrain.*

Proof. Follows from similar arguments of Lemma 3. □

Observe that the lines, mentioned in Lemma 3 and Lemma 4 are the candidates for defining left and right boundary of an optimal triangle whose base coincides with the base of the terrain. These lines are termed as *Candidate Lines*. Denote the set of all candidate lines of positive slope by $\mathcal{L}$ and the set of all candidate lines of negative slope by $\mathcal{R}$. The vertices through which a candidate line passes are called the *definers* of this candidate line. The upper one, which has the larger y-coordinate is called the *upper definer* and the other one is called the *lower definer*. A triangle in the terrain whose base lies on the terrain base and whose sides are segments of candidate lines, is called a *grounded triangle*.

Combining Lemma 3 and Lemma 4 we have the following result.

Lemma 5 *The cardinality of $\mathcal{L}$ and $\mathcal{R}$ are bounded by $O(n)$, where n is the total number of vertices of the terrain.*

4 An Exact Algorithm

We can check whether a line lies inside a terrain in $O(n)$ time. So in $O(n^2)$ time we can locate all the candidate lines which contribute to form an optimal triangle having the base coinciding with the terrain base. Then we compute $O(n^2)$ triangles in $O(n^2)$ time formed by these

$O(n)$ candidate lines and report the one with largest area. We can also use techniques like shortest path trees [10] (as described later) to find the candidate lines in $O(n)$ time, but that does not improve the total time of our search.

Theorem 6 *An optimal solution can be found in $O(n^2)$ time.*

5 An Approximation

We design an approximation algorithm with $O(n \log n)$ running time. The algorithm finds a triangle whose height is greater or equal to an optimal triangle and base length is at least half of an optimal triangle. Now to reduce the running time we first find $O(n)$ candidate lines efficiently. Recall that B_l and B_r are the left base vertex and right base vertex of the terrain. We consider a graph (V_T, E), where V_T denotes the vertices of the terrain and an edge $(v_i, v_j) \in E$ if and only if v_i and v_j are two definers of a candidate line belonging to the set $\mathcal{L}$ and $(B_l, B_r) \in E$. Note that the graph is connected. The cardinality of $\mathcal{L}$ is $(n-2)$ (follows from Lemma 3), the cardinality of the edge set E of the graph is $(n-1)$, hence this graph is a tree. We denote it by $\mathcal{T}$.

Lemma 7 *Let v be a vertex of $\mathcal{T}$. Then the path from v to B_l in $\mathcal{T}$ is the shortest path from v to B_l in the terrain.*

Proof. First we observe the structure of the paths in $\mathcal{T}$. Let (v_i, v_j) and (v_j, v_k) be two edges in this tree such that v_i lies below v_j and v_j lies below v_k in the terrain. Then, from the property of the candidate lines, the segment $\overline{v_i v_j}$ has lesser positive slope than the segment $\overline{v_j v_k}$. This means $\mathcal{P}$ is a outward convex chain, where $\mathcal{P} = (v, v_1, v_2, ..., v_s, B_l)$ be the path from v to B_l in $\mathcal{T}$. We show that no other path $\mathcal{Q}$ inside the terrain from B_l to v can have a shorter length than $\mathcal{P}$.

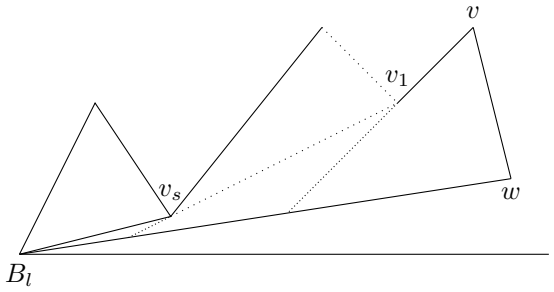


Figure 8: Shortest path tree and the candidate lines

Let $\mathcal{Q}$ be another path from B_l to v which is shorter than $\mathcal{P}$. As shown by Lee and Preparata [14], any path $\mathcal{Q}$ must pass through the vertices of the terrain to minimize its length. If no vertex of $\mathcal{Q}$ lies on the right of $\mathcal{P}$ and $\mathcal{Q}$ lies totally inside the terrain, then $\mathcal{Q}$ must pass through the vertices of $\mathcal{P}$. Since $\mathcal{P}$ is constructed by joining its vertices through straight line segments, $\mathcal{Q}$ cannot be shorter than $\mathcal{P}$. So $\mathcal{Q}$ must have a vertex on the right of $\mathcal{P}$. Let w be such a vertex of $\mathcal{Q}$ as shown in Figure 8.

Without loss of generality let B_l be visible from w. So, length of $\overline{B_l w} + \overline{wv}$ is at least length of $\mathcal{Q}$. Applying triangle inequality repeatedly we can say that length of $\mathcal{P}$ is shorter than this quantity. This implies that $\mathcal{P}$ is the shortest path in the terrain from v to B_l. □

Theorem 8 *The tree $\mathcal{T}$ is the shortest path tree of the terrain sourced at B_l.*

Proof. The proof follows from Lemma 7. □

We can conclude that the tree constructed by the members of $\mathcal{R}$ in a similar fashion, is the shortest path tree of the terrain sourced at B_r. So we can find out the candidate lines by constructing these two shortest path trees.

Let c_i be a candidate line which intersects the base of the terrain at b_i. Also let t_i denote the topmost point on c_i such that the segment $\overline{b_i t_i}$ lies completely inside the terrain. We denote the height of a point p in the terrain by $ht(p) = y$, where (x, y) is the coordinate of point p. So an optimal grounded triangle constructed with the segments of $c_i \in \mathcal{L}$ and $c_j \in \mathcal{R}$ can have height at most $\max\{ht(t_i), ht(t_j)\}$. Let p_v be the point on the terrain base where the perpendicular from point v on the terrain base, intersects the terrain base.

Lemma 9 *Let $c_i \in \mathcal{L}$, $c_j \in \mathcal{R}$ be two candidate lines forming an optimal grounded triangle and suppose c_i and c_j intersect at point a_{ij} inside the terrain. Then one of the triangles between $\triangle t_i p_{t_i} b_i$ and $\triangle t_j p_{t_j} b_j$ has base length at least half of the base length of the optimal triangle $\triangle b_i a i j b_j$.*

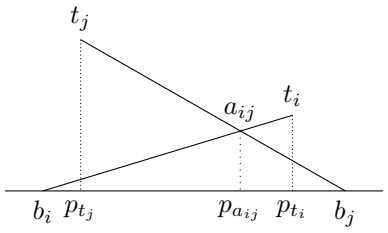


Figure 9: Capturing the base

Proof. The internal angles adjacent to the base of $\triangle b_i a_{ij} b_j$ are acute (by Observation 2), which implies that $p_{a_{ij}}$ lies between b_i and b_j. So either $\overline{b_i p_{a_{ij}}}$ (as shown in Figure 9) or $\overline{b_j p_{a_{ij}}}$ has length at least half of the length of $\overline{b_i b_j}$. Accordingly either $\triangle t_i p_{t_i} b_i$ or

$\triangle t_j p_{t_j} b_j$ has base length at least half of the base length of the triangle $\triangle b_i a_{ij} b_j$.

□

We find the candidate lines and the corresponding t_is as defined above. Then we calculate the area of the triangles $\triangle b_i t_i p_{t_i}$ and report that one with the maximum area. Since any optimal grounded triangle having a side lying on c_i, has lesser height than the height of t_i, we can conclude that the reported triangle has area at least half of an optimal triangle.

Using the shortest path tree method [10] we can locate the candidate lines. This takes $O(n)$ time. Then by using the ray-shooting query [7] we can find the t_i for a line c_i in $O(\log n)$ time. So total time complexity of the search is $O(n \log n)$. So we can conclude the following theorem.

Theorem 10 *A $\frac{1}{2}$-factor approximation for a largest area triangle inside a terrain can be found in $O(n \log n)$ time.*

6 Remarks

We presented an $O(n^2)$ time exact algorithm, and $O(n \log n)$ time $\frac{1}{2}$-approximation algorithm for computing a largest area triangle contained in a terrain. We expect to generalize the above approximation algorithm to an approximation scheme in case of simple polygon. We want to further investigate the complexity of computing a largest area convex polygon inside a given terrain.

References

[1] P. Bose and J.-L. De Carufel. Minimum enclosing area triangle with a fixed angle. *Computational Geometry*, 47:90–109, 2014.

[2] J. E. Boyce, D. P. Dobkin, R. L. Drysdale III, and L. J. Guibas. Finding extremal polygons. In *Proceedings of the 14th annual ACM symposium on Theory of computing*, pages 282–289, San Francisco, California, USA, 1982.

[3] S. Cabello, O. Cheong, C. Knauer, and S. Schlipf. Finding largest rectangles in convex polygons. *Computational Geometry*, 51:67–74, 2016.

[4] S. Cabello, J. Cibulka, J. Kynčl, M. Saumell, and P. Valtr. Peeling potatoes near-optimally in near-linear time. *SIAM Journal on Computing*, 46(5):1574–1602, 2017.

[5] S. Chandran and D. Mount. A parallel algorithm for enclosed and enclosing triangles. *International Journal of Computational Geometry and Applications*, 2(2):191–214, 1992.

[6] J. S. Chang and C. K. Yap. A polynomial solution for the potato-peeling problem. *Discrete & Computational Geometry*, 1(2):155–182, 1986.

[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.

[8] D. P. Dobkin and L. Snyder. On a general method for maximizing and minimizing among geometric problems. In *Proceedings of 20th Annual Symposium on the Foundations of Computer Science*, pages 9–17, San Juan, Puerto Rico, 1979.

[9] J. E. Goodman. On the largest convex polygon contained in a non-convex n-gon, or how to peel a potato. *Geometriae Dedicata*, 11:99–106, 1981.

[10] L. J. Guibss and J. Hershberger. Optimal shortest path queries in a simple polygon. In *Proceedings of the 3rd annual symposium on Computational geometry*, pages 50–63, Waterloo, Ontario, Canada, 1987.

[11] O. Hall-Holt, M. J. Katz, P. Kumar, J. S. B. Mitchell, A. Sityon. Finding large sticks and potatoes in polygons. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, Pages 474–483, Miami, Florida, 2006.

[12] Y. Kallus. A linear-time algorithm for the maximum-area inscribed triangle in a convex polygon. *arXiv*:1706.03049, 2017.

[13] V. Keikha, M. Löffler, A. Mohades, J. Urhausen, I. Hoog. Maximum-area triangle in a convex polygon, revisited. *arXiv*:1705.11035, 2017.

[14] D. T. Lee and F. P. Preparata. Euclidean shortest paths in the presence of rectilinear barriers. *Networks* 14(3): 393–410, 1984.

[15] E. A. Melissaratos and D. L. Souvaine. Shortest paths help solve geometric optimization problems in planar regions. *SIAM Journal of Computing*, 21(4):601–638, 1992.

Reconstructing a Polyhedron between Polygons in Parallel Slices

Therese Biedl* Pavle Bulatovic* Veronika Irvine*
Anna Lubiw* Owen Merkel* Anurag Murty Naredla*

Abstract

Given two n-vertex polygons, $P = (p_1, \ldots, p_n)$ lying in the xy-plane at $z = 0$, and $P' = (p'_1, \ldots, p'_n)$ lying in the xy-plane at $z = 1$, a *banded surface* is a triangulated surface homeomorphic to an open cylinder connecting P and P' such that the triangulation contains vertex disjoint paths π_i connecting p_i to p'_i for all $i = 1, \ldots, n$. The surface then consists of *bands*, where the ith band goes between π_i and π_{i+1}. We give a polynomial-time algorithm to find a banded surface without Steiner points if one exists. We explore connections between banded surfaces and linear morphs, where time in the morph corresponds to the z direction. In particular, we show that if P and P' are convex and the linear morph between them preserves planarity, then there is a banded surface without Steiner points.

1 Introduction

The problem of reconstructing a 3D polyhedral structure between two planar cross-sections has been heavily studied because of its many practical applications, e.g., in medicine for constructing models of body organs from MRI slices. Most approaches, e.g. [3], separate the problem into two steps, both of which are hard and are tackled via heuristics: choose a correspondence between the two cross-sections; and then construct a triangulated surface using extra Steiner points. The problem is considered to be well-solved by these heuristic methods, but many theoretical questions remain open. We focus on the second step, i.e., we assume that the correspondence is given. Also we focus on the case of two polygons, though the case of general graph drawings is also of interest.

Given two simple n-vertex polygons, $P = (p_1, \ldots, p_n)$ lying in the xy-plane at $z = 0$, and $P' = (p'_1, \ldots, p'_n)$ lying in the xy-plane at $z = 1$, we want to interpolate between them by constructing a non-self-intersecting triangulated surface $\mathcal{S}$ homeomorphic to an open-ended cylinder, with P at one end and P' at the other end. Vertices of $\mathcal{S}$ that are not vertices of P or P' are called *Steiner points*. We want the surface to be monotone, in the sense that any plane $z = t$ intersects the surface in one simple (non-self-intersecting) polygon. Furthermore, we want to maintain the correspondence between p_i and p'_i in the following strong sense: for each i there is a path π_i of edges in the triangulation of $\mathcal{S}$ from p_i to p'_i, and these paths are vertex disjoint. The paths then partition the surface $\mathcal{S}$ into interior-disjoint *bands* $B_1, \ldots, B_n$, where B_i is the subset of $\mathcal{S}$ between π_i and π_{i+1}. We call $\mathcal{S}$ a *banded surface* and we call this problem *banded surface reconstruction between parallel slices* or just "banded surface reconstruction". Figure 1 shows some examples.

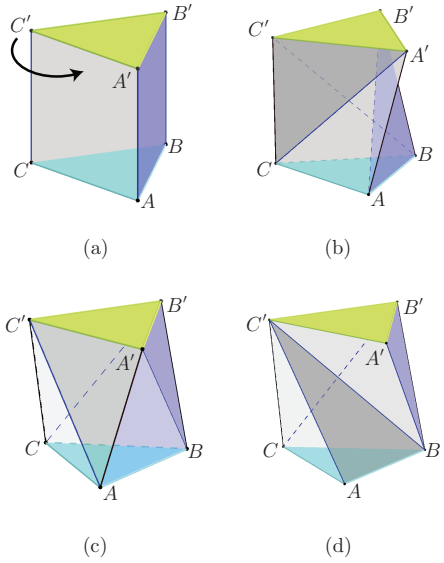


Figure 1: Examples of banded surfaces without Steiner points. (a) Starting with a triangular prism based on equilateral triangle $P = ABC$, rotate the top triangle to obtain $P' = A'B'C'$ used subsequently. (b) The Schönhardt polyhedron is a banded surface formed by bending each original rectangular face inward to form two triangles, using the "right" chords AB', BC', CA'. (c) Using the outward or "left" chords, AC', CB', BA' also yields a banded surface (an antiprism when P' is rotated by 60°). (d) An example of a triangulated surface that is not banded due to the lack of a path from A to A' disjoint from BB' and CC'.

Our conditions prevent some undesirable "solutions" such as placing one Steiner point X at $z = \frac{1}{2}$ and

*Cheriton School of Computer Science, University of Waterloo. Research of TB, VI and AL supported by NSERC. This research was initiated at the Algorithmic Problem Session group at the University of Waterloo

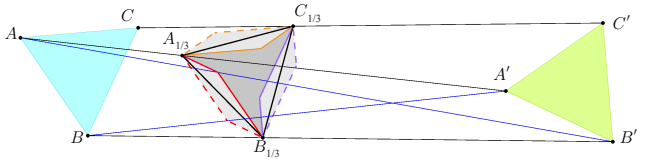


Figure 2: The examples of Figure 1(b,c) in top-view with triangle $A'B'C'$ translated horizontally. (Invariance under translation is proved in Lemma 3.) The cross-section at $z = 1/3$ shows the triangle $A_{1/3}B_{1/3}C_{1/3}$ of the linear morph, together with the inward (solid colour) and outward (dashed colour) choices for each edge. Observe that whereas the linear morph uses the edge $A_{1/3}B_{1/3}$ at $z = \frac{1}{3}$, the inward banded surface using chord AB' uses two edges (shown in solid red), the first parallel to $A'B'$ and the second parallel to AB, and the outward banded surface using chord $A'B$ uses two edges (shown in dashed red), the first parallel to AB and the second parallel to $A'B'$.

building cones from the configurations at $z = 0$ and $z = 1$ to X. (The fact that each of these cones does not self-intersect is proved in [8].) Furthermore, the condition about disjoint paths π_i prevents us from replacing such a central point X by a polygon with fewer than n vertices in the plane $z = \frac{1}{2}$.

In this paper we concentrate on the case where no Steiner points are allowed. Understanding this case may lead to more general solutions where we design $\mathcal{S}$ in layers using intermediate polygons (made of Steiner points) at a succession of z values, and build surfaces without Steiner points between successive layers.

When no Steiner points are allowed we must use the edges (p_i, p'_i), and our only choice is how to triangulate each quadrilateral $p_i, p_{i+1}, p'_{i+1}, p'_i$. There are two possible chords for each quadrilateral: the *right* chord (p_i, p'_{i+1}) or the *left* chord (p_{i+1}, p'_i). The difference between these two choices can be seen in Figure 1(b) and (c), and also in Figure 2. An example of two triangles with no banded surface is shown in Figure 3(a).

Our Results. We prove the following:

1. There is a polynomial time algorithm (using 2-SAT) to solve the banded surface reconstruction problem when no Steiner points are allowed.

2. The existence of a banded surface without Steiner points is preserved by translating P'.

3. If P and P' are convex and the linear morph from P to P' preserves planarity (these terms are defined on the next page) then there is a banded surface without Steiner points between P and P'. This no longer holds if P and P' are non-convex.

4. In the other direction, the existence of a banded surface without Steiner points does not imply that the linear morph preserves planarity, not even when P and P' are triangles. See Figure 3(b).

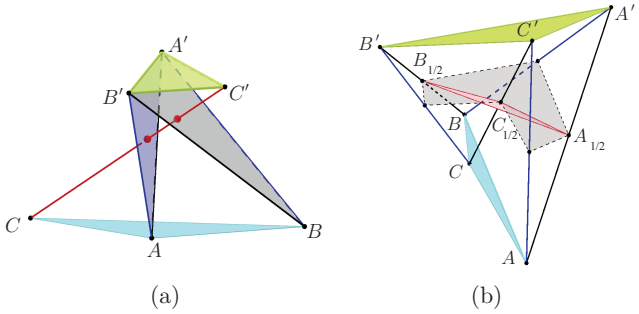


Figure 3: (a) Triangles ABC and $A'B'C'$ have no banded surface (without Steiner points): the edge $A'B'$ must be in a triangle with either vertex A or vertex B but both those triangles intersect the edge CC'. (b) A banded surface between ABC and $A'B'C'$ using chords AC', BA', and CB', showing the cross-section (dashed, shaded grey) at $z = \frac{1}{2}$. However, the linear morph does not preserve planarity since at $z = \frac{1}{2}$ the triangle $A_{1/2}C_{1/2}B_{1/2}$ (shown in red) is inverted.

Previous Work. Gitlin, O'Rourke and Subramanian [8] considered a similar problem of joining two polygons via a triangulated surface without adding Steiner points. However, they did not require disjoint paths from p_i to p'_i, which gives a lot more freedom, e.g., the two polygons can have different numbers of vertices. Essentially, every edge of P must be in a triangle with some vertex of P', and vice versa, and these triangles must be internally disjoint. Their main result was a construction of a pair of polygons on 63 vertices with no triangulated surface between them. Their proof involved a computer search. Barequet and Steiner [5] gave a slightly simpler example on 45 vertices. It is an open question whether this version of the problem is NP-complete.

There is a vast literature about interpolating between two families of nested polygons lying in parallel planes via a triangulated surface, see [3, 4]. Barequet and Steiner [4] write: "The primary concern in the literature has usually been to find fast heuristics for selecting a 'good' reconstruction among the many available solutions." There is little work analyzing the number of Steiner points, or examining when a solution with no Steiner points is possible.

Our problem is related to the problem of finding a piecewise linear embedding of a 2D simplicial complex in 3D, which was recently shown to be NP-hard [7]. (One dimension down this is easy, since it is the problem of

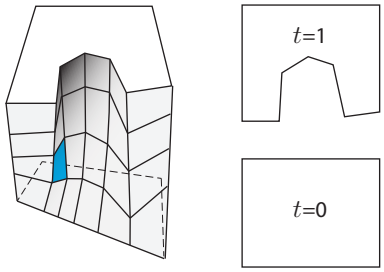


Figure 4: A morph from a rectangle at time (or z-coordinate) $t = 0$ to the "arch-shaped" polygon at time $t = 1$ yields a 3D interpolation by taking "snapshots" of the morph at some intermediate time points, and joining vertices between one snapshot and the next. Note that the quadrilateral patches formed this way (one of which is coloured blue) are not planar in general. This figure is loosely based on one by Surazhsky and Gotsman [10, Figure 10].

finding a (poly-line) planar drawing of a graph.) Our 2D complex consists of the quadrilaterals $p_i, p_{i+1}, p'_{i+1}, p'_i$. Our additional structure ensures that there always is a solution, so our goal is just to minimize the number of Steiner points.

Relationship to Morphing. A morph is a continuous transformation from one shape to another. In particular, a *morph* from an initial simple polygon (or planar straight-line graph drawing) P^0 to a final one, P^1, with the same labelled vertices, is a continuously changing family of polygons/drawings P^t indexed by time $t \in [0, 1]$. A morph *preserves planarity* if all intermediate polygons/drawings P^t are simple/planar. In a *linear morph* each vertex moves on a straight line from its initial position to its final position at constant speed (where the speed of a vertex depends on the distance it must travel), and an edge is always drawn as a line segment between its endpoints.

Our problem of reconstructing a 3D polyhedral structure between two planar drawings is very related to morphing—the z direction corresponds to time in the morph. In fact, it is claimed (for example, by Surazhsky and Gotsman [10]) that morphing algorithms solve 3D shape reconstruction. We will examine this claim more closely. Figure 4 shows how a morph between two polygons yields a 3D surface composed of quadrilateral "patches", each a ruled surface. In order to obtain a piece-wise linear surface we must replace each such quadrilateral by two triangles. It is not obvious how to do this—replacing quadrilaterals by pairs of triangles may cause the surface to self-intersect. It seems intuitive that this can be remedied by taking sufficiently many snapshots, but such analysis is lacking.

An algorithm by Alamdari et al. [1] finds "piece-wise linear" morphs that consist of a linear number of planarity-preserving linear morphs. This would provide a solution to banded surface reconstruction if we could show how to add Steiner points to turn a linear morph into a triangulated surface.

In the other direction, a banded surface (even one with Steiner points) can be interpreted as a morph between the polygons P and P', albeit a morph in which an edge becomes a poly-line. Such "morphs with bent edges" have been investigated [9] and come with small grid guarantees, unlike the piece-wise linear morphs of [1]. A banded surface without Steiner points provides a morph with the interesting property that in any intermediate drawing of the morph, an edge e appears as a path of two line segments, one in the direction of the initial version of e and the other in the direction of the final version of e. See Figure 2. Such morphs may be valuable for visualizations. We note that there is work on morphing while maintaining edge directions [6]—this only applies in the restricted situation where the initial and final polygons have corresponding edges with the same directions.

To summarize, it seems worth investigating to what extent linear morphs provide banded surfaces, and to what extent banded surfaces provide morphs.

2 Finding a Banded Surface without Steiner Points

In this section we give an algorithm using 2-SAT to find a banded surface without Steiner points between two n-vertex polygons, $P = (p_1, \ldots, p_n)$ lying in the xy-plane at $z = 0$, and $P' = (p'_1, \ldots, p'_n)$ lying in the xy-plane at $z = 1$. For each $i = 1, \ldots, n$ we have the choice of the right chord $p_i p'_{i+1}$ or the left chord $p_{i+1} p'_i$. Let the Boolean variable R_i be 1 if the right chord is chosen and 0 otherwise. Each chord choice determines two triangles of the surface, for example $R_i = 1$ determines triangles $p_i p_{i+1} p'_{i+1}$ and $p_i p'_{i+1} p'_i$. We say that chord choices for i and j *conflict* if the resulting open triangles intersect. Note that this can be tested, for given i, j, in constant time. The problem of choosing chords to form a non-self-intersecting surface can be formulated as a Boolean satisfiability problem by adding a clause to prohibit conflicts, e.g., if chord choices R_i and $\neg R_j$ conflict then we add the clause $\neg(R_i \wedge \neg R_j)$. Note that there are $O(n^2)$ clauses.

There is a banded surface without Steiner points if and only if the resulting clauses are satisfiable. Because all clauses have two variables, the result is a 2-SAT instance. Since 2-SAT can be solved in linear time [2], we have:

Lemma 1 *There is a quadratic time algorithm that either finds a banded surface without Steiner points, or declares that no such surface exists.*

3 Some Conditions for the Existence of a Banded Surface

One approach to banded surface reconstruction when Steiner points are required, is to subdivide the interval $z \in [0,1]$ into $0{=}z_0, z_1, \ldots, z_k{=}1$ and place an n-vertex polygon at each $z_i, 0 < i < k$ so that each successive pair of polygons admits a banded surface without Steiner points. Using this approach, the final solution would have nk Steiner points.

In order to design the intermediate polygons, it would be good to have conditions for when two polygons admit a banded surface without Steiner points. (Our polynomial-time test does not seem helpful when the polygons are not given).

In this section we explore two situations where we can guarantee the existence of a banded surface. We show:

1. Translation of P' in the $z = 1$ plane preserves the existence of a banded surface without Steiner points (Lemma 3).

2. If P is convex and P' is a rotation of P by an angle less than π, then a banded surface exists (Lemma 5). The example of Figure 8 shows that this property does not hold more generally, not even for a star-shaped polygon.

We first show how translation of the target-polygon affects the intermediate polygons in a linear morph:

Lemma 2 *Let P be an n-vertex polygon in the $z = 0$ plane and P' be an n-vertex polygon in the $z = 1$ plane. Let Q' be a translation of P' within the $z = 1$ plane. For any $0 < t < 1$, if P_t is the polygon at time t during the linear morph from P to P', and Q_t is the polygon at time t during the linear morph from P to Q', then Q_t is a translation of P_t within the $z = t$ plane.*

Proof. Set $s = Q' - P'$ to be the translation vector and consider an arbitrary point p of P that morphs to point p' of P' and q' of Q'. We have $q' = p' + s$, and hence

$$q_t = (1-t)p + tq' = (1-t)p + tp' + ts = p_t + t \cdot s$$

so polygon Q_t is a translation of P_t by $t \cdot s$. □

In particular, if the linear morph from P to P' preserves planarity, then the same holds for the linear morph from P to any translation of P'. We can argue the same for banded surfaces:

Lemma 3 *Assume that P, P' and Q' are as in Lemma 2. If there is a banded surface without Steiner points between P and P', then the same choice of chords yields a banded surface without Steiner points between P and Q'.*

Proof. We show that the banded surface between P and P' is the same as the linear morph between two modified polygons P_D and P'_D, which we now define. Initially start with P and P'. For each $i = 1, \ldots, n$, if we chose the right chord $p_i p'_{i+1}$, then duplicate vertex p_i in P_D (inserting an edge of length 0) and duplicate vertex p'_{i+1} in P'_D. Proceed symmetrically if we chose the left chord. Now consider the linear morph from P_D to P'_D, where vertices that have been inserted due to a chord correspond to each other. Say we chose the right chord $p_i p'_{i+1}$. Then the zero-length edge $p_i p_i$ in P_D morphs to edge $p'_i p'_{i+1}$ in P'_D, hence forms a triangle. Likewise edge $p_i p_{i+1}$ in P_D morphs to zero-length edge $p'_{i+1} p'_{i+1}$ in P'_D, and also forms a triangle. The two triangles together form exactly the part of the banded surface between edges $p_i p_{i+1}$ and $p'_i p'_{i+1}$ in P and P'.

Since the banded surface is the same as the linear morph from P_D to P'_D the result now follows from Lemma 2. □

We now turn to rotations, beginning with this result on linear morphs when the target-polygon is a rotation of the source-polygon:

Lemma 4 *Let P be a polygon and let P' be a rotation of P about an origin X by an angle α. For any $0 < t < 1$ let P_t be the polygon at time t during a linear morph from P to P'. If $\alpha \neq \pi$ or $t \neq \frac{1}{2}$ then P_t is a rotated copy of P that has been scaled by $s_t \neq 0$.*

Proof. We consider P, P' and P_t projected to the xy plane. If $\alpha = \pi$ then every point p of P maps to to $-p$ in P', which implies $p_t = (1-2t)p$. So $P_t = s_t P$ for $s_t = 1 - 2t$, which is non-zero for $t \neq \frac{1}{2}$.

Now suppose that $\alpha < \pi$ (the case $\alpha > \pi$ is symmetric). For any point p of P, consider the triangle $\Delta_p := \Delta p X p'$, where X is the center of the rotation. Note that Δ_p and Δ_q are similar for any two points p and q of P, since they both have angle α and two equal-length incident sides; in particular Δ_q is obtained from Δ_p by scaling by $||q||/||p||$ and (possibly) rotating. Also notice that p_t travels along the side of Δ_p opposite to angle α, and is at the point that divides the side at ratio $t/(1-t)$. We can view p_t as having been rotated by some angle θ_t and scaled by some $s_t > 0$. Both θ_t and s_t are independent of the choice of p since all triangles Δ_p are similar. Therefore P_t is obtained from P by scaling by s_t and rotating by θ_t. □

Lemma 5 *Let P be a convex polygon and let P' be a rotation of P about an origin X by an angle $\alpha < \pi$. Then there is a banded surface between P and P'.*

Proof. Observe first that the linear morph from P to P' preserves planarity since, by Lemma 4, each intermediate polygon is a rotated and scaled copy of P. By Theorem 6 (forthcoming, but there is no circularity) this implies the existence of a banded surface. □

4 Linear Morphing versus Banded Surface Reconstruction

In this section we compare the existence of a planarity-preserving linear morph from P to P' and the existence of a banded surface without Steiner points. In general, these two properties are independent, i.e., neither implies the other. Figure 3(b) shows an example of two triangles that have a banded surface, but the linear morph does not preserve planarity. Figure 8 shows an example of two stars that do not have a banded surface, but the linear morph preserves planarity.

When the polygons P and P' are convex, we do get an implication:

Theorem 6 *If P and P' are convex and the linear morph from P to P' preserves planarity, then there is a banded surface without Steiner points between P and P'.*

Proof. Let p_i^t be the position of the ith vertex at time (z-coordinate) t during the linear morph. In particular, $p_i^0 = p_i$ and $p_i^1 = p_i'$. Let P^t be the polygon at time t during the morph. By our convention of numbering polygons in counterclockwise order, the inside of P is to the left of p_ip_{i+1}, and the inside of P' is to the left of $p_i'p_{i+1}'$. Also, because the linear morph preserves planarity, the inside of P^t is to the left of $p_i^tp_{i+1}^t$.

We begin by defining the surface $\mathcal{S}$, i.e., which chords to use. Let v_i^0 be the vector $p_{i+1}-p_i$ in the xy plane, and let v_i^1 be the vector $p_{i+1}' - p_i'$ projected to the xy plane. Let θ_i be the angle between v_i^0 and v_i^1, measured towards the inside of P, as shown in Figure 5. We distinguish 3 cases:

- If $\theta_i < \pi$, use the left chord $p_{i+1}p_i'$. In the cross-section of $\mathcal{S}$ at z-coordinate (or time) t, the edge $p_i^tp_{i+1}^t$ is replaced by a segment in the direction v_i^0 followed by a segment in the direction v_i^1. We call the resulting triangle Δ_i^t and refer to it as a 01 triangle. Observe that Δ_i^t lies to the outside of the edge $p_i^tp_{i+1}^t$. See Figure 5(a).
- If $\theta_i > \pi$, use the right chord p_ip_{i+1}'. Then, in the cross-section at z-coordinate t, the edge $p_i^tp_{i+1}^t$ is replaced by a segment in the direction v_i^1 followed by a segment in the direction v_i^0. We refer to the resulting triangle Δ_i^t as a 10 triangle. Again, Δ_i^t lies to the outside of $p_i^tp_{i+1}^t$, see Figure 5(b).
- If $\theta_i = \pi$, use either chord—in this case the quadrilateral $p_i, p_{i+1}p_{i+1}'p_i'$ is coplanar, and Δ_i^t collapses to the edge $p_i^tp_{i+1}^t$.

We now prove that $\mathcal{S}$, as defined by the above chord choices, is non-self-intersecting, which proves that $\mathcal{S}$ is a banded surface without Steiner points. In particular, we will prove that $\mathcal{S}^t$, the cross-section of $\mathcal{S}$ at z-coordinate t is a simple polygon. By assumption, the polygon P^t with vertices $p_1^t, p_2^t, \ldots, p_n^t$ is simple. $\mathcal{S}^t$ consists of P^t plus triangles Δ_i^t added to the outside of each edge. See Figure 6. We will show that no two triangles intersect.

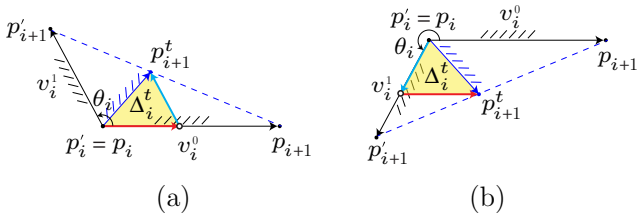


Figure 5: A top view to illustrate choosing chords in the proof of Theorem 6. In order to show the angles clearly, P' has been translated so that p_i is at the same xy-coordinates as p_i'. (Lemma 3 justifies this.) Note that p_i^t then remains at these same xy-coordinates. Hatching indicates the inside of the polygon on that edge. (a) If $\theta_i < \pi$, use a left chord to obtain a 01 triangle Δ_i^t. (b) If $\theta_i > \pi$, use a right chord to obtain a 10 triangle Δ_i^t. The segments that replace $p_i^tp_{i+1}^t$ are shown in red/cyan.

Claim 7 *Suppose that $\Delta_i^t, \Delta_{i+1}^t, \ldots, \Delta_j^t$ are all 01 triangles. Let r_i^0 be the ray from p_i^t in the direction v_i^0. Then none of these triangles cross r_i^0 from its left to its right.*

Proof. It suffices to prove that no triangle crosses the ray of the previous triangle, so consider triangle Δ_{i+1}^t and r_i^0. The apex of Δ_{i+1}^t lies on r_{i+1}^0. Rays r_i^0 and r_{i+1}^0 emanate from the endpoints of the edge $p_i^tp_{i+1}^t$ and the angle between r_i^0 and r_{i+1}^0 is positive (counterclockwise). Thus the apex of Δ_{i+1}^t lies to the left of r_i^0. □

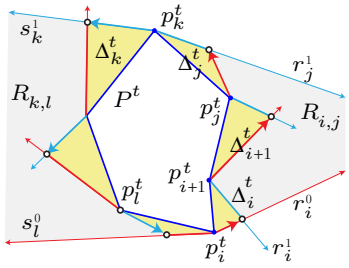


Figure 6: Polygon P^t (in blue) with 01 triangles $\Delta_i^t, \ldots, \Delta_j^t$ and 10 triangles $\Delta_k^t, \ldots, \Delta_l^t$.

Define r_i^1 to be the ray from p_{i+1}^t in the direction $-v_i^1$. Thus a 01 triangle Δ_i^t is bounded by r_i^0 and r_i^1. Symmetrically, for a 10 triangle, define s_i^1 to be the ray from p_i^t in the direction v_i^1, and s_i^0 to be the ray from p_{i+1}^t in the direction $-v_i^0$. Thus a 10 triangle Δ_i^t is bounded by s_i^1 and s_i^0.

From Claim 7, by symmetry, we obtain (see Figure 6):

Claim 8 *If $\Delta_i^t, \Delta_{i+1}^t, \ldots, \Delta_j^t$ are 01 triangles then none of them cross r_j^1 from right to left. If $\Delta_k^t, \Delta_{k+1}^t, \ldots, \Delta_l^t$ are 10 triangles then none of them cross s_k^1 from left to right and none of them cross s_l^0 from right to left.*

These two claims imply that $\mathcal{S}^t$ is simple if all the triangles are the same (all 01 or all 10). It remains to consider the possibility that there are triangles of both types.

Claim 9 *Suppose Δ_{i-1}^t is a 10 triangle and Δ_i^t is a 01 triangle. Then Δ_{i-1}^t and Δ_i^t are disjoint. Furthermore, P^t is convex at p_i^t.*

Proof. We analyze the top-view projection with P' translated so that p_i and p_i' are at the same xy-coordinates.

Consider the angle $\alpha_i^t = \angle p_{i+1} p_i p_{i+1}^t$. Because Δ_i^t is a 01 triangle, α_i^t goes from 0 to $\theta_i < \pi$. Similarly, because Δ_{i-1}^t is a 10 triangle, the angle $\alpha_{i-1}^t = \angle p_{i-1} p_i p_{i-1}^t$ goes from 0 to $2\pi - \theta_{i-1} < \pi$.

If $p_i p_{i-1}^t$ and $p_i p_{i+1}^t$ cross over each other, as in Figure 7(a), i.e., $\theta_i + 2\pi - \theta_{i-1} \geq \angle p_{i-1} p_i p_{i+1}$, then there must be some time t when $\alpha_i^t + \alpha_{i-1}^t = \angle p_{i-1} p_i p_{i+1}$, i.e., angle $\angle p_{i-1}^t p_i^t p_{i+1}^t$ becomes 0. But we assumed that P^t remains simple, so this cannot happen.

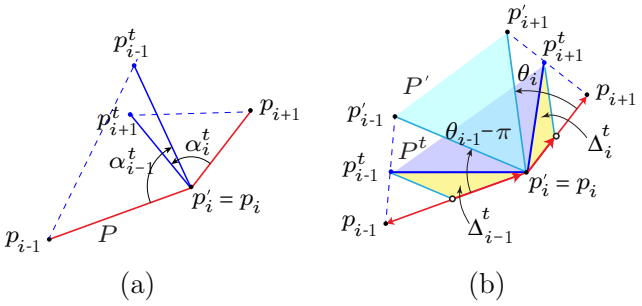


Figure 7: Illustration for the proof of Claim 9: (a) top view projection at $p_i = p_i'$ showing the angles α_i^t and α_{i-1}^t; (b) because $\theta_i + 2\pi - \theta_{i-1} < \angle p_{i-1} p_i p_{i+1}$, P' lies inside P at p_i.

Thus we must have the situation shown in Figure 7(b), so Δ_{i-1}^t and Δ_i^t are disjoint and P^t remains convex at p_i^t. □

With these claims in hand, we can complete the proof of the theorem. Divide the circular sequence $\Delta_1^t, \ldots, \Delta_n^t$ into maximal subsequences all of the same type (all 01 or all 10). If $D_{i,j} = \Delta_i^t, \ldots, \Delta_j^t$ is such a maximal subsequence then by Claims 7 and 8 no two triangles of $D_{i,j}$ intersect, and all the triangles of $D_{i,j}$ live in the region $R_{i,j}$ bounded by $p_i^t, \ldots, p_{j+1}^t$ and two bounding rays—r_i^0 and r_j^1 in the case of 01 triangles, as shown in Figure 6. Between one sequence $D_{i,j}$ and the next, $D_{j+1,l}$, Claim 9 implies that the regions $R_{i,j}$ and $R_{j+1,l}$ are disjoint. □

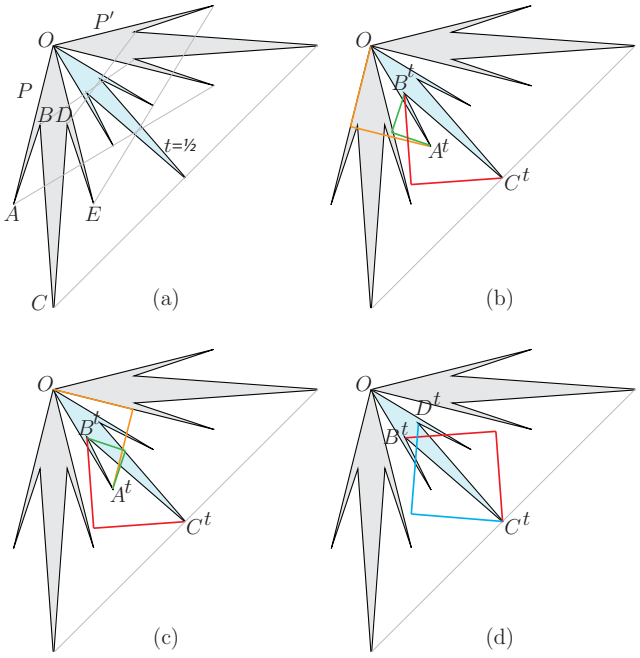


Figure 8: A star-shaped polygon P where the linear morph to its 90° rotation P' preserves planarity, but there is no banded surface: (a) P and P' and the intermediate position of the linear morph at $t = \frac{1}{2}$ (shaded blue); (b) If we choose the chord for edge BC that produces the "outward" triangle (shown in red) then at $t = \frac{1}{2}$ it intersects one choice for OA and one choice for BA; (c) The other choices for OA and BA intersect each other; (d) Thus we are forced to choose the chord for edge BC that produces the "inward" triangle (shown in red), and, by symmetry, the "inward" triangle for DC (shown in cyan)—but these intersect.

5 Conclusions

We have introduced the idea of a banded surface to construct a polyhedron between two polygons in parallel slices and have explored some connections between linear morphs and banded surfaces without Steiner points. Many questions remain:

1. What is a bound on the number of Steiner points needed to construct a banded surface between two n-vertex polygons? What if the polygons are convex?

2. Is the problem of minimizing the number of Steiner points NP-hard?

References

[1] S. Alamdari, P. Angelini, F. Barrera-Cruz, T. M. Chan, G. Da Lozzo, G. Di Battista, F. Frati, P. Haxell, A. Lubiw, M. Patrignani, V. Roselli, S. Singla, and B. T. Wilkinson. How to morph planar graph drawings. *SIAM J. Computing*, 46(2):29 pages, 2017.

[2] B. Aspvall, M. F. Plass, and R. E. Tarjan. A linear-time algorithm for testing the truth of certain quantified Boolean formulas. *Information Processing Letters*, 8(3):121–123, 1979.

[3] G. Barequet, M. T. Goodrich, A. Levi-Steiner, and D. Steiner. Contour interpolation by straight skeletons. *Graphical Models*, 66(4):245–260, 2004.

[4] G. Barequet and M. Sharir. Piecewise-linear interpolation between polygonal slices. *Computer Vision and Image Understanding*, 63(2):251–272, 1996.

[5] G. Barequet and A. Steiner. On the matability of polygons. *International Journal of Computational Geometry & Applications*, 18(05):469–506, 2008.

[6] T. Biedl, A. Lubiw, and M. J. Spriggs. Morphing planar graphs while preserving edge directions. In *International Symposium on Graph Drawing*, pages 13–24. Springer, 2005.

[7] A. De Mesmay, Y. Rieck, E. Sedgwick, and M. Tancer. Embeddability in R^3 is NP-hard. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1316–1329. Society for Industrial and Applied Mathematics, 2018.

[8] C. Gitlin, J. O'Rourke, and V. Subramanian. On reconstructing polyhedra from parallel slices. *International Journal of Computational Geometry & Applications*, 6(01):103–122, 1996.

[9] A. Lubiw and M. Petrick. Morphing planar graph drawings with bent edges. *Journal of Graph Algorithms and Applications*, 15:31–53, 2011.

[10] V. Surazhsky and C. Gotsman. High quality compatible triangulations. *Engineering with Computers*, 20(2):147–156, 2004.

On Multi-Dimensional Team Formation

Thomas Schibler* Ambuj Singh† Subhash Suri‡

Abstract

We consider a team formation problem in multi-dimensional space where the goal is to group a set of n agents into α teams, each of size β, to maximize their total performance. The performance of each team is measured by a score, which is the sum of h highest skill values in each dimension. We wish to maximize the sum of team scores. We prove that the problem is NP-hard if the dimension is $d = \Omega(\log n)$ even for $h = 1$ and $\beta = 4$. We then describe an efficient algorithm for solving the problem in two dimensions as well an algorithm for computing a single optimal team in any constant dimension.

1 Introduction

The problem of grouping a set of agents into teams with the objective of optimizing their collective performance is ubiquitous in a variety of organization settings, including team sports, project management, law, military, management consulting, academic ad hoc committees, to name a few. Mathematical models of team selection and performance, therefore, are an important area of research in social and management sciences. In these models, the skill set of each individual is typically modeled as an attribute vector. Research shows that while individual skills are clearly an important factor, the team's ability to search over vast and often ill-defined decision space crucially depends on its overall synergy and diversity [7, 14, 18, 20]. As a result, it is widely recognized that the performance of a team along a specific skill dimension should not depend on the average of the group members' values (so called weak synergy [6]) but rather on the skills of the best individuals on each dimension [1, 10, 19].

The selection of a single best team has been considered broadly in the literature [2, 3, 12, 13, 9, 11, 4, 15]. The more general problem of assembling multiple teams, however, is less well-understood, and has been studied mainly in the context of very specific performance objectives. For instance, Fitzpatrick and Askin [5] develop heuristics for assembling multiple 'multi-functional' teams using an integer programming formulation. The *coalition formation* problem in multi-agent (and multi-robot) systems also partitions agents into teams but the primary goal there is strategic utility maximization of completing a given set of tasks [16]. When the utility function is a simple sum of scalars, this becomes an easy-to-solve assignment problem in bipartite graphs [17], but under arbitrary set-valued functions the coalition formation is both NP-hard and inapproximable [16].

Against this backdrop, in this paper we investigate a simple and natural model for assembling multiple teams with multi-dimensional skills that allows us to explore the computational complexity of multi-team formation as a function of the problem's intrinsic parameters: number of agents n, number of teams α, team size β, and dimension d of the skill vector. We place no constraints on the team structure except its prescribed size—any subset of agents can form a team—and use a simple additive function over independent attributes to measure team performance, thereby isolating the combinatorial aspects of the problem.

Specifically, we have an agent pool of n candidates, each modeled as a d-dimensional point $\mathbf{p} = (p_1, p_2, \ldots, p_d)$, where each dimension represents an independent real-valued skill. We want to form α teams, each of size β, for some integer values α, β, with $\alpha\beta \leq n$, so as to *maximize* the total score of all the teams. Each agent belongs to at most one team. In formulating the team score, we combine the two important aspects of a team performance: *strength* and *robustness* [3]. We measure the team strength by its coordinate-wise maxima but in order to add some degree of robustness we take the *top* h values for each coordinate, for a user-specified parameter $h \geq 1$. Thus, a team's score is defined as the *sum of h highest values of all dimensions.* We use the notation $\text{score}_h(T)$ to denote the score of team T using the top h scoring rule, which can be formally defined as

$$\text{score}_h(T) \quad = \quad \sum_{j=1}^{d} \max_{S \subset T, |S|=h} \sum_{\mathbf{p}_i \in S} p_j^i,$$

where p_j^i is the jth coordinate of the ith point $\mathbf{p}_i$. Our problem then is the following: given a set of n agents, form α teams $T_1, T_2, \ldots, T_\alpha$, each of size β to maximize $\sum_i \text{score}_h(T_i)$. Figure 1 shows an example in two di-

*Computer Science Department, University of California, Santa Barbara, CA 93106, USA, tschibler@gmail.com

†Computer Science Department, University of California, Santa Barbara, CA 93106, USA, ambuj@cs.ucsb.edu

‡Computer Science Department, University of California, Santa Barbara, CA 93106, USA, suri@cs.ucsb.edu

mensions, where (A, C, D) is an optimal team of size 3 for the instance on the left using scoring parameter $h = 2$.

Our Results

We show that the multi-team formation problem is NP-hard for dimension $d = \Omega(\log n)$, even with $h = 1$ and $\beta = 4$. Specifically, we reduce the well-known NP-complete problem of 3-Dimensional Matching to the team formation problem in dimension $d = \Omega(\log n)$. (If we consider very large dimensions, namely, $d = \Omega(n)$, the problem becomes trivially hard because simply acquiring all necessary skills is a set covering problem. In most realistic settings, however, the dimension is much smaller than n, which is the focus of our work.) Our main result is a polynomial time algorithm for solving the 2-dimensional team formation problem optimally, for all scoring rules $h \geq 1$, using the following two-step algorithm. We first form a single team of size $\alpha \times \beta$, which we call a *league*, using a modified scoring rule. We prove that the total score of the league equals the score of the optimal team formation, and that an optimal league can be decomposed into an optimal solution of the team formation in polynomial time.

Our dynamic programming based algorithm can compute an optimal league in any fixed dimension. However, we show that a key structural result, called *league decomposition* lemma, fails in higher dimensions, and so the optimal league's score no longer equals the score of the optimal team formation problem. Thus, forming multiple teams in more than two, but a constant, dimension remains an open problem.

2 Hardness of Team Formation

We begin with a brief reintroduction of the multi-team formation problem. Given an agent pool of n candidates, each candidate modeled as a d-dimensional point $\mathbf{p} = (p_1, p_2, \ldots, p_d)$, we want to form α teams, each of size β, for some integer values α, β, with $\alpha\beta \leq n$, so as to *maximize the sum* of team scores. Each agent belongs to at most team, and the score of a team T is defined as

$$\text{score}_h(T) \quad = \quad \sum_{j=1}^{d} \max_{S \subset T, |S|=h} \sum_{\mathbf{p}_i \in S} p_j^i,$$

where p_j^i is the jth coordinate of the ith point $\mathbf{p}_i$, and $h \geq 1$ is the scoring parameter.

Theorem 1 *The multi-team formation problem is NP-hard.*

Proof. We reduce the well-known 3-dimensional matching (3DM) [8] problem to our problem. An instance of 3DM consists of three input sets X, Y, Z each of size n and a set of triples $W \subset X \times Y \times Z$. The problem is to decide if there exists a subset of n triples $T \subseteq W$ so that each element of $X \cup Y \cup Z$ is contained in exactly one of the triples.

Given an instance of 3DM, we create an instance of the team formation problem as follows. The number of dimensions in our problem will be $6\ell+4$, for a parameter $\ell = \Theta(\log n)$. For each element of $X \cup Y \cup Z$, we associate a unique bit string of length 2ℓ, containing ℓ zeros and ℓ ones. We call this string the *tag* of that element. The bitwise complement of a tag t will be denoted t'. In particular, we will use the following types of bit strings:

1. $\text{tag}(x)$ = a unique bit string of length 2ℓ containing ℓ bits of 0 and ℓ bits of 1
2. $\text{tag}'(x)$ = bitwise complement of $\text{tag}(x)$
3. 0_ℓ = a string of length ℓ containing all 0

Using the fact that $\binom{2\ell}{\ell} \geq 2^\ell$, the choice of $\ell = 2 \log n$ suffices for the creation of $3n$ distinct tags, one for each element of $X \cup Y \cup Z$. With the help of these tags we now create a point for each element of $X \cup Y \cup Z$ and each triple $t = (u, v, w)$ of W, in dimension $6\ell + 4$, as follows:

x :	$\text{tag}(x)$	$0_{2\ell}$	$0_{2\ell}$	1	0	0	0
y :	$0_{2\ell}$	$\text{tag}(y)$	$0_{2\ell}$	0	1	0	0
z :	$0_{2\ell}$	$0_{2\ell}$	$\text{tag}(z)$	0	0	1	0
t :	$\text{tag}'(u)$	$\text{tag}'(v)$	$\text{tag}'(w)$	0	0	0	1

Specifically, the first 2ℓ dimensions of $x \in X$ are its tag bits, followed by 4ℓ bits of 0's, and its last four bits are 1 0 0 0. The patterns for $y \in Y$ and $z \in Z$ are similar, as shown above. Next, the point corresponding to a triple $t = (u, v, w)$ has 6ℓ bits corresponding to the tag' strings of u, v, w, followed by the pattern 0 0 0 1. Altogether we have $3n + |W|$ points in dimension $O(\log n)$, which is polynomial in the input size.

We now prove the following: the input 3DM instance is a yes instance if and only if our constructed instance admits formation of $\alpha = n$ teams, each of size $\beta = 4$, with total score at least $n(6\ell + 4)$. To prove the forward direction, suppose the 3DM instance has a solution given by the set of triples T. For each $t = \{x, y, z\} \in T$, we create a team of size 4 using the points corresponding to x, y, z and t. Since $|T| = n$, and no element appears in more than one triple, we can form n disjoint teams. We now show that these teams achieve the target total score.

Each point's coordinate is either 0 or 1 along each of the $6\ell + 4$ dimensions, and so to reach the target score, each team must collect a 1 in each dimension, using its four points. Suppose the four points correspond to x, y, z and the triple (x, y, z). Then, by construction, in each of the first 6ℓ dimensions, we have a 1 in either tag() or tag′(), satisfying the requirement. Finally, the same holds for the last four dimensions, which is easy to check by inspection. Thus, assuming that the 3DM instance has a satisfying solution, we can construct n teams, each of size 4 with total score $n(6\ell + 4)$.

In the reverse direction, we show that any set of n teams with this score correspond to a perfect 3 dimensional matching. First, we observe that the optimal score requires that every team contributes exactly one 1 in each dimension. Considering the last 4 dimensions alone, this is only possible if the team contains exactly one point corresponding to a triple and each of the 3 elements in X, Y, and Z. Given this team structure, each of the first 3 sets of 2ℓ dimensions must collect a 1 from either the tag or tag′ of some element or triple respectively. To satisfy all 2ℓ dimensions, the tag and the tag′ must correspond to the same element, otherwise they will not be bitwise complements of each other. Consequently, we must have elements x, y, z and triple $t = \{u, v, w\}$ with $x = u$, $y = v$, and $z = w$. If this property holds for all teams, then all selected triples must exactly cover each of the element sets, proving the existence of a 3DM solution. This completes the proof. □

The hardness proof is easily extended to any team size $\beta \geq 4$ by introducing an appropriate number of agents with null skills, namely, points with all 0 coordinates. The argument requires increasing the number of dimensions by $\Omega(\beta)$ to avoid the use of multiple triples in a single team. If the dimension is $\Omega(n)$, then computing a *single* team is also intractable. The proof can also be extended to scoring rule with $h > 1$ by introducing an appropriate number of points whose all coordinates are 1s.

3 An Efficient Algorithm for 2 Dimensions

Given the NP-hardness of the general problem, we now consider optimal team formation in small dimensions. In one dimension, the problem can be easily solved in $O(n \log n)$ time, as follows. We sort the agents in the increasing order of the skill level, say, the x axis. We then repeatedly select the top h unassigned agents, and assign them to the next team, until each of the α teams has h agents. Clearly, this assignment has the maximum sum of team scores. If needed, we can make each team's size to be exactly $\beta \geq h$, by arbitrarily selecting any of the unassigned agents since their scores do not contribute to the team scores.

In fact, a similar greedy strategy also solves the team formation problem for any dimension $d \geq 1$ if the team size is $\beta \geq hd$: repeat the earlier one-dimensional algorithm independently for each dimension. (It is possible for an agent to contribute a top score in more than one dimension, in which case a team may reach its maximum possible score with fewer than hd agents.) For team size $\beta < hd$, however, the problem becomes non-trivial even in dimension $d = 2$ and $h = 2$. This is the focus of the following discussion, where we consider the team formation problem in two dimensions.

3.1 Forming 3-person teams in 2-dimensions

In the interest of simpler exposition and proofs, we describe our algorithm using the scoring rule $h = 2$, and then discuss the minor adaptations needed for generalization to higher values of h. Therefore, in the following we drop the subscript h from the scoring notation; it is always assumed to be $h = 2$. Specifically, we focus on the case of team size $\beta = 3$, which helps illustrate some of the main difficulties of the problem. The case of $\beta = 1$ or $\beta = 2$ is easily solved greedily in two dimensions, and thus omitted from our discussion.

Somewhat surprisingly, the problem of forming teams of size 3 turns out to be non-trivial even if we want to form a single team, namely, $\alpha = 1$ with the scoring parameter $h = 2$. It serves as a test case both for disproving greedy schemes, and for our polynomial time algorithm. Using x and y as coordinates in two dimensions, suppose a 3-person team has agents with coordinates $(x_1, y_1), (x_2, y_2), (x_3, y_3)$. (Recall that we are using scoring rule of top two values, namely, $h = 2$.) Since the team score is composed of top two x and top two y values, and there are only 3 agents, at least one of them contributes both x and y to the team score. Let us call such an agent a 2-contributor. Each of the other agents contributes its x or y values (possibly these two agents are the same).

This property of the optimal 3-person team suggests a natural greedy algorithm: sort the agents by their x, y, and $x + y$ values. First take the agent with the maximum $x + y$, remove it from all three lists, and then choose the agents with the maximum x and maximum y. Unfortunately, this simple algorithm is flawed. In fact, one can show that any algorithm that selects the team using only the *rank order* by x, y and $x + y$ coordinates fails. In particular, we construct two instances, each containing 4 agents, whose sorted orders by x, y, and $x+y$ are identical, yet their top scoring teams are different. The construction is shown in Figure 1.

On the left, we have an instance with four agents $A = (4, 11), B = (5, 5), C = (1, 8), D = (8, 1)$. The optimal 3-person team for this instance is (A, C, D) with score of $31 = x(A) + x(D) + y(A) + y(C)$, with A contributing both x and y, C contributing y and D contributing x.

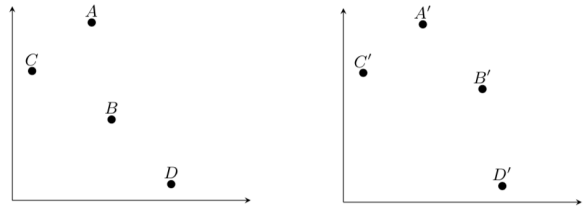


Figure 1: An example for $\alpha = 1, \beta = 3, h = 1$ and $d = 2$. On the left, we show an instance of team formation with four two-dimensional agents: $A = (4, 11), B = (5, 5), C = (1, 8), D = (8, 1)$. On the right, we show a closely related instance with $A' = (4, 11), B' = (7, 7), C' = (1, 8), D' = (8, 1)$. The two instances have exactly the same sorted orders along $x, y, x+y$, but they lead to different optimal 3-person team solutions. The optimal team for the left instance is (A, C, D) with score 31 while the team for the right instance is (A', B', D') with score 33.

On the right, we have another instance also with four agents, where only the coordinates of B' are different: $A' = (4, 11), B' = (7, 7), C' = (1, 8), D' = (8, 1)$, whose optimal team is (A', B', D') with score of $33 = x(B') + x(D') + y(A') + y(B')$. Yet, the two instances have the same ranking order by x, y, and $x+y$. *The crucial point of this example is that although A is an obvious choice for inclusion in the team, whether it contributes both x and y or just y depends on which other agents are in the team, namely, B or B'.*

Of course, since there are only $O(n^3)$ choices for a 3-person team, one can exhaustively find an optimal one. But what about forming α teams? Even for the simple sum-of-team-scores objective function, the greedy strategy of iteratively computing the best 3-person team among the remaining agents fails, as shown by the following example of six agents that we want to group into two teams of size 3.

$$A = (20, 20); B = (10, 20); C = (20, 10);$$

$$D = E = F = (0, 0)$$

The single optimal team is (A, B, C), with an score of 80, which leaves the remaining team of (D, E, F) with score 0. Instead, an optimal choice of two teams would be (A, B, D) and (C, E, F), which together have a score of 100.

3.2 A Polynomial Time Algorithm

In the following, we develop a polynomial time algorithm for solving the multi-team formation problem optimally in two dimensions. Our algorithm is based on the following idea:

1. First, identify the *union* of all the agents that are in the optimal set of teams, and then
2. Partition this union into individual teams while preserving the total score.

A 3-person team in dimension $d = 2$ involves a total of $2d = 4$ individual skill scores, namely, top two scores in each of the two dimensions. Across α teams, therefore, we have a total of 4α scores. Instead of forming these teams, let us consider a slightly different problem. Find a group of 3α agents whose score is computed as follows: for each dimension, we take the top 2α skill values, and the group score is the sum of these 4α values.

For ease of reference, let us call such a group of 3α agents with this new scoring rule a *league*. Given a league L, let score(L) be the total score of L. Suppose $\mathcal{T}$ is the optimal set of 3-person teams, with total score score($\mathcal{T}$). The question we ask is: what is the gap between score($\mathcal{T}$) and score(L)? Clearly, score($\mathcal{T}$) $\leq$ score(L), because the union of $\mathcal{T}$ is a valid league: a group of 3α agents, whose dimension-wise scores add up to score(L). But how much larger can the league score be compared to the team score? Our main result is that the two are always equal in two dimensions and, more importantly, (1) an optimal league can be partitioned in polynomial time into α teams of size $\beta = 3$ with the same total score, and (2) we can compute an optimal league in polynomial time. Together the two lead to a polynomial time algorithm.

3.3 Optimal League Decomposition

Let us first establish the league decomposition lemma.

Lemma 2 (League Decomposition) *Given an instance of multi-team formation in two dimensions, let $\mathcal{T}$ be an optimal solution of α teams of size 3 each, and let L be an optimal league of size 3α. Then,* score($\mathcal{T}$) = score(L). *We can also partition L into an optimal multi-team solution in time $O(n)$.*

Proof. The score of a league sums the top 2α values in each dimension. We label each point a 2-contributor, x-contributor, y-contributor, or none, depending on how many coordinate values it contributes to the league score. We then observe the following:

1. there are at least α 2-contributors.
2. there are an equal number of x and y-contributors, and this number is at most α.

The first claim follows from the pigeon hole principle: 4α values are summed in scoring the league, but there are only 3α points, and so at least α points must contribute both of their coordinates. This leaves at most 2α

values unaccounted for, which must be evenly split between x and y values. Thus, at most α values can come from an x-contributor, and α from a y-contributor.

```
1: procedure PARTITION 2D LEAGUE(p_1, ..., p_{3α})
2:     Initialize contributor lists X, Y, and XY.
3:     Initialize empty list of teams T.
4:     for i ∈ [0, len(X)] do
5:         Add team {XY[i], X[i], Y[i]} to T.
       j ← len(X).
6:     while j < len(XY) do
7:         Select any unused point p.
8:         Add team {XY[j], XY[j+1], p} to T.
9:         j ← j + 2.
10:    Return T.
```

Figure 2: Partitioning a league into optimal teams.

We use these two facts to design a simple greedy algorithm for partitioning the league. The algorithm is shown above in Fig. 2. We first pair any 2-contributor of the league with one x-contributor and one y-contributor. Because there are at least as many 2-contributors as x or y-contributors, we can continue this until there are no more 1-contributors left. By (2) above, we exhaust the x and y contributors at the same time. If any 2-contributors remain, we pair them arbitrarily together, along with an arbitrary extra point if we wish to maintain the team size.

To see that the resulting teams have the same total score as L, we note that exactly two x and two y values contributing to the league score are assigned to each team. Finally, the greedy algorithm only uses unsorted lists, and therefore runs in $O(n)$ time. This completes the proof. □

3.4 Computing an Optimal League

We now describe an algorithm for computing the optimal league L, using dynamic programming. Given a set of n d-dimensional points $\mathbf{p}_1, \cdots, \mathbf{p}_n$, we construct a 4-dimensional table A of size $n \times 3\alpha \times 2\alpha \times 2\alpha$ whose $A[i,j,k,l]$ entry stores $\text{score}_{k,l}(L_{i,j})$, where

$$\begin{aligned} L_{i,j} &= \text{an optimal league using at most } j \text{ points in } \{\mathbf{p}_1, \cdots, \mathbf{p}_i\} \\ \text{score}_{k,l}(L) &= \text{sum of top } k \text{ x-values and top } l \text{ y-values of } L \end{aligned}$$

The table is initialized as $L_{0,j} = L_{i,0} = 0$, for all i, j. Suppose we have computed all $L_{i-1,j-1}$ and want to compute $L_{i,j}$. Consider the new point $\mathbf{p}_i$. It is either not included in the league, or if it is included it serves in one of the three possible roles: x-contributor, y-contributor, or 2-contributor. We can, therefore, compute the table entry $L_{i,j}$ using the following dynamic program:

```
1: procedure 2D LEAGUE(p_1, ..., p_n, α)
2:     Initialize n × 3α × 2α × 2α table A
3:     Let A[0, j, k, l] = 0, ∀j, k, l
4:     Let A[i, 0, k, l] = 0, ∀i, k, l
5:     for i ∈ [1, n] do
6:         for j ∈ [1, 3α] and k, l ∈ [0, 2α] do
7:             s_x ← A[i−1, j−1, k−1, l] + p_i[x]
8:             s_y ← A[i−1, j−1, k, l−1] + p_i[y]
9:             s_{x,y} ← A[i−1, j−1, k−1, l−1] + p_i[x] +
   p_i[y]
10:            s_0 ← A[i−1, j, k, l]
11:            A[i, j, k, l] ← max(s_x, s_y, s_{x,y}, s_0)
12:    Return A[n, 3α, 2α, 2α]
```

Specifically, if $\mathbf{p}_i$ is an x-contributor, then $\text{score}_{k,l}(L_{i,j})$ is the x-coordinate of $\mathbf{p}_i$ plus the $\text{score}_{k-1,l}(L_{i-1,j-1})$; that is, the remaining points may only contribute $k-1$ x-values. We have similar cases for $\mathbf{p}_i$ being a y-contributor or a 2-contributor. The final optimal league score is found in the table entry $A[n, 3\alpha, 2\alpha, 2\alpha]$.

The table A has size $O(n\alpha^3)$, each entry can be computed in constant time, and so the algorithm runs in $O(n\alpha^3)$ time and space.

3.5 Extension to Top h Scoring Rule

The league decomposition lemma and the algorithm for computing the optimal league easily extend to scoring rule of top h, for all $h \geq 2$, as follows. Without loss of generality, we may assume that the team size satisfies $h \leq \beta < 2h$. Thus, the league size satisfies $\alpha\beta < 2\alpha h$. By the pigeonhole principle, the number of 2-contributors in the league is at least $\alpha(2h - \beta)$, and therefore we can assign to each team at least $(2h-\beta)$ of these 2-contributors, and fill the rest by 1-contributors arbitrarily. Similarly, the dynamic program algorithm is easily extended by changing the table size to $h\alpha$ instead of 2α. We summarize the main result of our paper.

Theorem 3 *The multi-team formation problem in two dimensions can be solved optimally in worst-case time and space $O(n\alpha^3\beta h^2)$, where α is the number of teams, β the team size, h the scoring parameter, and n is the number of agents.*

4 Team Formation in Higher Dimensions

The dynamic programming algorithm of Section 3.3 can be extended to form an optimal *single* team of size

$\beta < hd$ in polynomial time, for any fixed dimension d. Specifically, we compute a $(d+2)$-dimensional table A, whose first two dimensions are the same as before, namely, the first i points and the team size j. Each of the remaining d indices corresponds to the number of top scores in each dimension. In particular, $\text{score}_{k_1,\cdots k_d}$, where each $k_i \in [0,h]$, is the team score where top k_i values in dimension i have been accounted for. There are 2^d such combinations, so each table entry can be computed in $O(2^d)$ time. As mentioned earlier, when the team size $\beta \geq hd$, the problem can be easily solved in $O(dn)$ time using a greedy algorithm. We therefore have the following result.

Theorem 4 *We can compute an optimal single team of size β in d dimensions in time $O((2h)^d \beta n)$ time.*

The real difficulty in higher dimensions lies in forming *multiple teams*. In two dimensions, we used the League Decomposition Lemma as a key tool. Unfortunately, as we show below, in higher dimensions, this lemma no longer holds.

Theorem 5 *Let L be an optimal league, and $\mathcal{T}$ a set of optimal teams in dimensions $d \geq 4$. Then, there are instances for which* $\text{score}(\mathcal{T}) < \text{score}(L)$.

Proof. Consider the following set of 9 agents in four dimensions. $A = A' = (1,1,1,0), B = B' = (1,1,0,1), C = C' = (1,0,1,1), D = D' = (0,1,1,1)$, and $F = (0,0,0,0)$. Suppose our goal is to form $\alpha = 3$ teams, each of size $\beta = 3$. Then, trivially, our league consists of all p points, where the scoring rule sums the top $2\alpha = 6$ values in each dimension. By construction, we have six 1s in each dimension, and so $\text{score}(L) = 24$.

However, any partition of these nine agents into 3 teams must assign the all-zero point F to one of the teams, which can therefore have a score of at most 6. On the other hand, no team has score more than 8, since the sum of top two entries in each of the four dimensions is two. Thus, the optimal team formation has score 22, proving that $\text{score}(\mathcal{T}) < \text{score}(L)$. This completes the proof. □

One can also show that an optimal partition of an optimal league L may not give an optimal team formation solution $\mathcal{T}$. For instance, imagine introducing one more agent $G = (1,1,1,1)$ to the set of points in the previous example. Replacing F by G does not improve $\text{score}(L)$, so L remains an optimal league. On the other hand, replacing F by G does improve $\text{score}(\mathcal{T})$. Finding an efficient algorithm for optimal or approximately optimal team formation in a constant dimension larger than 2 remains an interesting open problem.

5 Concluding Remarks

Our work introduces a simple and natural model for multi-dimensional multi-team formation, and shows that computing optimal teams is NP-hard even in moderate dimensions. We show that the problem of forming multiple teams optimally can be efficiently solved in two dimensions, as is the problem of forming a single team in any dimension $d = O(\log n)$. The problem of forming multiple teams in higher than two dimension, either exactly or approximately, remains an interesting open problem.

There are several other natural objective functions for team optimization, such as maximizing the minimum team score, instead of maximizing the sum of team scores. For maximizing the minimum, unfortunately, we can show that the problem is NP-hard even in one dimension if we sum the top three scores of the team. The objective function can also be extended by considering other aspects of team formation that translate to more general constraints beyond individual-specific skills: for example, synergy between team members translates to edge-level requirements. Learning the skills and synergies based on past observations is another possible future extension of the research.

References

[1] Bryan L. Bonner, Michael R. Baumann, Austin K. Lehn, Daisy M. Pierce, and Erin C. Wheeler. Modeling collective choice: decision-making on complex intellective tasks. *European Journal of Social Psychology*, 36(5):617–633, 2006.

[2] Shi-Jie Chen and Li Lin. Modeling team member characteristics for the formation of a multifunctional team in concurrent engineering. *IEEE Transactions on Engineering Management*, 51(2):111–124, 2004.

[3] Chad Crawford, Zenefa Rahaman, and Sandip Sen. Evaluating the efficiency of robust team formation algorithms. In *Autonomous Agents and Multiagent Systems*, pages 14–29, Cham, 2016. Springer International Publishing.

[4] Christoph Dorn and Schahram Dustdar. Composing near-optimal expert teams: A trade-off between skills and connectivity. In *On the Move to Meaningful Internet Systems: OTM 2010*, pages 472–489, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[5] Erin L. Fitzpatrick and Ronald G. Askin. Forming effective worker teams with multi-functional skill requirements. *Computers & Industrial Engineering*, 48(3):593 – 608, 2005.

[6] Larson J.R., Jr. *In search of synergy: In small group performance*. Psychology Press, Taylor & Francis, New York, 2010.

[7] Jon Kleinberg and Maithra Raghu. Team performance with test scores. In *Proceedings of the Sixteenth ACM Conference on Economics and Computation*, EC '15, pages 511–528, 2015.

[8] Jon M. Kleinberg and Éva Tardos. *Algorithm design*. Addison-Wesley, 2006.

[9] Theodoros Lappas, Kun Liu, and Evimaria Terzi. Finding a team of experts in social networks. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, pages 467–476, 2009.

[10] Patrick R. Laughlin and Andrea B. Hollingshead. A theory of collective induction. *Organizational Behavior and Human Decision Processes*, 61(1):94 – 107, 1995.

[11] C. Li and M. Shan. Team formation for generalized tasks in expertise social networks. In *2010 IEEE Second International Conference on Social Computing*, pages 9–16, Aug 2010.

[12] Somchaya Liemhetcharat and Manuela Veloso. Modeling and learning synergy for team formation with heterogeneous agents. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS '12, 2012.

[13] Somchaya Liemhetcharat and Manuela Veloso. Weighted synergy graphs for effective team formation with heterogeneous ad hoc agents. *Artificial Intelligence*, 208:41 – 65, 2014.

[14] Scott Page. *The Difference: How the Power of Diversity Creates Better Groups, Firms, Schools, and Societies*. Princeton University Press, 2007.

[15] Habibur Rahman, Senjuti Basu Roy, Saravanan Thirumuruganathan, Sihem Amer-Yahia, and Gautam Das. Optimized group formation for solving collaborative tasks. *The VLDB Journal*, 28(1):1–23, February 2019.

[16] Tuomas Sandholm, Kate Larson, Martin Andersson, Onn Shehory, and Fernando Tohmé. Coalition structure generation with worst case guarantees. *Artif. Intell.*, 111(1-2):209–238, 1999.

[17] Travis C. Service and Julie A. Adams. Coalition formation for task allocation: theory and algorithms. *Autonomous Agents and Multi-Agent Systems*, 22(2):225–248, Mar 2011.

[18] Marjorie E. Shaw. A comparison of individuals and small groups in the rational solution of complex problems. *The American Journal of Psychology*, 44(3):491–504, 1932.

[19] I. D. Steiner. *Group process and productivity*. New York: Academic Press, 1972.

[20] Anita Williams Woolley, Christopher F. Chabris, Alex Pentland, Nada Hashmi, and Thomas W. Malone. Evidence for a collective intelligence factor in the performance of human groups. *Science*, 330(6004):686–688, 2010.

Simple Fold and Cut Problem for Line Segments

Guoxin Hu* Shin-ichi Nakano† Ryuhei Uehara* Takeaki Uno¶

Abstract

We investigate a natural variant of the fold-and-cut problem. We are given a long paper strip P and n line segments drawn on P such that each line segment is perpendicular to the two long edges of P and the distances between the line segments are not uniform. We cut all the line segments by one complete straight cut after overlapping all of them by a sequence of simple foldings. Our goal is to minimize the number of simple foldings. In this paper, we give algorithms for finding a shortest sequence of simple foldings for given n line segments. We first investigate the case that the distances are almost the same. In this case, our algorithm runs in $O(n^2)$ time and $O(n^2)$ space. Next we extend the algorithm for general distances. In general case, our algorithm runs in $O(n^3)$ time and $O(n^2)$ space.

1 Introduction

Take a sheet of paper, fold it flat, and then make one complete straight cut. What shapes can the unfolded pieces have? This fold-and-cut problem was introduced formally from the viewpoint of computational geometry in 1998 [5]. It is well known that there is a universal theorem for this problem, that is, any planar graph drawn by straight lines can be fold-and-cuttable. There are two major approaches to this problem in general form (see [7] for further details). However, in any of these approaches, to cut any shape, we need quite complicated folding operations so far, and hence it is quite difficult to realize by folding robots. From this viewpoint, the fold-and-cut problem for restricted ways of folding has been investigated. Demaine et al [4] only use a simple folding as a basic operation and investigate a connected simple polygon which can be a solution of the fold-and-cut problem. There are several models of simple folding [1], and precisely, they use all-layer simple folding, which is the simplest operation among them [2]. Demaine et al [4] focus on a connected simple polygon, and it was open for disconnected polygons.

In this paper, we consider the fold-and-cut problem for disconnected polygons, which is quite complicated in general form. Therefore, we start from the simplest problem in this framework. Intuitively, we introduce a one-dimensional version of this problem. (A similar idea is introduced for some paper folding problems; see [3, 6].) We are given a set of n parallel line segments on a long paper strip P. The given line segments are perpendicular to the two long edges of P, which give us the set of cut lines. A simple example is given in Figure 1(a). In this example, a paper strip P is of length 23, and 13 line segments are given as shown in the figure. We have the sequence of distances [1,1,2,2,2,1,2,2,2,2,2,1,2,1] as the input of this problem. In order to cut these line segments by one complete straight cut, we have to overlap all of these line segments on a line so that no other part overlaps on the line.

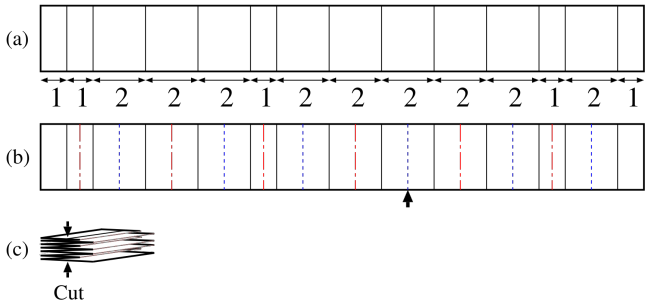


Figure 1: A simple example. The input of this problem is [1,1,2,2,2,1,2,2,2,2,2,1,2,1]. Mountain foldings and valley foldings are represented by dash-dot red lines and dash blue lines, respectively.

We employ the simplest folding model, which is called *all-layer simple folding*. Once we choose a crease line, all paper segments on the crease line are folded in the same direction. (See [1, 2] for further details of simple folding models.) From the practical viewpoint, all-layer simple folding is easier than the others, and hence there are some folding robots realizing it. Hereafter, we just use the term *simple folding* for simplicity.

We note that this problem always has a feasible solution achieved by a naive algorithm (Figure 1(b)) if the intervals satisfy some condition, which is discussed later: We first put crease lines halfway between two consecutive line segments, next fold along these crease lines alternately in mountain and valley folds, and obtain a pleat folding (Figure 1(c)). Then all given cut lines are on the same line, and we can cut them (and do not

*School of Information Science, Japan Advanced Institute of Science and Technology, Japan {huguoxin, uehara}@jaist.ac.jp
†Gunma University, Japan nakano@cs.gunma-u.ac.jp
¶National Institute of Informatics, Japan uno@nii.jp

cut any other paper) with one complete straight cut. Therefore, if we perform simple folding $n-1$ times, we obtain a valid folded state to be cut. However, we can sometimes reduce the number of simple foldings. For example, the crease line pointed by a bold arrow in Figure 1(b) can be folded first, and then the length of the paper strip is drastically reduced.

As we will see, the naive algorithm with $n-1$ foldings always works if the minimum distance $d_{\min}$ and the maximum distance $d_{\max}$ satisfy $d_{\max}/d_{min} \leq 2$. We say distances are *almost the same* when this condition is satisfied. (We note that the condition is sufficient, but not necessary.) However, for example, the input [10,1,1,10] does not satisfy this condition, and then the naive algorithm does not work anymore. However, even in this case, we can solve the problem: We first fold the leftmost paper segment in half 4 times. Then we have the paper strip represented by [0.625,1,1,10]. We fold again for the rightmost paper segment and obtain [0.625,1,1,0.625]. Then we can use the naive algorithm. We will see that any sequence of distances has a feasible solution for the simple fold and cut problem using this technique.

In this paper, we consider the problem for finding the optimal way of simple folding. That is, for any given set of n line segments on a paper strip, our aim is finding the shortest sequence of all-layer simple foldings to overlap all line segments on a line. When the distances are almost the same, by some observations, it is not difficult to construct a straightforward algorithm that runs in $O(n^4)$ time. We will give a non-trivial efficient algorithm for solving this problem in $O(n^2)$ time and $O(n^2)$ space. Next we extend this algorithm to the problem for general case. In general case, our extended algorithm finds a shortest simple foldings in $O(n^3)$ time and $O(n^2)$ space.

2 Preliminaries

Let $(d_0, d_1, \ldots, d_n)$ be the input of the problem. The paper strip P is of length $L = \sum_{i=0}^{n} d_i$. We regard the paper strip P as a line segment of length L which is placed on the interval $[0, L]$ on the x-coordinate. Let $\ell_0 = 0$ (and $\ell_{n+1} = L$) be the corresponding x-coordinate of the left (and right) endpoint of P, respectively. We also let $\ell_j = \sum_{i=0}^{j-1} d_i$ for $0 < j < n$. Namely, ℓ_j gives the x-coordinate of the j-th line segment. We sometimes call paper strip between ℓ_i and ℓ_{i+1} *the ith paper segment*.

We define the point f_j as the middle point between two consecutive line segments ℓ_j and ℓ_{j+1} with $0 \leq j \leq n$ (precisely, $f_j = (\ell_j + \ell_{j+1})/2$). Except the 0th and the nth segment, we can assume that our algorithm always folds P at some f_j (otherwise, we cannot make a simple fold and cut anymore).

We here note that ℓ_0 and ℓ_{n+1} are not the line segments to be cut. In a sense, they are already cut, and we do not need to fold along the line at f_0 and f_n. On the other hand, when d_0 and d_n are quite large, we cannot use the simple pleat folding algorithm stated in the introduction. Once we fold along a line f_i, the paper segment in $[0, d_0]$ or $[L - d_n, L]$ may cover some other crease lines, and we cannot make any simple fold anymore. When they cover no crease line, the line f_i becomes the edge of the paper strip and it plays the same role then. This issue will be discussed in Section 4. We first assume on the distances to avoid this issue. Let $d_{\min} = \min\{2d_0, d_1, d_2, \ldots, d_{n-1}, 2d_n\}$ and $d_{\max} = \max\{2d_0, d_1, d_2, \ldots, d_{n-1}, 2d_n\}$. When we have $1 \leq d_{\max}/d_{\min} \leq 2$, we can use the simple pleat folding algorithm as follows. When we fold the paper strip by a simple folding along the leftmost crease line f_i at the ith step, the left paper segment from f_i is of length at most $\max\{d_0, d_{\max}/2\}$, and the $(i+1)$st paper segment is of length at least $d_{\min} \geq \max\{d_{\max}/2, d_n\}$. Therefore, the leftmost paper segment does not cover the crease line f_{i+1}. We call the distances are almost the same in this situation. Now we are ready to state our problem:

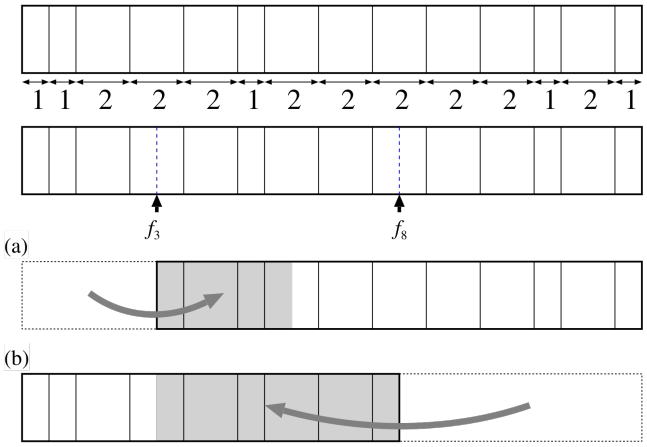


Figure 2: Two simple foldings along (a) f_3 and (b) f_8. Two sheets are overlapping in gray areas.

Input: A paper strip P with parallel n lines $\ell_1, \ldots, \ell_n$, where d_i is the distance between ℓ_i and ℓ_{i+1} for each $d = 0, \ldots, n$ (ℓ_0 and ℓ_{n+1} denote the left and right edges of P).

Operation: All-layer simple folding.

Goal: Finding a shortest sequence of simple foldings that overlaps all lines ℓ_i $(1 \leq i \leq n)$ on a line, and no other paper segment is on the line.

We first consider the case that the distances are almost the same. That is, we only make a simple folding along some crease line f_i. We assume that the (folded) paper strip P is on the interval $[i, j]$ for some $i < j$.

When our algorithm makes a simple fold P at f_k with $i < f_k < j$, it flips the left part of P at the crease f_k if $f_k \leq (i+j)/2$, and it flips the right part of P at f_k if $f_k > (i+j)/2$. Then, we can observe that P always shrinks without changing its position. Precisely, we have the following observation:

Observation 1 *Assume P is placed on the interval $[i, j]$ for some $i < j$. After making a simple folding, P is placed on the interval $[i', j']$ such that either (1) $i < i' \leq (i+j)/2$ and $j' = j$ or (2) $i' = i$ and $(i+j)/2 < j' < j$. Moreover, the sequence of line segments (or distances) in $[i', j']$ is not changed by the simple folding.*

Two examples are shown in Figure 2. We here note that in this simple folding, the direction (mountain or valley) of simple folding does not matter. Therefore, we do not consider the direction of each folding hereafter. Intuitively, we start from a paper strip P placed on $[0, L]$, the interval shrinks after each simple folding, and eventually, we obtain the paper strip placed on $[f_j, f_{j+1}]$ for some j. Then we can make one complete straight cut of all given line segments (without cutting any other part). Our goal is finding the shortest sequence of f_ks for it.

Let $(d_0, d_1, \ldots, d_n)$ be the input of the problem, and $(f_1, \ldots, f_{n-1})$ be the middle points. At the point f_i with $f_i \leq L/2$, we can fold P at f_i only if every pair of corresponding line segments overlaps. More precisely, we say that P is *simple foldable at f_i* if and only if $d_{i+j} = d_{i-j}$ for every $j = 1, 2, \ldots, i-1$ and $d_{2i} \geq d_0$ (since d_0 is the length of the leftmost paper segment)[1]. Such a simple folding at f_i is said to be *valid*. We can define it for a point f_i with $f_i > L/2$ in the same way. We can also define a valid simple folding for a folded P in the same manner. When a folded state of P is obtained by a sequence of valid simple foldings, the folded state is also said to be *valid*. Let f_i be a valid point of P with $f_i \leq L/2$. Then it is easy to see that the sequence $(d_1, \ldots, d_i, \ldots, d_{2i-1})$ is a *palindrome* of odd length, and $d_0 \leq d_{2i}$.

In this paper, palindromes of odd lengths play an important role. For a given string $S = (s_0, s_1, \ldots, s_n)$, a *maximal palindrome centered at s_i* is defined by a maximal palindrome of odd length at center s_i in S. We will use the following result by Manacher:

Theorem 1 ([8]) *For a given string $S = (s_0, s_1, \ldots, s_n)$ of length n, let p_i be the length of the maximal palindrome centered at s_i. Then the sequence $(p_0, p_1, \ldots, p_n)$ can be computed in $O(n)$ time and $O(n)$ space.*

[1]We may have a special case that $2i = n$ (see Figure 3(c)). In this case, the length of this paper segment is changed to $\max\{d_0, d_n\}$. The management of this special case is trivial and hence we do not consider in this paper hereafter.

We here note that, by Observation 1 and Theorem 1, we can solve the simple fold and cut problem for a given P in $O(n^4)$ time and $O(n^2)$ space if the distances are almost the same:

Proposition 2 *If the distances are almost the same, there is an algorithm for solving the simple fold and cut problem for P in $O(n^4)$ time and $O(n^2)$ space.*

Proof. For the input of the problem, let $\mathcal{P}$ be the set of intervals $[f_i, f_j]$ for each $0 \leq f_i < f_j \leq L$. By Observation 1, each valid folded state P can be represented by an interval $[f_i, f_j]$ for some $0 \leq f_i < f_j \leq L$. Now let $G = (\mathcal{P}, E)$ be a directed graph defined as follows. For each pair of $P_1, P_2 \in \mathcal{P}$, $(P_1, P_2) \in E$ if the folded state represented by P_2 can be achieved from the folded state represented by P_1 by a simple folding. Then the solution of the problem is given by a shortest path on G from the interval $[0, L]$ to an interval $[f_i, f_{i+1}]$ for some i. Since $|\mathcal{P}| = O(n^2)$ and each vertex P_i is of degree $O(n)$, the construction of G takes $O(n^3)$ time with $O(n^2)$ space with precomputation of the sequence of lengths of maximal palindromes by Theorem 1. Then the shortest path problem can be solved in $O(n^4)$ time with $O(n^2)$ space by the breadth first search. □

3 Almost the Same Case

We will improve the running time in Proposition 2 from $O(n^4)$ time to $O(n^2)$ time. That is, the main theorem in this section is as follows:

Theorem 3 *There is an algorithm for solving the simple fold and cut problem for P when the distances are almost the same in $O(n^2)$ time and $O(n^2)$ space.*

Here we show a simple but crucial lemma.

Lemma 4 *Let $(d_0, d_1, \ldots, d_n)$ be the input of the problem, and $(f_1, \ldots, f_{n-1})$ be the middle points. Let P' be a folded state of P placed on $[f_i, f_j]$. We assume that two simple foldings at f_k and $f_{k'}$ are both valid for some k and k' with $i < k < k' < (i+j)/2$. Then for any valid simple folding sequence $F = (f_k, \ldots)$, we have another valid simple folding sequence $F' = (f_{k'}, \ldots)$ that is as short as F.*

Proof. In F, after first valid simple folding at f_k, we obtain the folded state P'' in the interval $[f_k, f_j]$. On the other hand, after $f_{k'}$, the folded state P''' is in the interval $[f_{k'}, f_j]$ with $f_k < f_{k'}$. Then, by Observation 1, P''' is the subsequence of P''. Precisely, the latter part $[f_{k'}, f_j]$ in P'' is the same as P'''. Therefore, we first replace the first f_k in F by $f_{k'}$ and remove all valid simple foldings $f_{k''}$ with $f_k < f_{k''} < f_{k'}$, then we obtain a valid simple folding sequence F'. Then F' is as short as F, which completes the proof. □

We can obtain the same lemma for the right part of P. Therefore, when we consider the shortest sequence of valid simple foldings for any folded state P' of P placed on $[f_i, f_j]$, it is sufficient to consider two valid simple foldings at f_l and f_r, where f_l is the maximum valid simple foldable point with $f_l \leq (f_i + f_j)/2$ and f_r is the minimum valid simple foldable point with $(f_i + f_j)/2 < f_r$.

3.1 Data Structures for Our Algorithm

Let $(d_0, d_1, \ldots, d_n)$ be the input of the problem, and $(f_1, \ldots, f_{n-1})$ be the middle points on P. Our algorithm is based on dynamic programming. We will use the following tables.

Palindromes: To check valid simple folding, we will use the sequence $(p_0, p_1, \ldots, p_{n-1})$, where p_i is the length of the maximal palindrome centered at f_i for each $i = 0, \ldots, n-1$. By Theorem 1, the sequence can be computed in $O(n)$ time and $O(n)$ space.

LLine, RLine: For positive integers i and l, LLine$[i][l]$ indicates whether the paper segment $[\ell_i, \ell_{i+l}]$ can be folded to left along the line f_{i+l} or not. Intuitively, when LLine$[i][l] = 1$, the paper segment $[\ell_i, \ell_{i+2l}]$ gives us a palindrome with respect to the distances. More precisely, we define LLine$[i][l] = 1$ if $d_i = d_{i+2l}$, $d_{i+1} = d_{i+2l-1}$, ..., and $d_{i+l-1} = d_{i+l+1}$. Otherwise, we define LLine$[i][l] = 0$. We note that if $L < i + 2l$, we define LLine$[i][l] = 0$. For the sake of simplicity, we also define LLine$[i][0] = 1$ for any i.

In the same manner, we also define RLine$[i][l]$. Precisely, we define RLine$[i][l] = 1$ if $d_i = d_{i-2l}$, $d_{i-1} = d_{i-2l+1}$, ..., and $d_{i-l+1} = d_{i-l-1}$.

For the initial state of P where P is placed on an interval $[0, L]$, by Lemma 4, we can observe that the shortest valid simple folding sequence starts from either the crease line f_l or f_r such that (0) let f_m be the central crease line, that is, $m = \lfloor n/2 \rfloor$, (1) f_l is the maximum index in $0 \leq l \leq m$ such that LLine$[0][l - i] = 1$, and (2) f_r is the minimum index in $m < r \leq n$ such that RLine$[n][n - r] = 1$. This observation holds for general folded state of P as follows.

Lemma 5 *Let P' be the folded state of P which is obtained by some simple foldings in the manner stated in Observation 1. That is, we can assume P' is placed on an interval $[f_i, f_j]$ for some crease lines f_i and f_j with $i < j$. Let f_m be the central crease line given by $m = \lfloor (i + j)/2 \rfloor$. Then the shortest valid simple folding sequence starts from either the crease line $f_l(i, j)$ or $f_r(i, j)$ such that (1) $f_l(i, j)$ is the maximum index in $i \leq l \leq m$ such that* LLine$[i][l - i] = 1$*, and (2) $f_r(i, j)$ is the minimum index in $m < r \leq j$ such that* RLine$[j][j - r] = 1$.

Proof. By Lemma 4 and the definitions of LLine, RLine, it follows. □

Let $\mathsf{sf}[i][j]$ be the minimum number of simple foldings of the folded state P' placed on an interval $[f_i, f_j]$ of the paper strip P. Then by Lemma 5, we have the following observation.

Observation 2 *The minimum number of simple foldings is given by $\mathsf{sf}[0][n-1]$ which satisfies the following recursive definitions.*

$$\begin{aligned}
&\mathsf{sf}[i][j] = \infty \text{ if } i > j,\\
&\mathsf{sf}[i][j] = 0 \text{ if } j = i,\\
&\mathsf{sf}[i][j] = \min\{\\
&\qquad \mathsf{sf}[f_l(i,j)+1][j], \mathsf{sf}[i][f_r(i,j)-1]\\
&\qquad \} + 1 \text{ otherwise,}
\end{aligned}$$

where $f_l(i, j)$ and $f_r(i, j)$ are defined as shown in Lemma 5.

Hereafter, we will show the implementation of the computation of Observation 2 which runs in $O(n^2)$ time and $O(n^2)$ space.

3.2 Algorithm Description and Computational Complexity

In this section, we prove Theorem 3 by showing the details of implementation of our main algorithm. It computes the minimum number of simple foldings by computing $\mathsf{sf}[0][n-1]$ in Observation 2. The algorithm consists of initialization of auxiliary arrays and computation of $\mathsf{sf}[i][j]$.

3.2.1 Initialization step

For a given input $(d_0, d_1, \ldots, d_n)$, the algorithm first computes the middle points $(f_1, \ldots, f_{n-1})$. Next it computes the sequence $(p_0, p_1, \ldots, p_{n-1})$, where p_i is the length of the maximal palindrome centered at f_i for each $i = 0, \ldots, n-1$. By Theorem 1, the sequence can be computed in $O(n)$ time and $O(n)$ space. The algorithm next computes LLine$[i][l]$ for each i and l. (We prepare a two-dimensional array to store the table LLine$[i][l]$.) By the definition of LLine$[i][l]$, LLine$[i][l] = 1$ if and only if $p_{i+l} > 2l$. Therefore, it can be computed in $O(n^2)$ time with $O(n^2)$ space.

In order to compute the maximum and minimum indices in Lemma 5, we use two auxiliary tables (or arrays) FL(i, j) and FR(i, j). For $m = \lfloor (i + j)/2 \rfloor$, each element FL$(i, j)$ gives the maximum index l in $[i, m]$ such that LLine$[i][l - i] = 1$, and FR(i, j) gives the minimum index r in $[m + 1, j]$ such that RLine$[j][j - r] = 1$. For a fixed i, it is not difficult to see that FL(i, j) is changed from FL$(i, j - 1)$ only when the corresponding LLine$[i][m - i] = 1$, otherwise, FL(i, j) =FL$(i, j -$

1). More precisely, we have (0) $\textsc{Fl}(i,0) = i$ (since $\textsc{LLine}[i][0] = 1$), (1) $\textsc{Fl}(i,j) = m$ if $\textsc{LLine}[i][m-i] = 1$, and (2) $\textsc{Fl}(i,j) = \textsc{Fl}(i,j-1)$ if $\textsc{LLine}[i][m-i] = 0$. Therefore, the table $\textsc{Fl}(i,j)$ can be computed in $O(n^2)$ time and $O(n^2)$ space. For the table $\textsc{Fr}(i,j)$, we can use a symmetric argument, and it can be computed in the same manner.

3.2.2 Computation of $\mathsf{sf}[i][j]$

Now we turn to the computation of the minimum number of simple foldings. It can be done by computing the array $\mathsf{sf}[i][j]$. Using the auxiliary arrays in the previous section, we can describe this computation as follows: $\mathsf{sf}[i][i] = \mathsf{sf}[i][i+1] = 0$ for each i, and

$$\mathsf{sf}[i][j] = \min\{\mathsf{sf}[\textsc{Fl}(i,j)+1][j], \mathsf{sf}[i][\textsc{Fr}(i,j)-1]\} + 1 \quad \text{for each } j = i, \ldots, n-1$$

This equation is a recursive function with respect to the interval $[i,j]$. That is, $\mathsf{sf}[i][j]$ is defined by $\mathsf{sf}[i'][j]$ and $\mathsf{sf}[i][j']$ such that $i < i'$ and $j' < j$. Therefore, we can use the standard dynamic programming technique with respect to the length of the interval. That is, the algorithm first initializes all elements of $\mathsf{sf}[i][i] = \mathsf{sf}[i][i+1] = 0$ and next computes all elements of $\mathsf{sf}[i][i+2]$, then computes all elements of $\mathsf{sf}[i][i+3]$, and so on. This can be done in $O(n^2)$ time and $O(n^2)$ space, which completes the proof of Theorem 3.

4 General Case

In the general case, we have to take care of the case that the leftmost paper segment or the rightmost paper segment covers some lines ℓ_i to be cut after folding.

To avoid this, the leftmost or rightmost paper segment has to be simple folded locally to shrink its length. Some simple examples are given in Figure 3. We here note that any intermediate paper segment, say the ith paper segment between ℓ_i and ℓ_{i+1}, cannot be shrunk by any all-layer simple fold: If we fold along some crease line except f_i, either ℓ_i or ℓ_{i+1} should overlap with some paper inside of this segment. Then we cannot separate this cut line by any all-layer simple fold, and hence we will fail to perform one straight cut for this line. That is, if we want to shrink the ith paper segment, we have to fold along f_i first, and make it the leftmost or rightmost paper segment of length $d_i/2$ in the folded state. Then we can observe the following:

Observation 3 *For any given numbers d and D with $d < D$, the minimum number of simple foldings to make a paper segment of length D to one of length at most d is $\lceil \log_2 D - \log_2 d \rceil$.*

Proof. It is easy to see that to reduce the length of a paper segment of length D to at most d, the optimal way is to repeat in simple folding in half k times such that k is the minimum integer with $D/2^k < d$. Solving this, we obtain the claim. □

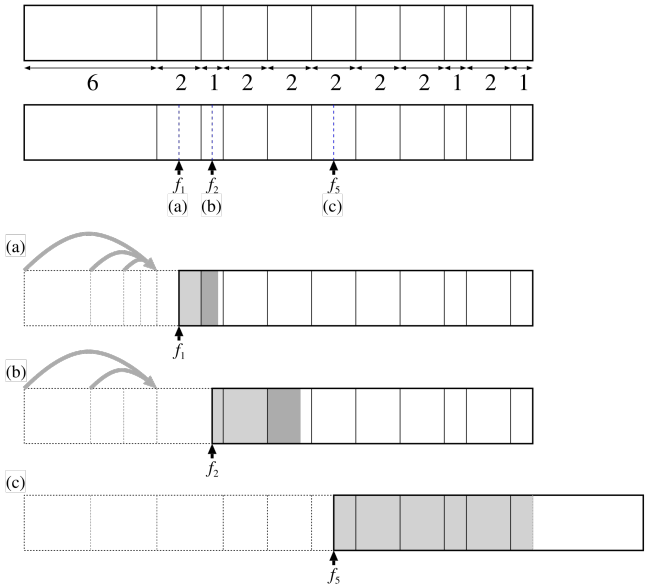


Figure 3: We have to roll up the leftmost paper segment of length 6; it should be folded in half (a) 3 times to make it of length $6/2^3 = 0.75 < 1$ when we fold it along the crease line f_1, (b) 2 times to make it of length $6/2^4 = 1.5 < 2$ when we fold it along the crease line f_2, and (c) 0 times when we fold it along the crease line f_5.

In the case that distances are almost the same, one simple folding can always be done at any f_i if it is valid. In general case, as observed above, it is not necessarily true anymore. We suppose that the folded state P' is on an interval $[i,j]$ as shown in Observation 1, and we want to make a simple folding along f_k for some k in (i,j). Then we have two basic operations that consist of two phases: we first shrink the leftmost paper strip and make a simple fold along f_k when $k \le (i+j)/2$, or we first shrink the rightmost paper strip and make a simple fold along f_k when $k > (i+j)/2$. The extra simple foldings can be estimated by Observation 3. In order to describe this extra foldings, we introduce a function $\mathsf{ef}(d_i, d_j)$ as follows. If d_i is short enough, we need no extra foldings. We have three cases; we define $\mathsf{ef}(d_i, d_j) = 0$ when (1) $0 < i < n$ and $(d_i/2) \le d_j$, (2) $i = 0$ and $j = n$, (3) $i = 0$, $j < n$, and $d_0 \le d_j$, or (4) $i = n$, $j > 0$, and $d_n \le d_j$. In the other cases, we need extra foldings as follows. When $0 < i < n$ and $(d_i/2) > d_j$, we define $\mathsf{ef}(d_i, d_j) = \lceil \log_2(d_i/2) - \log_2 d_j \rceil$. When $i = 0$ or $i = n$ (with $j \neq 0$ and $j \neq n$), we define $\mathsf{ef}(d_i, d_j) = \lceil \log_2 d_i - \log_2 d_j \rceil$ when $d_i > d_j$.

Now we are ready to show the main theorem in this section:

Theorem 6 *There is an algorithm for solving the simple fold and cut problem for P in general distances in $O(n^3)$ time and $O(n^2)$ space.*

Proof. The basic strategy is the same as the proof of Theorem 3. The key issue is that we cannot have the monotone property used in the proof of Lemma 4 anymore. When we have two valid foldings at f_k and $f_{k'}$ with $i < k < k' < (i+j)/2$ on the folded state P' on $[i, j]$ except the leftmost paper segment, f_k can be better choice than $f_{k'}$ if $\mathsf{ef}(d_i, d_{2k-i})$ is much smaller than $\mathsf{ef}(d_i, d_{2k'-i})$. Thus we cannot use Lemma 5.

However, we use the following weaker claim. Let P' be the folded state of P which is obtained by some simple foldings, in the manner stated in Observation 1. That is, we can assume P' is placed on an interval $[f_i, f_j]$ for some crease lines f_i and f_j with $i < j$. Let f_m be the central crease line given by $m = \lfloor (i+j)/2 \rfloor$. Then, since no intermediate paper segment can be folded by an all-layer simple folding along a line except f_is, the shortest valid simple folding sequence starts from either the crease line $f_l(i, j)$ or $f_r(i, j)$ such that (1) $f_l(i, j)$ is an index in $i \leq l \leq m$ such that $\textsc{LLine}[i][l - i] = 1$, and (2) $f_r(i, j)$ is an index in $m < r \leq j$ such that $\textsc{RLine}[j][j - r] = 1$.

Thus, we can use the same technique for finding the shortest sequence of simple foldings maintained by an array $\mathsf{sf}[i][j]$ which is defined by $\mathsf{sf}[i][i] = 0$ for each i, and

$$\begin{aligned}\mathsf{sf}[i][j] = \min\{ & \\ & \mathsf{sf}[i'][j] + \mathsf{ef}(d_i, d_{2i'-i}), \mathsf{sf}[i][j'] + \mathsf{ef}(d_j, d_{2j'-j}) \\ & \} + 1\end{aligned}$$

for each $i' = i+1, \ldots, m$ and $j' = m+1, m+2, \ldots, j-1$. The correctness follows from the above discussion. This equation is still a recursive function with respect to the interval $[i, j]$. That is, $\mathsf{sf}[i][j]$ is defined by $\mathsf{sf}[i'][j]$ and $\mathsf{sf}[i][j']$ such that $i < i'$ and $j' < j$. Therefore, we can use the standard dynamic programming technique with respect to the length of the interval. This can be done in $\mathrm{O}(n^3)$ time and $\mathrm{O}(n^2)$ space. □

5 Concluding Remarks

In this paper, we investigated a one-dimensional variant of the simple fold and cut problem. We use the simplest model for simple folding. When we use the other models for simple folding discussed in [1], we will have a different problem. For example, the naive algorithm stated in the introduction always works with $n - 1$ foldings for some simple folding models stronger than all-layer simple folding model. In this case, finding a shortest sequence of simple foldings seems to be a completely different problem, which is an interesting open problem.

On any simple folding model, the simple fold and cut problem for multiple simple polygons in two-dimensional space is still open. The authors conjecture that finding a shortest sequence of simple foldings for this general two-dimensional case is NP-complete.

Acknowledgements

The third author was supported in part by MEXT/JSPS KAKENHI Grant Number 17H06287 and 18H04091. This work was discussed at the 4th Workshop on Combinatorics and Enumeration Algorithms held on February 21–23, 2019 in Karuizawa, Japan supported by NII joint research fund. We thank the other participants of that workshop for providing a stimulating research environment.

References

[1] H. A. Akitaya, E. D. Demaine, and J. S. Ku. Simple Folding is Really Hard. *Journal of Information Processing*, 25:580–589, 2017.

[2] E. M. Arkin, M. A. Bender, E. D. Demaine, M. L. Demaine, J. S. Mitchell, S. Sethia, and S. S. Skiena. When can you fold a map? *Computational Geometry*, 29(1):23–46, 2004.

[3] J. Cardinal, E. D. Demaine, M. L. Demaine, S. Imahori, T. Ito, M. Kiyomi, S. Langerman, R. Uehara, and T. Uno. Algorithmic Folding Complexity. *Graphs and Combinatorics*, 27:341–351, 2010.

[4] E. D. Demaine, M. L. Demaine, A. Hawksley, H. Ito, P.-R. Loh, S. Manber, and O. Stephens. Making polygons by simple folds and one straight cut. In *Computational Geometry, Graphs and Applications*, pages 27–43. Springer, 2011.

[5] E. D. Demaine, M. L. Demaine, and A. Lubiw. Folding and cutting paper. In *Japanese Conference on Discrete and Computational Geometry*, pages 104–118. Springer, 1998.

[6] E. D. Demaine, D. Eppstein, A. Hesterberg, H. Ito, A. Lubiw, R. Uehara, and Y. Uno. Folding a Paper Strip to Minimize Thickness. In *The 9th Workshop on Algorithms and Computation (WALCOM 2015)*, pages 113–124. Lecture Notes in Computer Science Vol. 8973, Springer-Verlag, 2015.

[7] E. D. Demaine and J. O'Rourke. *Geometric Folding Algorithms: Linkages, Origami, Polyhedra*. Cambridge University Press, 2007.

[8] G. Manacher. A new linear-time"on-line"algorithm for finding the smallest initial palindrome of a string. *Journal of the ACM (JACM)*, 22(3):346–351, 1975.

Rectangular Unfoldings of Polycubes

Martin L. Demaine* Robert Hearn† Jason Ku‡ Ryuhei Uehara§

Abstract

In this paper, we investigate the problem that asks if there exists a net of a polycube that is exactly a rectangle with slits. For this nontrivial question, we show affirmative solutions. First, we show some concrete examples: (1) no rectangle with slits with fewer than 24 squares can fold to any polycube, (2) a 4×7 rectangle with slits can fold to a heptacube (nonmanifold), (3) both of a 3×8 rectangle and a 4×6 rectangle can fold to a hexacube (nonmanifold), and (4) a 5×6 rectangle can fold to a heptacube (manifold). Second, we show a construction of infinite family of polycubes folded from a rectangle with slits. The smallest one given by this construction is a 6×20 rectangle with slits that can fold to a polycube of genus 5. This construction gives us a polycube for any positive genus. Moreover, by this construction, we can show that there exists a rectangle with slits that can fold to k different polycubes for any given positive integer k.

1 Introduction

It is well known that a unit cube has eleven edge developments. When we unfold the cube, no overlap occurs on any of these eleven developments. In fact, any development of a regular tetrahedron is a tiling, and hence no overlap occurs [1]. However, this is not necessarily true for general polycube, which is a polyhedron obtained by face-to-face gluing of unit cubes. For example, we can have an overlap when we unfold a box of size $1 \times 1 \times 3$ (Figure 1), while we have no overlap when we unfold a box of size $1 \times 1 \times 2$ (checked by exhaustive search). On the other hand, even for the Dali cross (3-dimensional development of 4-dimensional hyper cube), there is a non-overlapping unfolding that is a polyomino with slits that satisfies Conway's criterion in the induced plane tiling [2].

In this context, we investigate a natural but nontrivial question that asks if we can fold a polycube from a rectangle with slits or not. We first note that a convex polycube (or a "box") cannot be folded from any rectangle with slits. In general, any slit has no meaning of a development of a convex polycube as proved in [3, Lemma 1]. Therefore, a rectangle cannot fold to any convex polycube even if we make slits in any way. That is, if the answer to the question is yes, the polycube should be concave.

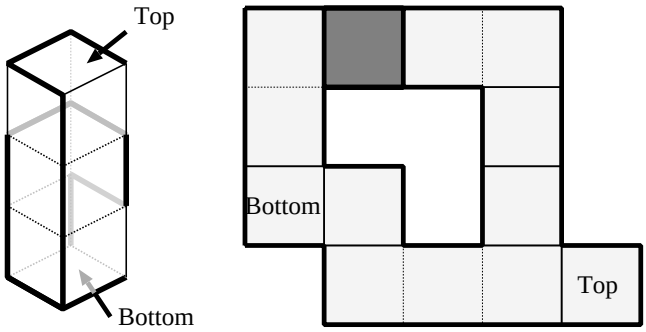


Figure 1: Cutting along the bold lines of the left box of size $1 \times 1 \times 3$, overlap occurs at the dark gray square on the right development. This development was first found by Takeaki Uno in 2008. We have four places to glue the top, however, this development is essentially unique way for this box to overlap except the place of the top, which is examined by exhaustive search.

In this paper, we show two series of affirmative answers to the question. First, we develop an algorithm that searches slits of a given rectangle to fold a polycube. Based on the algorithm, we find some concrete slit patterns:

Theorem 1 *(1) No rectangle with slits with fewer than 24 squares can fold to any polycube, (2) a 4×7 rectangle with slits can fold to a heptacube, which is nonmanifold, (3) both of a 3×8 rectangle and a 4×6 rectangle can fold to a hexacube, which is also nonmanifold, and (4) a 5×6 rectangle can fold to a heptacube, which is manifold.*

Second, we show a construction of infinite family of polycubes folded from a rectangle with slits.

Theorem 2 *For any positive integer g, there is a rectangle with slits that can fold to a polycube of genus g.*

As a result, we can conclude that there are infinite many polycubes that can be folded from a rectangle with slits. In the construction in Theorem 2, we use a series of

*Computer Science and Artificial Intelligence Lab, MIT, mdemaine@mit.edu

†bob@hearn.to

‡Electrical Engineering and Computer Science Department, MIT, jasonku@mit.edu

§School of Information Science, Japan Advanced Institute of Science and Technology, uehara@jaist.ac.jp

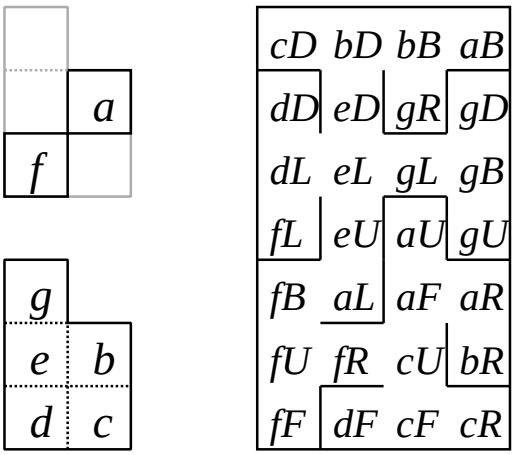


Figure 2: The right rectangle is an unfolding of the left polycube of volume 7. The left figures describe the polycube by slices of it.

gadgets that have many different ways of folding. Using this property, we also have the following corollary.

Corollary 3 *For any positive integer k, there is a rectangle with slits that can fold to k different polycubes.*

2 Proof of Theorem 1

We first show the results and give a brief idea of the algorithm that we used for finding the patterns.

2.1 Pattern 1: 4×7 rectangular unfolding of a heptacube

In Figure 2, we give a heptacube that has a 4×7 rectangular unfolding. Cubes a and f touch along a diagonal. In the unfolding, D, B, R, L, U, F mean Down, Back, Right, Left, Up, Front, respectively. This heptacube has 90 rectangular unfoldings.

2.2 Patterns 2 and 3: 3×8 and 4×6 rectangular unfoldings of a symmetric hexacube of genus 1

In Figure 3, we give a hexacube that has two rectangular unfoldings. One is of size 3×8 and the other is of size 4×6. This polycube has no diagonal touch, although it is genus 1 at the central point. There are 1440 rectangular unfoldings, and one of each type is shown in Figure 3. (This 24-face hexacube has 12 symmetries; therefore, the number of distinct rectangular unfoldings is 120 rather than 1440.)

2.3 Pattern 4: 5×6 rectangular unfolding of a symmetric heptacube

In Figure 3, we give a heptacube that has rectangular unfolding of size 5×6. This polycube has no diagonal

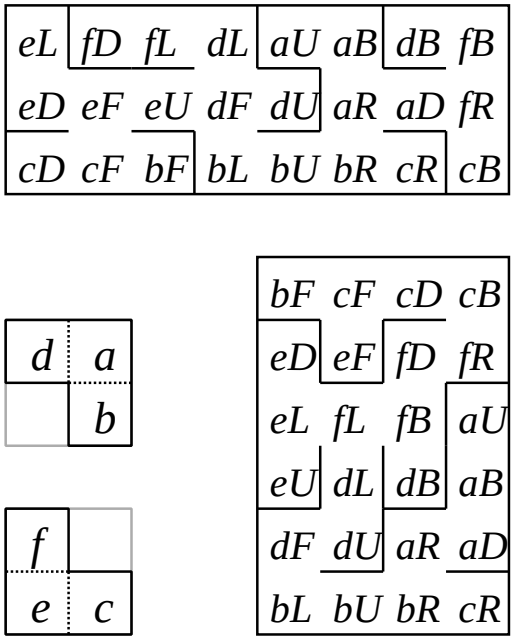


Figure 3: The left down polycube of volume 6 has two different rectangular unfoldings of size 3×8 and 4×6 with slits.

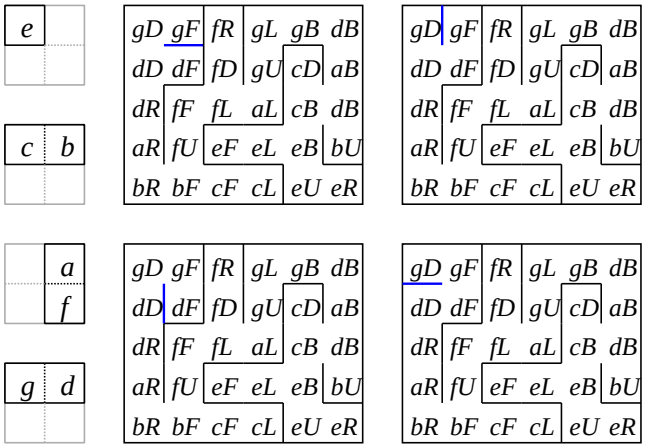


Figure 4: The left polycube of volume 7 has rectangular unfolding of size 5×6.

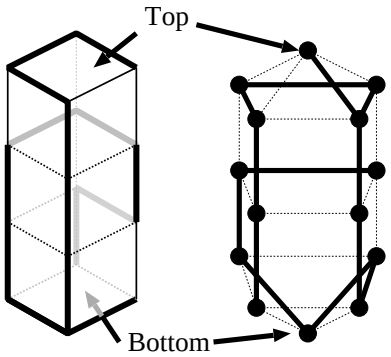


Figure 5: Spanning tree corresponding to the pattern in Figure 1.

touch with genus 0. Curiously, this heptacube has only 4 rectangular unfoldings. All unfoldings are shown in Figure 4. As you can observe, these 4 unfoldings are almost the same except the cut of unit length at the top left corner.

Our program confirmed that there are no rectangular polycube unfoldings with fewer than 24 faces, and the one shown in Figure 3 is unique for 24 faces. These facts complete the proof of Theorem 1. We here observe the algorithm used in this section.

2.4 Algorithm

The input of our algorithm is a polycube Q. We here consider the polycube Q of surface area n squares as a graph $G(Q) = (V, E)$; the set V of n unit squares and $E = \{\{u, v\} \mid$ the unit squares u and v share an edge on $Q\}$. On the graph $G(Q)$, a slit on Q cuts the corresponding edge. Then it is known that an unfolding of the polycube Q is given by a spanning tree of $G(Q)$ (see, e.g., [3]). For example, the cutting pattern in Figure 1 corresponds to the spanning tree in the right graph in Figure 5. Therefore, when a polycube Q is given, the algorithm can generate all unfoldings by generating all spanning trees for the graph $G(Q)$. (We note that, as mentioned in Introduction, some slits can be redundant in this context; however, we do not care about this issue. Therefore, some unfoldings in the figures contain redundant slits.)

For each spanning tree of $G(Q)$, the algorithm checks whether the corresponding unfolding overlaps or not. If not, it gives a valid net of Q. If, moreover, it forms a rectangle, it is a solution of our problem. For a given spanning tree, this check can be done in linear time. Since all spanning trees of a given graph G can be enumerated in $O(1)$ time per tree (see [5]), all unfoldings of a given polycube Q of area n can be done in $O(nT(G(Q)))$ time, where $T(G(Q))$ is the number of spanning trees of $G(Q)$.

We note that our algorithm runs for any given polycube Q, and it can check if Q has a valid net or not. By exhaustive check, we have the following theorem:

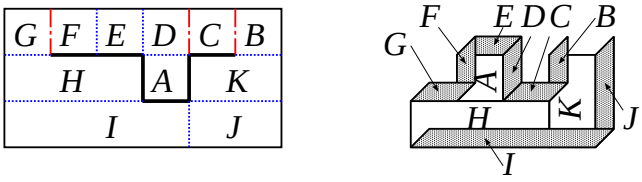


Figure 6: An F gadget.

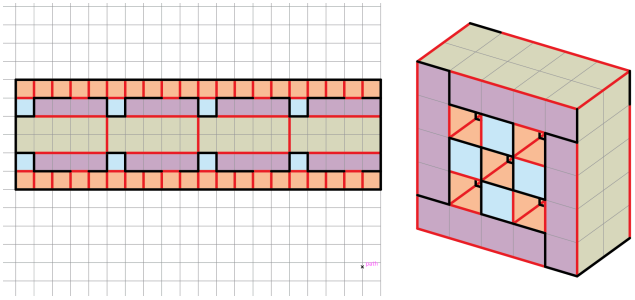


Figure 7: A construction of a rectangle of size 6×20. It can fold to a polycube of genus 5.

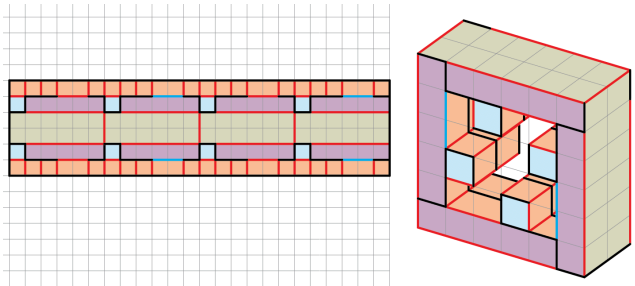


Figure 8: A construction of rectangle of size 6×24. It can fold to a polycube of genus 1.

Theorem 4 *All polycubes that consist of 12 or fewer cubes have an edge unfolding without overlapping.*

3 Proof of Theorem 2

Next we turn to a construction of family of polycubes.

We first introduce a gadget shown in Figure 6, which is called an *F gadget*. An F gadget is a rectangle of size 3×6 with some slits, which can be folded to F shape as shown in Figure 6. Gluing two copies of F gadgets (precisely, one is the mirror image), we can obtain an L-shaped pipe with hole of size 1×2. Therefore, joining four of the L-shaped pipes, we can construct a polycube as shown in Figure 7. By elongating the gadgets, we can change the size and genus as shown in Figure 8.

Now we introduce another series of gadgets in Figure 9, which is called *I gadgets*. An I gadget of size i

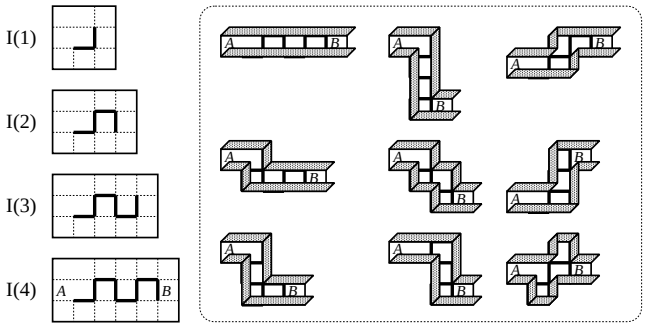


Figure 9: I gadgets.

is a rectangle of size $3 \times (i+2)$ with some slit. The I gadget of size i has a zig-zag slit of length $2i$ as shown in Figure 9. This gadget can be folded not only in the I shape in a natural way, but also many other ways. For example, I(4) has nine ways of folding in total as shown in the right in Figure 9. Therefore, in general, I(i) has exponentially many ways of foldings. (The exact value is open, but it is at least $9^{i/4}$ by joining $i/4$ of I(4)s.)

Combining these gadgets, it is easy to construct a rectangle with some slits for folding a polycube of any genus. In Figure 10(a), we give an example of a rectangle with some slits that can be folded to a polycube of genus 2. Figure 10(b) describes the polycube of genus 2 folded from (a) (since all polycubes folded in this manner are of thickness 2, we draw them by top-view).

We can observe that there are many polycubes folded from (a) by the property of the I gadget. That is, each I gadget in a rectangle can be folded to one of nine different shapes unless it intersects with others and the length has consistency. That is, choosing each way of folding properly, we can fold to (exponentially) many different polycubes from the rectangle of length $6 \times n$ with slits. For a rectangle in Figure 10(a), one of the variants is given in Figure 10(c). Now it is easy to see that Theorem 2 and Corollary 3 hold.

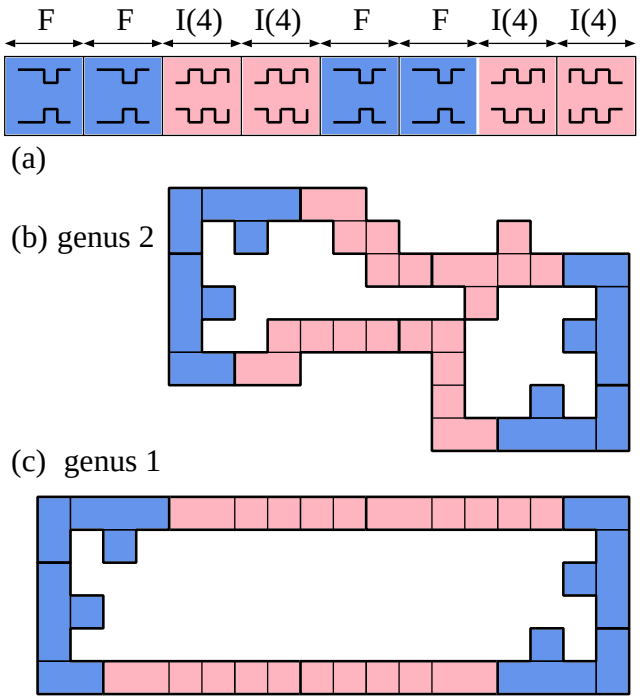


Figure 10: (a) A construction of rectangle of size 6×48. (b) A polycube of genus 2 folded from the rectangle. (c) Another polycube of genus 1 folded from the same rectangle.

4 Concluding Remarks

In this paper, we show some concrete polycubes folded from a rectangle with slits. Among them, there is a polycube of genus 0. We also show that there are infinitely many polycubes folded from a rectangle with slits. This construction gives us infinitely many polycubes of genus g for any positive integer g, and it also gives us infinitely many rectangles that can fold to (at least) k different polycubes for any positive integer k. However, so far, we have no construction that gives infinitely many polycubes of genus 0, which is an open problem.

The series of I gadgets in Figure 9 gives us interesting patterns. For a given i, the number of ways of folding of I(i) seems to be an interesting problem from the viewpoint of computational origami. From the viewpoint of puzzle, it is also an interesting problem to decide the kind of polyominoes folded from I(i) for general i. In the construction in Theorem 2 and Corollary 3, we use the rectangle of size $6 \times n$. It may be interesting whether we can use the rectangle of size $4 \times n$ or not.

In Theorem 4, we stated that all polycubes consisting of 12 or fewer cubes have an edge unfolding without overlap. This theorem begins to address an open problem that asks whether there exists a polycube that has no non-overlapping edge unfolding. It seems very challenging to find such an "ununfoldable" polycube by brute-force search: our program is able to quickly find solutions for randomly sampled polycubes with as many as 1000 cubes (as well as for hand-constructed polycubes that appear hard to unfold), and exhaustive search becomes infeasible at much smaller numbers. This problem sometimes appears as "grid unfoldings" in the context of unfolding of orthogonal polyhedra. See [6, 7, 8, 9, 10] for further details.

Acknowledgements

The fourth author was supported in part by MEXT/JSPS KAKENHI Grant Number 17H06287 and 18H04091. This work was discussed at the 34th Bellairs Winter Workshop on Computational Geometry, co-

organized by Erik D. Demaine and Godfried Toussaint, held on March 22–29, 2019 in Holetown, Barbados. We thank the other participants of that workshop for providing a stimulating research environment.

References

[1] Jin Akiyama. Tile-Makers and Semi-Tile-Makers. *American Mathematical Monthly*, Vol. 114, pp. 602–609, 2007.

[2] Stefan Langerman and Andrew Winslow. Polycube Unfoldings Satisfying Conway's Criterion. *Japan Conference on Discrete and Computational Geometry and Graphs (JCDCG3 2016)*, 2016.

[3] Jun Mitani and Ryuhei Uehara. Polygons Folding to Plural Incongruent Orthogonal Boxes. *Canadian Conference on Computational Geometry (CCCG 2008)*, pp. 39–42, 2008.

[4] Erik D. Demaine and Joseph O'Rourke. *Geometric Folding Algorithms: Linkages, Origami, Polyhedra*. Cambridge University Press, 2007.

[5] David Avis and Komei Fukuda. Reverse search for enumeration. *Discrete Applied Mathematics*, Vol. 65, pp. 24–46, 1996.

[6] Therese Biedl, Erik Demaine, Martin Demaine, Anna Lubiw, Mark Overmars, Joseph O'Rourke, Steve Robbins, and Sue Whitesides. Unfolding Some Classes of Orthogonal Polyhedra. *10th Canadian Conference on Computational Geometry (CCCG'98)*, 1998.

[7] Erik D. Demaine, John Iacono, and Stefan Langerman. Grid Vertex-Unfolding Orthostacks. *International Journal of Computational Geometry and Applications*, Vol. 20(3), pp. 245–254, 2010.

[8] Mirela Damian, Robin Flatland, and Joseph O'Rourke. Grid Vertex-Unfolding Orthogonal Polyhedra. *Discrete and Computational Geometry*, Vol. 39, pp. 213–238, 2008.

[9] Greg Aloupis, Prosenjit K. Bose, Sebastien Collette, Erik D. Demaine, Martin L. Demaine, Karim Douieb, Vida Dujmović, John Iacono, Stefan Langerman, and Pat Morin. Common Unfoldings of Polyominoes and Polycubes. *China-Japan Joint Conference on Computational Geometry, Graphs and Applications (CGGA 2010)*, LNCS Vol. 7033, pp. 44–54, 2010.

[10] Joseph O'Rourke. Unfolding orthogonal polyhedra. Surveys on Discrete and Computational Geometry: Twenty Years Later, Contemporary Mathematics 453, AMS, 2008.

Folding Polyominoes with Holes into a Cube

Oswin Aichholzer* Hugo A. Akitaya† Kenneth C. Cheung‡ Erik D. Demaine§ Martin L. Demaine§
Sándor P. Fekete¶ Linda Kleist¶ Irina Kostitsyna‖ Maarten Löffler** Zuzana Masárová††
Klara Mundilova‡‡ Christiane Schmidt♣

Abstract

When can a polyomino piece of paper be folded into a unit cube? Prior work studied tree-like polyominoes, but polyominoes with holes remain an intriguing open problem. We present sufficient conditions for a polyomino with hole(s) to fold into a cube, and conditions under which cube folding is impossible. In particular, we show that all but five special *simple* holes guarantee foldability.

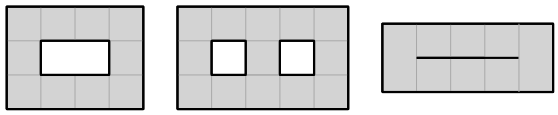


Figure 1: Three polyominoes that fold along grid lines into a unit cube, from puzzles by Nikolai Beluhov [4].

1 Introduction

Given a piece of paper in the shape of a polyomino (i.e., a polygon in the plane formed by unit squares on the square lattice that are connected edge-to-edge), does it have a folded state in the shape of a unit cube? The standard rules of origami apply; in particular, we allow each unit square face to be covered by multiple layers of paper. Examples of this decision problem are given by the three puzzles by Nikolai Beluhov [4] shown in Figure 1. We encourage the reader to print out the puzzles and try folding them.

Prior work [2] studied this decision problem extensively, introducing and solving several different models of folding. This gave rise to a model that matches the puzzles in Figure 1: Fold only along grid lines of the polyomino; allow only orthogonal folding angles ($\pm 90°$ and $\pm 180°$); and forbid folding material strictly interior to the cube. In this model, the prior work [2] characterizes which tree-shaped polyominoes lying within a $3 \times n$ strip can fold into a unit cube.

Notably, however, the polyominoes in Figure 1 are not tree-shaped or even simple: One puzzle has a hole, another puzzle has two holes, and a third puzzle has a degenerate hole (a slit). Arguably, these holes are what makes the puzzles fun and challenging. Therefore, in this paper, we embark on characterizing which polyominoes with hole(s) fold into a unit cube in this model. Although we do not obtain a complete characterization, we give many interesting conditions under which a polyomino does or does not fold into a unit cube.

The problem is sensitive to the choice of model. In the more flexible model allowing half-grid folds and 45° diagonal folds between grid points, the prior work [2] shows that *all* polyominoes of at least ten unit squares can fold into a unit cube, and lists all smaller polyominoes that fold into a cube. Thus this model already has a complete characterization of polyominoes that fold into a cube, including those with holes. Therefore, we focus on the grid-fold model described above.

Specific to polyominoes and polycubes, there is extensive work in this model on finding polyominoes that fold into many different polycubes [3] and into multiple different boxes [1, 5, 6, 7, 8].

Our Results.

1. We identify which polyominoes with a single hole are foldable; see Theorem 1, Section 3.1. In fact, all but five *simple* holes already guarantee foldability.
2. We identify combinations of two (of the remaining five) holes that allow the polyomino to fold into a cube; see Section 3.2.
3. We show that certain of the remaining five simple holes or their combinations do not allow a foldable polyomino; see Section 4.

*Institute for Software Technology, Graz University of Technology, oaich@ist.tugraz.at
†Department of Computer Science, Tufts University, Hugo.Alves_Akitaya@tufts.edu
‡NASA Ames Research Center, kenny@nasa.gov
§CSAIL, Massachusetts Institute of Technology, {edemaine,mdemaine}@mit.edu
¶Department of Computer Science, TU Braunschweig, s.fekete@tu-bs.de, kleist@ibr.cs.tu-bs.de
‖Mathematics and Computer Science Department, TU Eindhoven, i.kostitsyna@tue.nl
**Department of Information and Computing Science, Universiteit Utrecht, m.loffler@uu.nl
††IST Austria, Klosterneuburg, zuzana.masarova@ist.ac.at
‡‡Institute of Discrete Mathematics and Geometry, TU Wien, klara.mundilova@tuwien.ac.at
♣Department of Science and Technology, Linköping University, christiane.schmidt@liu.se

4. We present an algorithm that checks a necessary local condition for foldability; see Section 4.4.

2 Notation

A *polyomino* is a polygon P in the plane formed by a union of $|P| = n$ *unit squares* on the square lattice that are connected edge-to-edge. We do not require a connection between every pair of adjacent squares; that is, we allow *slits* along the edges of the lattice subject to the condition that the polyomino is connected.

We call a set h of connected missing squares and slits a *hole* if the dual has a cycle containing h in its interior. We call a hole of a polyomino *simple* if it is one of the following: a unit square, a slit of size 1, slits of size 2 (corner or straight), or a U-slit of size 3, see Figure 2 for an illustration.

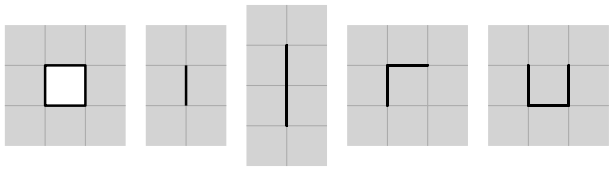


Figure 2: The five simple holes.

A connected three-dimensional polyhedron formed by a union of *unit cubes* on the cubic lattice that are connected face-to-face is called a *polycube*. If the polycube Q is a *unit cube*, we denote it by $Q = \mathcal{C}$.

In this paper, we study the problem of folding a given polyomino P with holes to form $\mathcal{C}$, allowing only 90° and 180° folds along the lattice. We illustrate mountain folds in red, and valley folds in blue. Whenever we show numbers on faces in crease patterns these refer to a "real" die, i.e., opposite faces sum up to 7.

3 Polyominoes That Do Fold

In this section, we present polyominoes that fold. We start with polyominoes that contain a hole guaranteeing foldability.

3.1 Polyominoes with Single Holes

In this section, we show that all holes different from a simple hole guarantee foldability.

Theorem 1 *If a polyomino P contains a hole h that is not simple, then P folds into a cube.*

Proof. It is easy to see that because the hole h is non-simple, it must be a superset of one of the holes in Figure 4, that is, we distinguish the cases where h contains

- Two unit squares sharing an edge
- Two unit squares sharing a vertex

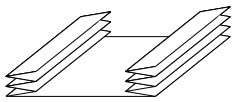


Figure 3: Folding strategy to reduce to seven cases.

- A unit square and an incident slit
- A slit of length at least 3 (straight, zigzagged, L-shaped, or T-shaped)

In a first step, we show that if h contains one of the four above holes, we may assume that it contains exactly this hole. Let h be a hole containing a hole h' of the above type. By definition of a hole, h needs to be enclosed by solid squares. Thus we can sequentially fold the squares of P in columns to the left and right of h' on top of the columns directly left and right of h', respectively, as illustrated in Figure 3. Afterwards, we fold the squares of P in rows to the top and bottom of h' on top of the rows directly top and bottom of h', respectively. We call the resulting polyomino P'. Note that because h is a hole of P, all neighbouring squares of h' exist in P'. Thus we may assume that we are given one of the seven polyominoes depicted in Figure 4, where striped squares may or may not be present.

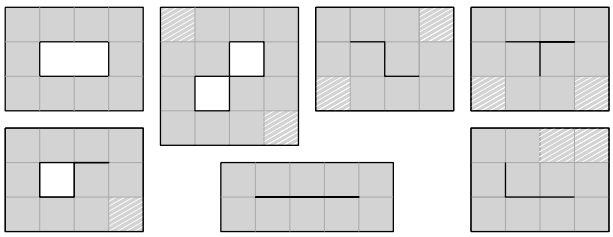


Figure 4: Any polyomino with a hole that is not simple can be reduced to one of the seven illustrated cases; striped squares may or may not be present.

Secondly, we present folding strategies. Note that the case if h' consists of two squares sharing only a vertex, we can fold the top row on its neighboring row and obtain the case where h' consist of a square and an incident slit. For an illustration of the folding strategies for the remaining cases consider Figure 5. □

Are simple holes ever helpful? In fact, four of the five simple holes sometimes allow foldability, as illustrated in Figure 6. Note that the U-slit of size 3 reduces to the square hole. In Lemma 11, we show that the slit of size 1 never helps to fold a rectangular polyomino. Lemma 7 implies that the polyominoes without the holes cannot be folded.

3.2 Combinations of Two Simple Holes

In this section we consider combinations of two simple holes that fold.

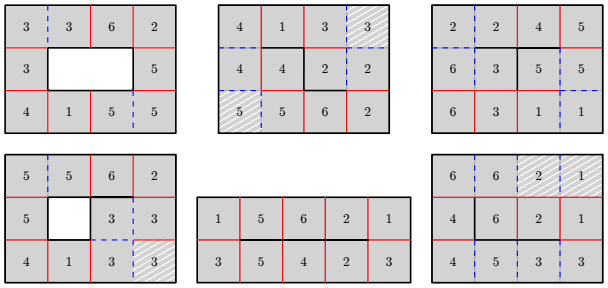


Figure 5: Crease pattern of cube foldings; mountain folds (solid red), valley folds (dashed blue). Squares with the same number cover the same face of the cube.

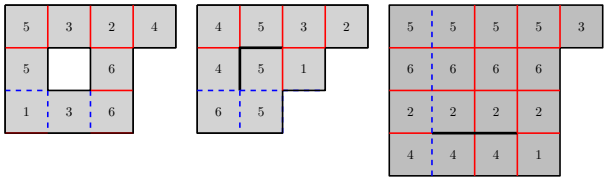


Figure 6: Four simple holes may be helpful.

Theorem 2 *A polyomino with two vertical straight size-2 slits with at least two columns and an odd number of rows between them folds.*

Proof. As in the previous section, we first fold all rows between the slits together to one row; this is possible because there is an odd number of rows between the slits. Then, all the surrounding rows and columns are folded towards the slits. Finally, we fold the columns between the slits to reduce their number to two or three. Depending on whether the number of columns between the slits was even or odd, this yields a shape similar to the one shown in Figure 7 (a) and (b), respectively. Striped squares may be (partially) present. In all cases, the two shapes fold as indicated by the illustrated crease pattern. Note that in Figure 7 (b) the polyomino is of course connected, which implies that for sure at least one square of the central column is part of the polyomino, i.e., a square with label 6 is used. □

If the two slits have only one or no column between them, then the shape cannot be folded as can be verified by the algorithm of Section 4.4.

In the following theorems we call a U-slit which has the open part at the bottom an A-slit. If the orientation of the U-slit is not relevant, then we call it a C-slit.

Theorem 3 *A polyomino with an A-slit and a unit square hole/C-slit in the same column above it, having an even number of rows between them, folds.*

Proof. We can assume that the upper hole is a unit square, as the flaps generated by a C-slit can always be folded away. Similar to before we fold away all surrounding rows and columns and reduce the number of rows between the A-slit and the unit square hole to two. This yields the shape of Figure 7 (c), which can be folded. □

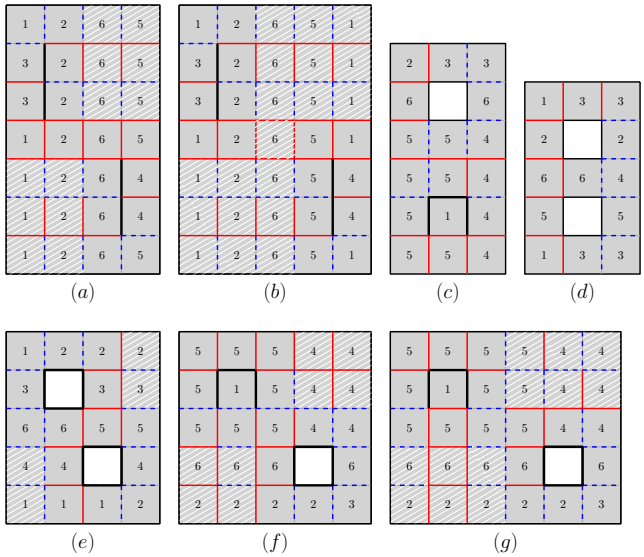


Figure 7: Combinations of two simple holes that are foldable with and without (part of) the striped region.

Note that if the bottom slit is a U-slit, then the shape of Figure 7 (c) does not fold, again verified by the algorithm of Section 4.4.

Theorem 4 *A polyomino with an A-slit and a unit square hole/C-slit below it and separated by an odd number of rows, folds, regardless in which columns they are.*

Proof. As before we assume that the lower hole is a unit square, fold away all surrounding rows and columns, and reduce the number of rows between the two slits/holes to one. Furthermore we fold the columns between the slits/holes to minimize their number. In this way the number of columns between the two slits/holes is at most two, and we obtain one of the shapes shown in Figure 7 (d) to (g). All of them fold, with or without the striped region. Note that the upper unit square holes in Figure 7 (d) and (e) can be replaced by an A-slit which can be folded away. □

Note that if the two holes are in the same or neighboured column(s) (Figure 7 (d) and (e)), then it does not matter which orientation the U-slits have or whether they are unit square holes—any combination folds. We thus get the following statement.

Theorem 5 *A polyomino with two unit square holes which are in the same or in neighboured column(s) and have an odd number of rows between them folds.*

4 Polyominoes That Do Not Fold

In this section, we identify simple holes and simple hole combinations that do not allow the polyomino to fold.

First, we present some results that show how the paper is constrained around an interior vertex.

Lemma 6 *Four faces around a polyomino vertex v for which the dual graph is connected cannot cover more than three faces of $\mathcal{C}$.*

Proof. v is incident to 4 faces in P. As vertices of P are mapped to vertices of $\mathcal{C}$ and all vertices of $\mathcal{C}$ are incident to 3 faces, v is incident to only 3 faces in $\mathcal{C}$. □

Lemma 7 *Four faces around a vertex v not in the boundary of P cannot cover more than two faces of $\mathcal{C}$. In particular, at least two collinear incident creases must be folded by* 180°.

Proof. Let A, B, C, and D be the faces around v in circular order. By Lemma 6, at most three faces of $\mathcal{C}$ are covered by A, B, C, and D. Hence, at least two faces map to the same face of $\mathcal{C}$. These can either be edge-adjacent of diagonal. For the first case, let this w.l.o.g. be A and B. Hence, the crease between them must be folded by 180°. Then C and D must also map to the same face of $\mathcal{C}$ to maintain the paper connected. The crease between C and D must also be folded by 180°. In the latter case, w.l.o.g. A and C map to the same face of $\mathcal{C}$. As they are both incident to v, only two options of folding those two faces on top of each other exist. Either the edge between A and B gets folded on top of the edge between B and C, this leaves a diagonal fold on B, a contradiction, or the edge between A and D gets folded on top of the edge between B and C, which results in D being mapped to C, and those are two adjacent faces, in which case we already argued that two collinear incident creases must be folded by 180°. □

Corollary 1 *Let v, w be two vertices in P's interior, which share a horizontal edge. If we fold horizontally through v, i.e., if the two collinear incident creases of v folded by* 180° *are horizontal, then we also have to fold* 180° *horizontally through w.*

4.1 Polyominoes with Unit Square Holes, L-Shaped Holes and U-Shaped Holes

We begin by examining the possible foldings of a polyomino containing a unit square hole. Suppose that a given polyomino P with a unit square hole h folds into a cube. Furthermore, let the shape of h no longer be a square in the folded state. That is, the hole h is folded in a *non-trivial* way. Then, in the folded state, either all edges of h are glued together, or two pairs of edges are glued forming an L-shape. We will argue that if P folds into a cube, the first case is impossible, while the second produces a specific crease pattern around h.

Lemma 8 *The four edges of a unit square hole of a polyomino that folds into a cube cannot be all glued together in the folded state.*

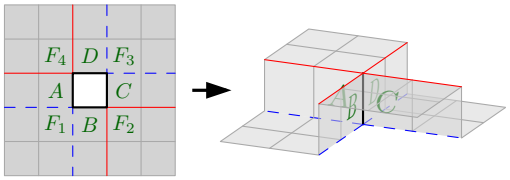


Figure 8: Four edges of a square hole glued together.

Proof. Let the four faces of the polyomino edge-adjacent to the hole be A, B, C, and D, and the four faces vertex-adjacent to the hole be F_1, F_2, F_3, and F_4, see Figure 8. Consider A, F_1, and B in the folded state. As the two corresponding edges of the hole are glued together, the three faces must be pairwise perpendicular. The similar statement holds for the triples $\{B, F_2, C\}$, $\{C, F_3, D\}$, and $\{D, F_4, A\}$. Therefore, if P folds into a cube, face A must be glued to face C, face B must be glued to D, F_1 to F_3, and F_2 to F_4. Suppose, w.l.o.g., in the folded state face A lies in a more outer layer than C. Then, F_1 and F_4 are in a more outer layer than F_3 and F_2, respectively. Thus, face B connects the more inner layer of F_2 to the more outer layer of F_1, and at the same time D connects the inner layer of F_3 to the outer layer of F_4. Hence, faces B and D intersect, which is impossible. Therefore, if the polyomino folds into a cube, the four edges of a square hole cannot all be glued together. □

It follows that the only non-trivial way to glue the edges of a square hole of a polyomino folded into a cube is to form an L-shape. This effectively amounts to gluing a pair of diagonal vertices of the hole.

Let a, b, c, and d be the four vertices of the hole, and suppose a and c are glued together when folding the polyomino into a cube, see also Figure 9 (left). Consider the crease pattern around the hole. We shall only focus on the angles of the creases and not the type of the fold, as there may be (and will be) other creases in P affecting the type of the creases under our consideration. Observe that the three faces incident to each of the vertices b and d are pairwise perpendicular, they form a corner of a

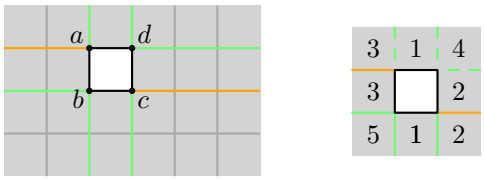


Figure 9: Left: crease pattern around a hole folding into an L-shape when gluing vertices a and c; 90° creases are shown in green, and 180° creases in orange. Right: numbers indicate the face of the cube in the folded state; mountain folds are shown in solid, and valley folds as dashed lines.

cube. Thus, the creases emanating from b and d are all 90°. Further observe that the three faces around each of the vertices a and c fold into two faces of a cube, thus leading to one of the creases being 90° and the other 180°. Finally, the two 180° creases are parallel to each other. Indeed, consider the right side of Figure 8. For a crease to form an L-shape one of the two dashed blue lines must fold to 180°, which corresponds to two parallel creases in the unfolded state. Therefore, the crease pattern in Figure 9 (left) is the only pattern of creases (up to rotation and reflection) around a non-trivially folded square hole. Figure 9 (right) shows the faces of the corresponding crease pattern.

Note that the arguments above extend to an L-shape slit of size 2, and a U-slit of size 3.

Theorem 9 *Two holes, which are either unit square, L-slit of size* 2*, or U-slit of size* 3*, of a polyomino P such that (1) P contains all the other cells of the bounding box of the two holes, (2) P folds into a cube, cannot be both folded non-trivially if the number of rows and the number of columns between the holes is at least* 1*.*

Proof. It follows from the above observations that if there were two unit square holes, both folded non-trivially, with positive number of rows and columns between them, there would be two intersecting 90° creases. □

Theorem 10 *A rectangle with two unit square holes in the same row does not fold into a cube if the number of columns between the holes is even.*

Proof (sketch). We prove that a 3×6 rectangle with two unit squares holes as in Figure 10 does not fold into a cube. From that it follows that any $3 \times (4 + 2k)$ rectangle with two unit square holes in the same row separated by $2k$ columns does not fold into a cube. Note that both holes must be folded non-trivially, otherwise the polyomino cannot be folded into a cube.

The vertical fold in the middle of two holes must be a 180° fold as depicted in Figure 10; otherwise there would be two perpendicular 90° creases. There are two types of crease patterns for this polyomino: when pairs of parallel 90° creases run vertical, and when there is one pair of horizontal parallel 90° creases. In both cases, the faces in between of those creases all map to the same face on $\mathcal{C}$, which implies that the face opposite to the one on $\mathcal{C}$ cannot be covered. □

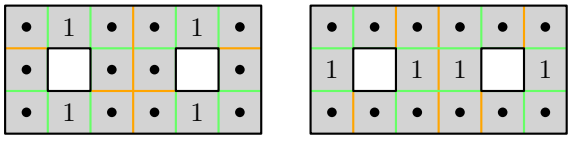


Figure 10: A polyomino that does not fold into a cube.

4.2 Polyominoes with a Single Slit of Size 1

The following Lemma shows that slit holes of size one do not help in folding a rectangular polyomino into $\mathcal{C}$.

Lemma 11 *A rectangular polyomino P with a single slit hole of size 1 does not fold into $\mathcal{C}$.*

Proof. Because of Corollary 1 we can restrict to the polyomino in Figure 11. Let A, B, C, D, E and F be the faces adjacent to h as in Figure 11. Because the paper must remain connected, the endpoints of h must map to adjacent vertices of $\mathcal{C}$. Then the paper behaves exactly as if the slit were not there as follows. If E and B maps to the same face of $\mathcal{C}$, then A (resp., C) must map to the same face as F (resp., D). Otherwise, E and B maps to adjacent faces. Then, A and C (resp., F and D) maps to the same face as B (resp., E). By the successive application of Lemma 7 in a rectangular polyomino P, without loss of generality only the front, left, back and right faces of $\mathcal{C}$ can be covered. □

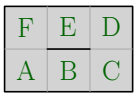


Figure 11: A polyomino with a slit hole of size one.

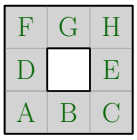


Figure 12: A polyomino with a single square hole.

4.3 Rectangles with a Single Square Hole

In this section, we show the following fact:

Theorem 12 *If P is a rectangle with a single square hole h, then P does not fold into a unit cube $\mathcal{C}$.*

Proof. Let P' denote the 3×3 rectangle with a central unit square hole depicted in Figure 12. By Corollary 1 any polyomino P needs to be reduced to P':

Claim 1 *Every rectangle with a single unit square hole is foldable (if and) only if P' is foldable.*

Consequently, it remains to show that

Claim 2 *The polyomino P' does not fold into $\mathcal{C}$.*

We label the eight faces of P' by (A,B,C,D,E,F,G,H) as depicted in Figure 12. Without loss of generality assume that A maps to the top face of $\mathcal{C}$. First, we argue that C cannot map to the same face. If that was true, then B also maps to the top face and by the number of faces, every remaining face must map to a different face of $\mathcal{C}$. However, if D maps to the back face of $\mathcal{C}$, so does E, a contradiction. Consequently, A and C do not map to the same face. By symmetry, F does not map to the top face (and neither C nor F map to the same face as H).

Next, we argue that the only faces that can map to the bottom face of $\mathcal{C}$ are C and F: If E would map to the bottom face, any of (B,C) or (D,F,G,H) must cover the front face and right face, respectively. For B to cover the front face, C must cover the bottom face. (Analogously, the argument for G.) H has odd number of squares to A, if H would map to the bottom face, one face would have to be between A and H, hence, we would need to reduce the number by folding, this folds H onto C or F. A contradiction to the first fact. Hence, only faces that can map to the bottom face of $\mathcal{C}$ are C and F.

W.l.o.g., let F be the bottom face, and D the back face. Then the only faces that can cover the left face are C and H; in particular, if E covers the left face then the right face remains uncovered. Thus, we assume w.l.o.g. that C covers the left face. Hence, B maps to the top face. Now, if E maps to the back face, both G and H must map to the right face of $\mathcal{C}$, and the front face is uncovered. If E maps to the left face, because the top and left faces are doubly covered, every remaining face must be singly covered. Then, H must map to the front face. But face G can only map to the top face, which cannot happen because A and B already cover this face. The only remaining case is when both C and F map to the bottom face, thus, B and D maps to the right and back faces of $\mathcal{C}$ respectively. However, both E and G can only cover faces of $\mathcal{C}$ that are already covered (bottom, back and right faces), and $\mathcal{C}$ would not have all its faces covered. □

4.4 An Algorithm to Check a Necessary Local Condition for Foldability

Consider the following local condition: let s be a square in a polyomino P such that the mapping between vertices of s and vertices of a face of $\mathcal{C}$ has been fixed. Then, for every adjacent square s' of s, there are two possibilities how to map its four vertices onto $\mathcal{C}$: the two vertices shared by s and s' must be mapped consistently and for the other two vertices there are two options depending on whether s' is folded at 90° angle to an adjacent face of $\mathcal{C}$, or whether it is folded at 180° to the same face of $\mathcal{C}$.

The algorithm below checks whether there exists a mapping between all vertices of squares of P to vertices of $\mathcal{C}$ such that the above condition holds for every pair of adjacent polyomino squares of P.

1. Run a breadth-first-search on the polyomino squares, starting with the leftmost square in the top row of P and continue via adjacent squares. This produces a numbering of polyomino squares in which each but the first square is adjacent to at least one square with smaller number.
2. Map vertices of the first square to the bottom face of $\mathcal{C}$. Extend the mapping one square at a time according to the numbering, respecting the local condition (that is, in up to two ways). Track all such partial mappings.

The algorithm is exponential, because unless inconsistencies are produced, the number of possible partial mappings doubles with every polyomino square. Nevertheless, it can be used to show non-foldability for small polyominoes: if no consistent mapping exists for a polyomino, then the polyomino cannot be folded onto $\mathcal{C}$. On the other hand, any consistent vertex mapping covering all faces of $\mathcal{C}$ obtained by the algorithm that we tried could in practice be turned into a folding. However, we have not been able to prove that this is always the case.

The algorithm above was used to prove that polyominoes in Figure 13 do not fold, as well as it aided us to find the foldings of polyominoes in Figure 7. An implementation of the algorithm is available at `http://github.com/zuzana-masarova/cube-folding`.

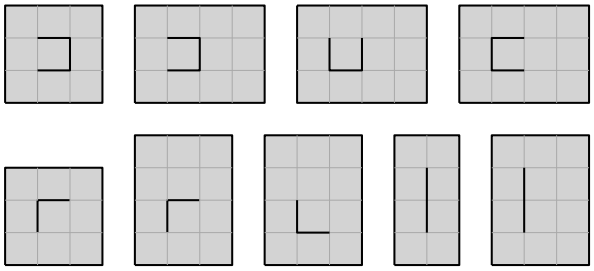


Figure 13: These polyominoes with single L, U and straight size-2 slits do not fold.

5 Conclusion

We showed that, if a polyomino P does contain a non-simple hole, then P folds into $\mathcal{C}$. Moreover, we showed that a unit square hole, size 2 slits (straight or corner), and a size-3 U-slit sometimes allow for foldability.

Based on the presented results, we created a font of 26 polyominoes with slits that look like each letter of the alphabet, and each fold into $\mathcal{C}$. See Figure A in the appendix, and `http://erikdemaine.org/fonts/cubefolding/` for a web app.

We conclude with a list of interesting open problems:

- Does a consistent vertex mapping output by the algorithm in Section 4.4 imply that the polyomino is foldable? If so, is the folding uniquely determined?
- Is any rectangular polyomino with one L-shape, U-shape or straight size-2 slit foldable? Currently, we only know that the small polyominoes in Figure 13 do not fold.

6 Acknowledgments

This research was performed in part at the 33rd Bellairs Winter Workshop on Computational Geometry. We thank all other participants for a fruitful atmosphere.

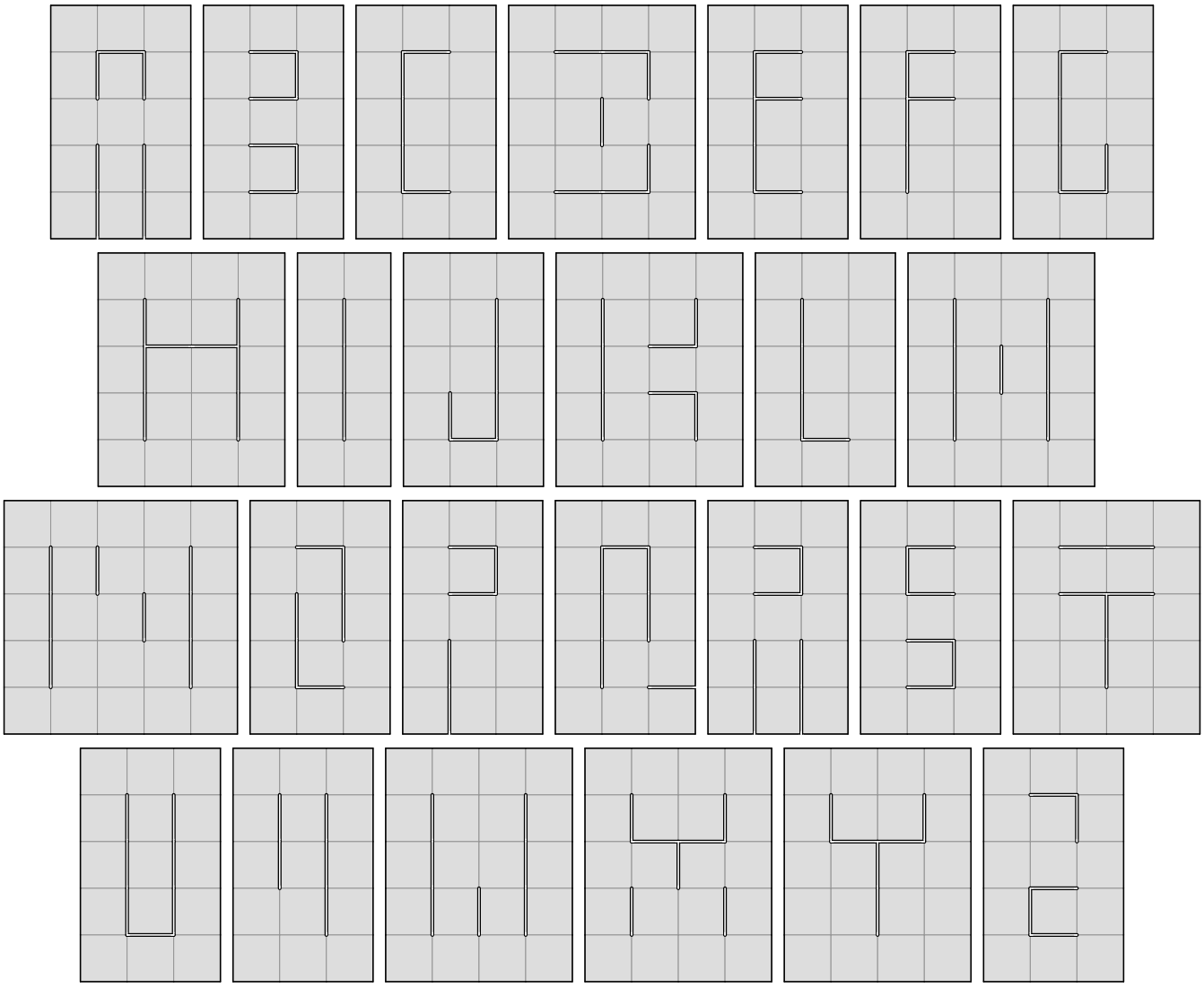


Figure A: Cube-folding font: the slits representing each letter enable each rectangular puzzle to fold into a cube.

References

[1] Z. Abel, E. Demaine, M. Demaine, H. Matsui, G. Rote, and R. Uehara. Common developments of several different orthogonal boxes. In *Proc. 23rd Can. Conf. Comp. Geom. (CCCG)*, 2011. Paper 49.

[2] O. Aichholzer, M. Biro, E. D. Demaine, M. L. Demaine, D. Eppstein, S. P. Fekete, A. Hesterberg, I. Kostitsyna, and C. Schmidt. Folding polyominoes into (poly)cubes. *Int. J. Comp. Geom. & Applic.*, 28(03):197–226, 2018.

[3] G. Aloupis, P. K. Bose, S. Collette, E. D. Demaine, M. L. Demaine, K. Douïeb, V. Dujmović, J. Iacono, S. Langerman, and P. Morin. Common unfoldings of polyominoes and polycubes. In *Proc. 9th Int. Conf. Comp. Geom., Graphs & Appl. (CGGA)*, volume 7033 of *LNCS*, pages 44–54, 2011.

[4] N. Beluhov. Cube folding. `https://nbpuzzles.wordpress.com/2014/06/08/cube-folding/`, 2014.

[5] J. Mitani and R. Uehara. Polygons folding to plural incongruent orthogonal boxes. In *Proc. 20th Can. Conf. Comp. Geom. (CCCG)*, 2008.

[6] T. Shirakawa and R. Uehara. Common developments of three incongruent orthogonal boxes. *Int. J. Comp. Geom. & Applic.*, 23(1):65–71, 2013.

[7] R. Uehara. A survey and recent results about common developments of two or more boxes. In *Origami^6: Proc. 6th Int. Meeting Origami in Sci., Math. and Educ. (OSME 2014)*, volume 1, pages 77–84. American Mathematical Society, 2014.

[8] D. Xu, T. Horiyama, T. Shirakawa, and R. Uehara. Common developments of three incongruent boxes of area 30. In R. Jain, S. Jain, and F. Stephan, editors, *Proc. 12th Ann. Conf. Theory and Applic. of Models of Comput. (TAMC)*, pages 236–247, 2015.

Minimum Forcing Sets for Single-Vertex Crease Pattern

Koji Ouchi* Ryuhei Uehara*

Abstract

We propose an algorithm for finding a minimum forcing set of a given flat-foldable *single-vertex crease pattern* (SVCP). SVCP consists of straight lines called *creases* that can be labeled as mountains or valleys, and the creases are incident to the center of a disk of paper. A *forcing set* is a subset of given creases that forces all other creases to fold according to the given labels. Our algorithm is a modification of a conventional algorithm for 1D origami. We show that the size of a minimum forcing set of an SVCP is $n/2$ or $n/2+1$ where n is the number of the creases in the SVCP.

1 Introduction

In an origami application called self-folding origami, a thin material folds into an intended shape by rotating the planes around creases according to the label mountain or valley assigned on the creases [6, 8, 11, 12]. The cost of such application can be reduced if it is enough to put actuators on a subset of creases. Finding such a subset of creases can be modeled as a forcing set problem. In applications, a material is often desirable to satisfy *flat foldablity* in order to make the size small. A material is *flat-foldable* if we can transform it from the completely unfolded state to the flat state that all creases are completely folded.

Forcing set problem is a new topic in computational origami, which was considered in [1, 2, 4]. Especially, minimum forcing set for flat foldability was studied for 1D origami [4] and 2D Miura-ori [2]. In a forcing set problem for flat foldability, a flat-foldable *crease pattern* C and a flat-foldable *mountain-valley assignment* (or MV assignment briefly) μ is given, where μ is a function from creases to $\{\mathrm{M}, \mathrm{V}\}$. MV assignment $\mu(c)$ on a crease $c \in C$ determines the direction of rotation of the planes around c when folding. A *forcing set* F is a subset of C where $c \in F$ is assigned the value $\mu(c)$, and F makes the other creases $c' \in C \backslash F$ to be assigned the value $\mu(c')$: F is not a forcing set if c' can have the assignment opposite to $\mu(c')$ to make the given crease pattern foldable. A forcing set F is called *minimum* if there is no other forcing set with size less than $|F|$.

This paper focuses on minimum forcing sets for flat-foldable *single-vertex crease pattern* (SVCP). An SVCP is a crease pattern whose creases are incident to the center of the paper to be folded. We consider the paper of SVCP is a disk. If $|C|$ is two, we are to fold the paper in half, and it is obvious that the size of the minimum forcing set is one. Figure 1 is an example of flat-foldable SVCP with MV assignment. The minimum forcing sets for the flat-foldable SVCP in Figure 1 are depicted in Figure 2. A crease pattern is *flat-foldable* if and only if there exists an MV assignment so that the paper settles into a flat shape without penetrating itself after folding the creases along the assignment. Bern and Hayes developed an algorithm to determine flat foldablity of a given SVCP with MV assignment [3]. Flat-foldable SVCPs were studied in [13] from the viewpoint of enumeration as well. A crease pattern of SVCP is a sequence of creases $C = (c_0, c_1, \ldots, c_{n-1})$ which are put clockwisely on the disk incident to the center. θ_i denotes the clockwise angle from c_i to $c_{i+1 \bmod n}$. We call (C, μ) an *MV pattern*.

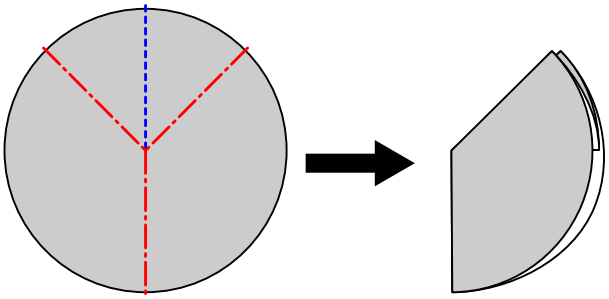


Figure 1: An example of flat-foldable SVCP with MV assignment.

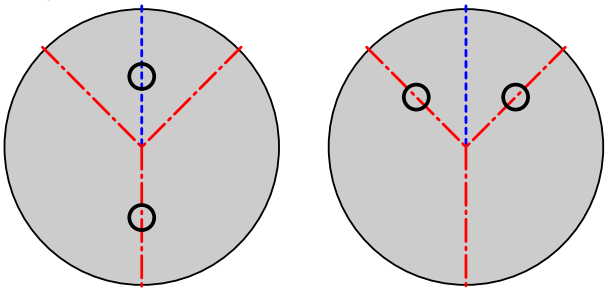


Figure 2: Examples of minimum forcing sets for the flat-foldable SVCP in Figure 1. Each crease with a small circle is the crease of the forcing set.

*School of Information Science, Japan Advanced Institute of Science and Technology, {k-ouchi, uehara}@jaist.ac.jp

In this paper, we develop an algorithm to find a minimum forcing set of a given flat-foldable SVCP in $O(n^2)$ time. As far as the authors know, our algorithm is the first one for finding a minimal forcing set of flat-foldable SVCP, even though SVCP is an important component of origami. Our algorithm is based on that for 1D origami in [4] because the structure of SVCP is similar to 1D origami if we regard it as a ring by cutting away the inner space of the paper: the creases reduce to points on the ring, and the paper becomes 1D origami if we cut the ring at some point. We also reveal that the size of F is $n/2$ or $n/2+1$. Precisely, $|F|$ is $n/2$ if the SVCP is of generic angles, which is a case that the angles to be operated always differ. In the case when all the angles in the SVCP are equal, $|F|$ is $n/2$ if $n = 2$, otherwise $|F|$ is $n/2+1$. For a general SVCP, which does not have any constraints, the size of F is $n/2+1$ if the crease pattern can be reduced to an SVCP of equal angles with size four or more by repeatedly crimping consecutive two creases $(c_i, c_{i+1 \bmod k})$ with different MV assignment where θ_i is minimal, otherwise $|F| = n/2$.

2 Preliminaries

This section introduces some terminology and preliminary results following [4]. Throughout the paper we work with a flat-foldable MV pattern (C, μ), where $C = (c_0, c_1, \ldots, c_{n-1})$ is an SVCP and μ is a flat-foldable MV assignment.

2.1 Crimpable Sequences [4]

We slightly change the definition of crimpable sequence to fit the assumption that C is circular. A *crimpable* sequence in SVCP is composed of consecutive creases where the angles between the creases are equal, with the property that the two angles adjacent to the left and right end of the sequence are strictly larger than the equal angles. Formally, for integers $0 \le i < n$ and $0 < k < n$, a sequence of consecutive creases $(c_i, c_{i+1 \bmod n}, \ldots, c_{i+k \bmod n})$ is crimpable if $\theta_i = \theta_{i+1 \bmod n} = \cdots = \theta_{i+k-1 \bmod n}$ and $\theta_{i-1 \bmod n} > \theta_i < \theta_{i+k \bmod n}$. We note that we have to take a mod on the index for circulation. Thus we may have $(i-1) \bmod n = (i+k) \bmod n$.

A *monocrimp* operation is defined as a fold about a single pair of consecutive creases of opposite MV parity in a crimpable sequence.

A *crimp* operation is a set of monocrimps repeatedly conducted on a crimpable sequence while the sequence is crimpable. The following theorem will be needed in Section 5.

Theorem 1 (Theorem 4 from [7]) *Let α be a crimpable sequence in a foldable MV pattern. The difference in the number of M and V assignments for the creases in α is zero (one) if α has an even (odd) number of creases.*

In the case of a crimpable sequence α of odd length, we say that the crease remaining after a crimp operation on α *survives* the crimp. We note that the surviving crease in α is with *majority assignment* in α ([4, Observation 1]). Majority assignment denotes the assignment M or V which is major in a sequence or a set of creases.

2.2 End Creases [4]

End creases are the remains after *exhaustive crimps*. Exhaustive crimps mean crimping repeatedly until there is no crimpable sequence. The following lemma holds for SVCP.

Lemma 2 *The end creases of an SVCP form a flat-foldable SVCP of equal angles.*

To prove this lemma, we need the following lemma and theorem:

Lemma 3 (Corollary 12.2.11 from [5]) *An equal-angle SVCP is flat-foldable iff $|\#M - \#V| = 2$. This implies that any MV assignment satisfying the condition is flat-foldable.*

Theorem 4 (The Maekawa Theorem) *In a flat-foldable SVCP with MV assignment defined by angles $\theta_0 + \theta_1 + \cdots + \theta_{n-1} = 360°$, the number of mountains and the number of valleys differ by ± 2.*

Details about the Maekawa Theorem can be found in [5, Chapter 12]. Now let us prove Lemma 2.

Proof. We will make exhaustive crimps, that is, we will repeat crimps while processed C satisfies $\theta_{i-1 \bmod n} > \theta_i = \theta_{i+1 \bmod n} = \cdots = \theta_{i+k-1 \bmod n} < \theta_{i+k \bmod n}$ for some i and k. After this repetition, the crease pattern becomes one that consists of all equal angles.

The original foldable (C, μ) satisfies the equation in Lemma 3 by the Maekawa Theorem. A monocrimp does not change the difference of Ms and Vs. Therefore after crimping all crimpable sequences in (C, μ), the crease pattern satisfies $|\#\mathrm{M} - \#\mathrm{V}| = 2$. By Lemma 3, the obtained equal-angle SVCP is flat-foldable. □

3 The Size of a Minimum Forcing Set of an SVCP

This section is devoted to proof of the theoretical minimum size of forcing sets.

3.1 SVCP of Generic Angles

In this section, let a given SVCP be of generic angles, that is, consecutive angles to be operated always differ. The following lemma is important in our proof.

Lemma 5 (from [9, 10]) *If an angle θ_i is strictly minimal, that is, $\theta_{i-1 \bmod n} > \theta_i < \theta_{i+1 \bmod n}$ holds, then the two consecutive creases c_i and $c_{i+1 \bmod n}$ forming θ_i have assignment different from each other in any flat-foldable MV pattern.*

In this section, first we show the existence of F with size $n/2$, then we prove that F with size $n/2-1$ or less does not exist.

Lemma 6 *There is a forcing set of an SVCP of generic angles, whose size is $n/2$.*

Proof. First we use a contradiction in order to show that there are always consecutive three angles which satisfy Lemma 5. Assume there are consecutive different angles $\theta_0, \theta_1, \ldots, \theta_{n-1}$, and any consecutive three of them do not satisfy Lemma 5. Then, for example, we can assume $\theta_0 > \theta_1 > \theta_2$. By the condition and assumption, $\theta_1 > \theta_2 > \theta_3$ holds. Similarly, $\theta_0 > \theta_1 > \theta_2 > \theta_3 > \theta_4 > \cdots$ holds and the sequence monotonically decreases. However, $\theta_{n-1} > \theta_0$ could never happen, which is a contradiction. Thus, we can always apply Lemma 5.

Applying Lemma 5 on $(\theta_{i-1 \bmod n}, \theta_i, \theta_{i+1 \bmod n})$ repeatedly, we can fold flat the paper. Let $(c_i, c_{i+1 \bmod n})$ be the pair of creases between the three angles. If we determine the assignment on one of $(c_i, c_{i+1 \bmod n})$, the assignment on the other of the pair is also determined. Hence we can make a forcing set by picking a crease in each pair as an element of the forcing set. We have $n/2$ such pairs because generic angles are the worst case of the number of such pairs. Therefore the size of the forcing set is $n/2$. □

Lemma 7 *There is no forcing set of an SVCP of generic angles whose size is less than $n/2$.*

Proof. The proof is by contradiction. Assume a forcing set F with size $n/2-1$ or less exists.

We monocrimp $(\theta_{i-1 \bmod n}, \theta_i, \theta_{i+1 \bmod n})$ according to Lemma 5. Every pair $(c_i, c_{i+1 \bmod n})$ is isolated from other pairs and there are $n/2$ pairs, thus every crease appears in a pair only once. Because $|F| < n/2$, there is an index i such that both in $(c_i, c_{i+1 \bmod n})$ are not in F. This contradicts the definition of F because the paper folds flat in the following two cases: we assign (M, V) on $(c_i, c_{i+1 \bmod n})$, or (V, M) on $(c_i, c_{i+1 \bmod n})$. □

By Lemma 6 and Lemma 7, we obtain the following theorem.

Theorem 8 *The size of an minimum forcing set for SVCP of generic angles is $n/2$.*

3.2 SVCP of Equal Angles

In this section, let a given SVCP be of equal angles, or *equal-angle* SVCP. Hence $\theta_i = \theta_{i+1 \bmod n}$ holds for any integer i.

Lemma 9 *There is a forcing set of an equal-angle SVCP whose size is $n/2+1$ if $n \geq 4$. Furthermore, the forcing set is composed of all creases with majority assignment.*

Proof. Assume that F consists of all majority M creases (thus all V creases are not in F). If F is not a forcing set then we can choose some crease in $C \setminus F$ to be M, contradicting Lemma 3. □

Lemma 10 *There is no forcing set of an equal-angle SVCP whose size is less than $n/2+1$ if $n \geq 4$.*

Proof. We prove it by contradiction. Assume F is a forcing set of an equal-angle SVCP, whose size is $n/2$ or less. Then there may be a pair of an M crease and a V crease which are not in F (Let M be the majority in the crease pattern). We denote such pair by p. We note that the creases in p do not have to be consecutive.

If all V creases are in F, p does not exist. In this case, we can invert the assignment of a pair of M creases in $C \setminus F$ to Vs, where the pair is not necessary to be consecutive. This operation holds Lemma 3, a contradiction.

Otherwise we can swap the MV assignment in p, and the resulting SVCP is flat-foldable by Lemma 3. This is a contradiction to our assumption that F is forcing. □

Theorem 11 *Assume that a given SVCP is of equal angles. If the number of creases in the SVCP is two, then the minimum forcing set consists of one crease. Otherwise the size of the minimum forcing set of the SVCP is $n/2+1$. By Iverson's convention, it can be described as $n/2 + [n \geq 4]$.*

Proof. It is obvious if the number of creases in an equal-angle SVCP is two. Lemma 9 and Lemma 10 imply that $n/2+1$ is the minimum size of F if $n \geq 4$. □

3.3 General SVCP

Here we consider that a given SVCP has no constraints.

Theorem 12 *Let m be the number of monocrimps performed until the given SVCP becomes a flat-foldable equal-angle SVCP (cf. Lemma 2). F denotes a minimum forcing set of the given SVCP. Then $|F| = n/2 + [n - 2m \geq 4]$.*

Proof. As the case of generic angles, we crimp the creases in crimpable sequences as many as possible. For each monocrimp, one of the creases in the pair must be in F. Such monocrimps contribute to m elements in F.

After monocrimping m times, the crease pattern has become a flat-foldable equal-angle SVCP (cf. Lemma 2). This equal-angle SVCP is composed of $n - 2m$ creases because two creases are consumed per one monocrimp. By Theorem 11, the size of a minimum forcing set of the equal-angle SVCP is $(n-2m)/2+[n-2m \geq 4]$.

The minimum size of F is the sum of the sizes of the two sets of forcing creases obtained above. This is because the sets do not have intersection and both are minimum. Thus, $|F| = m + (n - 2m)/2 + [n - 2m \geq 4] = n/2 + [n - 2m \geq 4]$. □

4 Constructing a Minimum Forcing Set

4.1 Crimp Forest Construction

We convert the crimp forest algorithm in [4] to an algorithm for SVCP by allowing circulation of the index of creases when finding a crimpable sequence. The converted algorithm is shown in Algorithm 1. A circulating crimpable sequence $(c_i, c_{i+1}, \ldots, c_0, c_1, \ldots, c_k)$ may occur when the algorithm finds and crimps crimpable sequences, but it does not change the behavior of the other parts of the algorithm. The algorithm constructs a forest in bottom-up manner. The edges are added if the sequence in the parent node includes the crease surviving the crimp on the sequence in child node.

Algorithm 1: CrimpForestSVCP(C, μ)

```
Initialize W ← ∅
while C has a crimpable sequence do
    Let s be the crimpable sequence in C with the
      smallest starting index. // modified from
      [4].
    create a node v corresponding to s, and add v
     to W.
    Make v the parent of each root node in W
     whose crimpable sequence has a surviving
     crease that is in s.
    Apply the crimp operation to s.
    Update C to be the resulting crease pattern.
end
return W
```

A straightforward implementation of Algorithm 1 takes $O(n^2)$ time because a naive way to find a crimpable sequence takes $O(n)$ time: start searching from c_0 clockwisely; skip monotonically nonincreasing angles; stop at the right side crease c_r which satisfies $\theta_{r-1 \bmod n} < \theta_r$; counterclockwisely from c_r, search the left side crease c_l which satisfies $\theta_{l-1 \bmod n} > \theta_l$; other operations can be done in constant time; since the algorithm loops at most n times, the time complexity of the algorithm is $O(n^2)$.

The following lemma describing the properties of crimp forest holds for SVCP as well.

Lemma 13 (Lemma 4 from [4]) *Given a crease pattern C and two foldable MV assignments μ_1 and μ_2, let W_1 and W_2 be the crimp forests corresponding to (C, μ_1) and (C, μ_2), respectively. Then the following properties hold:*

(1). W_1 and W_2 are structually identical.

(2). Corresponding nodes in W_1 and W_2 have crimpable sequences of the same size and the same interval distances between adjacent creases.

(3). Creases involved for the first time in a crimpable sequence at a node in W_1 have the same position in the crimpable sequence at the corresponding node in W_2.

4.2 Forcing Set Algorithm

We convert the forcing set algorithm in [4] by three modifications: switch CrimpForest(C, μ) to CrimpForestSVCP(C, μ); initialize F to the majority of end creases according to Lemma 9 instead of all end creases; remove one crease from F if $|F| = 2$ in the initialization according to Theorem 11. See Algorithm 2 for the detail.

The preorder traversal takes $O(n)$ time because each node is visited only once and the sum of lengths of the sequences in the nodes is n. Thus the main factor of computation time is CrimpForestSVCP, which takes $O(n^2)$ time.

We need the following lemma for the proof in Section 5:

Lemma 14 (Lemma 6 from [4]) *Let (C, μ_1) be a foldable MV pattern, and let F be the forcing set generated by Algorithm 2 with input (C, μ_1). Let (C, μ_2) be a foldable pattern such that μ_2 agrees with μ_1 on the forcing set F, that is, $\mu_2(c) = \mu_1(c)$ for $c \in F$. Let T_1 and T_2 be two structurally equivalent trees generated by the forcing set algorithm (C, μ_1) and (C, μ_2), respectively. If a crease c in a crimpable sequence $\alpha_1 \in T_1$ is in F, then a crease (not necessarily c) with the same MV assignment occurs in the corresponding crimpable sequence $\alpha_2 \in T_2$, in the same position as in α_1.*

5 Proof of Correctness

This section proves that F created by Algorithm 2 is forcing and minimum. The proof is almost the same as [4] because Damian et al. use local properties of crimpable sequence and abstract properties of crimp forest, which are not affected by the change from 1D to SVCP. In this section, we organize the proof in [4] to follow and prove the different points.

Algorithm 2: ForcingSetSVCP(C, μ)

```
Initialize W to the output generated by
    CrimpForestSVCP(C, μ) // modified
    from [4].
Initialize F to the all creases with majority
    assignment in end creases that remain after
    running CrimpForestSVCP(C, μ)
    // modified from [4].
if |F| = 2 then // added to [4]
    Remove one crease from F.
end
foreach tree T ∈ W do
    foreach node v in a preorder traversal of T do
        if v's crimpable sequence has even length
         then
            Add to F all creases from v's crimpable
            sequence having M assignment.
        else if the surviving crease from v's
         crimpable sequence is already in F then
            Add to F all creases from v's crimpable
            sequence having the majority MV
            assignment.
        else
            Add to F all creases from v's crimpable
            sequence having the minority MV
            assignment.
        end
    end
end
```

Assume that there exists a different foldable MV assignment μ_2 for C such that $\mu_2(c) = \mu(c)$ for $c \in F$. For symmetry, let $\mu_1 = \mu$. We obtain W_1 and W_2 by running ForcingSetSVCP with input (C, μ_1) and (C, μ_2), respectively. As stated in Lemma 13, W_1 and W_2 are structurally identical. Let corresponding nodes $v_1 \in T_1$ and $v_2 \in T_2$ be a pair of maximal depth in the trees whose assignments differ. We call two crimpable sequences α_1 and α_2 *similar* if they have the same size, the same MV assignment read from left to right, and same interval angles.

The proof in [4] for forcing property is by contradiction with a case analysis as follows:

1. v_1 and v_2 are dissimilar. Let l be the length of the crimpable sequences corresponding to v_1 and v_2.
   (a) l is even.
   (b) l is odd.
      i. The creases of v_1 with majority MV assignment are in F.
      ii. The creases of v_1 with minority MV assignment are in F.
2. All corresponding nodes in W_1 and W_2 have similar crimpable sequences.

Case 1a leads to a contradiction as shown in [4]. In this case, v_1 and v_2 are root nodes in T_1 and T_2 because they do not have surviving crease. $l/2$ creases of v_1 are put into F with M assignment by the algorithm. The creases with M assignment in F have a copy in v_2 by Lemma 14, and remaining creases must have V assignment by Theorem 1. Then v_1 and v_2 are similar, which is a contradiction to the assumption that v_1 and v_2 are dissimilar.

Case 1(b)i also contradicts as shown in [4]. By Lemma 14, the creases with majority MV assignment of v_1 have a copy in v_2 with the same MV assignment and located in the same positions. All other creases in v_1, v_2 must have the opposite assignment by Theorem 1. Thus v_1 and v_2 are similar, a contradiction.

The difference is in Case 1(b)ii and Case 2. In Case 1(b)ii, we have two new cases due to the second and third steps of Algorithm 2:

A. The survivor of the root node is in F. (hence the majority in the root node are in F.)

B. The survivor of the root node is not in F. (hence the minority in the root node are in F.)

The proof for Case A is the same as the proof of Case 1(b)ii shown in [4]. Assume without loss of generality that the minority assignment of v_1 is M. In Case B, we must encounter a node with majority assignment V, or an equal number of M and V assignment. Assume to the contrary that we encounter nodes with majority M assignments only. At the root node r_1 of T_1, V creases are selected as a part of F since we assume surviving crease is not in F. Similarly, on each node from r_1 to v_1, V creases are selected as elements of F, which contradicts the assumption that the minority M creases of v_1 are in F. Let w_1' be the first node encountered on the path from v_1 to r_1 of T_1 having majority assignment V or an equal number of M and V assignments. As addressed in [4], the differences in v_1, v_2's crimpable sequences must be in first-time creases.

Let p_1 be the parent node of v_1. In Case B, if $p_1 \neq w_1'$, p_1's majority are M (by definition of w_1') and its creases with M should be in F (otherwise it contradicts the assumption that minority of v_1 are in F). The difference of first-time creases in v_1 and v_2 causes a contradiction of Theorem 1 on p_2 in the following cases: (1) w_1' have majority assignment V; (2) w_1' have equal M and V assignments. Figure 3 shows the first case. Assume c_1 and c_1' in Figure 3 are first-time creases and c_0' survives a crimp operation. Then c_3 and c_4 are copied to c_3' and c_4' by Lemma 14. This contradicts Theorem 1 on p_2.

In Case 2, the end creases form a flat-foldable equal-angle SVCP (cf. Lemma 2). Because Forc-

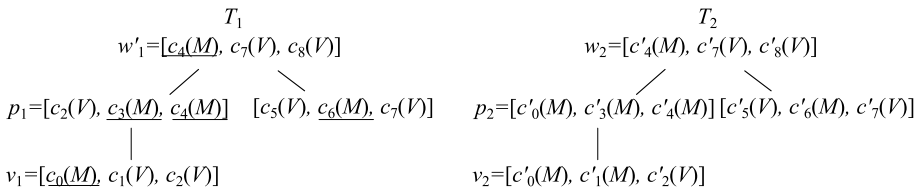


Figure 3: The case that first-time creases are different between v_1 and v_2, and w'_1 have majority assignment V.

ingSetSVCP puts all majority creases of the equal-angle SVCP into F, the remains of the creases in the equal-angle SVCP are forced to be with minority assignment, and it is not possible that μ_1 and μ_2 differ on the equal-angle SVCP. It follows $\mu_1 = \mu_2$, and therefore F is a forcing set.

We have shown that the theoretical minimum size of F is $n/2 + [n - 2m \geq 4]$ where m is the number of monocrimps performed in exhaustive crimps (cf. Theorem 12). Here we show how F yields $n/2+[n-2m \geq 4]$ creases by ForcingSetSVCP. The creases from the end equal-angle SVCP added to F in the second and third steps in the algorithm contributes to $(n-2m)/2+[n-2m \geq 4]$ creases. The same argument as [4] can be applied for the crimped creases: corresponding to each crimpable sequence α with size l, the forcing set algorithm adds to F precisely $\lfloor l/2 \rfloor$ creases; summing up over all crimp performed by the algorithm, we get m creases contributed to F.

6 Conclusion

We have developed an algorithm to find a minimum forcing set of flat-foldable SVCP in $O(n^2)$ time. We have shown that the size of such forcing set is $n/2$ or $n/2+1$. It is an open problem to find a minimum forcing set of arbitrary 2D origami. Enumeration of minimum forcing sets of a given MV pattern is an interesting problem as well. We believe that our result will help us to solve such open problems.

Acknowledgements

The second author was supported in part by MEXT/JSPS KAKENHI Grant Number 17H06287 and 18H04091.

References

[1] Z. Abel, J. Cantarella, E. D. Demaine, D. Eppstein, T. C. Hull, J. S. Ku, R. J. Lang, and T. Tachi. Rigid origami vertices: conditions and forcing sets. *Computational geometry*, 7(1), 2016.

[2] B. Ballinger, M. Damian, D. Eppstein, R. Flatland, J. Ginepro, and T. Hull. Minimum forcing sets for Miura folding patterns. In *Proceedings of the twenty-sixth annual ACM-SIAM symposium on Discrete algorithms*, pages 136–147. Society for Industrial and Applied Mathematics, 2015.

[3] M. Bern and B. Hayes. The complexity of flat origami. In *Proc. 7th Ann. ACM-SIAM Symp. on Discrete Algorithms*, pages 175–183, 1996.

[4] M. Damian, E. Demaine, M. Dulieu, R. Flatland, H. Hoffman, T. C. Hull, J. Lynch, and S. Ramaswami. Minimum forcing sets for 1D origami. *arXiv preprint arXiv:1703.06373v1*, 2015.

[5] E. D. Demaine and J. O' Rourke. *Geometric Folding Algorithms: Linkages, Origami, Polyhedra*. Cambridge University Press, 2007.

[6] E. Hawkes, B. An, N. M. Benbernou, H. Tanaka, S. Kim, E. D. Demaine, D. Rus, and R. J. Wood. Programmable matter by folding. *Proceedings of the National Academy of Sciences*, 107(28):12441–12445, 2010.

[7] T. Hull. Counting mountain-valley assignments for flat folds. *Ars Combinatoria*, 67:175–187, 2003.

[8] L. Ionov. 3D microfabrication using stimuli-responsive self-folding polymer films. *Polymer Reviews*, 53(1):92–107, 2013.

[9] J. Justin. Towards a mathematical theory of origami. In *Proceedings of 2nd international meeting origami science, technology*, pages 15–29, 1994.

[10] T. Kawasaki. On the relation between mountain-creases and valley-creases of a flat origami. In *Proceedings of 1st international meeting origami science, technology*, pages 229–237, 1989.

[11] T. G. Leong, P. A. Lester, T. L. Koh, E. K. Call, and D. H. Gracias. Surface tension-driven self-folding polyhedra. *Langmuir*, 23(17):8747–8751, 2007. PMID: 17608507.

[12] L. Mahadevan and S. Rica. Self-organized origami. *Science*, 307(5716):1740–1740, 2005.

[13] K. Ouchi and R. Uehara. Efficient enumeration of flat-foldable single vertex crease patterns. *IEICE Transactions on Information and Systems*, E102-D(3):416–422, 2019.

Efficient Segment Folding is Hard

Takashi Horiyama* Fabian Klute† Matias Korman‡ Irene Parada§ Ryuhei Uehara¶
Katsuhisa Yamanaka‖

Abstract

We introduce a computational origami problem which we call the *segment folding* problem: given a set of n line-segments in the plane the aim is to make creases along all segments in the minimum number of folding steps. Note that a folding might alter the relative position between the segments, and a segment could split into two. We show that it is NP-hard to determine if n line segments can be folded in n simple folding operations.

1 Introduction

Origami designers around the world struggle with the problem for finding a better way to fold an origami model. Recent advanced origami models require substantial *precreasing* of a prescribed mountain-valley pattern (getting each crease folded slightly in the correct direction), and then folding all the creases at once. For example, for folding the MIT seal *Mens et Manus* in "three easy steps", Chan [3] spent roughly three hours precreasing, three hours folding those creases, and four hours of artistic folding. The precreasing component is particularly tedious. Thus, we wonder if this process can be automated by folding robots. As of the writing of this paper, the most recent robots for folding paper can achieve only quite simple foldings (see, e.g., [2]), but we consider a future setting in which more difficult ones are possible. In such a setting, we have a series of portions of the sheet that need to be folded in some specified locations, and we would like to do it as fast as possible (that is, in the minimum possible number of foldings).

We consider one of the simplest folding operations possible called *all-layers simple fold*. An all-layers simple fold starts with a sheet in a flat folded state and consists of the following three steps: (1) choose a crease line, (2) fold all paper layers along this line, and (3) crease to make all paper layers flat again.

In our model we start with a plain sheet of paper with no folds or creases (say, a unit square). From that sheet we do all-layers simple folds one at a time until we have reached our desired shape. We note that this is one of the simplest folding models, and that many variants have been considered in the literature (see [1] for the other folding models).

This situation leads us to the natural *segment folding problem*: given n line segments $\ell_1, \ell_2, \ldots, \ell_n$ on a sheet of paper, we consider executing, one at a time, an all-layers simple fold operation along a line containing one or more segments ℓ_i for some $i \leq n$. The goal is to make a sequence of such folds in a way that all segments ℓ_i end in crease lines. Notice that, when we fold along a line L, the location of all segments in one side of the line are reflected (via line symmetry) onto the other halfplane. In particular, if L intersects the interior of some segment ℓ_i, it may create two segments that form a V shape and meet at the folded line (it will create two segments if and only if L and ℓ_i do not meet at a right angle and $\ell_i \not\subset L$). Whenever this happens, we have to fold the two subsegments since the original segment has been split into two.

Because folding through a line may increase the number of folds that are needed, we wonder if every instance can be folded with a finite number of fold operations[1]. We say that an instance is *foldable* if it has a solution to the segment folding problem in a finite number of moves. To date, we do not know if every problem instance is foldable; in Figure 1 we show an example admitting an infinite sequence of folds. Another natural question is related to efficiency: if an instance is foldable, can we find the sequence of folds? Ideally, one that minimizes that number of fold operations that are needed? In this paper we answer this question negatively by showing that it is NP-hard to decide if we can finish in at most n folds:

Theorem 1 *Deciding whether a segment folding problem instance of n segments can be solved with n folds is* NP*-hard.*

*Graduate School of Science and Engineering, Saitama University, horiyama@al.ics.saitama-u.ac.jp

†TU Vienna, fklute@ac.tuwien.ac.at

‡Tufts University, Matias.Korman@tufts.edu. Supported by MEXT Kakenhi No. 17K12635 and the NSF award CCF-1422311.

§Graz University of Technology, iparada@ist.tugraz.at. Supported by the Austrian Science Fund (FWF): W1230.

¶School of Information Science, JAIST, uehara@jaist.ac.jp

‖Faculty of Science and Engineering, Iwate University, yamanaka@cis.iwate-u.ac.jp

[1] We thank Takeshi Tokuyama for posing this question.

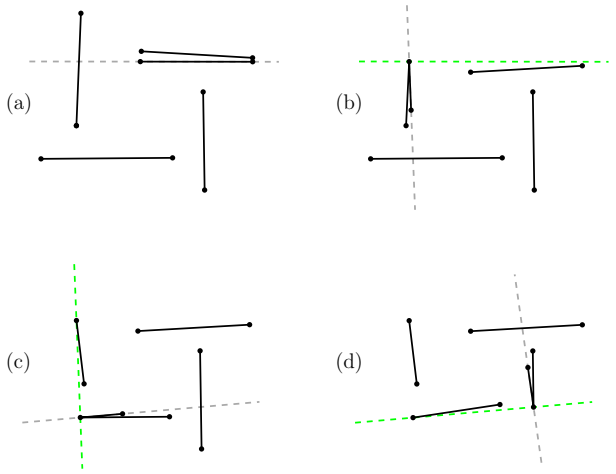


Figure 1: A problem instance that can be folded arbitrarily many times through non-folded input segments. If you fold along the dashed lines, the instance cycles through the four patterns and never ends. Note that in all cases we have two segments forming a V shape and three almost but not exactly orthogonal segments. Each folding changes the length and the position of the V shape, but the overall structure remains the same. Note that it is possible to fold this instance with a finite number of moves (folding any of the segments that is not part of the V shape).

Preliminaries. Let S be a given set of n segments in $\mathbb{R}^2$. Let ℓ_s be the supporting line of a segment $s \in S$ and let R_s be the right closed half plane bounded by ℓ_s. For a set $U \subseteq \mathbb{R}^2$ and a segment $s \in S$, we denote as $\mathcal{R}_s(U)$ the reflection of set U along the supporting line ℓ_s of segment s. A fold along the supporting line ℓ_s of a segment $s \in S$ creates a new set of segments $S' := \{(S \cap R_s) \cup \mathcal{R}(S \setminus R_s)\} \setminus \{s\}$ where the segments (or parts of them) from S in the left half plane defined by ℓ_s are reflected into the right half plane. We denote that operation a *fold along* s, or simply, *folding* s. Note that we only allow folding through segments of S.

When a folding line intersects another segment $s' \in S$, this second segment is split into two segments that share an endpoint. Both segments are now part of S and must be folded as usual. The goal of the *folding segment problem* is to find a sequence of folds such that S becomes empty. Specifically, we are interested in finding the shortest possible sequence, that is, the one with the fewest folds.

2 Reduction

We will prove Theorem 1 by reducing from 3SAT. A 3SAT formula is given by a set $\{x_1, \dots x_n\}$ of boolean variables and $\{c_1, \dots, c_m\}$ of clauses. Each clause contains the disjunction of three literals. A literal is a positive or negative occurrence of a variable. A formula $F(x_1, \dots, x_n)$ is the conjunction of the clauses $c_1, \dots, c_m$. In 3SAT we say F is satisfied if and only if for each clause c_j at least one of its literals evaluates to true and the other two to false. This problem is well known to be NP-complete[4]. For the reduction we will construct a set S of segments from a given CNF formula F.

We start by making straight-forward observations about basic folds.

Observation 1 *If a segment s lies on the boundary of the convex hull of our set of segments a fold along it does not change any of the remaining segments.*

Let S be a set of segments in $\mathbb{R}^2$, we say a segment $s \in S$ *stabs* $t \in S$, $s \neq t$, if the supporting line δ_s intersects t, but s and t do not share a common point. We say a segment $s \in S$ is stabbing, if there exists a $t \in S$ such that s stabs t.

Consider an instance in *general position*, that is, no two segments of S lie on the same line or on perpendicular lines, even after we have performed up to n folds. In that case each fold can only decrease the size of S by one.

Observation 2 *A solution sequence of a problem instance that is in general position cannot make a fold along a stabbing segment.*

This claim follows from (i) the fact that, in general position, folding along a stabbing segment does not decrease the number of segments of and (ii) we are interested in determining if a problem instance can be solved with n folds.

Observation 3 *Let S be a set of segments in general position in $\mathbb{R}^2$ and $s \in S$ be a non-stabbing segment, then if a fold along s does not produce a crossing between another two segments, no segment is split.*

The reduction we present uses for simplicity two perpendicular directions for the segments, and thus, constructs a set of segments that is not in general position. However, we could perturb the endpoints such that we obtain a set of segments in general position and whose endpoints have rational coordinates that can be represented in polynomially many bits.

Overview. We first give a brief overview of the reduction. Let $F(x_1, \dots, x_n)$ be a 3SAT formula. As usual, our reduction will construct variable and clause gadgets using $F(x_1, \dots, x_n)$ in a way so that any solution must first fold all variable gadgets. In each such gadget we will have the choice to fold one of two segments as the first one in the sequence, which will in turn encode the truth value of a variable. The folds themselves will

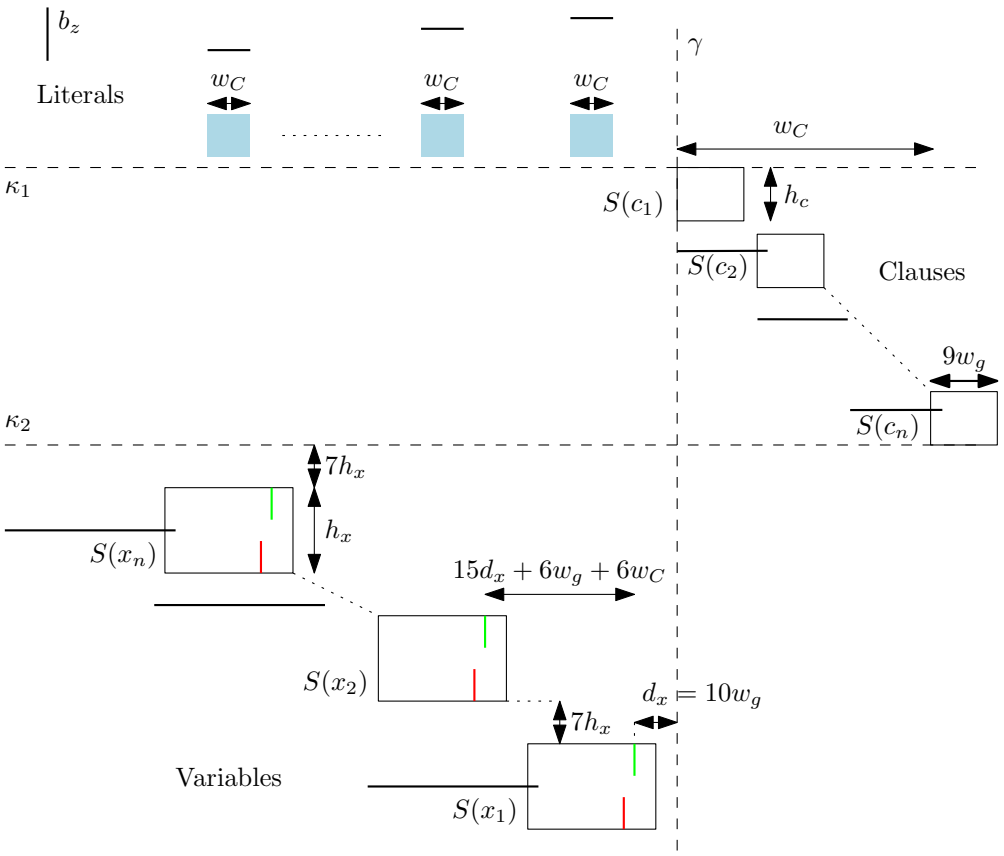


Figure 2: Overview of the reduction. Note that the lengths are not up to scale.

shift the position of some literal segments. These segments will enable the folding of a clause gadget if an only if the truth assignment of the associated variable satisfies the clause. Only after all variables are folded the clause gadgets can be folded. This will be possible in the required number of steps if and only if for each clause gadget at least one variable was folded in a way that the corresponding literal segment was placed in the enabling part of the clause gadget.

Global Layout. We define a vertical line γ and two horizontal lines κ_1 and κ_2. We call the region between κ_1 and κ_2 and to the right of γ the *clause region*. The region above κ_1 and to the left of γ is the *literal region* and the region below κ_2 and to the left of γ is the *variable region*. As the names indicate, we will place the clause, literal, and variable gadgets in the corresponding regions. See Figure 2 for an illustration, the details will become clear in the following sections. Furthermore, we define w_C as the sum of the width of all clause gadgets and $m_C = w_C/2$.

Variable Gadget. For each variable x the corresponding variable gadget consists of thirteen segments. We call these segments *true segment* t, *false segment* f, *true helper* t_h, *false helper* f_h, *true blocker one* t_b^1, *true blocker two* t_b^2, *true blocker three* t_b^3, *false blocker one* f_b^1, *false blocker two* f_b^2, *false blocker three* f_b^3, *next blocker one* b_1, *next blocker two* b_2, and *next blocker three* b_3. For a variable x we denote with $S(x)$ the set of the thirteen segments corresponding to this variable. The positioning of them is drawn in Figure 3.

Let w_g be the horizontal distance between the t_h and f_h segment. For a variable x, the gadget $S(x)$ without the next blocker segment b_2 has width $w_x = 37/2w_g + w_C$ and constant height h_x. Nearly all horizontal space is between the two helper segments and the true and false segment. It includes the *literal strip* of variable gadget $S(x_i)$, see the light-blue region in Figure 3. This strip has width w_C.

The key property of the placement of the segments is that if we start folding t, the segment t_b^2 is stabbed by the segment f, forcing us to fold (t_b^1 and) t_h before f. Symmetrically, folding first along f forces to fold (f_b^1 and) f_h before t.

It remains to explain the function of the three next blocker segments b_1, b_2, and b_3 of a variable gadget $S(x_i)$ and the placement of b_2. These segments guarantee that no segment in the variable gadget $S(x_{i+1})$ can be folded before all the segments in $S(x_i)$ have been folded along.

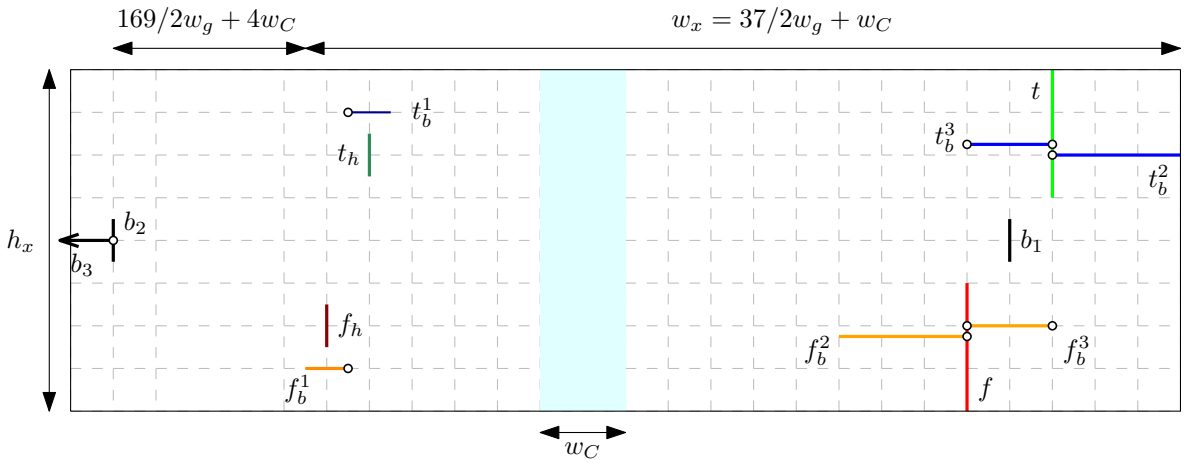


Figure 3: A variable gadget. (The drawing is not up to scale.) The literal strip is shown in lightblue. The marked points indicate that the corresponding segments are strictly separable with a vertical line.

The segment $b_3 \in S(x_i)$ is a long horizontal segment, and above it lie the segments in $S(x_{i+1}) \setminus \{b_3 \in S(x_{i+1})\}$. As we will see, it is the last segment of the variable gadget to be folded, and it prevents segments in $S(x_{i+1})$ from being folded before. The b_2 segment is there to reset the following variable gadgets properly, i.e., it makes sure the next variable is at the same distance of γ as the ones before.

We now describe how the variable gadgets are placed relative to each other. We begin with variable x_1 and place it such that the true segment $t \in S(x_1)$ is $d_x = 10w_g$ units to the left of the vertical line γ and the top-point of $t \in S(x_1)$ is $(n-1)8h_x + 7h_x$ units below the horizontal line κ_2. The next variable gadgets $S(x_2), \ldots, S(x_n)$ are placed to the left and above $S(x_1)$ always leaving $15d_x + 6w_g + 6w_C$ units of horizontal space between t in $S(x_{i-1})$ and the t segment of $S(x_i)$, as well as $2h_x$ units vertical space between the upper end-point of $t \in S(x_{i-1})$ and the lower end-point of $f \in S(x_i)$. We place $b_2 \in S(x_i)$ at $10d_x + 5w_C$ horizontally to the left of $t \in S(x_i)$. Finally, b_3 is placed directly to the left of b_2 and extended by $10w_C$ units to the left.

We now show that there are only four ways to fold a variable gadget.

Lemma 2 *There are exactly four ways in which a variable gadget can be folded in thirteen steps, namely:*

$$t \to t_b^1 \to t_h \to t_b^2/t_b^3 \to f \to f_b^1 \to f_h \to f_b^2/f_b^3 \to b_1 \to b_2 \to b_3 \text{ or}$$
$$f \to f_b^1 \to f_h \to f_b^2/f_b^3 \to t \to t_b^1 \to t_h \to t_b^2/t_b^3 \to b_1 \to b_2 \to b_3,$$

where t_b^2/t_b^3 means that we fold the two segments at that point in any relative order.

Proof. Let $S = \{t, t_h, t_b^1, t_b^2, t_b^3, f, f_h, f_b^1, f_b^2, f_b^3, b_1, b_2, b_3\}$ be the segments of a variable gadget constructed as above. Then the only three non stabbing segments are b_2, t and f. Notice that b_2 cannot be folded before b_1 and b_3, since b_1 would intersect b_3 after the folding. By Observation 2 this means the only possible folds are along t and f, respectively.

W.l.o.g. assume we first fold t. Let S' be the new set of segments. Note that, since t was not stabbing and no crossings were created, by Observation 3, we did also not create a new segment and no two segments are sharing a common supporting line. Identify the segments in S with their reflected counterparts in S'. Then f and b_1 are now stabbing t_b^2, but t_b^1 does not stab any segment anymore. All other segments are still stabbing the same segments as before. Hence the only possible fold is along the supporting line of t_b^1. By the same argumentation we get that the following folds must be in order t_h, t_b^2/t_b^3, f, f_b^1, f_h, f_b^2/f_b^3, b_1, b_2, and b_3. □

Next, we will show that the folding sequences of length $13n$ for the variable gadgets all fold the gadgets in order of the corresponding indices. The proof of the lemma is omitted due to space constraints.

Lemma 3 *Let $x_1, \ldots, x_n$ be the variables of a* 3SAT *formula and consider the variable gadgets $S(x_1), \ldots, S(x_n)$ as above, then the variable gadgets can be folded in $13n$ steps if and only if they are folded in order of their indices.*

Clause Gadget. A clause gadget consists of four segments such that each one stabs the next in a cyclic way. See Figure 4(a). Thus, by Observation 2, the four segments cannot be folded unless there is another segment whose supporting line splits the gadget. If one or more lines goes through the gadget along the shaded region

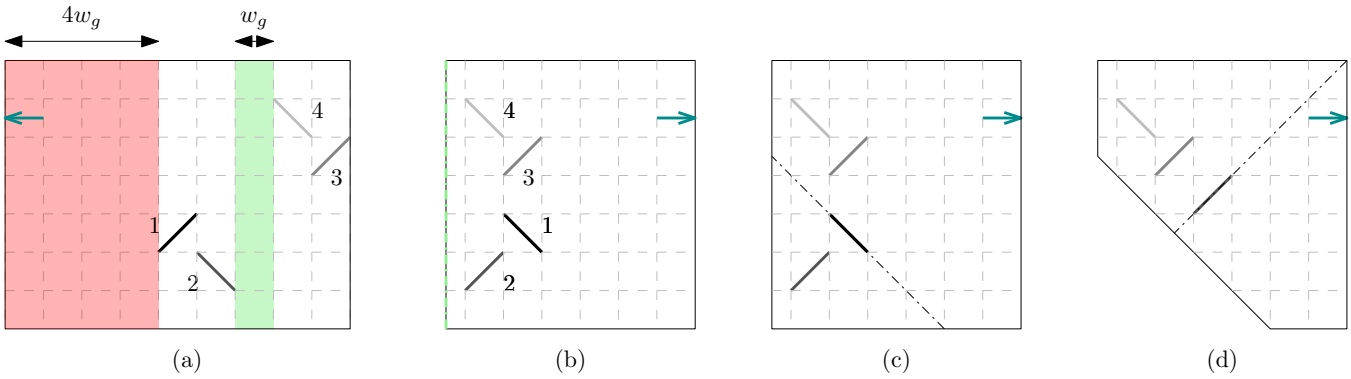


Figure 4: Clause gadget.

in Figure 4(a), this gadget can be folded as shown in Figure 4(b-d) (the remaining folds follow from Observation 1). Notice that it does not matter how many lines go through the shaded region. As for the variable gadgets, we denote with $S(c_j)$ the four segments corresponding to the clause c_j in S. Additionally, we call the gap between the four segments, of width w_g, the *good zone* of the clause gadget $S(c_j)$ and the vertical strip of width $4w_g$, starting at the left-most point of any segment in $S(c_j)$, the *bad zone*. Finally, to guarantee the clauses get folded in order of their indices, we also introduce a blocker segment $b_j \in S(c_j)$ for each clause c_j except c_1, represented by the green arrow in Figure 4. In total, one clause gadget without its blocker, but including the good and bad zones, has width $9w_g$. We obtain the following lemma.

Lemma 4 *A clause gadget $S(c)$ in a setting where no horizontal segment can be folded can be folded in four steps if and only if there is a segment whose supporting line goes through its good zone.*

Let h_c be the height of one clause gadget. We position the gadgets such that the bad zone of $S(c_1)$ starts at γ and the bottommost point of $S(c_1)$ is $(m-1)2h_c$ units above κ_2 and h_c units below κ_1. The next clause $S(c_2)$ is placed h_c units below the bottommost segment in $S(c_1)$ and such that its bad zone aligns with the rightmost point of the four segments of $S(c_1)$. The horizontal segment $b_j \in S(c_j)$ is $10w_g$ units long, w_g of it lying in the bad zone of $S(c_j)$ and the rest placed below $S(c_{j+1})$. Its y-coordinate makes it stab segment 4 in $S(c_j)$, see Figure 4. We will see in the following section how these blocker segments guarantee that the order on the clauses is fixed by their indices.

Literal gadget. As explained above, the clause gadgets can not be folded without the supporting line of another segment separating the cyclically intersecting segments. Let c_j be a clause and x_i a variable that occurs in c_j. Then, we create one vertical segment $S(z_{i,j})$ in the literal gadget corresponding to that literal $z_{i,j}$. For each clause c_j and variable x_i, the segment $S(z_{i,j})$ is placed w_g above κ_1 and inside the literal strip of the gadget corresponding to variable x_i, i.e., the corresponding light-blue strip in Figure 3.

More precisely, $S(z_{i,j})$ is placed with an offset to the true segment t of $S(x_i)$ of $(16+\frac{1}{4})w_g + (j-1)w_c$ if x_i appears positive in c_j and at $(20+\frac{3}{4})w_g + (j-1)w_c$ if x_i appears negated in c_j. Additionally, we give a literal segment $S(z_{i,j})$ a negative horizontal offset $\delta = \frac{1}{10}w_g$ if $z_{i,j}$ is the literal with smallest index i in the clause, and a positive horizontal offset of δ if the literal appears with the largest index i.

For each $z_{i,j}$ we further place a blocker segment $w_g(n-i+1)$ units above the toppoint of $S(z_{i,j})$, compare Figure 2. Observe that no $S(z_{i,j})$ can now be folded before the corresponding blocker.

Finally we introduce one vertical segment b_z. Horizontally we place it $11d_x + 5w_C$ units of horizontal distance to the left of $t \in S(x_n)$. Vertically we put its bottommost point w_g units above the topmost point of the $a_{i,j}$ segments. The segment b_z then extends $(n+1)w_g$ units to the top.

Observation 4 *Let $F(x_1,\ldots,x_n)$ be a* 3SAT *formula and consider the variable gadgets $S(x_1),\ldots,S(x_n)$ and literal gadgets $S(z_{i,j})$ as above, then for all literals $z_{i,j}$ the segments in $S(z_{i,j})$ are folded after all segments in $S(x_1),\ldots,S(x_n)$.*

Correctness. It remains to argue the correctness of our reduction. The proofs of Lemmas 5 and 6 are omitted for space reasons. Theorem 1 then follows directly from Lemma 8.

Lemma 5 *Given a* 3SAT *formula $F(x_1,\ldots,x_n)$, let $S(x_1),\ldots,S(x_n)$ be the variable gadgets constructed as above. Then, after folding $S(x_1),\ldots,S(x_i)$, the horizontal distance between $t \in S(x_{i+1})$ and γ is d_x.*

Lemma 6 *Given a* 3SAT *formula $F(x_1, \ldots, x_n)$, let S be the set of segments constructed by the reduction. Let $\mathcal{S}$ be a folding sequence of the variable gadgets up to $S(x_{i-1})$. Let S' be the segments after a fold along $s \in \{t, f\} \subset S(x_i)$. If $s = t$, then for every literal gadget $S(z_{i,j})$ with clause $c_j \in F(x_1, \ldots, x_n)$ we have that $S'(z_{i,j})$ is in the good zone of $S'(c_j)$ if $z_{i,j}$ is a positive literal, and in the bad zone otherwise. If instead $s = f$, then for every literal gadget $S(z_{i,j})$ with clause $c_j \in F(x_1, \ldots, x_n)$ we have that $S'(z_{i,j})$ is in the good zone of $S'(c_j)$ if $z_{i,j}$ is a negative literal, and in the bad zone of $S'(c_{j+1})$ otherwise. Moreover, it is the only literal segment reflected to this coordinate.*

We are now ready to state the same ordering lemma for the clauses as for the variables.

Lemma 7 *Let $F(x_1, \ldots, x_n)$ be a CNF formula and consider the segments S as above, then after all variable gadgets $S(x_1), \ldots, S(x_n)$ are folded the clause gadgets can only be folded in order $S(c_n), \ldots, S(c_1)$.*

Proof. The lemma follows as for Lemma 3 once we realize that, by Lemma 6, after folding all variable gadgets $S(x_1), \ldots, S(x_n)$, we find that the segments $S(z_{i,j})$ are reflected such that they are either in the good or bad zone of the corresponding $S(c_j)$. This means that for every clause gadget $S(c_j)$, aside from $S(c_n)$, we find that the literal segments $S(z_{i,j})$ are stabbing the blocker segments $b \in S(c_{j+1})$. □

The reduction can be computed in polynomial time in n. More precisely, for a given CNF formula $F(x_1, \ldots, x_n)$, we can construct the set S of segments in polynomial time of n using these gadgets with suitable spaces. Then we can prove that $F(x_1, \ldots, x_n)$ is satisfiable if and only if the constructed set S can be folded in $|S|$ steps. Theorem 1 follows directly from the following lemma.

Lemma 8 *Let $F(x_1, \ldots, x_n)$ be a formula of a* 3SAT *instance with bounded degree five and S the set of segments constructed from F as above, then F is satisfiable if and only if S can be folded in $|S|$ steps.*

Proof. Let S be the set of segments constructed as above from a 3SAT CNF formula $F(x_1, \ldots, x_n)$, and let $\mathcal{S}$ be a folding sequence of length $|S|$, folding S. Combining Lemmas 3, 4 and 7, as well as Observation 4, we get that in S the variable gadgets must all be folded first, then the literal and clause gadgets. Furthermore, by Lemma 2, each fold of a variable gadget starts with the t or f segment for each variable. By Lemma 6 we know that a literal segment gets reflected into the good zone of a clause if and only if it appears positive and the t segment is folded first or it appears negated and the f segment of the corresponding variable is folded first. Additionally, no clause gadget can be folded in the required number of steps if there is no segment inside its good zone. Finally observe that in such a folding sequence no two segments ever cross. As a result, we find a satisfying assignment of the variables of F by simply setting x_i to true if in $\mathcal{S}$ the true segment $t \in S(x_i)$ was folded before $f \in S(x_i)$ and to false if the converse holds.

For the reverse direction, let $F(x_1, \ldots, x_n)$ be again a 3SAT CNF formula and $\mathcal{F}$ a fulfilling assignment. Now let S be the set of segments constructed as above. We construct a folding sequence $\mathcal{S}$ as follows. For every variable gadget $S(x_i)$ pick $t \in S(x_i)$ to be folded first if x_i is set to true in $\mathcal{F}$ and pick $f \in S(x_i)$ otherwise. It follows from Lemmas 3, 4, and 7, and Observation 4, that this fixes the folding sequence. Assume that$|\mathcal{S}| \neq |S|$ and there is a fold in $\mathcal{S}$ which splits another segment, aligns them, or leads to a crossing. By the way we picked $\mathcal{S}$ none of these three cases can occur when we fold the n variable gadgets $S(x_1), \ldots, S(x_n)$. Hence, the fold splitting, aligning, or crossing two segments must happen when the clause gadgets are folded, but then, with Lemma 4, this can only happen in a clause c_j, for which none of the corresponding literal segments $S(z_{i,j})$ was in the good zone of $S(c_j)$. However, since $\mathcal{F}$ is satisfying, this can not be the case by Lemma 6. □

3 Concluding Remarks

We showed that deciding if there is a solution to the Segment Folding Problem with at most n folding operations is NP-hard when we don't allow folds along stabbing segments. This is always the case for an instance in general position. For simplicity, the reduction presented uses two perpendicular directions and constructs an instance that is not in general position. However, it is possible to modify the reduction to be in general position. Intuitively, duplicate each segment at a small angle such that the copy stabs the original segment. The length of the copied segment is such that the same segments which stab the original stab the copied segment. In that way, the original segment is always folded first. Furthermore, after this fold, the copied segment is on the boundary of the convex hull and can be deleted by Observation 1.

Acknowledgements

This work was started during the FWF-JSPS Japan-Austria Bilateral Seminar "Computational Geometry Seminar with Applications to Sensor Networks" held in Zao (Japan) in 2018. The authors want to thank all participants for the fruitful discussions.

References

[1] H. A. Akitaya, E. D. Demaine, and J. S. Ku. Simple folding is really hard. *Journal of Information Processing*, 25:580–589, 2017.

[2] D. Balkcom. Origami robot folding a paper hat. `https://www.youtube.com/watch?v=djPdzCj4k14`, 2004. Accessed: 2019-06-19.

[3] B. Chan. The making of mens et manus (in origami), vol. 1. `http://techtv.mit.edu/collections/chosetec/videos/361-the-making-of-mens-et-manus-in-origami-vol-1`, 2007. Accessed: 2019-06-19.

[4] M. R. Garey and D. S. Johnson. *Computers and intractability: a guide to the theory of NP-completeness*. W. H. Freeman & Co, New York, NY, USA, 1979.

Nurimisaki and Sashigane are NP-complete

Chuzo Iwamoto* Tatsuya Ide†

Abstract

Nurimisaki and Sashigane are Nikoli's pencil puzzles. We study the computational complexity of Nurimisaki and Sashigane puzzles. It is shown that deciding whether a given instance of each puzzle has a solution is NP-complete.

1 Introduction

The Nurimisaki puzzle is played on a rectangular grid of cells (see Fig. 1(a)). Initially, some of the cells contain circles, where each circle contains a number or no number. The purpose of the puzzle is to fill in cells in black (see Fig. 1(i)) according to the following rules [1]: The set of all white cells is regarded as a land, and sets of connected black cells are seas. (In Fig. 1(i), there are eight seas and one land.) (1) Cells with a circle remain white and must be a "Misaki" (Promontory), while cells without a circle cannot be a "Misaki." A Misaki cell has only one of the cells next to it remaining white and the rest have to be black. Each white cell without a circle has at least two white cells next to it. (Namely, a cell has a circle if and only if it is at a Misaki position.) (2) The numbers in the circles indicate how many white cells form a straight line from the Misaki cell (see the four red cells starting from ④ in Fig. 1(d)). At the cells with empty circles, any number of white cells may form a straight line. (3) The set of all white cells is connected. (Namely, there is exactly one land on the grid.) (4) Neither black cells nor white cells can be linked to be a 2×2 square or larger. (The Japanese word "Nuri" in Nurimisaki means "painting.")

Figure 1(a) is the initial configuration of a Nurimisaki puzzle. In this figure, there are seven circles, three of which contain numbers 4, 2, and 3. From Figs. 1(b)–(i), the reader can understand the basic technique for finding a solution. (b) Since there is a red circle in the yellow area, white cells cannot form a straight line downward from number ④. Thus, cell a must be colored black (see (c)), since a is 4 cells away from number ④. (c) The three cells between number ④ and cell a must not be colored black; such cells are indicated by • in Fig. 1. Since ④ is a promontory (Misaki), cells b and c must be colored black. Since there is a circled number ② in the blue area, white cells must form a straight line downward from number ③ (see the green area in (d)). (d) Cells d and e must be colored black. (e) Since red and green cells belong to one land, two blue cells are not colored black. Since white cells cannot form a 2×2 square, cell f is black (see (f)). (f) Since ② is a promontory (Misaki), either cell g or h must be colored black. (g) is an invalid placement of black cells. (If cell g is colored black, then cell i becomes a promontory (Misaki), but it has no circle.) Therefore, cells g and i are white, and cell h is black (see (h)). (h) Since the yellow cell is a promontory (Misaki), cells j and k are black. Since black cells cannot form a 2×2 square, at least one of the three cells l, m, and n is white. (i) is one of the multiple solutions.

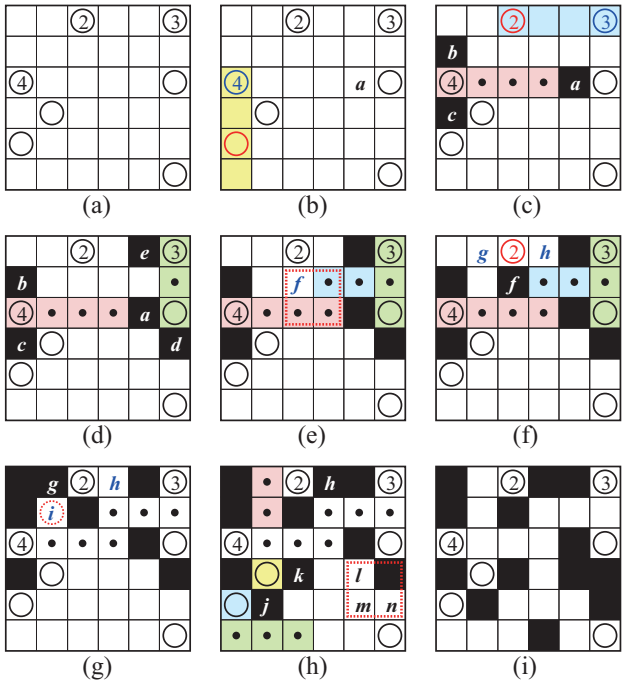


Figure 1: (a) Initial configuration of a Nurimisaki puzzle. (b)–(i) are the progress from the initial configuration to a solution. (g) is an invalid placement of black cells.

The Sashigane puzzle is played on a rectangular grid of cells (see Fig. 2(a)). Initially, there are circled numbers, empty circles, and arrows. The purpose of the puzzle is to divide the grid into L-shaped blocks ac-

*Graduate School of Engineering, Hiroshima University, chuzo@hiroshima-u.ac.jp

†School of Integrated Arts and Sciences, Hiroshima University, b161622@hiroshima-u.ac.jp

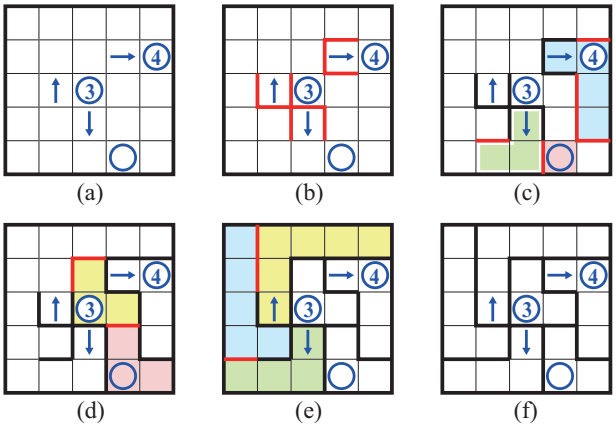


Figure 2: (a) Initial configuration of a Sashigane puzzle. (b)–(f) are the progress from the initial configuration to a solution.

cording to the following rules [2]: (1) Cells with circles form the knee (bend) in a block (see Fig. 2(f)). (2) The number in a circle shows the number of cells in its block. Circles without numbers may have any number of cells. (3) Cells with arrows form one end of its block; the arrow points towards the knee of this block. (The Japanese word "Sashigane" means "steel square," a tool used in carpentry [3].)

Figure 2(a) is the initial configuration of a Sashigane puzzle. In this figure, there are three arrows and three circles, two of which contain numbers 3 and 4. (b) Each arrow is surrounded by a "wall," which becomes one end of an L-shaped block. (c) Since there is a circle in the red cell, the three green cells must form a ⅃-shaped block. The blue cells form a ⅂-shaped block, where ④ is the knee of the block. (d) Circled number ③ and empty circle ○ are the knees of three-cell blocks. (e) The remaining cells are divided into three L-shaped blocks. (f) is the solution.

In this paper, we study the computational complexity of the decision version of the Nurimisaki and Sashigane puzzles. The instance of each problem is a rectangular grid of cells. In the *Nurimisaki puzzle problem*, some of the cells of the grid contain circles, and each circle contains a number or no number. In the *Sashigane puzzle problem*, there are circled numbers, empty circles, and arrows on the grid. Each problem is to decide whether there is a solution to the instance. Now we are ready to present our main theorem.

Theorem 1 *The Nurimisaki and Sashigane puzzle problems are NP-complete.*

The proofs are given in Section 2.2 and Appendix. It is clear that the Nurimisaki puzzle problem belongs to NP, since the game ends when all empty cells are colored black or white. The Sashigane puzzle problem also belongs to NP, since the game ends when all edges (of cells) become a thick line (wall) or a thin line (no wall).

There has been a huge amount of literature on the computational complexities of games and puzzles. In 2009, a survey of games, puzzles, and their complexities was reported by Hearn and Demaine [11]. After the publication of this book, the following Nikoli's pencil puzzles were shown to be NP-complete: Dosun-Fuwari [16], Fillmat [23], Hashiwokakero [6], Hebi, Satogaeri, and Suraromu [18], Herugolf and Makaro [15], Kurodoko [19], Kurotto and Juosan [17], Norinori and LITS [7], Numberlink [4], Pipe link [24], Shakashaka [9], Shikaku and Ripple Effect [22], Usowan [14], Yajilin and Country Road [12], and Yosenabe [13]. Last year, Nikoli's pencil puzzles, Pencils [20] and Sto-Stone [5], were shown to be NP-complete in CCCG 2018.

2 NP-completeness of Nurimisaki

2.1 3SAT Problem

The definition of 3SAT is mostly from [10]. Let $U = \{x_1, x_2, \ldots, x_n\}$ be a set of Boolean *variables*. Boolean variables take on values 0 (false) and 1 (true). If x is a variable in U, then x and $\overline{x}$ are *literals* over U. The value of $\overline{x}$ is 1 (true) if and only if x is 0 (false). A *clause* over U is a set of literals over U, such as $\{\overline{x_1}, x_3, x_4\}$. It represents the disjunction of those literals and is *satisfied* by a truth assignment if and only if at least one of its members is true under that assignment.

An instance of PLANAR 3SAT is a collection $C = \{c_1, c_2, \ldots, c_m\}$ of clauses over U such that (i) $|c_j| \leq 3$ for each $c_j \in C$ and (ii) the bipartite graph $G = (V, E)$, where $V = U \cup C$ and E contains exactly those pairs $\{x, c\}$ such that either literal x or $\overline{x}$ belongs to the clause c, is planar.

The PLANAR 3SAT problem asks whether there exists some truth assignment for U that simultaneously satisfies all the clauses in C. This problem is known to be NP-complete. For example, $U = \{x_1, x_2, x_3, x_4\}$, $C = \{c_1, c_2, c_3, c_4\}$, and $c_1 = \{x_1, x_2, x_3\}$, $c_2 = \{\overline{x_1}, \overline{x_2}, \overline{x_4}\}$, $c_3 = \{\overline{x_1}, \overline{x_3}, x_4\}$, $c_4 = \{\overline{x_2}, \overline{x_3}, \overline{x_4}\}$ provide an instance of PLANAR 3SAT. For this instance, the answer is "yes," since there is a truth assignment $(x_1, x_2, x_3, x_4) = (0, 1, 0, 0)$ satisfying all clauses. It is known that PLANAR 3SAT is NP-complete even if each variable occurs exactly once positively and exactly twice negatively in C [8] (this restriction is used in Appendix).

2.2 Transformation from an Instance of 3SAT to a Nurimisaki Puzzle

We present a polynomial-time transformation from an arbitrary instance C of PLANAR 3SAT to a Nurimisaki

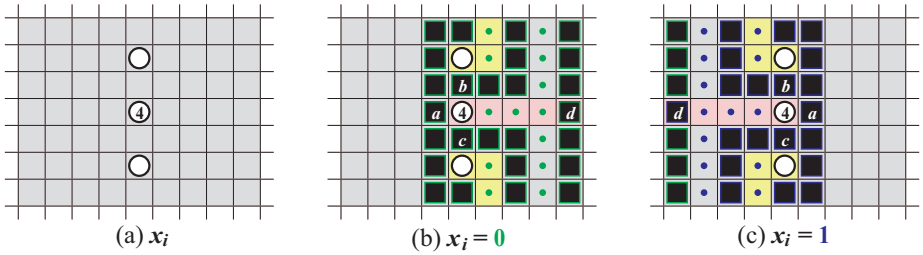


Figure 3: (a) Variable gadget of Nurimisaki transformed from x_i. (b) Assignment $x_i = 0$. (c) Assignment $x_i = 1$.

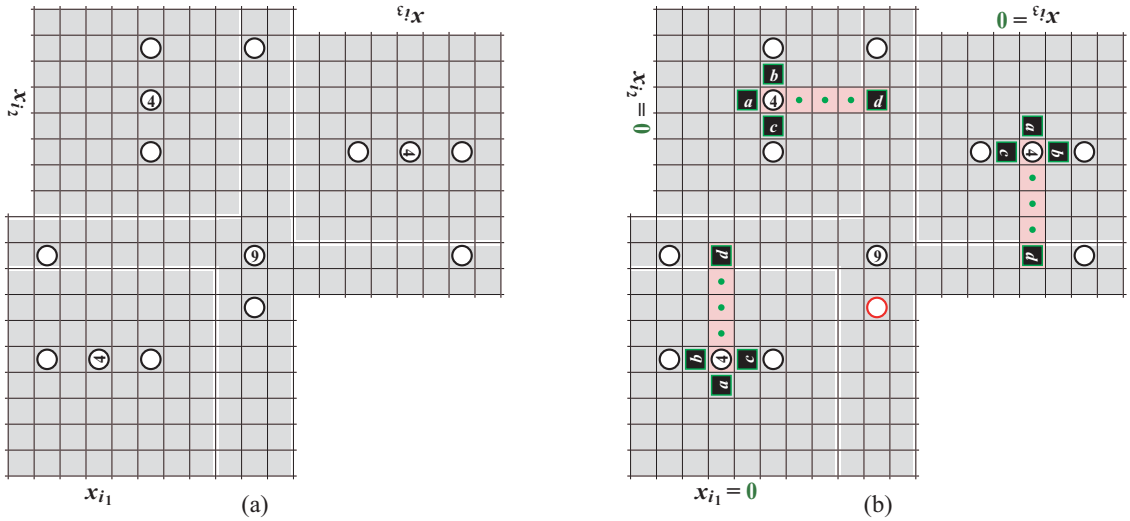


Figure 4: (a) Clause gadget of Nurimisaki transformed from $c_j = \{x_{i_1}, x_{i_2}, x_{i_3}\}$. (b) is an invalid placement of black cells. If $x_{i_1} = x_{i_2} = x_{i_3} = 0$, there is no solution to number ⑨.

puzzle such that C is satisfiable if and only if the puzzle has a solution.

Each variable $x_i \in \{x_1, x_2, \ldots, x_n\}$ is transformed into the variable gadget as illustrated in Fig. 3(a), which is composed of three circles, one of which contains number 4.

There are two possible solutions to circled number ④. In Figs. 3(b) and 3(c), the three red cells between number ④ and cell d remain white. (In this paper, all non-black cells are called "white" cells, although they are colored red, yellow, grey, green, blue, or orange in Figs. 1–9.) Since circled number ④ is a promontory (Misaki), cells a, b, c and d are colored black. Figures 3(b) and 3(c) correspond to assignment $x_i = 0$ and $x_i = 1$, respectively.

Clause $c_j \in \{c_1, c_2, \ldots, c_m\}$ is transformed into the clause gadget as illustrated in Fig. 4(a), which contains three variable gadgets of Fig. 3(a). A clause gadget is composed of 14 circles, where three circles contain number 4 and one circle contains number 9. There is a circle which is two cells down from number ⑨ (see the red empty circle in Fig. 4(b)). Thus, number ⑨ cannot form a straight line of white cells downward.

Let $c_j = \{x_{i_1}, x_{i_2}, x_{i_3}\}$. Suppose $x_{i_1} = x_{i_2} = x_{i_3} = 0$ (see Fig. 4(b)). In this case, there are three black cells d, which are six cells away from number ⑨. Therefore, there is no way to form a straight line of nine white cells from number ⑨. Hence, Fig. 4(b) is an invalid placement of black cells. On the other hand, if at least one of the variables x_{i_1}, x_{i_2}, and x_{i_3} is 1 (see Fig. 5), there is a valid placement of black cells.

Fig. 6(a) is a connection gadget, where circled numbers ④ appear at regular intervals of length four. If the leftmost ④ forms a straight line of four white cells to the right (see the red area of Fig. 6(b)), then the second and third ④ must form straight lines of four white cells to the right. Thus, signal "$x_i = 0$" is transmitted to the right.

On the other hand, if the leftmost ④ forms a straight line of four white cells to the left (see Fig. 6(c)), then the second and third ④ can form straight lines of four white cells to the left. In this case, signal "$x_i = 1$" is transmitted to the right.

If you want the distance between two circled numbers ④ to be even (see Figs. 6(d) and 6(e)), then a pair of circled numbers ④ are placed at an interval of length five.

Figure 7 is the right branch gadget, where a signal

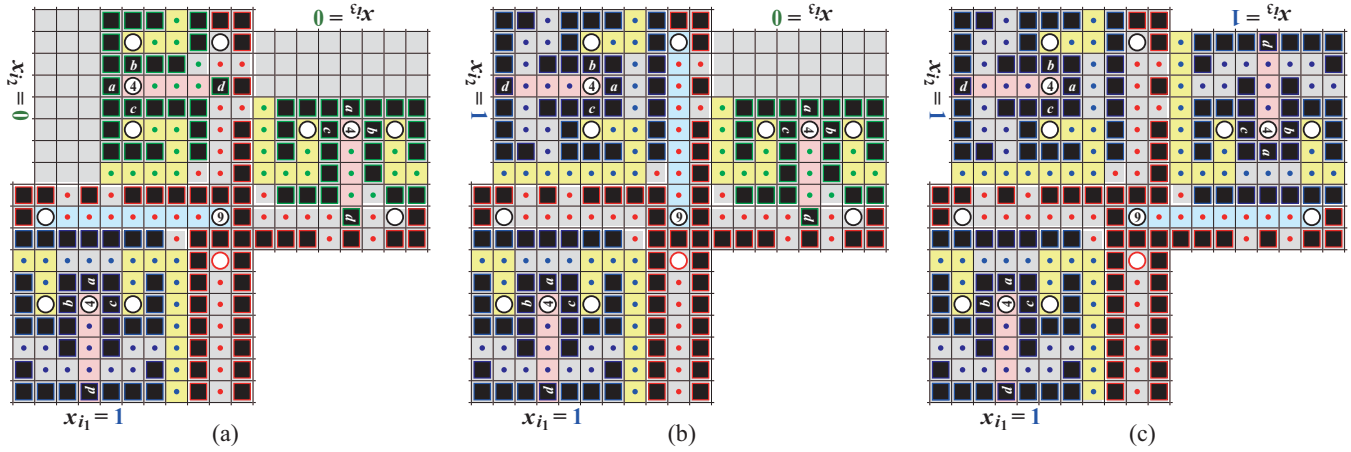


Figure 5: If at least one of the three variables x_{i_1}, x_{i_2}, and x_{i_3} is 1, then there is a solution.

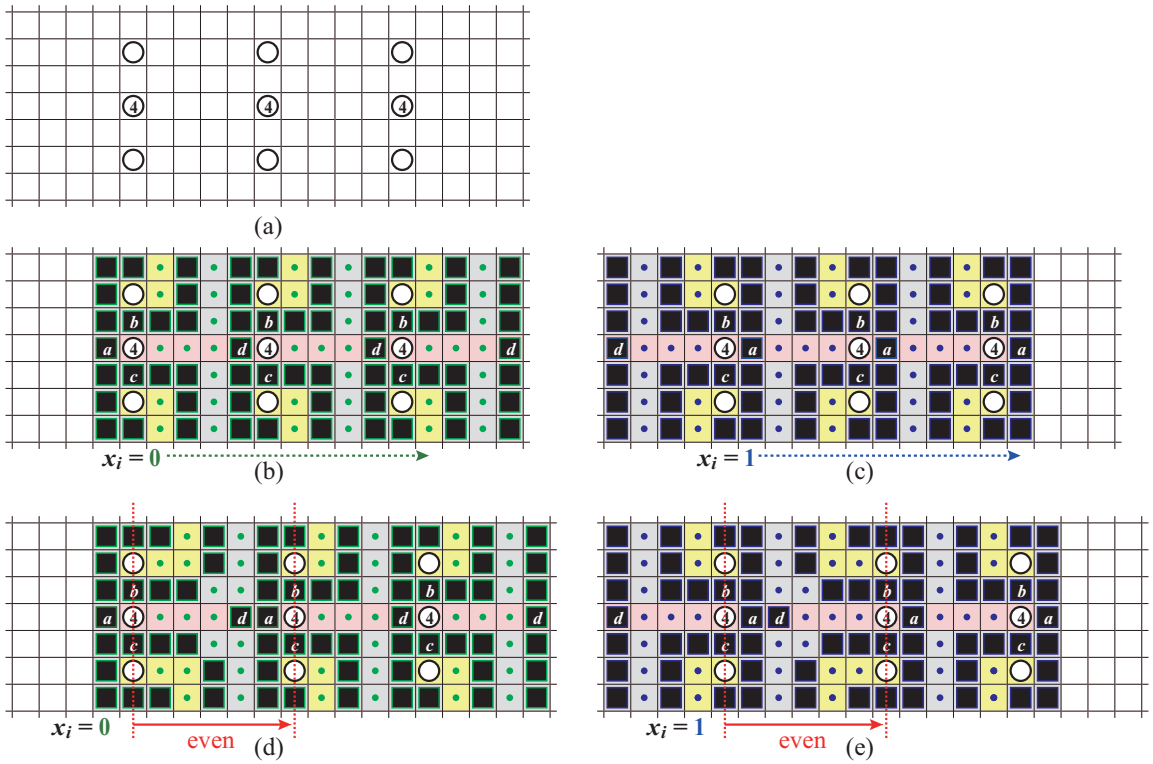


Figure 6: (a) Connection gadget. (b) Assignment $x_i = 0$. If the leftmost ④ forms a straight line of four white cells to the right (see the red area), then the second and third ④ must form straight lines of four white cells to the right. (c) Assignment $x_i = 1$. (d,e) Connection gadget of even length.

is separated and transmitted to two directions. (In Fig. 9, this gadget is used between variable x_4 and clauses c_1, c_2.) The left branch gadget can be constructed similarly. Figure 7 is also used as a right turn gadget. (In Fig. 9, right (resp. left) turn gadgets are used between variable x_1 and clause c_1 (resp. x_2 and c_2).)

Figure 9 is a Nurimisaki puzzle transformed from $C = \{c_1, c_2\}$ and $U = \{x_1, x_2, x_3, x_4\}$, where $c_1 = \{\overline{x_1}, x_2, x_4\}$ and $c_2 = \{\overline{x_2}, \overline{x_3}, x_4\}$. In this figure, there are two large white areas separated by connection, variable, and clause gadgets. Figure 8 is an enlarged illustration of variable gadget x_1 and its surroundings. The border of each white area is filled with white cells (see ∘ in the orange area of Fig. 8). In each white area in the orange border, a black-cell row and a white-cell row appear alternately (see □□ · · · □ and ∘∘ · · · ∘ in the white area). From this construction, the instance C of PLA-

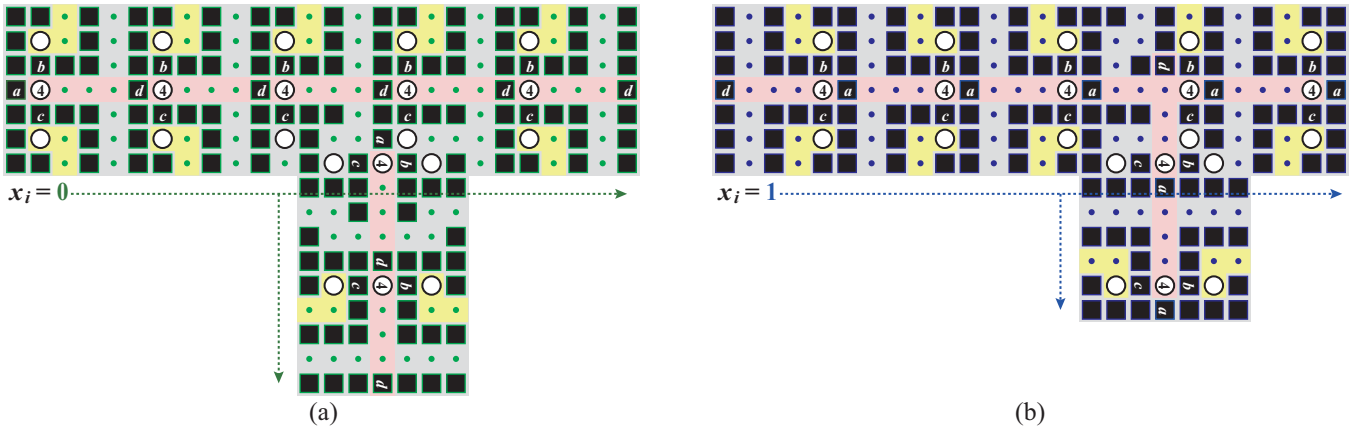


Figure 7: Right branch gadget. This gadget is also used as a right turn gadget. Left branch gadget can be constructed similarly.

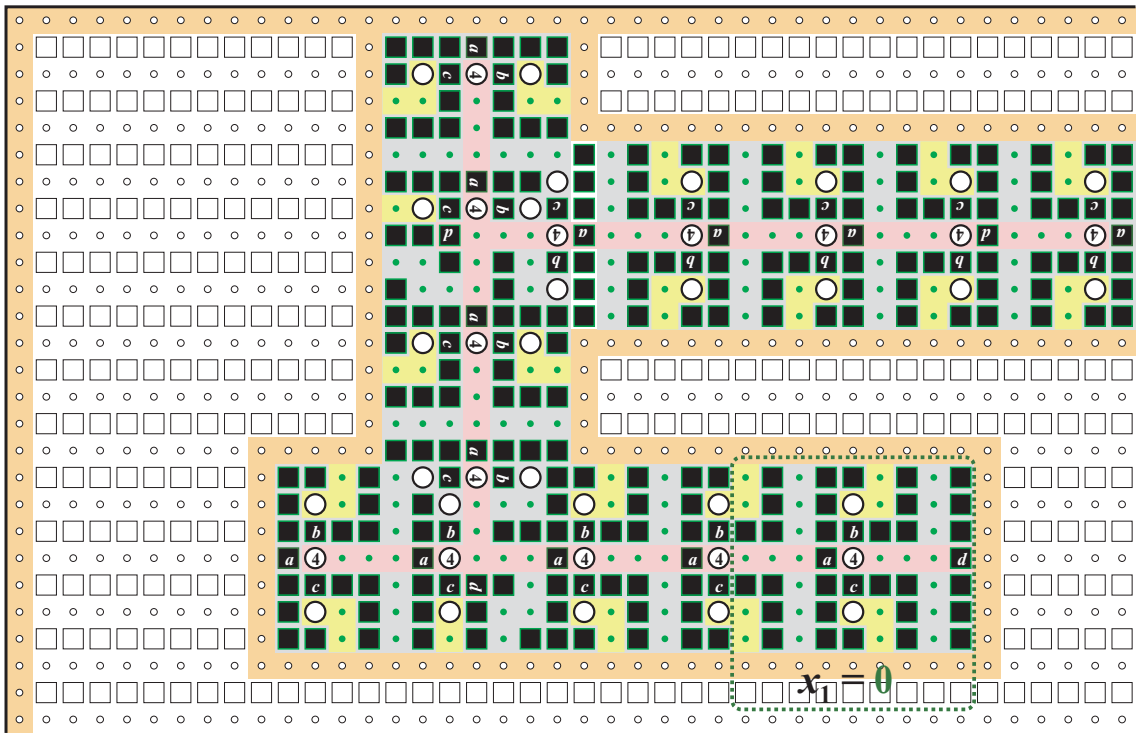


Figure 8: Enlarged illustration of variable gadget x_1 and its surroundings of Fig. 9. The border of white areas are filled with white cells (see ○ in the orange area). In each white area separated by the orange border, a black-cell row and a white-cell row appear alternately (see □□ ··· □ and ○○ ··· ○ in the white area).

NAR 3SAT is satisfiable if and only if the corresponding
Nurimisaki puzzle has a solution.

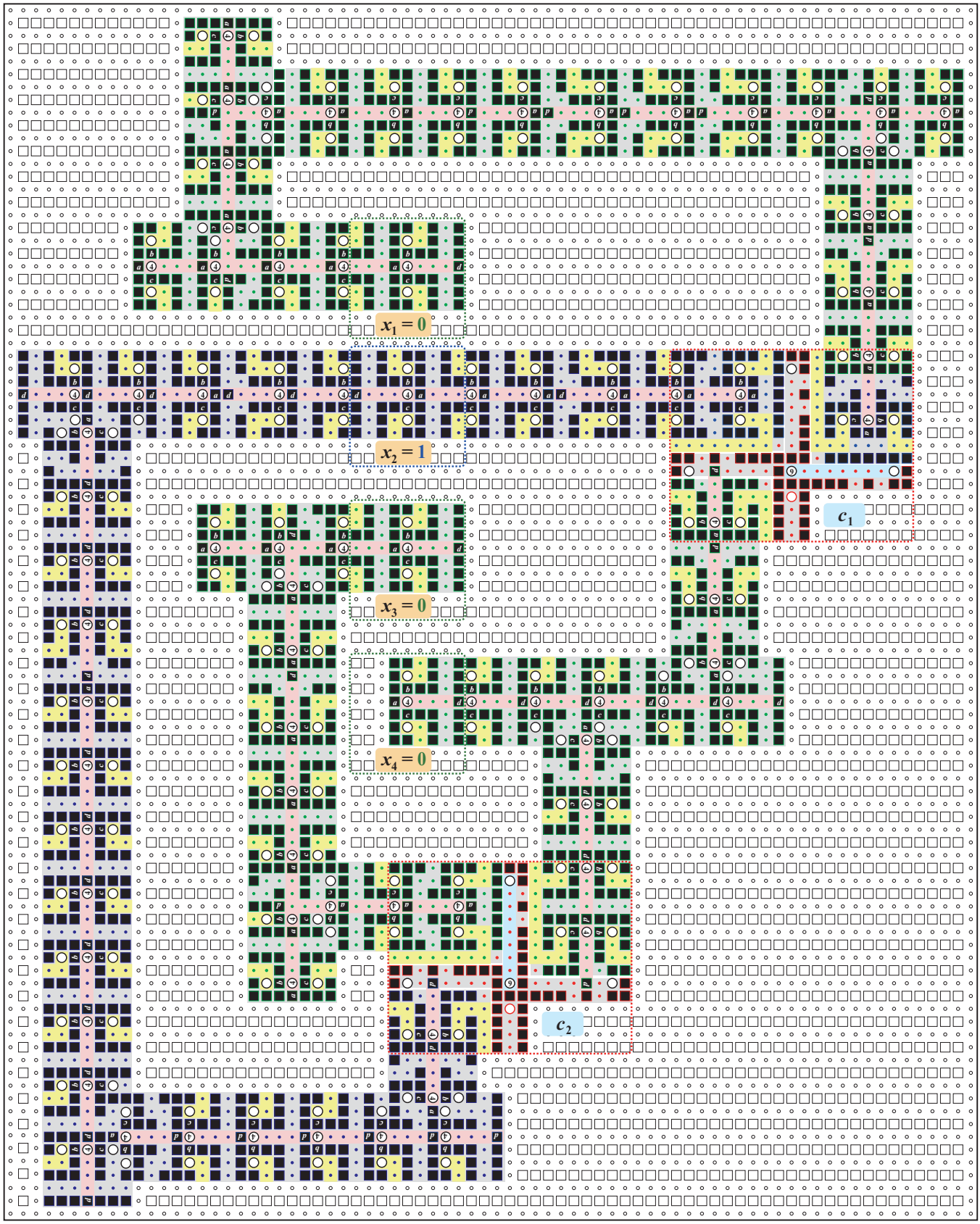


Figure 9: A Nurimisaki puzzle transformed from $C = \{c_1, c_2\}$, where $c_1 = \{\overline{x_1}, x_2, x_4\}$ and $c_2 = \{\overline{x_2}, \overline{x_3}, x_4\}$. From the solution of the puzzle, one can see that the assignment $(x_1, x_2, x_3, x_4) = (0, 1, 0, 0)$ satisfies all clauses of C.

References

[1] https://nikoli.co.jp/en/puzzles/nurimisaki.html

[2] https://nikoli.co.jp/en/puzzles/sashigane.html

[3] https://en.wikipedia.org/wiki/Steel_square

[4] A.B. Adcock, E.D. Demaine, M.L. Demaine, M.P. O'Brien, F. Reidl, F.S. Villaamil, and B.D. Sullivan. Zig-zag numberlink is NP-complete. *J. Inf. Process.*, 23(3):239–245, 2015.

[5] A. Allen and A. Williams, Sto-Stone is NP-Complete. *Proceedings of the 30th Canadian Conference on Computational Geometry*, pp. 28–34, 2018.

[6] D. Andersson, Hashiwokakero is NP-complete. *Inf. Process. Lett.*, 109:1145–1146, 2009.

[7] M. Biro and C. Schmidt. Computational complexity and bounds for Norinori and LITS. *Proceedings of the 33rd European Workshop on Computational Geometry*. pp. 29–32, 2017.

[8] M.R. Cerioli, L. Faria, T.O. Ferreira, C.A.J. Martinhon, F. Protti, and B. Reed. Partition into cliques for cubic graphs: planar case, complexity and approximation. *Discrete Appl. Math.*, 156(12):2270–2278, 2008.

[9] E.D. Demaine, Y. Okamoto, R. Uehara, and Y. Uno. Computational complexity and an integer programming model of Shakashaka. *IEICE T. Fund. Electr.*, E97A(6):1213–1219, 2014.

[10] M.R. Garey and D.S. Johnson. Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman, NY, 1979.

[11] R.A. Hearn and E.D. Demaine. Games, Puzzles, and Computation. A K Peters Ltd., 2009.

[12] A. Ishibashi, Y. Sato, and S. Iwata. NP-completeness of two pencil puzzles: Yajilin and Country Road. *Utilitas Mathematica*, 88:237–246, 2012.

[13] C. Iwamoto. Yosenabe is NP-complete. *J. Inf. Process.*, 22(1):40–43, 2014.

[14] C. Iwamoto and M. Haruishi. Computational complexity of Usowan puzzles. *IEICE Trans. Inf. Syst.* E101-A(9):1537–1540, 2018.

[15] C. Iwamoto, M. Haruishi, and T. Ibusuki. Herugolf and Makaro are NP-complete. *Proceedings of the 9th International Conference on Fun with Algorithms*, 100(23):23:1–23:11.

[16] C. Iwamoto and T. Ibusuki. Dosun-Fuwari is NP-complete. *J. Inf. Process.*, 26:358–361, 2018.

[17] C. Iwamoto and T. Ibusuki. Kurotto and Juosan are NP-complete. *Proceedings of the 21st Japan Conference on Discrete and Computational Geometry, Graphs, and Games*. pp.46–48, 2018.

[18] S. Kanehiro and Y. Takenaga. Satogaeri, Hebi and Suraromu are NP-complete. *Proceedings of the 3rd International Conference on Applied Computing and Information Technology*. pp.47–52, 2015.

[19] J. Kölker. Kurodoko is NP-complete. *J. Inf. Process.*, 20(3):694–706, 2012.

[20] D. Packer, S. White, and A. Williams. A paper on pencils: A pencil and paper puzzle – Pencils is NP-Complete. *Proceedings of the 30th Canadian Conference on Computational Geometry*, pp. 35–41, 2018.

[21] C.H. Papadimitriou. Computational Complexity. Addison-Wesley, 1994.

[22] Y. Takenaga, S. Aoyagi, S. Iwata, and T. Kasai. Shikaku and Ripple Effect are NP-complete. *Congressus Numerantium*, 216:119–127, 2013.

[23] A. Uejima and H. Suzuki. Fillmat is NP-complete and ASP-complete. *J. Inf. Process.*, 23(3):310–316, 2015.

[24] A. Uejima, H. Suzuki, and A. Okada. The complexity of generalized pipe link puzzles. *J. Inf. Process.*, 25:724–729, 2017.

Appendix

2.3 Transformation from an Instance of 3SAT to a Sashigane Puzzle

We present a polynomial-time transformation from an instance C of PLANAR 3SAT to a Sashigane puzzle such that C is satisfiable if and only if the puzzle has a solution.

Each variable $x_i \in \{x_1, x_2, \ldots, x_n\}$ is transformed into the variable gadget as illustrated in Fig. 10(a), which is composed of four circled numbers and ten arrows in a grey (9×5)-cell area. Note that the instances of 3SAT considered in this section have the restriction explained at the end of Sect. 2.1.

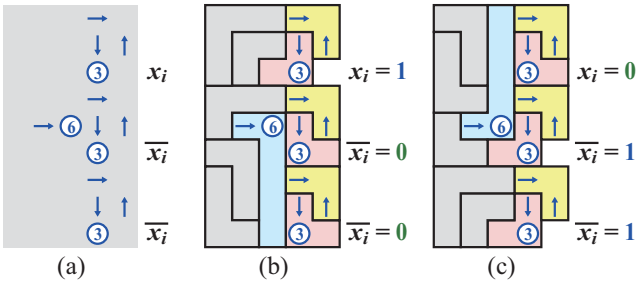


Figure 10: (a) Variable gadget of Sashigane transformed from x_i. (b) Assignment $x_i = 1$. (c) Assignment $x_i = 0$.

There are two possible solutions to circled number ⑥ (see blue blocks of Figs. 10(b) and 10(c)). In Fig. 10(b), the middle ③ and bottom ③ must be the knees of red "L-shaped blocks," and the top ③ can be the knee of the red "⅃-shaped block." This configuration corresponds to $x_i = 1$. (The empty cell at the (3,5) position of Fig. 10(b) will play a key role.) On the other hand, in Fig. 10(c), the top ③ must be the knee of the red L-shaped block, and the middle ③ and bottom ③ can be the knees of red ⅃-shaped blocks. This corresponds to $x_i = 0$.

Clause $c_j \in \{c_1, c_2, \ldots, c_m\}$ is transformed into the clause gadget as illustrated in Fig. 11, which is composed of four circled numbers and 11 arrows. Let $c_j = \{x_{i_1}, x_{i_2}, x_{i_3}\}$. If $x_{i_1} = x_{i_2} = x_{i_3} = 0$, there is no way to place an L-shaped block of size 6 (see Fig. 12(a)). However, if at least one of the three variables x_{i_1}, x_{i_2}, and x_{i_3} is 1, there is a solution (see Figs. 12(b)–(h)).

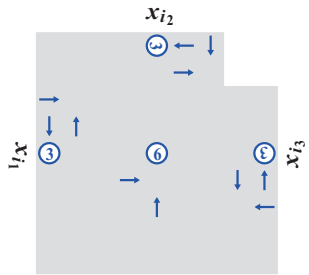


Figure 11: Clause gadget of Sashigane transformed from $c_j = \{x_{i_1}, x_{i_2}, x_{i_3}\}$.

If c_j consists of two literals, then the corresponding clause gadget is Fig. 13. The gadget of Fig. 13 is essential, since it is known that 3SAT WITH EXACTLY THREE OCCURRENCES PER VARIABLE is polynomial-time solvable if every clause c_j has three literals [21].

Fig. 14(a) is the connection gadget. If the leftmost ③ is the knee of an L-shaped block (see Fig. 14(b)), the remaining circled numbers ③ must become knees of L-shaped blocks. On the other hand, if the leftmost ③ is the knee of a ⅃-shaped block (see Fig. 14(c)), the remaining circled numbers ③ can be knees of ⅃-shaped blocks.

If you want the distance between two circled numbers ③ to be odd (see Fig. 15(a)), then two circled numbers ④ are used in a connection gadget. Four and six orange L-shaped blocks

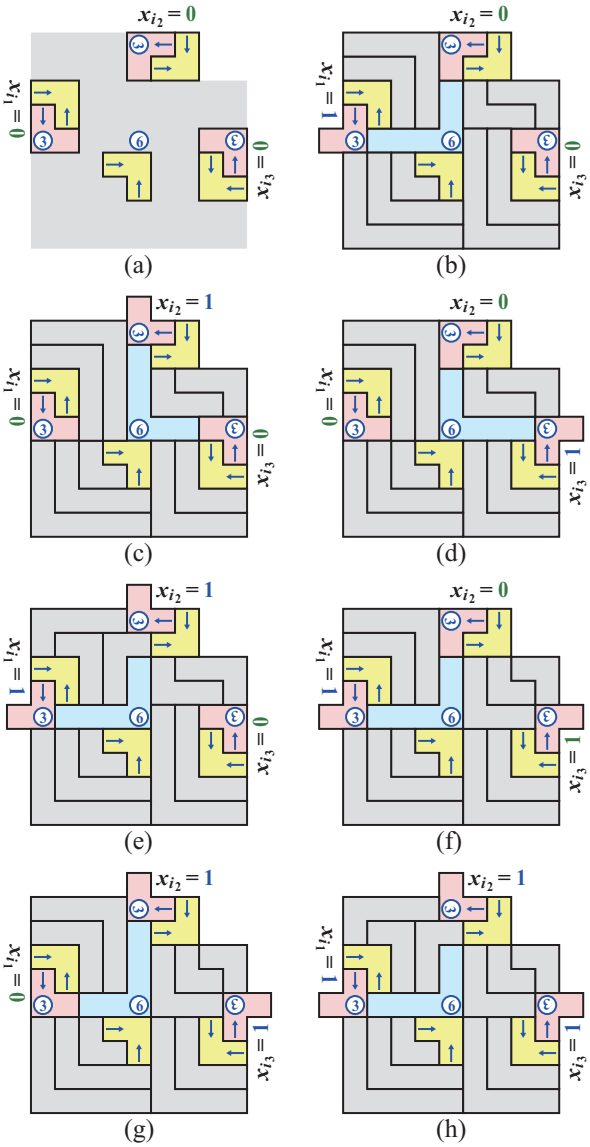


Figure 12: (a) If $x_{i_1} = x_{i_2} = x_{i_3} = 0$, there is no way to place an L-shaped block of size 6. (b)–(h) If at least one of the three variables x_{i_1}, x_{i_2}, and x_{i_3} is 1, there is a solution.

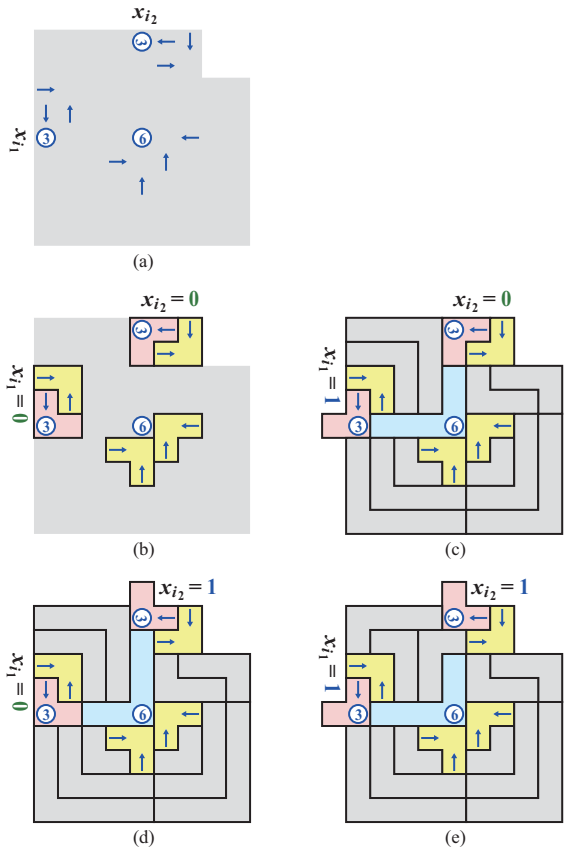


Figure 13: Clause gadget for $c_j = \{x_{i_1}, x_{i_2}\}$, which consists of only two literals.

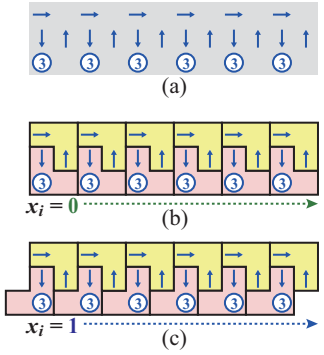


Figure 14: (a) Connection gadget. (b) Signal "$x_i = 0$" is transmitted to the right. (c) Signal "$x_i = 1$."

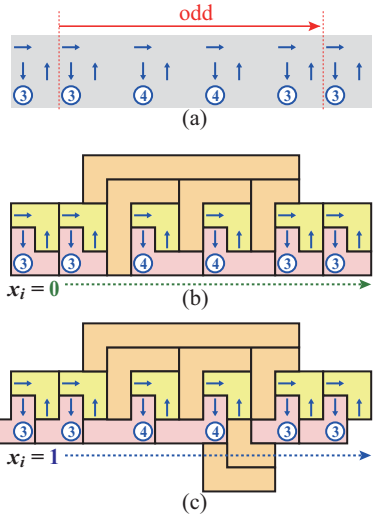


Figure 15: Connection gadget of odd length.

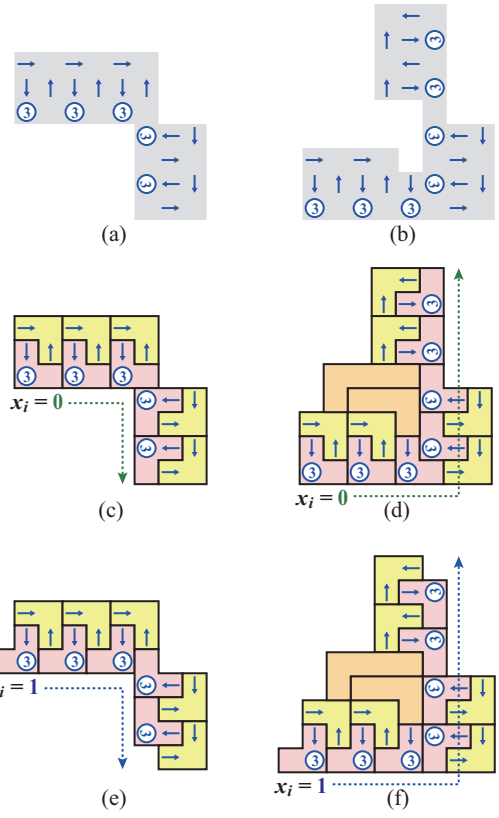


Figure 16: Right and left turn gadgets.

in Figs. 15(b) and 15(c), respectively, will be explained in the next paragraph. Figures 16(a,c,e) and 16(b,d,f) are the right and left turn gadgets, respectively.

Figure 17 is a Sashigane puzzle transformed from $C = \{c_1, c_2, c_3, c_4\}$, and $U = \{x_1, x_2, x_3, x_4\}$, where $c_1 = \{x_1, x_2, x_3\}$, $c_2 = \{\overline{x_1}, \overline{x_2}, \overline{x_4}\}$, $c_3 = \{\overline{x_1}, \overline{x_3}, x_4\}$, and $c_4 = \{\overline{x_2}, \overline{x_3}, \overline{x_4}\}$. In this figure, there are six large white areas separated by connection, variable, and clause gadgets. Those white areas can be filled up with $l_1 \times 2$ and $l_2 \times 3$ rectangles (see Fig. 18). Here, $l_1 \times 2$ and $l_2 \times 3$ rectangles are composed of two and three L-shaped blocks, where $l_1 \geq 3$ and $l_2 \geq 4$, respectively. Orange L-shaped blocks were used in Figs. 15 and 16 so that white areas of Fig. 18 can be filled up with rectangles of width 2 and 3. From this construction, the instance C of PLANAR 3SAT is satisfiable if and only if the corresponding Sashigane puzzle has a solution.

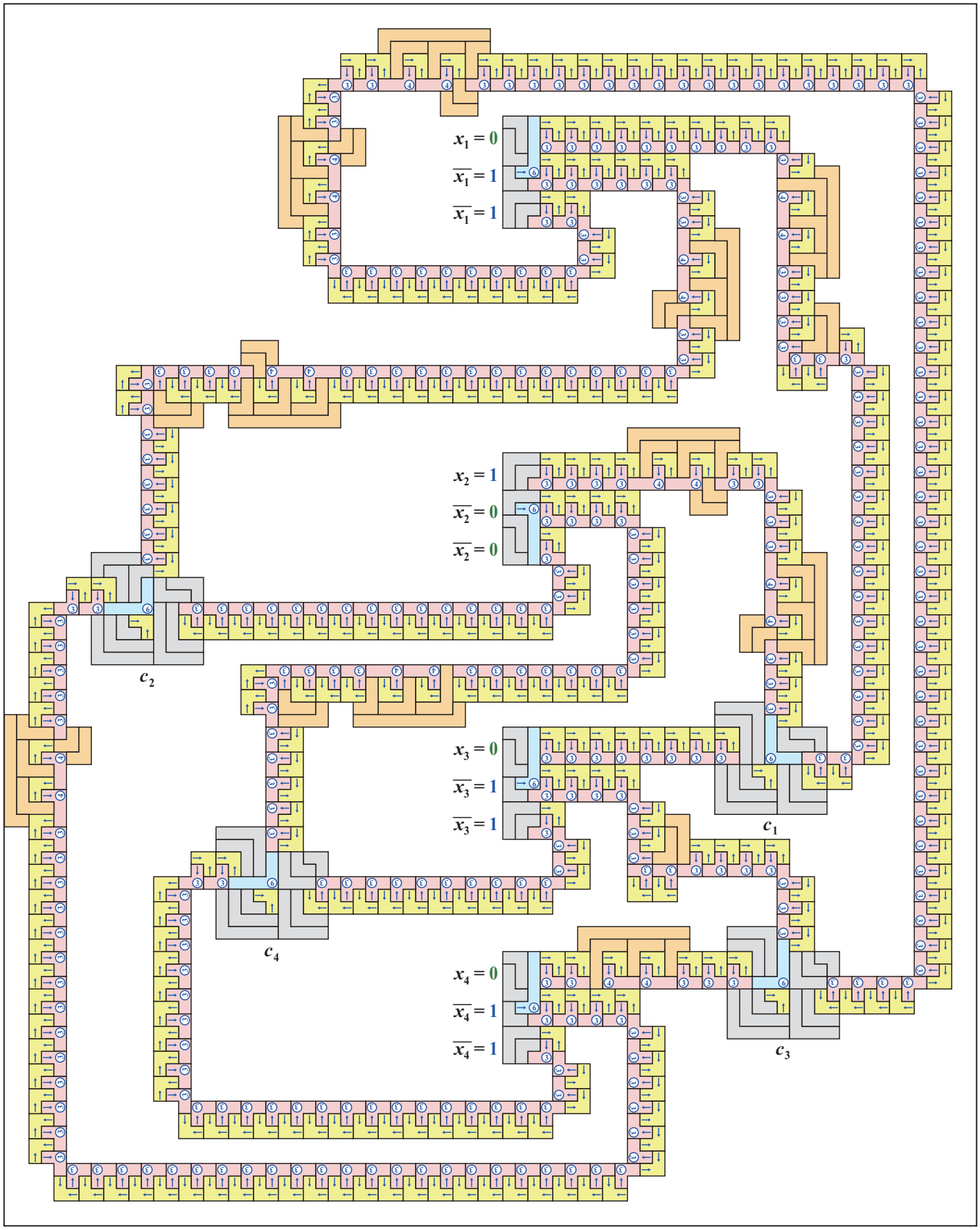


Figure 17: A Sashigane puzzle transformed from $C = \{c_1, c_2, c_3, c_4\}$, where $c_1 = \{x_1, x_2, x_3\}$, $c_2 = \{\overline{x_1}, \overline{x_2}, \overline{x_4}\}$, $c_3 = \{\overline{x_1}, \overline{x_3}, x_4\}$, $c_4 = \{\overline{x_2}, \overline{x_3}, \overline{x_4}\}$. From the solution of the puzzle, one can see that the assignment $(x_1, x_2, x_3, x_4) = (0, 1, 0, 0)$ satisfies all clauses of C.

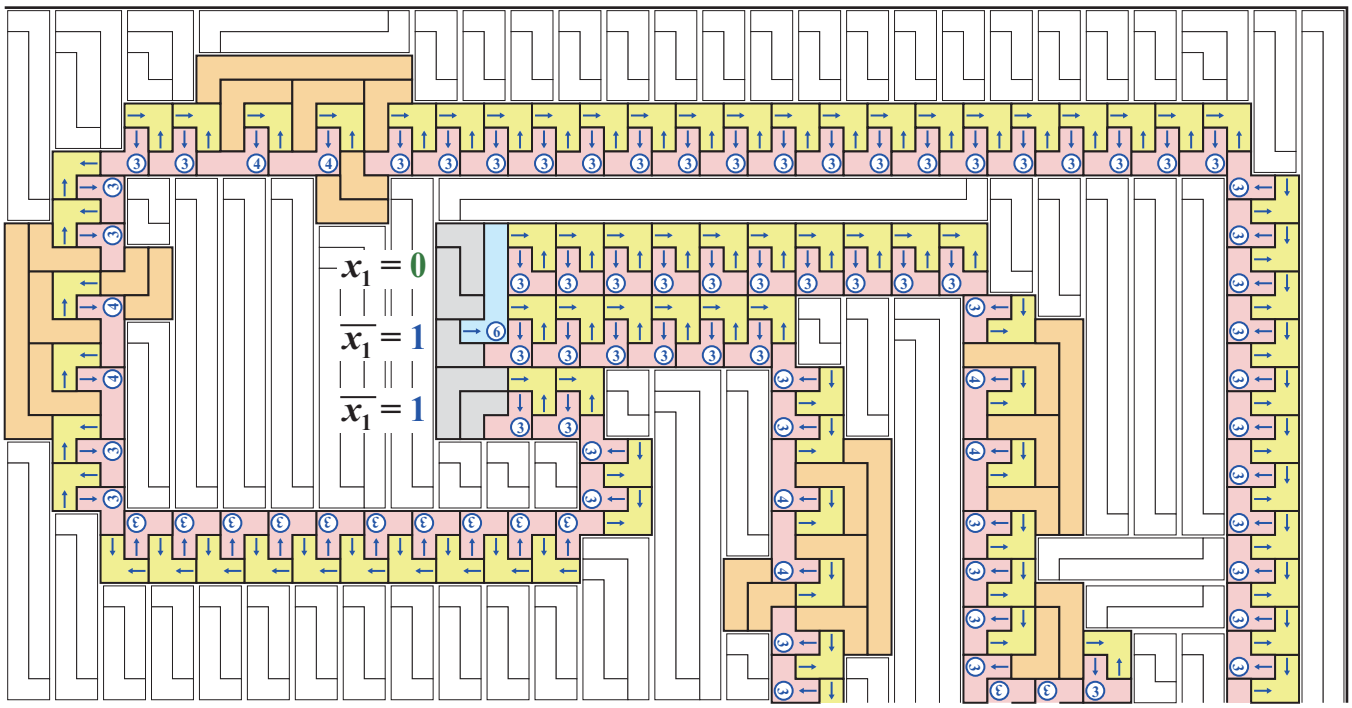


Figure 18: Enlarged illustration of variable gadget x_1 and its surroundings of Fig. 17. White areas of Fig. 17 are filled with $l_1 \times 2$ and $l_2 \times 3$ rectangles, where $l_1 \geq 3$ and $l_2 \geq 4$.

Constrained Orthogonal Segment Stabbing

Sayan Bandyapadhyay* Saeed Mehrabi†

Abstract

Let S and D each be a set of orthogonal line segments in the plane. A line segment $s \in S$ *stabs* a line segment $s' \in D$ if $s \cap s' \neq \emptyset$. It is known that the problem of stabbing the line segments in D with the minimum number of line segments of S is NP-hard [8]. However, no better than $O(\log |S \cup D|)$-approximation is known for the problem. In this paper, we introduce a constrained version of this problem in which every horizontal line segment of $S \cup D$ intersects a common vertical line. We study several versions of the problem, depending on which line segments are used for stabbing and which line segments must be stabbed. We obtain several NP-hardness and constant approximation results for these versions. Our finding implies, the problem remains NP-hard even under the extra assumption on input, but small constant approximation algorithms can be designed.

1 Introduction

Let S and D be two sets of orthogonal line segments in the plane. In this paper, we study the *orthogonal segment stabbing* problem, where the goal is to find a minimum-cardinality subset $S' \subseteq S$ such that every line segment in D is *stabbed* by at least one line segment in S'. A line segment $s \in D$ is stabbed by a line segment $s' \in S$ if and only if $s \cap s' \neq \emptyset$. Let H and V denote the set of input horizontal and vertical line segments, respectively, and $n = |H \sqcup V|$[1].

The orthogonal segment stabbing problem was studied by Katz et al. [8] who proved that the problem is NP-hard even in the case when S contains only vertical line segments and D contains only horizontal line segments, i.e, $S = V$ and $D = H$. Notice that the problem is trivial when $S, D \subseteq V$ or $S, D \subseteq H$. Moreover, an $O(\log n)$-approximation algorithm for the problem is straightforward (by reducing the problem to set cover). To the best of our knowledge, no other approximation or inapproximabilty results are known for the general version of the problem. However, two special versions of the problem have been studied very recently. Bandyapadhyay and Basu Roy [2] studied a related art gallery problem, and from their work it follows that one can get a PTAS for the orthogonal segment stabbing problem if $S \subseteq H$ or $S \subseteq V$. Mehrabi [11] considered the version where no two horizontal (resp. vertical) line segments intersect each other (i.e., the intersection graph of the line segments in $S \cup D$ is bipartite, but $S, D \subseteq H \sqcup V$), and obtained a constant approximation.

S \ D	H	V	$H \sqcup V$
H	Polytime (trivial)	NP-hard PTAS [2]	NP-hard PTAS [2]
V	NP-hard[1] PTAS [2]	Polytime (trivial)	PTAS [2]
$H \sqcup V$	5-approx. PTAS	NP-hard 2-approx.	NP-hard 7-approx. $(3+\epsilon)$-approx.

Table 1: A summary of our results for the (S, D)-stabbing problem. Each row corresponds to a set S and each column corresponds to a set D. ([1] when each horizontal line segment must be intersected by exactly one selected line segment)

In this paper, we introduce a constrained version of the orthogonal segment stabbing problem in which every horizontal line segment intersects a vertical line. More formally, for $S, D \subseteq H \sqcup V$, we define the (S, D)-*stabbing problem* to be the problem of stabbing all the line segments in D with the minimum number of line segments in S with the constraint that all the line segments in H intersect a vertical line L_v. In this work, we study different versions of the (S, D)-stabbing problem. Considering these versions, we obtain the following results (see Table 1 for a summary of our results).

- The (H, V)-stabbing, $(H, H \sqcup V)$-stabbing, $(H \sqcup V, V)$-stabbing and $(H \sqcup V, H \sqcup V)$-stabbing problems are all NP-hard.
- There exists an $O(n^6)$-time 5-approximation algorithm, and a local-search based PTAS for the $(H \sqcup V, H)$-stabbing problem.
- There exists an $O(n^5)$-time 2-approximation algorithm for the $(H \sqcup V, V)$-stabbing problem.

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

†School of Computer Science, Carleton University, Ottawa, Canada. saeed.mehrabi@carleton.ca.

[1]Throughout the paper we use $\sqcup$ to denote disjoint union.

- There exists an $O(n^6)$-time 7-approximation algorithm, and an $n^{O(\frac{1}{\epsilon^2})}$-time $(3+\epsilon)$-approximation algorithm (for any $\epsilon > 0$) for the $(H \sqcup V, H \sqcup V)$-stabbing problem.
- The (V, H)-stabbing problem is NP-hard when each horizontal line segment must be stabbed by exactly one line segment.

The (S, D)-stabbing problem is closely related to the minimum dominating set problem on axis-parallel polygonal chains in the plane. Among the previous work, the papers of Bandyapadhyay et al. [1] and Chakraborty et al. [3] are perhaps the most related to ours, as they consider the minimum dominating set problem on the intersection graph of "L-shapes" such that each L intersects a vertical line. The former showed that this problem is APX-hard, and an 8-approximation algorithm for the problem was given by the latter. For more related work, see [1, 3] and the references therein.

Organization. In Section 2, we define some notations that we use throughout the paper, and make a few observations that will be useful later. Then, we discuss our hardness results in Section 3, and the approximation results in Sections 4 and 5. The proofs of the results marked with $(*)$ are given in the appendix due to space constraints.

2 Preliminaries

For a point p, let x_p and y_p be its x- and y-coordinates, respectively. For a horizontal line segment h and a vertical line segment v that intersect each other, let $I(h, v)$ denote their intersection point. The y-coordinate (resp., x-coordinate) of a horizontal (resp., vertical) line segment is defined to be the y-coordinate (resp., x-coordinate) of its endpoints. For a horizontal line segment h with left endpoint p, call the line $y = y_p$, denoted by $L(h)$, the line-extension of h. Consider the line-extension $L(h)$. Let V' be any subset of vertical line segments that intersect $L(h)$. For $v, v' \in V'$, v and v' are called consecutive w.r.t. V' if there is no other line segment $v_1 \in V'$ such that $I(L(h), v_1)$ lies in the open interval $(I(L(h), v), I(L(h), v'))$. We say a set of line segments S_1 *hits* a set of line segments S_2 if for any line segment $s \in S_2$, there is a line segment $s' \in S_1$ such that s' stabs s; i.e, $s \cap s' \neq \emptyset$.

Our PTAS is based on the local search technique, which was introduced to computational geometry independently by Chan and Har-Peled [4], and Mustafa and Ray [12]. Consider a minimization problem in which the objective is to compute a feasible subset A of a ground set S whose cardinality is minimum over all such feasible subsets of S. Moreover, it is assumed that computing some initial feasible solution and determining whether a subset $A \subseteq S$ is a feasible solution can be done in polynomial time. The local search algorithm for a minimization problem is as follows. Fix some parameter k, and let A be some initial feasible solution. Now, if there exist $A' \subseteq A$, $M \subseteq S \setminus A$ such that $|A'| \leq k$, $|M| < |A'|$ and $(A \setminus A') \cup M$ is a feasible solution, then we set $A = (A \setminus A') \cup M$. The above is repeated until no such "local improvement" is possible and we return A as the final solution.

Let $\mathcal{B}$ and $\mathcal{R}$ be the solutions returned by the algorithm and an optimum solution, respectively. The following theorem establishes the connection between local search technique and obtaining a PTAS.

Theorem 1 ([4, 12]) *Consider the solutions $\mathcal{B}$ and $\mathcal{R}$ for a minimization problem, and suppose that there exists a planar bipartite graph $H = (\mathcal{B} \cup \mathcal{R}, E)$ that satisfies the local exchange property: for any subset $\mathcal{B}' \subseteq \mathcal{B}$, $(\mathcal{B} \setminus \mathcal{B}') \cup N_H(\mathcal{B}')$ is a feasible solution, where $N_H(\mathcal{B}')$ denotes the set of neighbours of $\mathcal{B}'$ in H. Then, the local search algorithm yields a PTAS for the problem.*

The following simple observation will be useful in the next sections.

Observation 1 $(*)$ *Suppose $D = D_1 \sqcup D_2$. If there is an α-approximation algorithm A_1 for (S, D_1)-stabbing that runs in $f(n)$ time and a β-approximation algorithm A_2 for (S, D_2)-stabbing that runs in $g(n)$ time, then there is an $(\alpha+\beta)$-approximation algorithm A for (S, D)-stabbing that runs in $f(n) + g(n)$ time.*

3 Hardness Results

In this section, we first prove that the $(H \sqcup V, H \sqcup V)$-stabbing problem is NP-hard, and then we will show how to use or modify the construction for proving the hardness of other variants claimed in Table 1 except (V, H)-stabbing. To prove the NP-hardness of (V, H)-stabbing, we will show a completely different reduction from the Positive Planar Cycle 1-In-3SAT problem [5]. Note that NP-hardness of any of these variants does not directly imply the NP-hardness of any other variant. In the following, we first give some definitions and then we describe the reduction.

Consider an instance I of the 3SAT problem with n variables and m clauses. The instance I is called *monotone* if each clause is monotone; that is, each clause consists of only positive literals (called *positive clauses*) or only negative literals (called *negative clauses*). The 3SAT problem is NP-hard even when restricted to monotone instances [7].

We can associate a bipartite *variable-clause graph* $G_I = (V, E)$ with I, where the vertices in one partition of G_I correspond to the variables in I and the vertices in the other partition of G_I correspond to the clauses of I. Each clause vertex is adjacent to the variable vertices it contains. The instance I is called *planar* if G_I

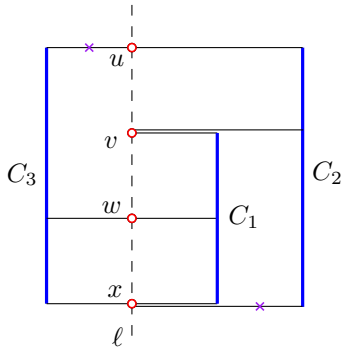


Figure 1: An instance of the planar 3SAT in the comb-shaped form of Knuth and Raghunathan [9]. Crosses on the edges indicate negations; e.g., see $C_2 = (u \vee v \vee \overline{x})$.

is planar; it is known that the Planar 3SAT problem is NP-hard [10]. Moreover, when I is an instance of the Planar 3SAT problem, Knuth and Raghunathan [9] showed that G_I can be drawn on a grid such that all variable vertices are on a vertical line ℓ and clause vertices are connected from left or right of that line in a comb-shaped form without any edge crossing. Moreover, in such a drawing of G_I, each variable vertex is drawn as a point in the plane and each clause vertex is drawn as a vertical line segment that is spanned from its lowest variable to its highest variable. More precisely, if a clause C contains three variables u, v and w such that v is between u and w on ℓ, then the clause vertex is drawn as a vertical line segment $s : x_s \times [y_u, y_w]$. The clause vertex is then connected to its three variable vertices using horizontal line segments. See Figure 1 for an example. In the Planar Monotone 3SAT problem, for any instance I, G_I can be drawn as described above. Moreover, all positive clauses (resp., negative clauses) lie to the left (resp., right) of the vertical line ℓ. de Berg and Khosravi [6] showed that Planar Monotone 3SAT is NP-hard.

Reduction. We reduce the Planar Monotone 3SAT problem to the $(H \sqcup V, H \sqcup V)$-stabbing problem. For the rest of this section, let I be an instance of the Planar Monotone 3SAT problem with n variables and m clauses. First, we consider a planar monotone drawing of the variable-clause graph G_I. As mentioned before, this is similar to the non-crossing comb-shaped form of Knuth and Raghunathan [9], where (i) variable vertices are all on the vertical line $\ell : x = 0$, (ii) the clause vertices are drawn as vertical line segments as described above, and (iii) all the positive clauses (resp., negative clauses) are to the left (resp., right) of ℓ. Next, we replace each variable vertex v with three horizontal line segments v_l, v_r and $s(v)$. First, $v_l : [x_s, 0] \times y_v$ (resp., $v_r : [0, x_{s'}] \times y_v$), where s (resp., s') is the vertical line segment corresponding to the left-most positive (resp., right-most negative) clause that contains v. Moreover,

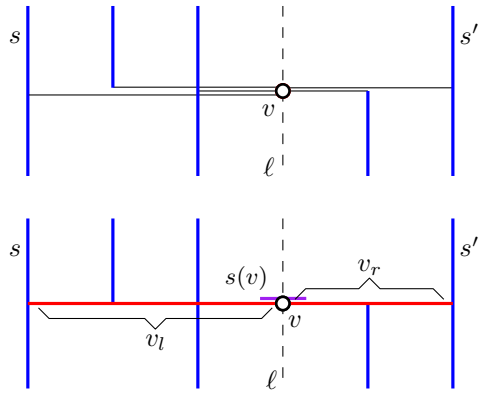


Figure 2: A variable v with all the clauses that contain a literal of v (top). The three horizontal line segments v_l, v_r and $s(v)$ corresponding to v (bottom).

$s(v) : [-\epsilon, \epsilon] \times y_v$ for some $\epsilon > 0$ such that $s(v)$ does not intersect any vertical line segment corresponding to a clause. See Figure 2 for an example. This forms the set of horizontal line segments. Finally, we take the line segments corresponding to clause vertices as the set of vertical line segments. This concludes our instance I' of the $(H \sqcup V, H \sqcup V)$-stabbing problem. Clearly, every horizontal line segment intersects the vertical line ℓ. Also, there are exactly $3n$ horizontal line segments and m vertical line segments in I', and the instance I' can be constructed in polynomial time.

Lemma 2 $(*)$ *The instance I is satisfiable if and only if the instance I' has a feasible solution of size n, where n is the number of variables in I.*

By Lemma 2, we have the following theorem.

Theorem 3 *The $(H \sqcup V, H \sqcup V)$-stabbing problem is* NP*-hard.*

We prove the other NP-hardness results claimed in Table 1 in the appendix.

4 $(H \sqcup V, H)$-Stabbing

In this section, we design approximation algorithms for $(H \sqcup V, H)$-stabbing.

4.1 A 5-approximation for $(H \sqcup V, H)$-stabbing

Our algorithm is based on a reduction to three simpler problems via LP. The approach is very similar to the one used in [3]. Recall that L_v is the common vertical line that the horizontal segments intersect. Let $V(l)$ and $V(r)$ be the vertical line segments that lie on the left and right of L_v, respectively. For simplicity, suppose $V = V(l) \sqcup V(r)$. Fix an optimum solution OPT. Let H_1 (resp. H_2) be the set of horizontal line segments that get hit by $V(l)$ (resp. $V(r)$) in OPT. Note

that the line segments in $H_3 = H \setminus (H_1 \cup H_2)$ get hit by horizontal line segments in OPT. If we knew the three sets H_1, H_2 and H_3, we could solve three subproblems $(V(l), H_1)$-stabbing, $(V(r), H_2 \setminus H_1)$-stabbing and (H, H_3)-stabbing, and return the union of these three solutions. It is not hard to see that the returned solution has size $|\mathsf{OPT}|$. We do not know those three sets, but one can guess (modulo constant approximation) these sets from a fractional LP solution to $(H \sqcup V, H)$-stabbing.

For an LP or ILP M, let $\mathsf{OPT}(M)$ be its optimum cost. First, we consider the ILP (denoted by ILP1) of $(H \sqcup V, H)$-stabbing. For each line segment $j \in H \sqcup V$, we take a standard 0-1 variable y_j denoting whether j is chosen in the solution or not. For $i \in H$, Let $V(l)^i$ (resp. $V(r)^i$ and H^i) be the line segments in $V(l)$ (resp. $V(r)$ and H) that stab i. Following is the LP relaxation of the ILP.

$$\begin{aligned} \text{minimize} \quad & \sum_{j \in H \sqcup V} y_j && \text{(LP1)} \\ \text{subject to} \quad & \sum_{j \in V(l)^i} y_j + \sum_{j \in V(r)^i} y_j \\ & + \sum_{j \in H^i} y_j \geq 1 && \forall i \in H \\ & y_j \in [0,1] && j \in H \sqcup V \end{aligned}$$

We first solve this LP to obtain a fractional optimal solution $y^* = \{y_j^* : j \in H \sqcup V\}$. Let $H_l = \{i \in H : \sum_{j \in V(l)^i} y_j^* \geq \frac{2}{5}\}$, $H_r = \{i \in H : \sum_{j \in V(r)^i} y_j^* \geq \frac{2}{5}\}$, and $H_h = \{i \in H : \sum_{j \in H^i} y_j^* \geq \frac{1}{5}\}$. Now, consider the following ILPs.

$$\begin{aligned} \text{minimize} \quad & \sum_{j \in V(l)} y_j && \text{(ILP2)} \\ \text{subject to} \quad & \sum_{j \in V(l)} y_j \geq 1 && \forall i \in H_l \\ & y_j \in \{0,1\} && j \in V(l) \end{aligned}$$

$$\begin{aligned} \text{minimize} \quad & \sum_{j \in V(r)} y_j && \text{(ILP3)} \\ \text{subject to} \quad & \sum_{j \in V(r)} y_j \geq 1 && \forall i \in H_r \\ & y_j \in \{0,1\} && j \in V(r) \end{aligned}$$

$$\begin{aligned} \text{minimize} \quad & \sum_{j \in H} y_j && \text{(ILP4)} \\ \text{subject to} \quad & \sum_{j \in H} y_j \geq 1 && \forall i \in H_h \\ & y_j \in \{0,1\} && j \in H \end{aligned}$$

Note that the problems corresponding to ILP2 and ILP3 are precisely the problem of stabbing horizontal rays using vertical line segments [8]. Also, ILP4 is corresponding to the stabbing problem of horizontal line segments using horizontal line segments where all line segments intersect L_v. By definition of H_l, H_r and H_h, there must be a feasible solution for each of these three ILPs. We use the algorithm in [8] to obtain optimum solutions S_1 and S_2 for ILP2 and ILP3, respectively. Also, an optimum solution S_3 of ILP4 can be obtained using a simple greedy selection scheme. Finally, we return the solution $S_1 \cup S_2 \cup S_3$.

It is not hard to see that $S_1 \cup S_2 \cup S_3$ hits H. Next, we argue that $|S_1 \cup S_2 \cup S_3| \leq 5 \cdot \mathsf{OPT}$. Let $LP2$ (resp. $LP3$ and $LP4$) be the LP relaxation of $ILP2$ (resp. $ILP3$ and $ILP4$). Consider the optimum solution $y^* = \{y_j^* : j \in H \sqcup V\}$ of LP1. First, we have the following simple observation.

Observation 2 $\mathsf{OPT}(LP2) \leq 2.5 \cdot \sum_{j \in V(l)} y_j^*$, $\mathsf{OPT}(LP3) \leq 2.5 \cdot \sum_{j \in V(r)} y_j^*$, *and* $\mathsf{OPT}(LP4) \leq 5 \cdot \sum_{j \in H} y_j^*$.

From the work of [3], it follows that the integrality gap of the stabbing problem of horizontal rays using vertical line segments is 2. In particular, their result can be summarized as follows.

Lemma 4 *[3]* $\mathsf{OPT}(ILP2) \leq 2 \cdot \mathsf{OPT}(LP2)$ *and* $\mathsf{OPT}(ILP3) \leq 2 \cdot \mathsf{OPT}(LP3)$.

We will use Lemma 4 to prove our approximation bound. Before that we prove the following lemma.

Lemma 5 $\mathsf{OPT}(ILP4) \leq \mathsf{OPT}(LP4)$.

Proof. Given a fractional solution S to LP4, we show how to round it to an integral solution S' such that the cost of S' is at most the cost of S. First, we partition the set H to a collection of subsets such that each subset contains line segments having the same y-coordinates. Then, we round the subsets independent of each other. Consider a particular subset T and a line segment $h \in H_h$ whose y-coordinate is the same as that of the line segments in T. If there is no such h, we set the y-values of all line segments in T to 0. Otherwise, as S is a feasible solution, $\sum_{j \in T} y_j \geq 1$. We pick any line segment h_1 from T arbitrarily, set its y-value to 1, and set the y-values of all remaining line segments in T to 0.

As h_1 hits all line segments in T, it is a feasible solution for this subset. Hence, the lemma follows. □

Lemma 6 $|S_1 \cup S_2 \cup S_3| \leq 5 \cdot \mathsf{OPT}$.

Proof. First note that $|S_1 \cup S_2 \cup S_3| = \mathsf{OPT}(ILP2) + \mathsf{OPT}(ILP3) + \mathsf{OPT}(ILP4)$. From Lemmas 4 and 5, it follows that $|S_1 \cup S_2 \cup S_3| \leq 2 \cdot \mathsf{OPT}(LP2) + 2 \cdot \mathsf{OPT}(LP3) + \mathsf{OPT}(LP4)$. Then, as the sets $V(l), V(r)$ and H are pairwise disjoint, by Observation 2, $|S_1 \cup S_2 \cup S_3| \leq 5 \cdot \mathsf{OPT}(LP1) \leq 5 \cdot \mathsf{OPT}$. □

To solve LP1 one can use the LP solver in [13] that runs in $O(n^5)$ time. The algorithm to compute S_1, S_2 and S_3 takes $O(n^6)$ time in total [8]. Hence, our approximation algorithm runs in $O(n^6)$ time.

Theorem 7 *There is a 5-approximation for $(H \sqcup V, H)$-stabbing that runs in $O(n^6)$ time.*

4.2 A PTAS for $(H \sqcup V, H)$-stabbing

We want to hit all the line segments of H using a minimum size subset of $H \sqcup V$. We design a local search based PTAS for this problem. Note that if a horizontal line segment h_1 in a solution hits a horizontal line segment, then they must have the same y-coordinates, and h_1 hits all the line segments that have the same y-coordinates as h_1. Thus, for a subset S of horizontal line segments with the same y-coordinates, we can identify one of them (say h) and assume that if the line segments in S are being hit by a horizontal line segment, then that line segment is h. We preprocess the set H to compute a subset H' so that for each subset S of horizontal line segments with the same y-coordinates, $|H' \cap S| = 1$. Thus, we want to find a minimum size subset of $H' \sqcup V$ that hit all the line segments in H.

We use a standard local search algorithm as the one in [1], which runs in $n^{O(\frac{1}{\epsilon^2})}$ time. Let $\mathcal{B}$ (blue) and $\mathcal{R}$ (red) be the local and global optimum solutions, respectively and w.l.o.g., assume $\mathcal{B} \cap \mathcal{R} = \emptyset$. Then, by Theorem 1, to say that the local search algorithm is a PTAS it is sufficient to prove the existence of a bipartite local exchange graph $G = (\mathcal{B}, \mathcal{R}, E)$ that is also planar. Note that $G = (\mathcal{B}, \mathcal{R}, E)$ is a local exchange graph if for each $h \in H$, there exists $s_1 \in \mathcal{B}, s_2 \in \mathcal{R}$ such that $s_1 \cap s_2 \cap h \neq \emptyset$ and $(s_1, s_2) \in E$. We will construct a plane graph $G = (\mathcal{B}, \mathcal{R}, E)$ in the following, which satisfies the local exchange property.

For each line segment $h \in H' \cap (\mathcal{B} \cup \mathcal{R})$, we select the intersection point $h \cap L_v$ to draw the vertex for h. For each line segment $v \in V \cap (\mathcal{B} \cup \mathcal{R})$, we select v itself to draw the vertex for v. Later we will contract each such v to a single point. For each red (resp. blue) $h \in H' \cap \mathcal{R}$ (resp. $h \in H' \cap \mathcal{B}$), let v_1 and v_2 be the first line segments of color blue (resp. red) on the left and right of L_v, respectively that intersect the line-extension $L(h)$. Note that v_1 and v_2 might not exist. We add two edges (h, v_1) and (h, v_2) (or horizontal line segments), between $h \cap L_v$ and $I(L(h), v_1)$ and between $h \cap L_v$ and $I(L(h), v_2)$, respectively. For each $h \in H' \setminus (\mathcal{B} \cup \mathcal{R})$, let V' be the line segments in $\mathcal{R} \cup \mathcal{B}$ that intersect $L(h)$. We add an edge between each consecutive (w.r.t. V') pair of line segments (v, v') such that $v \in \mathcal{R}$ and $v' \in \mathcal{B}$. In particular, we draw a horizontal line segment between $I(L(h), v)$ and $I(L(h), v')$.

Note that each edge of G is a horizontal line segment and any pair of those can be drawn in a non-overlapping manner. Also, an input vertical segment can intersect any such edge only at one of its vertices. Thus, the planarity of the graph follows. Lastly, we contract each vertical line segment v to a point, which does not violate the planarity. Now, consider any line segment $h_1 \in H$. Suppose h_1 is hit by a horizontal line segment $h \in \mathcal{R} \cup \mathcal{B}$. W.l.o.g., let $h \in \mathcal{R}$. Let v_1 and v_2 be the first line segments of color blue on the left and right of L_v, respectively that intersect $L(h)$. Then, either v_1 or v_2 must hit h_1, as h_1 intersects the line L_v. As we add the edges (h, v_1) and (h, v_2), the local exchange property holds for h_1. Now, suppose h_1 does not get hit by a horizontal line segment in $\mathcal{R} \cup \mathcal{B}$. Let V' be the line segments in $\mathcal{R} \cup \mathcal{B}$ that intersect $L(h_1)$. Then, there must be two consecutive (w.r.t. V') vertical line segments $v \in \mathcal{R}$ and $v' \in \mathcal{B}$ both of which hit h_1. As we add the edge (v, v'), the local exchange property holds for h_1 in this case as well. It follows that the local search algorithm is a PTAS.

Theorem 8 *There is a $(1+\epsilon)$-approximation for $(H \sqcup V, H)$-stabbing that runs in $n^{O(\frac{1}{\epsilon^2})}$ time for any $\epsilon > 0$.*

Remark. One can show that $(H \sqcup V, H)$-stabbing is a special case of (V, H)-stabbing. Construct an instance I' of (V, H)-stabbing from any given instance I of $(H \sqcup V, H)$-stabbing by taking the vertical and horizontal segments in I along with some special vertical segments. For each maximal cluster of horizontal segments having same y-coordinates, add one special vertical segment such that the only segments it intersects are the segments in the cluster. Then, given a solution for one instance, a solution for the other of the same size can be computed in polynomial time. As (V, H)-stabbing admits a PTAS [2], one can also obtain a PTAS for $(H \sqcup V, H)$-stabbing using this alternative approach.

5 $(H \sqcup V, V)$-Stabbing and $(H \sqcup V, H \sqcup V)$-Stabbing

In this section, we obtain approximations for $(H \sqcup V, V)$-stabbing and $(H \sqcup V, H \sqcup V)$-stabbing.

5.1 A 2-approximation for $(H \sqcup V, V)$-stabbing

First, we assume that the line segments of V are lying only on the right side of L_v. We design a polynomial

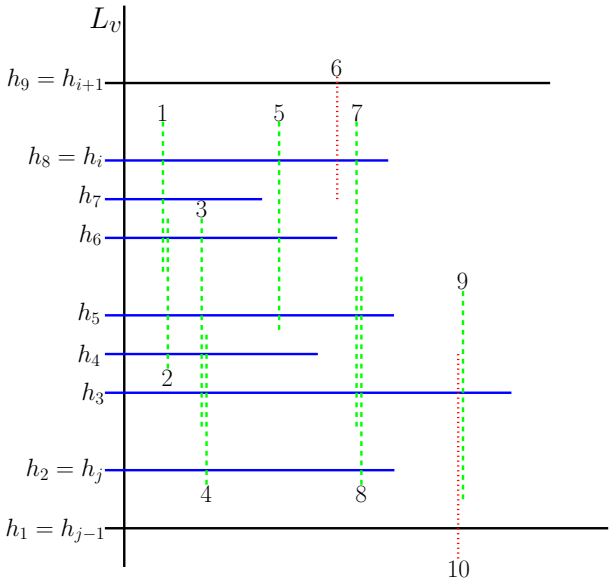


Figure 3: An example demonstrating a subproblem (i, j, k, k') with $i = 8, j = 2, k = 10$ and $k' = 9$. The line segments in $V(i, j, k')$ are shown using dashed (green) line segments. The dotted (red) line segments either intersect h_{i+1} or has index more than 9, and thus are not in $V(i, j, k')$.

time algorithm for this case, and later we will show how to obtain a 2-approximation for the general case of $(H \sqcup V, V)$-stabbing by using this algorithm as a subroutine. Now, as the line segments of V are lying only on the right side of L_v, w.l.o.g., we can assume that the y-coordinates of the line segments in H are distinct. Also, for simplicity, we assume that the endpoints of all the line segments are distinct.

Our algorithm is based on dynamic programming. Let $h_1, h_2, \ldots, h_t$ be the line segments in H in increasing order of their y-coordinates. Also, let $v_1, v_2, \ldots, v_m$ be the line segments in V in non-decreasing order of their x-coordinates. If two vertical line segments have the same x-coordinates, order them in decreasing order of y-coordinates of their bottom endpoints. Now, we describe the subproblems we consider. Each subproblem (i, j, k, k') is defined by four indexes i, j, k and k', where $1 \leq j \leq i \leq t$, and $1 \leq k, k' \leq m$. Let $H(i, j) = \{h_i, h_{i-1}, \ldots, h_j\}$. Also, let $V_l = \{v_1, v_2, \ldots, v_l\}$, and $V(i, j, l)$ be the line segments of V_l each of which intersects at least one line segment in $H(i, j)$ and does not intersect h_{i+1} or h_{j-1} (if exists). In subproblem (i, j, k, k'), we would like to hit $V(i, j, k')$ using a minimum size subset of $H(i, j) \sqcup V_k$ (see Figure 3). We define $f(i, j, k, k')$ to be the size of an optimum solution of the subproblem (i, j, k, k'). Note that we are interested in computing $f(t, 1, m, m)$.

We use the following recursive structure to compute $f(i, j, k, k')$. Let v be the line segment in $V(i, j, k')$ having the maximum index. Now, there can be two cases. v gets hit by a horizontal line segment in optimum solution or v gets hit by only vertical line segments in the optimum solution. For the first case, we guess a line segment $h_{i'}$ that hits v. Let k'_1 be the maximum index of the vertical line segments in $V(i, j, k')$ that intersect at least one line segment in $H(i, i'+1)$ and do not intersect h_{i+1} or $h_{i'}$. Similarly, let k'_2 be the maximum index of the vertical line segments in $V(i, j, k')$ that intersect at least one line segment in $H(i'-1, j)$ and do not intersect $h_{i'}$ or h_{j-1}. Then, the two new subproblems that we need to solve are $(i, i'+1, k, k'_1)$ and $(i'-1, j, k, k'_2)$. For the second case, we guess the maximum index l of the line segments in V_k that hit v in the optimum solution. Let k'_3 be the maximum index of the vertical line segments in $V(i, j, k')$ that does not intersect v_l. From the definition of v and the fact that x-coordinates of v and v_l are same, it follows that there is no line segment in $V(i, j, k')$ with index larger than k'_3 that does not intersect v_l. Thus, the new subproblem that we need to solve is $(i, j, l-1, k'_3)$. Note that the number of guesses for $h_{i'}$ and v_l is $O(n)$. Thus, to solve (i, j, k, k') the total number of subproblems that we need to solve is $O(n)$. Using a dynamic programming based scheme these subproblems can be evaluated easily. Since the number of distinct subproblems (i, j, k, k') is $O(n^4)$, all the subproblems can be solved in $O(n^5)$ time.

$(H \sqcup V, V)$-stabbing. Let $V(l)$ and $V(r)$ be the vertical line segments that lie on the left and right of L_v, respectively. Set $S = H \sqcup V$, $D_1 = V(l)$ and $D_2 = V(r)$. Then, note that the problem (S, D_1)-stabbing is same as $(H \sqcup V(l), D_1)$-stabbing. Similarly, (S, D_2)-stabbing is same as $(H \sqcup V(r), D_2)$-stabbing. Thus, we can use the above algorithm to solve these two problems, and hence there exists exact algorithms for (S, D_1)-stabbing and (S, D_2)-stabbing. From Observation 1, we obtain the following theorem.

Theorem 9 *There is a 2-approximation for $(H \sqcup V, V)$-stabbing that runs in $O(n^5)$ time.*

$(H \sqcup V, H \sqcup V)$-stabbing. Set $S = H \sqcup V$, $D_1 = H$ and $D_2 = V$. Then, we know that there are two algorithms for the (S, D_1)-stabbing problem: an $O(n^6)$-time 7-approximation algorithm and a PTAS. Moreover, we have an $O(n^5)$-time 2-approximation algorithm for the (S, D_2)-stabbing problem. Therefore, by Observation 1, we have the following theorem.

Theorem 10 *There is a 7-approximation for $(H \sqcup V, H \sqcup V)$-stabbing that runs in $O(n^6)$ time and a $(3+\epsilon)$-approximation that runs in $n^{O(\frac{1}{\epsilon^2})}$ time for any $\epsilon > 0$.*

Acknowledgements. We are grateful to the reviewers for their helpful comments.

References

[1] S. Bandyapadhyay, A. Maheshwari, S. Mehrabi, and S. Suri. Approximating dominating set on intersection graphs of rectangles and l-frames. In *43rd International Symposium on Mathematical Foundations of Computer Science (MFCS 2018), Liverpool, UK*, pages 37:1–37:15, 2018.

[2] S. Bandyapadhyay and A. B. Roy. Effectiveness of local search for art gallery problems. In *15th International Symposium on Algorithms and Data Structures (WADS 2017), St. John's, NL, Canada*, pages 49–60, 2017.

[3] D. Chakraborty, S. Das, and J. Mukherjee. Approximating minimum dominating set on string graphs. *CoRR*, abs/1809.09990, 2018.

[4] T. M. Chan and S. Har-Peled. Approximation algorithms for maximum independent set of pseudo-disks. *Discrete & Computational Geometry*, 48(2):373–392, 2012.

[5] S. Chaplick, K. Fleszar, F. Lipp, A. Ravsky, O. Verbitsky, and A. Wolff. The complexity of drawing graphs on few lines and few planes. In *15th International Symposium on Algorithms and Data Structures (WADS 2017)*, pages 265–276, 2017.

[6] M. de Berg and A. Khosravi. Optimal binary space partitions for segments in the plane. *Int. J. Comput. Geometry Appl.*, 22(3):187–206, 2012.

[7] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[8] M. J. Katz, J. S. B. Mitchell, and Y. Nir. Orthogonal segment stabbing. *Comput. Geom.*, 30(2):197–205, 2005.

[9] D. E. Knuth and A. Raghunathan. The problem of compatible representatives. *SIAM J. Discrete Math.*, 5(3):422–427, 1992.

[10] D. Lichtenstein. Planar formulae and their uses. *SIAM J. Comput.*, 11(2):329–343, 1982.

[11] S. Mehrabi. Approximating domination on intersection graphs of paths on a grid. In *15th International Workshop on Approximation and Online Algorithms (WAOA 2017), Vienna, Austria*, pages 76–89, 2017.

[12] N. H. Mustafa and S. Ray. Improved results on geometric hitting set problems. *Discrete & Computational Geometry*, 44(4):883–895, 2010.

[13] É. Tardos. A strongly polynomial algorithm to solve combinatorial linear programs. *Operations Research*, 34(2):250–256, 1986.

Appendix

Proof of Observation 1

Proof. We show the existence of such an algorithm A for the (S, D)-stabbing problem. Given an instance of the (S, D)-stabbing problem, consider an algorithm A that uses A_1 to compute a solution for (S, D_1)-stabbing and uses A_2 to compute a solution for (S, D_2)-stabbing. Then, it returns the union of these two solutions as its solution. Clearly, the running time of A is then $f(n) + g(n)$ as claimed. Moreover, since the size of an optimum solution for each of the (S, D_1)-stabbing and (S, D_2)-stabbing problems is at most the size of an optimum solution for (S, D)-stabbing and $D_1 \cap D_2 = \emptyset$, we get the desired approximation factor. □

Proof of Lemma 2

Proof. First, suppose that there exists an assignment that satisfies all clauses in I. We construct a feasible solution S for I' as follows. For each variable v, if v is set to true (resp., false), then we add the line segment v_l (resp., v_r) to S. Clearly, $|S| = n$ and every horizontal line segment is stabbed by some line segment in S. Now, take any vertical line segment $s \in I'$. If s corresponds to a positive (resp., negative) clause C, then at least one of the variables v in C is set to true (resp., false) and so we have added v_l (resp., v_r) to S. So, every line segment in I' is stabbed by some line segment in S.

Now, suppose that there exists a feasible solution S for I' such that $|S| = n$. We assume w.l.o.g. that $s(v) \notin S$ for all variables v of I; this is because we can always replace such a line segment $s(v)$ with v_l, and still have a feasible solution for I' with the same size n. Since $|S| = n$, we must have $v_l \in S$ or $v_r \in S$ for all variables v of I, because if there is a variable v for which $v_l \notin S$ and $v_r \notin S$, then no line segment in S can dominate $s(v)$ — a contradiction to feasibility of S. This implies that exactly one of v_l and v_r is in S for all variables v of I, and so no vertical line segment can be in S, as $|S| = n$. We now obtain a true assignment for I as follows. For each variable v, we set v to true (resp., false) if $v_l \in S$ (resp., $v_r \in S$). To see why this is a true assignment for I, take any clause C and let $s \in S$ be a line segment that dominates the vertical line segment corresponding to C. If C is a positive (resp., negative) clause, then we have set the variable corresponding to s to true (resp., false) and so C is satisfied. □

Other NP-Hardness Results

Consider the construction in Section 3. Observe that the set S in the proof of Lemma 2 consists of only horizontal line segments; that is, the problem is NP-hard even if we are restricted to selecting a minimum number of horizontal line segments only to stab all line segments.

Theorem 11 *The $(H, H \sqcup V)$-stabbing problem is* NP*-hard.*

The construction can be modified to show the hardness of the (H, V)-stabbing and $(H \sqcup V, V)$-stabbing problems.

To this end, for every variable vertex v, we remove the horizontal line segment $s(v)$ and instead add a small vertical line segment $s'(v)$ such that it intersects v_l and v_r only. (To this end, we can e.g. extend v_r slightly to the left of ℓ and place $s'(v)$ to the left of ℓ and very close to it.) Considering the resulting set of line segments to be an instance of the (H, V)-stabbing problem, one can prove a result similar to Lemma 2: the first direction is true as every vertical line segment is stabbed by some horizontal line segment. Moreover, we need to stab the small vertical line segment $s'(v)$, for every variable vertex v, which leads to a truth assignment for the instance I.

We notice that this also shows the NP-hardness of the $(H \sqcup V, V)$-stabbing problem. This is because the vertical line segments are pairwise disjoint; hence, given a feasible solution for the $(H \sqcup V, V)$-stabbing problem, we can obtain a feasible solution of the same size by replacing each vertical line segment by a horizontal one. Hence, we have the following theorem.

Theorem 12 *The (H, V)-stabbing problem is NP-hard. Moreover, the $(H \sqcup V, V)$-stabbing problem is also NP-hard.*

Now, we show that the (V, H)-stabbing problem is NP-hard when each horizontal line segment must be stabbed by exactly one line segment. We show a reduction from the following variant of the 3SAT problem, which was shown to be NP-hard by Chaplick et al. [5]. In an instance I of the Positive Planar Cycle 1-In-3SAT problem, we have n variables and m clauses together with an embedding of $G_I + C$, where C is a cycle through all clause vertices. Moreover, each clause contains exactly three variables and all literals are positive. The problem is to decide whether I is *satisfiable*; here, being satisfiable means whether there exists an assignment of the variables such that exactly one variable in each clause is true. Notice that the problem remains NP-hard if we are also given an integer $1 \leq k \leq n$ and the problem is to decide if there exists an assignment of the variables such that at most k variables are set to true and exactly one variable in each clause is true.

Reduction. Given an instance of the Positive Planar Cycle 1-In-3SAT problem, consider the embedding of $G_I + C$. We first transform the embedding into another one in which the cycle C is a half-circle and all the clauses are positioned on the vertical diameter of this half-circle. Let ℓ be the vertical line through the diameter. Consider the clause vertices on ℓ from top to bottom. Next, we transform the embedding into a "comb-shaped" form, where clause vertices are drawn as horizontal line segments and variable vertices are drawn as vertical line segments. We do this transformation in such a way that the vertical line segment corresponding to a vertex v passes through v and spans from the top-most clause to the bottom-most clause that contain it. Moreover, the horizontal line segment corresponding to a clause C passes through C and spans from the left (resp., right) to the left-most (resp., right-most) variable vertex that it contains. This ensures that the horizontal line segment corresponding to a clause C intersects exactly those vertical line segments that correspond to variables that C contain. Notice that this is similar to the "comb-shaped" form of Knuth and Raghunathan [9] except the position of variable and clause vertices are swapped. The set of all vertical and horizontal line segments forms our instance I' of the (V, H)-stabbing problem. Observe that I' consists of n vertical and m horizontal line segments, and that it can be constructed in polynomial time.

Lemma 13 *The instance I is satisfiable with k variables set to true if and only if there exists a feasible solution of size k for I'.*

Proof. First, suppose that I is satisfiable with k variables set to true. Then, we select the k vertical line segments corresponding to these variables as the solution for I'. Each horizontal line segment s of I' is stabbed by exactly one selected line segment: the one that satisfies the clause corresponding to s. Now, suppose that I' has a feasible solution S of size k. Notice that each line segment of S is vertical and so we set to true exactly those variables that correspond to the line segments in S. Clearly, k variables are set to true. Moreover, since S is a feasible solution for I', each horizontal line segment is stabbed by exactly one vertical line segment; that is, exactly one variable per clause is set to true. □

By Lemma 13, we have the following theorem.

Theorem 14 *The (V, H)-stabbing problem is NP-hard when every horizontal line segment must be stabbed by exactly one line segment.*

Watchtower for k-crossing Visibility

Yeganeh Bahoo* Prosenjit Bose§ Stephane Durocher**

Abstract

Given a 1.5D terrain T, consisting of an x-monotone polygonal chain with n vertices in the plane, and a positive integer k, we propose an algorithm to place one point, called a *watchtower*, whose vertical height above T is minimized, such that every point x on T is k-crossing visible from the watchtower w. That is, the line segment from w to any point x on T crosses T at most k times. Our algorithm runs in $O((n^2 + h) \log n)$ time, where h denotes the number of vertices on the boundary of the k-kernel of T. For arbitrary k, $h \in O(n^4)$, and for $k = 2$, $h \in O(n^2)$. We present an $O(n^3)$-time algorithm for the discrete version of the problem, in which the watchtower is restricted to being positioned over vertices of T.

1 Introduction

A *terrain* T in $\mathbb{R}^2$ is an x-monotone polygonal chain consisting of a sequence of vertices $v_0, v_1, \ldots, v_{n-1}$, each of which is a point in $\mathbb{R}^2$, such that v_i is to the left of v_j for all $i < j$ and $v_i v_{i+1}$ is an edge for $i \in \{0, \ldots, n-2\}$. See Figure 1. As defined by Chang et al. [5], "two paths [polygonal chains], P and Q, are *weakly disjoint* if, for all sufficiently small $\epsilon > 0$, there are disjoint paths $\tilde{P}$ and $\tilde{Q}$ such that $d_{\mathcal{F}}(P, \tilde{P}) < \epsilon$ and $d_{\mathcal{F}}(Q, \tilde{Q}) < \epsilon$", where $d_{\mathcal{F}}(A, B)$ denotes the Fréchet distance between A and B. As also defined by Chang et al. [5], "two paths [polygonal chains] *cross* if they are not weakly disjoint." We say two polygonal chains P and Q *cross k times*, if there exist partitions $P_1, \ldots, P_k$ of P and $Q_1, \ldots, Q_k$ of Q such that P_i and Q_i cross, for all $i \in \{1, \ldots, k\}$. Two points p and q are *k-crossing visible* if and only if the line segment pq crosses T at most k times. When $k = 0$, k-crossing visibility corresponds to the traditional definition of visibility.

A *watchtower* w is a point on or above T. Given a terrain T and a positive integer k, the goal in the *1-watchtower* problem is to place a watchtower w with minimum height on or above T (length of the vertical line segment from w to T) such that the entire terrain T is k-crossing visible from w. This definition can be generalized to the *m-watchtower* problem where the goal is to assign positions to a set $W = \{w_1, \ldots, w_m\}$ of m watchtowers, such that each w_i is a point on or above T, and for each point p on T, there exists a watchtower $w \in W$ such that p is k-crossing visible from w.

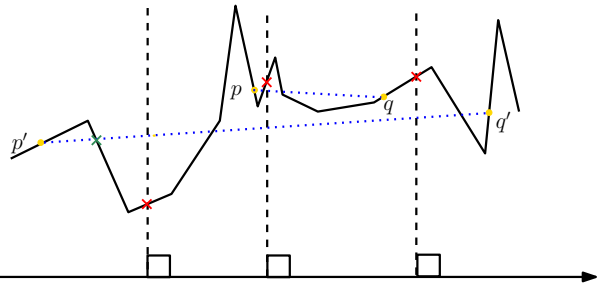


Figure 1: The points p and q mutually 2-crossing visible, while p' and q' are not.

The watchtower problem presents itself in two forms: discrete and continuous. In the discrete version, the watchtower must be located on a vertical line through a vertex of the terrain, while in the continuous version the watchtower can be located anywhere above the terrain. Solutions to the discrete and continuous watchtower problems can vary significantly. Figure 2 shows an instance for which the solution to the continuous 1-watchtower problem has height zero (on the terrain), whereas the solution to the discrete 1-watchtower problem on the same terrain requires a watchtower to be positioned significantly higher.

This paper examines algorithms for the 1-watchtower problem, for both the discrete and continuous cases, under k-visibility. We also describe faster algorithms for the case $k = 2$ and $k = 0$.

2 Related Work

The original terrain watchtower problem was introduced by Sharir for polyhedral terrains [11]. The minimum height for one watchtower can be found in $O(n \log n)$ time for both the continuous and discrete problems under 0-crossing visibility on an x-monotone polyhedral terrain in $\mathbb{R}^3$ [12].

Bespamyatnikh et al. [4] proposed an $O(n^4)$-time algorithm for the discrete 2-watchtower problem under 0-crossing visibility on a 2.5D terrain. They also gen-

*Department of Computer Science, University of Manitoba, bahoo@cs.umanitoba.ca

§School of Computer Science, Carleton University, jit@scs.carleton.ca

**Department of Computer Science, University of Manitoba, durocher@cs.umanitoba.ca

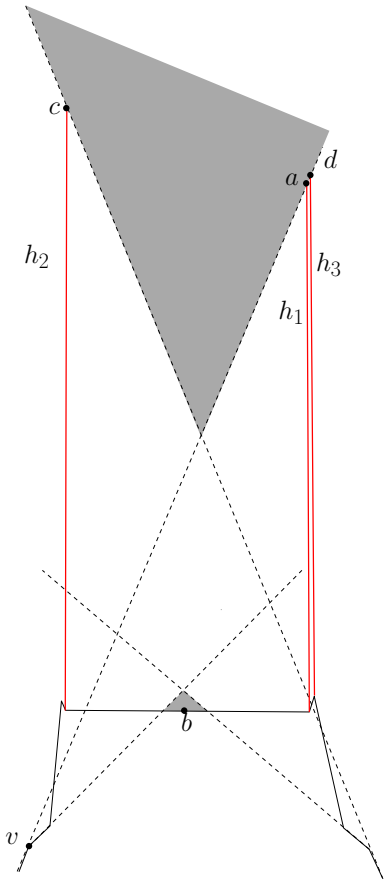


Figure 2: When $k = 2$, the solution of the 1-watchtower problem for the continuous version is much smaller than the discrete version. The points b and d represent the locations of the watchtower in the continuous and discrete versions, respectively (suppose $h_3 < h_1$). In the continuous version, the tower is located on the edge of the terrain with height zero, while in the discrete version it must be located above the terrain with height h_3, significantly bigger than zero. Notice that the points below d cannot see the edges adjacent to the vertex v.

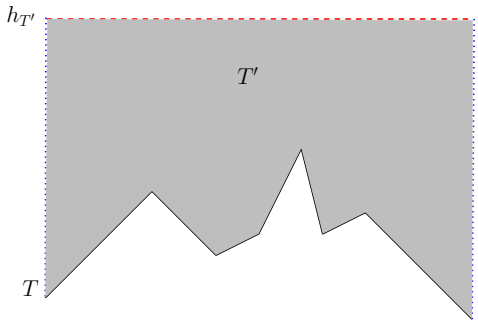


Figure 3: The shaded region is a simple polygon T' constructed for a given terrain.

eralized their approach to the continuous version of the problem with assumptions on the time required to solve a specific cubic equation with three bounded variables. Under the assumption that the equation can be solved in $O(f_3)$ time, their approach takes $O(n^4 + n^3 f_3)$ time. Using parametric search, they show that the discrete and continuous versions of the problem can be solved in $O(n^3 \log^2 n)$ and $O(n^4 \log^2 n)$ time, respectively. Ben-Moshe et al. [2] improved the time to $O(n^{3/2}\sqrt{m'(n)})$ for the discrete 2-watchtower problem, where $m'(n)$ denotes the time required to multiply two $n \times n$ matrices, resulting in a time of $O(n^{2.37+\epsilon})$ using the current fastest matrix multiplication algorithm [8]. Using parametric search, Agarwal et al. [1] improved the time complexity of the discrete and continuous 2-watchtower problems for 0-crossing visibility to $O(n^2 \log^4 n)$ and $O(n^3 \alpha(n) \log^3 n)$ respectively, where $\alpha(n)$ denotes the inverse Ackermann function.

The watchtower problem generalizes to the setting of k-crossing visibility for any k. We consider the problem of placing one watchtower. In Section 3, we present an algorithm for the continuous problem, and then propose an algorithm for the discrete problem in Section 4. For both algorithms we describe how the running time can be decreased when $k = 2$ and $k = 0$.

3 The Continuous Case

In this section, we solve the continuous 1-watchtower problem under k-visibility for general k, and then describe how the running time can be reduced when $k = 2$ and $k = 0$.

Consider a simple polygon T', bounded from above by a horizontal line segment $h_{T'}$ that lies above T, and on its sides by vertical line segments aligned with the respective left and right endpoints of T; see Figure 3. We first find the k-kernel of T'. The k-kernel of a given polygon P is the set of all points p such that every point in P is k-crossing visible from p; see Figure 4. The algorithm of Evans and Sember [6] finds the k-kernel of T' in $O(n^2 \log n + h)$ time, where h denotes the complexity

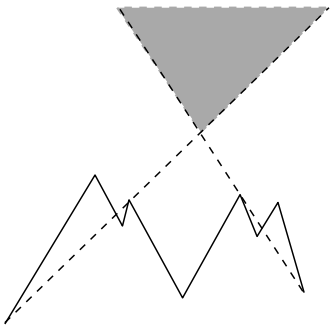


Figure 4: 2-kernel

(the number of boundary vertices) of the k-kernel. The k-kernel consists of $O(n^4)$ disjoint simple polygons. The worst-case number of vertices of the k-kernel is $\Theta(n^4)$. For $k = 2$, the complexity of the k-kernel is $\Theta(n^2)$, and for $k = 3$, the complexity of the k-kernel is $O(n^4)$ and $\Omega(n^2)$ [6].

The lower envelope of the portion of the k-kernel above T is the locus of feasible locations for the top of the watchtower from which the entire terrain T is k-crossing visible. Finding the minimum-length vertical line segment between this lower envelope and T yields the optimal solution for the 1-watchtower problem; see Figure 6. Notice that given line segments s_1 and s_2 that intersect a vertical line, the distance between s_1 and s_2 along the vertical line is minimized at a vertex of s_1 or a vertex of s_2. Hence, to find the optimal height for the continuous 1-watchtower problem, it suffices to examine vertical line segments from the vertices of the lower envelope of the k-kernel to T, and vertical line segments from the vertices of T to the lower envelope of the k-kernel. The minimum length of these line segments is the minimum height of the continuous 1-watchtower problem.

The minimum height of a watchtower can be found by partitioning the edges of the k-kernel into those that lie above T and those that lie below T. Following this partition, the lower envelope of the edges above T is computed. By sweeping a vertical line across T and the lower envelope, we stop at all vertices to evaluate the distance on the sweep line between these two x-monotone chains, maintaining the minimum distance thus far. These steps can be implemented in a single sweep using a modification of the algorithm of Bentley and Ottmann [3]. At each event during the sweep, it suffices to measure the distance along the sweep line between T and the closest line segment above T. If this distance is less than the previously recorded minimum, we update the minimum distance and the current x-coordinate of the sweep line. Observe that no two edges of the k-kernel cross, and that no two edges of T cross. Furthermore, if any edge of the k-kernel crosses T, then this point of intersection corresponds to the location of a watchtower of height zero: this is the solution, and

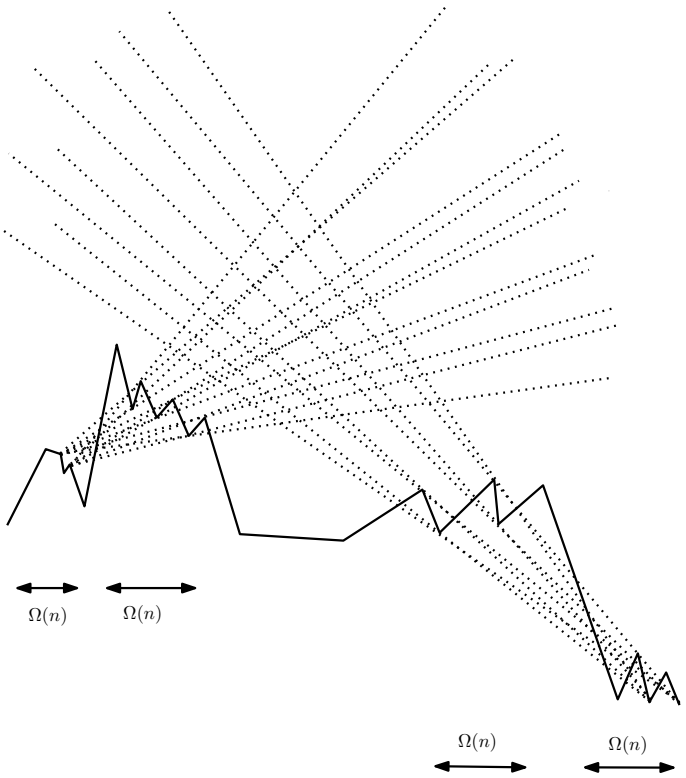


Figure 5: The 4-kernel of a monotone chain has $\Omega(n^4)$ vertices. There are $\Omega(n^2)$ cells in the arrangement of dotted lines that form the v-regions of the vertices on the terrain. These lines have $\Omega(n^2)$ points of intersection.

the algorithm terminates. Consequently, the number of intersection events processed is at most 1. Since the number of edges in the k-kernel is $h \in O(n^4)$ and the number of edges in T is n, the total running time of the algorithm is $O((n^2 + h) \log n))$.

Although we seek the k-kernel in a restricted type of polygon, i.e., a monotone polygon, the k-kernel for a monotone polygon has $\Theta(n^4)$ complexity in the worst case when $k \geq 4$; see Figure 5. The complexity of the k-kernel when $k = 3$ is unknown [6]. When $k = 2$ its complexity is $O(n^2)$. Since the watchtower must be located above the terrain, it must be inside T'.

When $k = 0$, the 0-kernel corresponds to the kernel of the polygon T'. This kernel is a convex polygon with $O(n)$ vertices from which the entire polygon is 0-crossing visible. Additionally, the kernel is the feasible region for the watchtower, and can be determined in $O(n)$ time [9, 10]; see Figure 6. As mentioned above, to find the solution for the continuous 1-watchtower problem, it is sufficient to examine the vertical line segments from the vertices of the kernel to T, and the vertical line segments from the vertices of T to the kernel. The boundary of the 0-kernel is an x-monotone chain consisting of $O(n)$ vertices given in order. The terrain T is

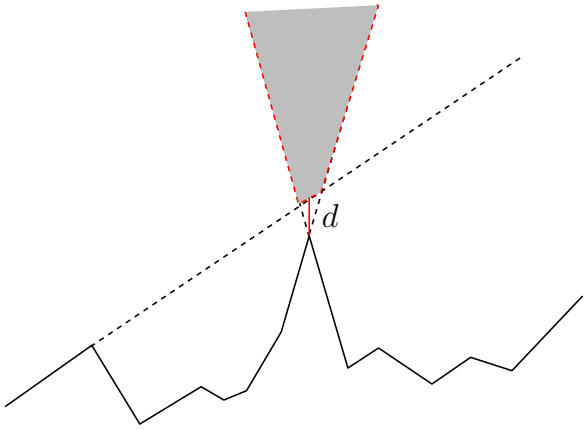


Figure 6: The shaded region is the intersection of the visible part of the plane for each vertex when $k = 0$; dotted lines show the boundaries of some of these regions.

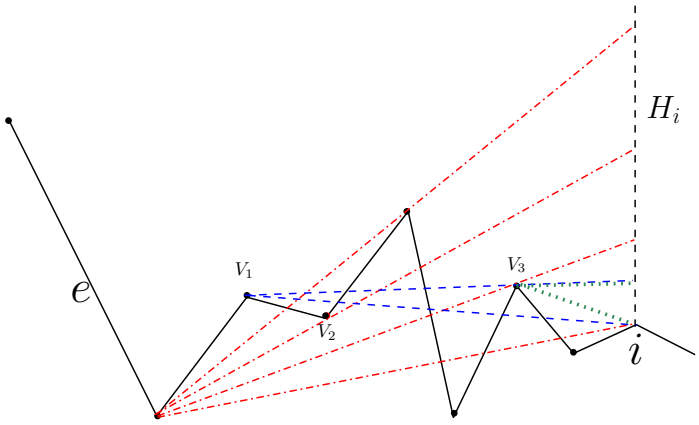


Figure 7: The v-regions and their intersection with H_i for three vertices V_1, V_2 and V_3 are shown in dashed, dotted, dashed and dotted respectively.

an x-monotone chain of n vertices given in order. By merging the two sets of sorted vertices of T and of the kernel in O(n) time, for each vertex in the merged sorted list the corresponding edge intersected by the vertical line segment can be found in $O(1)$ time by comparing the current vertex against the previous vertex in the list. If the previous vertex is on the same chain, then the current vertex intersects the same edge as the previous vertex. Otherwise, if the previous vertex is not on the same chain, then the edge that starts from the previous vertex is the intersected edge. At each step, the minimum vertical line segment encountered is maintained. Thus, the minimum length segment can be found in $O(n)$ time.

When $k = 2$, the boundary of the 2-kernel has $O(n^2)$ vertices [6]. Consequently, we can find the minimum length vertical line segment between the 2-kernel and the terrain T in $O(n^2 \log n)$ time, so the continuous 1-watchtower problem for 2-visibility can be solved in $O(n^2 \log n)$ time.

Theorem 1 *The continuous 1-watchtower problem can be solved in $O((n^2 + h) \log n)$ time under k-crossing visibility, where $h \in O(n^4)$ is the size of the k-kernel. For $k = 0$ and $k = 2$, the continuous 1-watchtower problem can be solved in $O(n)$ and $O(n^2 \log n)$ time, respectively.*

4 The Discrete Case

In this section, we propose an $O(n^3)$-time algorithm for the discrete k-crossing visible 1-watchtower problem on a terrain T.

As defined by Evans and Sember [6], "The v-region for vertex v of a polygon P, is the set of points q for which q is k-visible to every point of P on ray $\rightarrow qv$". The boundary of each v-region is a simple polygon with $O(n)$ vertices [6]. Computing the v-region of each vertex of the polygon takes $O(n \log n)$ time. We compute the v-region for each vertex of T' in $O(n \log n)$ time per vertex using the algorithm of Evans and Sember [6], using $O(n^2 \log n)$ total time. The intersection of v-regions of the polygon P is the k-kernel of the polygon P [6]. In other words, the intersection of v-regions of the vertices of P is the locus for the watchtower.

Observation 1 *The intersection of the v-regions of the vertices of T corresponds to the set of feasible locations for the top of the watchtower.*

Proof. The intersection of the v-regions is the k-kernel of T' [6], which is the region where the entire T' including T is k-crossing visible from. So, T is k-crossing visible from a watchtower located in this region. □

In the discrete problem, the watchtower must be located on a vertical line emanating from a vertex of the terrain. Consider a vertical line passing through a vertex of the terrain. We find the intersection of the v-regions of the vertices of T with this vertical line.

Lemma 2 *Any vertical line crosses the boundaries of the v-regions of the vertices of T $O(n^2)$ times.*

Proof. The number of vertices on the boundary of each v-region is $O(n)$. So each v-region may intersect a vertical line $O(n)$ times. As there exist n v-regions, so the number of intersections between v-regions and any given vertical line is $O(n^2)$. □

Let V_i denote the v-region of vertex v_i in T. We have the following lemma:

Lemma 3 *The intersection of any v-region with any vertical line is a set of at most n disjoint intervals on the line, where the topmost interval is open.*

Proof. Considering a bounding box around T'. The v-region of a vertex v_i is a closed Jordan curve with $O(n)$ complexity. The intersection between the vertical line and the inside of this closed Jordan curve is a set of $O(n)$ intervals. The last interval is open as after moving sufficiently high above the terrain T all of T will be visible while looking toward the vertex v_i. □

Consider a vertical line ℓ_i passing through a given vertex v_i of T, and the intersections with the v-regions $V_1, \ldots, V_n$ for the vertices $v_1, \ldots, v_n$ of T. Let each v-region be determined by a specific color i. As a result, we have n different colors of intervals on the line ℓ_i. Each color is a set of $O(n)$ pairwise disjoint intervals. If the optimal watchtower lies on this vertical line, it is in the interval which intersects all n v-regions with the lowest y-coordinate. We define depth-n intervals as the intervals on ℓ_i on which all n v-regions intersect.

Lemma 4 *The minimum height of a watchtower located above the vertex v_i is the closest depth-n interval.*

Proof. Intervals with the same color do not intersect each other. So, the maximum number of intersection is n where n v-regions intersect. So, a depth-n interval is in the k-kernel and T is k-crossing visible from such intervals. Among all such depth-n intervals we look for the one that has the smallest distance from the terrain. □

As a result of Lemma 4, we can remove the color on the intervals. This transforms the problem to that of finding the depth-n intervals among $O(n^2)$ intervals.

Lemma 5 *Given a v-region of a vertex of the terrain T, finding and sorting the intersections of this v-region with a given vertical line takes $O(n)$.*

Proof. We can find the intersection of a v-region with the vertical line ℓ_i in $O(n)$ time. This gives a set of $O(n)$ intervals on ℓ_i. We can sort these intervals in $O(n)$ time as the v-region is a Jordan arc [7]. □

We find the sorted list of the intersections of each polygon V_i with a line ℓ_i in $O(n^2)$ time by Lemma 5. So we have n sorted lists each containing $O(n)$ intervals. Let these lists be labeled as $L_1, L_2, ..., L_n$. We have the following lemma:

Lemma 6 *The deepest interval with the minimum height for a set of $O(n^2)$ intervals on a given line ℓ_i can be found in $O(n^2)$ time.*

Proof. As mentioned in Lemma 5, each set of n intervals in the list L_i can be sorted in linear time. There exist n lists, so it takes $O(n^2)$ time to sort all $L_1, \ldots, L_n$. Consider two list L_1 and L_2. First, we find the intersections between L_1 and L_2. Given two sets of sorted intervals X and Y, their intersection can be found in $O(|X|+|Y|+h)$, where h denotes the number of output intervals [13]. As X and Y are of size $O(n)$ for the lists L_1 and L_2. h is also of size $O(n)$. This is because if an interval in L_1 intersects m intervals of L_2, remaining intervals in L_1 can intersect at most $n-m+2$ intervals in L_2. As a result, finding the intersection between L_1 and L_2 takes $O(n)$ time; let the output list be called L'_1. Next, we find the intersection of L'_1 and L_3 (called L'_2) in $O(n)$ time. Repeating this process, the intersection between L'_{n-1} and L_n results in the intersections of $L_1, L_2, ..., L_n$. There are n steps, each taking $O(n)$ time. The algorithm takes $O(n^2)$ total time. □

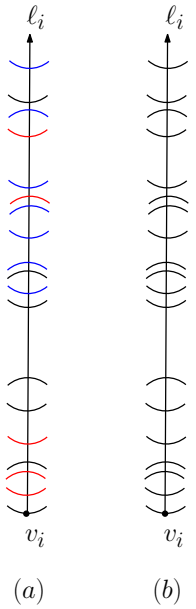


Figure 8: a. Colored intervals on a vertical line ℓ_i. b. Intervals can be considered as a set of $O(n^2)$ intervals without color.

Theorem 7 *The discrete 1-watchtower problem can be solved in $O(n^3)$ time under k-crossing visibility.*

Proof. There are n vertices in T corresponding to n vertical lines as the candidates for the location of the watchtower. By Lemmas 4 and 6, finding the minimum height of a watchtower located at the vertex v_i takes $O(n^2)$ time. So, the total required time is $O(n^3)$.

□

Considering 0-crossing visibility, the kernel is the potential location of the top of the watchtower as described for the continuous version. The difference between the discrete and continuous versions is that in the discrete version, the algorithm restricts the possible watchtowers to those whose x-coordinates coincide with a vertex of T. As a result, the discrete 1-watchtower problem under 0-crossing visibility can also be solved in $O(n)$ time.

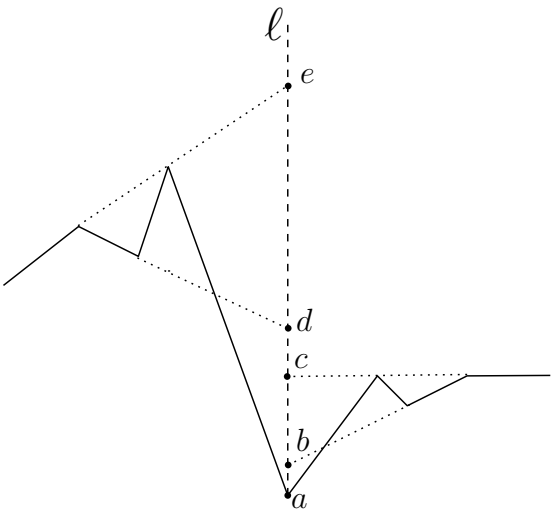


Figure 9: Going up and losing visibility: On point a, the entire terrain T is 2-crossing visible. At point b, the rightmost edge of T is not 2-crossing visible anymore. At point c, the entire terrain T becomes 2-crossing visible, while on d, the leftmost edge of T is not 2-crossing. At point e, T is 2-crossing visible again.

In the case of 2-crossing visibility, we apply the same approach as for the continuous version. The key difference is that only the vertical line segments emanating from vertices of the terrain are of interest as the possible location for the watchtower. As a result, the discrete version of the 2-watchtower problem can also be solved in $O(n^2 \log n)$ time.

4.1 Comparison between k-visibility and 0-visibility

As mentioned, both the discrete and continuous versions of the 1-watchtower problem for 0-crossing visibility can be solved in $O(n)$ time, while for k-crossing visibility the time complexity increases significantly when $k > 0$. The main reason is the fact that when $k \neq 0$, the k-kernel can be disconnected. Under 0-visibility, increasing the height of a watchtower always increases its visibility; that is, if p and q are two points on a vertical line above T, where p lies above q, then the region of T visible to q is contained in the region of T visible to p. This property does not hold when $k > 0$; q could see all of T (i.e., q is in the k-kernel), whereas p does not see all of T, even though p lies above q. See Figure 9.

5 Possible Directions for Future Research

The 1-watchtower problem generalizes to the m-watchtower problem, where instead of positioning a single watchtower to guard the terrain T, an algorithm must select positions for m watchtowers. The goal is to minimize the maximum height of any watchtower, while ensuring that each point on T is k-crossing visible from at least one watchtower. To solve the continuous 1-watchtower problem, it suffices to consider candidate locations for the watchtower whose x-coordinate coincides with that of a vertex of T or a vertex of the k-kernel of T. This property is not true in general for the continuous m-watchtower problem, even when $m = 2$; see Figure 10. It remains open to find an efficient algorithm to solve the (discrete or continuous) m-watchtower problem under k-crossing visibility, even for $m = 2$.

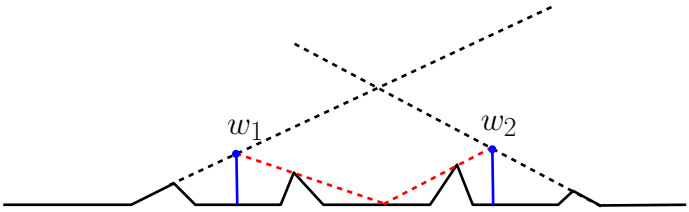


Figure 10: Even when $k = 0$ in the 2-watchtower problem, the x-coordinates of watchtowers do not coincide with those of vertices of the terrain, vertices of the k-kernel, nor of the intersections of the $\Theta(n^2)$ lines determined by pairs of vertices of the terrain.

References

[1] P. Agarwal, S. Bereg, O. Daescu, H. Kaplan, S. Ntafos, and B. Zhu. Guarding a terrain by two watchtowers. In *Proceedings of the Symposium on Computational Geometry (SOCG)*, pages 346–355. ACM, 2005.

[2] B. Ben-Moshe, P. Carmi, and M. Katz. Computing all large sums-of-pairs in R^n and the discrete planar two-watchtower problem. *Information Processing Letters*, 89(3):137–139, 2004.

[3] J. Bentley and T. Ottmann. Algorithms for reporting and counting geometric intersections. *IEEE Transactions on computers*, (9):643–647, 1979.

[4] S. Bespamyatnikh, Z. Chen, K. Wang, and B. Zhu. On the planar two-watchtower problem. In *Proceedings of the International Computing and Combinatorics Conference (COCOON)*, volume 2108 of *Lecture Notes in Computer Science*, pages 121–130. Springer, 2001.

[5] H. Chang, J. Erickson, and C. Xu. Detecting weakly simple polygons. In *Proceedings of the SIAM Symposium on Discrete Algorithms (SODA)*, pages 1655–1670. ACM-SIAM, 2015.

[6] W. Evans and J. Sember. k-star-shaped polygons. In *Proceedings of the Canadian Conference on Computational Geometry (CCCG)*, pages 215–218, 2010.

[7] K. Fung, T. Nicholl, R. Tarjan, and C. Van Wyk. Simplified linear-time Jordan sorting and polygon clipping. *Information Processing Letters*, 35(2):85–92, 1990.

[8] F. Le Gall. Powers of tensors and fast matrix multiplication. In *Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation*, pages 296–303. ACM, 2014.

[9] D. Lee and F. Preparata. An optimal algorithm for finding the kernel of a polygon. Technical report, University of Illinois at Urbana Champaign, Coordinated Science lab, 1977.

[10] D. Lee and F. Preparata. An optimal algorithm for finding the kernel of a polygon. *Journal of the Association for Computing Machinery*, 26(3):415–421, 1979.

[11] M. Sharir. The shortest watchtower and related problems for polyhedral terrains. *Information Processing Letters*, 29(5):265–270, 1988.

[12] B. Zhu. Computing the shortest watchtower of a polyhedral terrain in $O(n \log n)$ time. *Computational Geometry*, 8(4):181–193, 1997.

[13] A. Zomorodian and H. Edelsbrunner. Fast software for box intersections. *International Journal of Computational Geometry & Applications*, 12:143–172, 2002.

Rock Climber Distance: Frogs versus Dogs*

Hugo A. Akitaya† Leonie Ryvkin‡ Csaba D. Tóth†§

Abstract

The classical measure of similarity between two polygonal chains in Euclidean space is the Fréchet distance, which corresponds to the coordinated motion of two mobile agents along the chains while minimizing their maximum distance. As computing the Fréchet distance takes near-quadratic time under the Strong Exponential Time Hypothesis (SETH), we explore two new distance measures, called **rock climber distance** and **k-station distance**, in which the agents move alternately in their coordinated motion that traverses the polygonal chains. We show that the new variants are equivalent to the Fréchet or the Hausdorff distance if the number of moves is unlimited. When the number of moves is limited to a given parameter k, we show that it is NP-hard to determine the distance between two curves. We also describe a 2-approximation algorithm to find the minimum k for which the distance drops below a given threshold.

1 Introduction

Recognizing similarity between geometric objects is a classical problem in pattern matching, and has recently gained renewed attention due to its applications in artificial intelligence and robotics. Statistical methods and the Hausdorff distance have proved to be good similarity measures for static objects, but are insensitive to spatio-temporal data, such as individual trajectories or clusters (flocks) of trajectories. The **Fréchet distance** (defined below) is considered to be one of the best similarity measures between curves in space. Between two polygonal chains with a total of n vertices, the Fréchet distance can be computed in $O(n^2 \text{ polylog } n)$ time [4, 14]. Under the Strong Exponential Time Hypothesis (SETH), there is a lower bound of $\Omega(n^{2-\delta})$, for any $\delta > 0$, for computing the Fréchet distance [12], or even approximating it within a factor of 3 [16]. Without SETH, the current best lower bound for the time complexity under the algebraic decision tree model is $\Omega(n \log n)$ [13].

Applications, however, call for efficient algorithms for massive trajectory data. This motivates the quest for new variants of the Fréchet distance that may bypass some of its computational bottlenecks but maintain approximation guarantees.

In this paper, we introduce the **rock climber** distance. It combines properties of the continuous and the discrete Fréchet distance, and is closely related to the recently introduced k-Fréchet distance [2]. The classic **Fréchet distance** corresponds to coordinated motion, where two agents follow the polygonal paths P and Q, so that they minimize the maximum distance between the agents (intuitively, the agents are a man and a dog, and they minimize the length of the leash between them). The **discrete Fréchet distance** considers discrete motion on the vertices of the two chains (i.e., walking a frog [24], pun intended). The **rock climber distance** corresponds to a coordinated motion of two agents along P and Q that is continuous, but only one agent moves at a time, hence it can be described by an axis-parallel path in a suitable parameter space (the so-called free space diagram, described below).

Definitions Given two polygonal chains, parameterized by piecewise linear curves, $P : [0,1] \to \mathbb{R}^2$ and $Q : [0,1] \to \mathbb{R}^2$, the Hausdorff distance is defined as

$$\delta_{\mathrm{H}}(P,Q) = \max\{ \max_{s\in[0,1]} \min_{t\in[0,1]} \|P(s) - Q(t)\|, \max_{t\in[0,1]} \min_{s\in[0,1]} \|P(s) - Q(t)\|\}.$$

and the Fréchet distance is defined as

$$\delta_{\mathrm{F}}(P,Q) = \inf_{\sigma,\tau} \max_{t\in[0,1]} \|P(\sigma(t)) - Q(\tau(t))\|,$$

where $\sigma, \tau : [0,1] \to [0,1]$ range over all orientation-preserving homeomorphisms of $[0,1]$. The standard machinery for finding nearby points in the two polygonal chains, introduced by Alt and Godau [4] uses the so-called **free space diagram**. For every $\varepsilon > 0$, the **free space** is defined as

$$F_\varepsilon(P,Q) = \{(s,t) \in [0,1]^2 \colon \|P(s) - Q(t)\| \le \varepsilon\}.$$

Note that $F_\varepsilon(P,Q) \subset [0,1]^2$, where a point $(s,t) \in [0,1]^2$ corresponds to the positions $P(s)$ and $Q(t)$ on the two chains. The Fréchet distance between P and Q is at most ε if and only if the free space contains a strictly x- and y-monotone path from $(0,0)$ to $(1,1)$; namely, $\gamma : [0,1] \to [0,1]^2$, $\gamma(t) = (\sigma(t), \tau(t))$.

*Research supported in part by NSF CCF-1422311 & 1423615.
†Dept. Comp. Sci., Tufts University, Medford, MA, USA.
‡Dept. Mathematics, Ruhr University Bochum, Germany
§Dept. Mathematics, CSUN, Los Angeles, CA, USA.

We define further terms connected to the free space diagram below: A **component** of a free space diagram is a connected subset $c \subseteq F_\varepsilon(P,Q)$. A set S of components **covers** a set $I \subseteq [0,1]_P$ of the parameter space (corresponding to the curve P) if I is a subset of the projection of S onto said parameter space, i.e., $\forall x \in I\colon \exists c \in S, y \in [0,1]_Q\colon (x,y) \in c$. Covering on the second parameter space is defined analogously.

Rock Climbers Distance. Assume that two rock climbers each choose a route on a vertical wall, represented by polygonal chains P and Q. They secure each other with a rope: While one endpoint of the rope is firmly attached to the rock, the other endpoint may move. Both climbers must be secured at all times, and so only one climber can move at a time. The **rock climber distance** is the minimum length of a rope that allows them to traverse the routes P and Q, that is,

$$\delta_{\mathrm{rock}}(P,Q) = \inf_{\gamma} \max_{t\in[0,1]} \|P(\sigma(t)) - Q(\tau(t))\|, \quad (1)$$

where $\gamma : [0,1] \to [0,1]^2$, $\gamma(t) = (\sigma(t), \tau(t))$, ranges over all x- and y-monotonically increasing axis-parallel paths from $(0,0)$ to $(1,1)$.

We show that $\delta_{\mathrm{rock}}(P,Q) = \delta_{\mathrm{F}}(P,Q)$ (cf. Theorem 5), albeit the number of turns of the path γ may far exceed the number of vertices of P and Q. This indicates that the number of axis-parallel segments in γ is a crucial parameter. For every $k \in \mathbb{N}$, we define $\delta_{\mathrm{rock}}(k,P,Q)$ by equation (1) with the additional condition that the path γ consists of at most k line segments.

Rock Climber Distance with k Stations. The main focus of this paper is a variant of the rock climber distance, where the number of axis-parallel segments is a fixed parameter k, but these segments need not form a continuous path from $(0,0)$ to $(1,1)$. Assume that a rock climber club decides to install permanent safety ropes along the routes P and Q for training purposes. Each rope has one fixed endpoint on P or Q, and its other endpoint can move freely on some subcurve of the other polygonal chain (Q or P, respectively). The mobile endpoint of a rope, however, cannot pass through the fixed endpoint of another rope. The club decides to install $k \in \mathbb{N}$ identical ropes: What is the minimum length of a rope that allows safe traversal on both P and Q? More formally, we arrive at the following definition.

Definition 1 *For two polygonal chains, P and Q, and an integer $k \in \mathbb{N}$, the **k-station distance**, denoted $\delta_{\mathrm{station}}(k,P,Q)$, is the infimum of all $\varepsilon > 0$ such that there exist two subdivisions $0 = a_0 < a_1 < \ldots < a_p = 1$ and $0 = b_0 < b_1 < \ldots < b_q = 1$ into a total of $p+q = k$ intervals such that*

$$\min_{j\in\{1,\ldots,q\}} \min_{s\in[a_{i-1},a_i]} \|P(s) - Q(b_j)\| \le \varepsilon \text{ for } i = 1,\ldots,p;$$

$$\min_{i\in\{1,\ldots,p\}} \min_{t\in[b_{j-1},b_j]} \|P(a_i) - Q(t)\| \le \varepsilon \text{ for } j = 1,\ldots,q.$$

Every subcurve $P[a_{i-1},a_i]$ of P has some closest point $Q(b_{j(i)})$ in Q; and every subcurve $Q[b_{j-1},b_j]$ of Q has a closest point $P(b_{i(j)})$ in P. In the free space diagram $F_\varepsilon(P,Q)$, where $\varepsilon = \delta_{\mathrm{station}}(k,P,Q)$, the union of horizontal segments $[a_{i-1},a_i]\times\{b_{j(i)}\}$ and vertical segments $\{a_{i(j)}\}\times[b_{j-1},b_j]$ projects surjectively to the unit interval $[0,1]$ on each coordinate axis.

Fréchet Distance with k Jumps. The k-station distance can also be considered as a variant of the **k-Fréchet distance**, introduced by Buchin and Ryvkin [18] (see also [2]). Intuitively, it measures the similarity between two polygonal chains after k "mutations." Formally, $\delta_{\mathrm{cut}}(k,P,Q)$ is the infimum of $\varepsilon > 0$ such that P and Q can each be subdivided into k subcurves, P_i and Q_i ($i = 1,\ldots,k$), where $\delta_{\mathrm{F}}(P_i, Q_{\pi(i)}) \le \varepsilon$ for some permutation $\pi : [k] \to [k]$. Importantly, the chains P and Q can be subdivided at any point, not only at vertices. Determining the minimum $k \in \mathbb{N}$ for which $\delta_{\mathrm{cut}}(k,P,Q) \le \varepsilon$ for a given ε is NP-hard, and conjectured to be $\exists\mathbb{R}$-hard. The k-station distance can be considered as a restricted version of the k-Fréchet distance, where either P_i or $Q_{\pi(i)}$ is required to be a single point (i.e., a trivial curve) for $i = 1,\ldots,k$. By definition, we have $\delta_{\mathrm{cut}}(k,P,Q) \le \delta_{\mathrm{station}}(k,P,Q)$ for all $k \in \mathbb{N}$.

Unit Disk Cover (UDC). The rock climber k-station distance is also reminiscent of the unit disk cover problem: Given a point set $S \subset \mathbb{R}^2$, find a minimum set $\mathcal{D}$ of unit disks such that $S \subset \bigcup \mathcal{D}$. When S is finite, UDC is known to be NP-hard [21], one can find a 4-approximation in $O(n \log n)$ time [11]. In the **Discrete Unit Disk Cover** problem, S is finite, and the disks are restricted to a finite set of possible centers [10]; the discretized version admits a PTAS via local search [29, 30]. These results extend to the cases where S is confined to a narrow strip [22], or S is a finite union of line segments [9]. Finding the minimum $k \in \mathbb{N}$ such that $\delta_{\mathrm{station}}(k,P,Q) \le 1$ can be considered as a variant of UDC, where P (resp., Q) must be covered by disks centered at Q (resp., P), and each disk can cover at most one contiguous arc of a curve.

Our Results. We prove the following results.

1. We show that $\delta_{\mathrm{rock}}(P,Q) = \delta_{\mathrm{F}}(P,Q)$ and $\delta_{\mathrm{station}}(P,Q) = \delta_{\mathrm{H}}(P,Q)$ for a sufficiently large k (that depends on P and Q). It follows that for any two polygonal chains, P and Q, there exists a positive integer k such that $\delta_{\mathrm{cut}}(k,P,Q) \le \delta_{\mathrm{F}}(P,Q)$. The first identity implies that $\delta_{\mathrm{rock}}(P,Q)$ can be computed in $O(n^2\sqrt{\log n}(\log\log)^{3/2})$ time [14], where P and Q jointly have n vertices (Section 2).

2. We prove that it is NP-complete to decide whether $\delta_{\text{station}}(k, P, Q) \leq \varepsilon$ for two given polygonal chains, P and Q, and parameters k and $\varepsilon > 0$ (Section 3).

3. We also give a 2-approximation algorithm for finding the minimum $k \in \mathbb{N}$ such that $\delta_{\text{station}}(P, Q, k) \leq \varepsilon$ for given polygonal chains P and Q, and a threshold $\varepsilon > 0$. We reduce the problem to a variant of the set cover problem over axis-parallel line segments, for which a greedy strategy yields a 2-approximation (Section 4).

Further Related Previous Work. Alt, Knauer, and Wenk [5] compared the Hausdorff to the Fréchet distance and discussed κ-bounded curves as a special input instance. In particular, they showed that for convex closed curves Hausdorff distance equals Fréchet distance. For curves in one dimension Buchin et al. [13] proved equality of Hausdorff and weak Fréchet distance using the well-known Mountain Climbing theorem [23]. Recently, Driemel et al. [20] gave bounds on the VC-dimension of curves under Hausdorff and Fréchet distances. Buchin [17] characterized these measures in terms of the free space, which motivated the study of the variants of the k-Fréchet distance; see also Har-Peled and Raichel [25] for a treatment using product spaces. The k-station distance is also related to partial curve matching, studied by Buchin, Buchin, and Wang [15], who presented a polynomial-time algorithm to compute the "partial Fréchet similarity." A variation of this similarity was considered by Scheffer [31].

2 Relations to Other Distance Measures

In this section, we compare the rock climber distance and the k-station distance to the Fréchet and Hausdorff distances, as well as the cut distance.

Preliminaries. Let $P : [0,1] \to \mathbb{R}^2$ and $Q : [0,1] \to \mathbb{R}^2$ two piecewise linear curves. That is, there are subdivisions $0 = a_0 < a_1 < \ldots < a_m = 1$ and $0 = b_0 < b_1 < \ldots < b_n = 1$ such that P, Q are linear on each subinterval $[a_{i-1}, a_i]$ and $[b_{j-1}, b_j]$, respectively. Recall that for every $\varepsilon > 0$, the free space is defined as $F_\varepsilon(P, Q) = \{(s,t) \in [0,1]^2 : \|P(s) - Q(t)\| \leq \varepsilon\}$, which is a subset of the configuration space $U = [0,1]^2$. We can subdivide U into mn **cells** of the form $C_{i,j} = [a_{i-1}, a_i] \times [b_{j-1}, b_j]$, for $i = 1, \ldots, m$ and $j = 1, \ldots, n$. It is known that $C_{i,j} \cap F_\varepsilon(P, Q) = C_{i,j} \cap E_{i,j}$, where $E_{i,j}$ is either an ellipse or a slab parallel to the diagonal of $C_{i,j}$ (in case $P([a_{i-1}, a_i])$ and $Q([b_{j-1}, b_k])$ are parallel line segments).

Geometric Properties. We prove a few elementary properties for monotone curves passing through a cell of the free space diagram. We start with an easy observation. Omitted proofs are available in the full paper [3].

Lemma 2 *Let E be an ellipse with maximal curvature κ. Then for every point $p \in \partial E$, there are horizontal and vertical segments H_p and V_p, respectively, such that $p \in H_p \subset E$, $p \in V_p \subset E$, and $\|H_p\| + \|V_p\| \geq 2/\kappa$.*

Lemma 3 *Let C be an axis-aligned rectangle and E an ellipse such that $C \cap E \neq \emptyset$. Let $\alpha : [0,1] \to C \cap E$ be an x- and y-monotone increasing curve. Then there exists an x- and y-monotone increasing curve $\beta : [0,1] \to C \cap E$ such that $\beta(0) = \alpha(0)$, $\beta(1) = \alpha(1)$, and (the image of) β is a polygonal chain consisting of a finite number of axis-parallel edges.*

For a set $S \subset \mathbb{R}^2$, let $\mathsf{proj}_x(S)$ and $\mathsf{proj}_y(S)$ denote the orthogonal projection of S onto the x- and the y-axis, respectively.

Lemma 4 *Let C be an axis-aligned rectangle and E an ellipse such that $C \cap E \neq \emptyset$. Then there exists a finite set $\mathcal{S}$ of axis-parallel line segments in $C \cap E$ such that $\mathsf{proj}_x(C \cap E) = \mathsf{proj}_x(\bigcup \mathcal{S})$ and $\mathsf{proj}_y(C \cap E) = \mathsf{proj}_y(\bigcup \mathcal{S})$.*

Relation to the Fréchet Distance. We show that the rock climber distance equals the Fréchet distance.

Theorem 5 *For two polygonal chains, P and Q, it holds that $\delta_{\text{rock}}(P, Q) = \delta_{\text{F}}(P, Q)$.*

For two polygonal chains, P and Q, with a total of n segments, $\delta_{\text{F}}(P, Q)$ can be computed in $O(n^2 \sqrt{\log n} (\log \log)^{3/2})$ time [14]. Consequently, $\delta_{\text{rock}}(P, Q)$ can be computed in the same time, regardless of the complexity of the path γ in $F_{\varepsilon+\delta}(P, Q)$.

Relation to the Hausdorff and k-Fréchet Distances. The k-station distance between P and Q equals their Hausdorff distance for a sufficiently large integer k.

Theorem 6 *For two polygonal chains, P and Q, and for $\varepsilon > 0$, there exists a $k \in \mathbb{N}$ such that $\delta_{\text{station}}(k, P, Q) = \delta_{\text{H}}(P, Q)$.*

Remark. In the proofs of Theorems 5 and 6 (see [3]), we "inflate" the free space $F_\varepsilon(P, Q)$ into $F_{\varepsilon+\delta}(P, Q)$, $\delta > 0$, to avoid the case that P and Q contain parallel segments at distance ε. This step is necessary, as the free space $F_\varepsilon(P, Q)$, where $\varepsilon = \delta_{\text{F}}(P, Q)$, need not contain an axis-parallel path from $(0,0)$ to $(1,1)$. In the simplest example, P and Q are two parallel segments: The free space consists only of the straight line segment at the diagonal of $[0,1]^2$.

Figure 1 shows an example where three segments in P are parallel to two segments in Q at distance ε apart. It

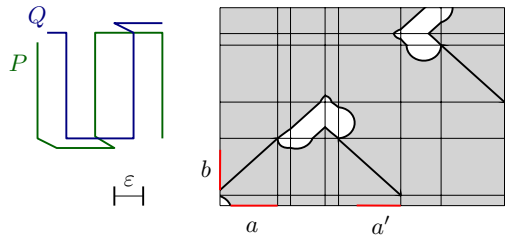


Figure 1: To project onto intervals a and a' we need to use the two straight line components above them, but then b has two preimages for its projection.

is impossible to cut P and Q into $k \in \{2, 3\}$ pieces such that $\delta_{\text{cut}}(k, P, Q) \approx \delta_{\text{H}}(P, Q)$. However, if we allow an arbitrarily large $k \in \mathbb{N}$, it is possible to place multiple cuts within a tiny distance in order to make sure that both parameter spaces can be covered by tiny slices of components.

Remark. For two polygonal chains, P and Q, with a total of n segments, the free space $F_\varepsilon(P, Q)$ is bounded by $N = O(n^2)$ line segments and elliptical arcs for every $\varepsilon > 0$. Mitchell et al. [27, 28] proved that the rectilinear link distance between two points in a rectilinear polygonal domain with N vertices can be computed in $O(N \log N)$ time. Perhaps this method can be adapted to decide whether the rectilinear link distance between $(0, 0)$ and $(1, 1)$ in the free space $F_\varepsilon(P, Q)$ does not exceed a given parameter in time polynomial in k and n. One could then find the infimum of $\varepsilon > 0$ such that $F_\varepsilon(P, Q)$ contains such a path with k or fewer links by parametric search [32], and compute $\delta_{\text{rock}}(k, P, Q)$ in polynomial time.

3 NP-Hardness

The k-station distance raises several optimization problems.

- Can we find the minimum $\varepsilon > 0$ such that $\delta_{\text{station}}(k, P, Q) \leq \varepsilon$ for two polygonal chains P and Q, and an integer k?
- Can we find the minimum $k \in \mathbb{N}$ for a given threshold $\varepsilon > 0$?

In this section, we show that the decision versions of these problems are NP-hard. That is, it is NP-hard to decide whether $\delta_{\text{station}}(k, P, Q) \leq \varepsilon$. Our reduction will produce weakly simple polygonal chains P and Q. A polygonal chain is **weakly simple** if its vertices can be moved by some arbitrary small amount to produce a Jordan arc [1, 19].

We reduce from Planar-Rectilinear-3SAT which is NP-complete [26]. An instance of Planar-Rectilinear-3SAT is defined by a boolean formula Φ in 3-CNF with n variables and m clauses. The formula is accompanied by a planar rectilinear drawing of the bipartite graph between variables and clauses in an integer grid where all variables are represented by points on the x-axis, and edges do not cross this axis. The problem asks whether there is an assignment from the variable set to {`true, false`} such that Φ evaluates to `true`.

Theorem 7 *It is NP-hard to decide whether $\delta_{\text{station}}(k, P, Q) \leq \varepsilon$ for given $k > 0$ and $\varepsilon > 0$, even when P and Q are weakly simple polygonal chains.*

Proof. We present here only an overview of the proof. The details are available in the full paper [3]. Given an instance A of Planar-Rectilinear-3SAT, we build an instance B of our problem producing two polygonal chains, P and Q, as shown in Figure 2. The chain P (Q) is represented by a blue (red) curve. Black edges represent overlap between P and Q. We set $\varepsilon := 1$, and design P and Q so that the length of almost every edge is an integer. That allows us to compute locally optimal solutions along the black edges that require a consistent choice of station placement alternating between blue and red stations, which in turn establishes a lower bound on the number of stations. We set the parameter k so that every solution must meet that lower bound. In the variable gadget, a concatenation of **literal gadgets** (Figure 3 (a)) must alternate consistently in order to achieve this lower bound. The choice of whether to start with a blue or a red station encodes the truth value of the variable. The **separation gadget** (Figure 3 (c)) allows choosing truth values for each variable independently. In the **clause gadget** (Figure 3 (d)) a subchain of Q (near p_5) can be covered by a blue station of the alternation of a literal gadget if the literal evaluates to `true`. If all literals in the clause evaluate `false`, then either an additional station is needed or ε has to be increased. Hence, $\delta_{\text{station}}(k, P, Q) \leq \varepsilon$ if and only if the instance A admits a positive solution. □

4 Approximation Algorithms

In this section, we show that for two polygonal chains, P and Q, and a threshold $\varepsilon > 0$, we can approximate the minimum $k \in \mathbb{N}$ for which $\delta_{\text{station}}(k, P, Q) \leq \varepsilon$ up to a factor of 2. Recall that $\delta_{\text{station}}(k, P, Q) \leq \varepsilon$ if and only if there exist a set $\mathcal{S}$ of k axis-parallel line segments in the free space $F_\varepsilon(P, Q)$ such that $\mathsf{proj}_x(\bigcup \mathcal{S}) = \mathsf{proj}_x(F_\varepsilon(P, Q))$, $\mathsf{proj}_y(\bigcup \mathcal{S}) = \mathsf{proj}_y(F_\varepsilon(P, Q))$, and the projections of the segments onto the two coordinate axes have pairwise disjoint relative interiors.

The condition that the projections of segments in $\mathcal{S}$ are interior-disjoint is crucial. Without this condition, the problem would be separable, and we could find an optimal solution efficiently: Let OPT_x be a minimum

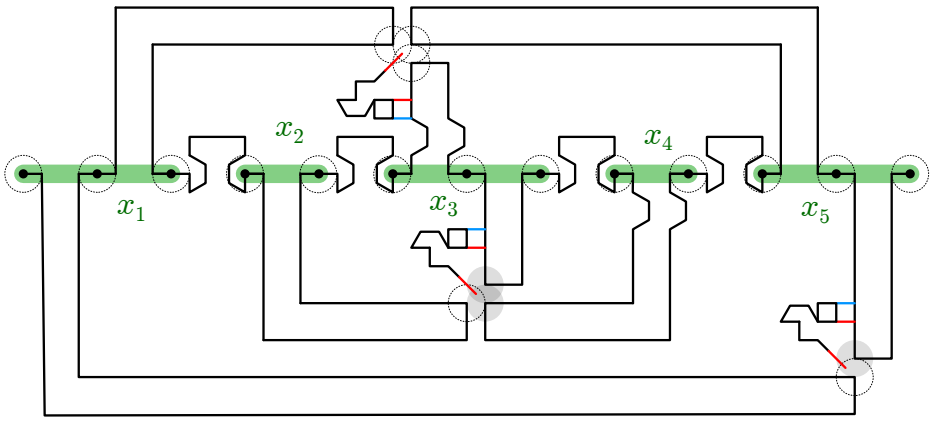


Figure 2: Reduction from the instance $(x_1 \vee x_3 \vee x_5) \wedge (x_1 \vee \overline{x_5}) \wedge (x_2 \vee \overline{x_3} \vee \overline{x_4})$. The segments in the x-axis corresponding to the five variables are shown in green.

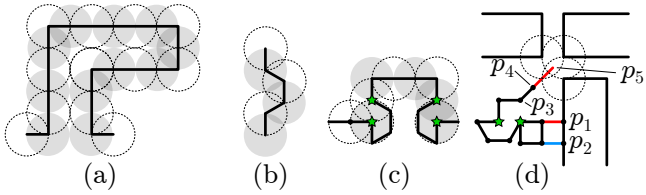


Figure 3: (a) Literal, (b) negation, (c) separation, and (d) clause gadgets.

cardinality set of horizontal segments in $F_\varepsilon(P,Q)$ such that $\mathsf{proj}_x(\bigcup \mathsf{OPT}_x) = \mathsf{proj}_x(F_\varepsilon(P,Q))$, and OPT_y a minimum set of vertical segments in $F_\varepsilon(P,Q)$ such that $\mathsf{proj}_y(\bigcup \mathsf{OPT}_y) = \mathsf{proj}_y(F_\varepsilon(P,Q))$.

Observation 1 *The set $\mathcal{S} = \mathit{OPT}_x \cup \mathit{OPT}_y$ is a minimum set of axis-parallel segments such that $\mathit{proj}_x(\bigcup \mathcal{S}) = \mathit{proj}_x(F_\varepsilon(P,Q))$, and $\mathit{proj}_y(\bigcup \mathcal{S}) = \mathit{proj}_y(F_\varepsilon(P,Q))$.*

Proof. Suppose $\mathsf{OPT}_x \cup \mathsf{OPT}_y$ is not minimal, i.e., there exists a smaller such set $\mathcal{S}'$ of axis-parallel segments whose x- and y-projection equals that of $F_\varepsilon(P,Q)$. Partition $\mathcal{S}'$ into subsets of horizontal and vertical segments, say $\mathcal{S}'_x$ and $\mathcal{S}'_y$. Then $|\mathcal{S}|' < |\mathsf{OPT}_x| + |\mathsf{OPT}_y|$ implies $|\mathcal{S}'_x| < |\mathsf{OPT}_x|$ or $|\mathcal{S}'_y| < |\mathsf{OPT}_y|$, contradicting the minimality of OPT_x or OPT_y. □

Given a set of axis-parallel line segments, we can eliminate intersections between the relative interiors of their x- and y-projections at the expense of increasing the number of segments by a factor of at most 2.

Lemma 8 *There exists a set $\mathcal{S}$ of at most $2(|\mathit{OPT}_x| + |\mathit{OPT}_y|)$ axis-parallel segments in $F_\varepsilon(P,Q)$ such that $\mathit{proj}_x(\bigcup \mathcal{S}) = \mathit{proj}_x(F_\varepsilon(P,Q))$, $\mathit{proj}_y(\bigcup \mathcal{S}) = \mathit{proj}_y(F_\varepsilon(P,Q))$, and the projections of the segments onto the two coordinate axis have pairwise disjoint relative interiors.*

Proof. We may assume, by truncating the segments in OPT_x and OPT_y, if necessary, that the x-projections of segments in OPT_x are interior-disjoint, and the y-projections of segments in OPT_y are also interior-disjoint. Then the supporting line of each horizontal segment in OPT_x intersects the interior of at most one vertical segment in OPT_y, and vice versa. Consequently, the supporting lines of the segments in OPT_x (resp., OPT_y) jointly subdivide the segments in OPT_y (resp., OPT_x) into at most $|\mathsf{OPT}_x| + |\mathsf{OPT}_y|$ pieces. The total number of resulting axis-parallel segments is $2(|\mathsf{OPT}_x| + |\mathsf{OPT}_y|)$, as required. □

It remains to show how to compute OPT_x and OPT_y efficiently. We first observe that a greedy strategy finds OPT_x (resp., OPT_y) from a set of maximal horizontal (resp., vertical) segments in $F_\varepsilon(P,Q)$.

A Greedy Strategy. Input: A set $\mathcal{H}$ of horizontal line segments in $\mathbb{R}^2$. Output: a subset $\mathcal{S} \subset \mathcal{H}$ such that $\mathsf{proj}_x(\bigcup \mathcal{S}) = \mathsf{proj}_x(\bigcup \mathcal{H})$. Initialize $\mathcal{S} := \emptyset$; and let L be a vertical line through the leftmost points in $\bigcup \mathcal{H}$. Let L^- be the closed halfplane on the left of L. While $\mathsf{proj}_x(\bigcup \mathcal{S}) \neq \mathsf{proj}_x(\bigcup \mathcal{H})$, do: Let $s \in \mathcal{H}$ be a segment whose left endpoint is in L^- and whose right endpoint has maximal x-coordinate. Put $\mathcal{S} \leftarrow \mathcal{S} \cup \{s\}$; let $L \leftarrow$the vertical line through the right endpoint of s, and $\mathcal{H} \leftarrow \{h \in \mathcal{H} : h \not\subset L^-\}$.

Observation 2 *Given a set $\mathcal{H}$ of horizontal segments, the above greedy algorithm returns a minimum subset $\mathcal{S} \subset \mathcal{H}$ such that $\mathit{proj}_x(\bigcup \mathcal{S}) = \mathit{proj}_x(\bigcup \mathcal{H})$.*

Proof. At each iteration of the while loop, we maintain the following invariant: $\mathcal{S}$ is a minimal subset of $\mathcal{H}$ such that $\mathsf{proj}_x(\bigcup \mathcal{S}) = \mathsf{proj}_x((\bigcup \mathcal{H}) \cap L^-)$. □

The implementation of the above greedy algorithm is straightforward when $\mathcal{H}$ is finite. However, the set $\mathcal{H}$ of

maximal horizontal segments in the free space $F_\varepsilon(P,Q)$ may be infinite.

Lemma 9 *Let P and Q be polygonal chains with m and n segments, respectively, and let $\varepsilon > 0$. Then a set OPT_x can be computed in output-sensitive $O((|\mathsf{OPT}_x|+m)n)$ time.*

Proof. Let $\mathcal{H}$ be the set of maximal horizontal segments in the free space $F_\varepsilon(P,Q)$. To implement the greedy algorithm above, we describe a data structure that supports the following query: Given a vertical line L, find a segment $s \in \mathcal{H}$ whose left endpoint is in L^- and whose right endpoint has maximal x-coordinate.

Recall from Section 2 that the parameter space $[0,1]^2$ is subdivided into mn axis-parallel cells $C_{i,j}$. In each cell, $C_{i,j} \cap F_\varepsilon(P,Q) = C_{i,j} \cap E_{i,j}$, where $E_{i,j}$ is either an ellipse or a slab parallel to the diagonal of $C_{i,j}$.

Let a vertical line L be given, and assume that it intersects the cells $C_{i,j}$, for $j = 1,\ldots,n$. In each of these n cells, compute the intersections $\ell_{i,j} = L \cap C_{i,j} \cap E_{i,j}$, and the set $R_{i,j}$ of points in $C_{i+1,j} \cap E_{i,j}$ that can be connected to $\ell_{i,j}$ by a horizontal line segment within $C_{i,j} \cap E_{i,j}$. If none of the sets $R_{i,j}$ touches the right edge of the cell $C_{i,j}$, then take a rightmost point r in $\bigcup_{j=1}^{n} R_{i,j}$, and report a maximal horizontal line segment in $F_\varepsilon(P,Q)$ whose right endpoint is r; this takes $O(n)$ time. Otherwise consider the vertical line L' passing through the right edges of the cells $C_{i,j}$ $(j = 1,\ldots,n)$; and let $\ell'_{i,j} = L' \cap R_{i,j}$. We can repeat the above process in cells $C_{i+1,j}$ $(j = 1,\ldots,n)$ with lines $\ell'_{i,j}$ in place of $\ell_{i,j}$. Ultimately, we find a rightmost point $r \in F_\varepsilon(P,Q)$ that can be connected to a point in L within $F_\varepsilon(P,Q)$.

Each query L takes $O(n)$ time if it finds r within a cell $C_{i,j}$ stabbed by L; and $O(nt)$ time if it finds r in a cell $C_{i+t,j}$ for some $t = 1,\ldots,m-i$. Since the x-coordinates of the query lines are strictly increasing, the total running time for $|\mathsf{OPT}_x|$ queries is $O((|\mathsf{OPT}_x|+m)n)$ time, as claimed. □

Theorem 10 *Let P and Q be polygonal chains with m and n segments, respectively, and let $\varepsilon > 0$. Then we can approximate the minimum k such that $\delta_{\text{station}}(k,P,Q) \le \varepsilon$ within a factor of 2 in output-sensitive $O(k(m+n)+mn)$ time.*

Proof. Compute the free space $F_\varepsilon(P,Q)$ in $O(mn)$ time. If $\mathsf{proj}_x(F_\varepsilon(P,Q)) \neq [0,1]$ or $\mathsf{proj}_y(F_\varepsilon(P,Q)) \neq [0,1]$, then report that $\delta_{\text{station}}(k,P,Q) > \varepsilon$ for every $k \in \mathbb{N}$. Otherwise, compute OPT_x and OPT_y by Lemma 9 in $O(mn + |\mathsf{OPT}_x|n + m|\mathsf{OPT}_y|)$ time. We have $k \le |\mathsf{OPT}_x| + |\mathsf{OPT}_y|$ by Observation 1. Lemma 8 yields a set $\mathcal{S}$ of at most $2(|\mathsf{OPT}S_x| + |\mathsf{OPT}S_y|)$ axis-parallel segments in $F_\varepsilon(P,Q)$ such that $\mathsf{proj}_x(\bigcup \mathcal{S}) = \mathsf{proj}_x(F_\varepsilon(P,Q))$, $\mathsf{proj}_y(\bigcup \mathcal{S}) = \mathsf{proj}_y(F_\varepsilon(P,Q))$, and the projections of the segments onto the two coordinate axes have pairwise disjoint relative interiors. In particular, $\delta_{\text{station}}(|\mathcal{S}|,P,Q) \le \varepsilon$, and so $k \le |\mathcal{S}| \le 2k$, as required. The running time of our algorithm is $O(mn + |\mathsf{OPT}_x|n + m|\mathsf{OPT}_y|) \subset O(mn + k(m+n))$. □

5 Conclusion

We have introduced the rock climber distance $\delta_{\text{rock}}(k,P,Q)$ and the k-station distance $\delta_{\text{station}}(k,P,Q)$ between two polygonal chains in the plane. The rock climber distance combines properties of the continuous and discrete Fréchet distance: It corresponds to a coordinated motion of two agents traversing the two chains where only one agent moves at a time. Our results raise several open problems, we present some of them here.

- Can we efficiently approximate $\delta_{\text{station}}(k,P,Q)$ for a given k and given polygonal chains P and Q?
- In Section 4, we described a 2-approximation algorithm for finding the minimum k for which $\delta_{\text{station}}(k,P,Q) \le \varepsilon$. Can the approximation ratio be improved? Does the problem admit a PTAS?

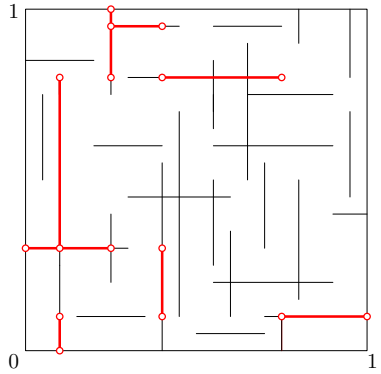


Figure 4: An instance of the compatible axis-parallel segment cover problem. A solution of size 10, is shown in red (bold).

- A discretization of the previous problem leads to the **compatible axis-parallel segment cover problem**: Instead of the free space $F_\varepsilon(P,Q)$, we are given a set $F \subset [0,1]^2$ as a union of n axis-aligned line segments, and ask for the minimum $k \in \mathbb{N}$ such that F contains k axis-parallel line segments whose vertical and horizontal projections, respectively, have pairwise disjoint relative interiors, and jointly cover the unit interval $[0,1]$. See Fig. 4. The conditions on disjoint relative interiors is crucial, and can be formulated as a geometric set cover problem with conflicts [8], or with unique coverage [6, 7]. Our NP-hardness and 2-approximation results extend to this problem. Can the approximation ratio be improved? Is the problem APX-hard?

References

[1] Hugo A. Akitaya, Greg Aloupis, Jeff Erickson, and Csaba D. Tóth. Recognizing weakly simple polygons. *Discrete & Computational Geometry*, 58(4):785–821, 2017. doi:10.1007/s00454-017-9918-3.

[2] Hugo A. Akitaya, Maike Buchin, Leonie Ryvkin, and Jérôme Urhausen. The k-Fréchet distance. *CoRR*, abs/1903.02353, 2019. arXiv:1903.02353.

[3] Hugo A. Akitaya, Leonie Ryvkin, and Csaba D. Tóth. Rock climber distance: Frogs versus dogs. *CoRR*, abs/1906.08141, 2019. arXiv:1906.08141.

[4] Helmut Alt and Michael Godau. Computing the Fréchet distance between two polygonal curves. *Internat. J. Comput. Geom. Appl.*, 5(1-2):75–91, 1995. doi:10.1142/S0218195995000064.

[5] Helmut Alt, Christian Knauer, and Carola Wenk. Comparison of distance measures for planar curves. *Algorithmica*, 38(1):45–58, 2004. doi:10.1007/s00453-003-1042-5.

[6] Pradeesha Ashok, Sudeshna Kolay, Neeldhara Misra, and Saket Saurabh. Unique covering problems with geometric sets. In *Proc. 21st Computing and Combinatorics Conference (COCOON)*, volume 9198 of *LNCS*, pages 548–558, Cham, 2015. Springer. doi:10.1007/978-3-319-21398-9_43.

[7] Pradeesha Ashok, Aniket Basu Roy, and Sathish Govindarajan. Local search strikes again: PTAS for variants of geometric covering and packing. In *Proc. 23rd Computing and Combinatorics Conference (COCOON)*, volume 10392 of *LNCS*, pages 25–37, Cham, 2017. Springer. doi:10.1007/978-3-319-62389-4_3.

[8] Aritra Banik, Fahad Panolan, Venkatesh Raman, Vibha Sahlot, and Saket Saurabh. Parameterized complexity of geometric covering problems having conflicts. In *Proc. 15th Workshop on Algorithms and Data Structures (WADS)*, volume 10389 of *LNCS*, pages 61–72, Cham, 2017. Springer. doi:10.1007/978-3-319-62127-2_6.

[9] Manjanna Basappa. Line segment disk cover. In *Proc. 4th Confference on Algorithms and Discrete Applied Mathematics (CALDAM)*, volume 10743 of *LNCS*, pages 81–92, Cham, 2018. Springer. doi:10.1007/978-3-319-74180-2_7.

[10] Manjanna Basappa, Rashmisnata Acharyya, and Gautam K. Das. Unit disk cover problem in 2D. *J. Discrete Algorithms*, 33:193–201, 2015. doi:10.1016/j.jda.2015.05.002.

[11] Ahmad Biniaz, Paul Liu, Anil Maheshwari, and Michiel H. M. Smid. Approximation algorithms for the unit disk cover problem in 2D and 3D. *Comput. Geom.*, 60:8–18, 2017. doi:10.1016/j.comgeo.2016.04.002.

[12] Karl Bringmann. Why walking the dog takes time: Fréchet distance has no strongly subquadratic algorithms unless SETH fails. In *Proc. 55th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 661–670, 2014. doi:10.1109/FOCS.2014.76.

[13] Kevin Buchin, Maike Buchin, Christian Knauer, Günther Rote, and Carola Wenk. How difficult is it to walk the dog? In *Abstracts of 23rd Europ. Workshop on Comput. Geom. (EuroCG)*, pages 170–173, Graz, 2007.

[14] Kevin Buchin, Maike Buchin, Wouter Meulemans, and Wolfgang Mulzer. Four Soviets walk the dog: Improved bounds for computing the Fréchet distance. *Discrete & Computational Geometry*, 58(1):180–216, 2017. doi:10.1007/s00454-017-9878-7.

[15] Kevin Buchin, Maike Buchin, and Yusu Wang. Exact algorithms for partial curve matching via the Fréchet distance. In *Proc. 20th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 645–654. SIAM, 2009. URL: http://dl.acm.org/citation.cfm?id=1496770.1496841.

[16] Kevin Buchin, Tim Ophelders, and Bettina Speckmann. SETH says: Weak Fréchet distance is faster, but only if it is continuous and in one dimension. In *Proc. 30th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2887–2901, 2019. doi:10.1137/1.9781611975482.179.

[17] Maike Buchin. *On the Computability of the Fréchet Distance Between Triangulated Surfaces*. PhD thesis, Free University, Berlin, 2007. URL: http://www.diss.fu-berlin.de/diss/receive/FUDISS_thesis_000000002618.

[18] Maike Buchin and Leonie Ryvkin. The k-Fréchet distance of polygonal curves. In *Abstracts of 34th Europ. Workshop on Computational Geometry (EuroCG)*, Berlin, 2018. Article 43. URL: conference.imp.fu-berlin.de/eurocg18/.

[19] Hsien-Chih Chang, Jeff Erickson, and Chao Xu. Detecting weakly simple polygons. In *Proc. 26th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1655–1670, 2015. doi:10.1137/1.9781611973730.110.

[20] Anne Driemel, Jeff M. Phillips, and Ioannis Psarros. The VC dimension of metric balls under Fréchet and Hausdorff distances. In *Proc. 35th Symposium on Computational Geometry (SoCG)*, volume 129 of *LIPIcs*, pages 28:1–28:16. Schloss Dagstuhl, 2019. doi:10.4230/LIPIcs.SoCG.2019.28.

[21] Robert J. Fowler, Mike Paterson, and Steven L. Tanimoto. Optimal packing and covering in the plane are NP-complete. *Inf. Process. Lett.*, 12(3):133–137, 1981. doi:10.1016/0020-0190(81)90111-3.

[22] Robert Fraser and Alejandro López-Ortiz. The within-strip discrete unit disk cover problem. *Theor. Comput. Sci.*, 674:99–115, 2017. doi:10.1016/j.tcs.2017.01.030.

[23] Jacob E. Goodman, János Pach, and Chee-K. Yap. Mountain climbing, ladder moving, and the ring-width of a polygon. *Amer. Math. Monthly*, 96(6):494–510, 1989.

[24] Sariel Har-Peled. Fréchet distance: How to walk your dog. In *Geometric Approximation Algorithms*, chapter 30. 2014. URL: https://sarielhp.org/book/.

[25] Sariel Har-Peled and Benjamin Raichel. The Fréchet distance revisited and extended. *ACM Trans. Algorithms*, 10(1):3:1–3:22, 2014. doi:10.1145/2532646.

[26] Donald E Knuth and Arvind Raghunathan. The problem of compatible representatives. *SIAM Journal on Discrete Mathematics*, 5(3):422–427, 1992. doi:10.1137/0405033.

[27] Joseph S. B. Mitchell, Valentin Polishchuk, and Mikko Sysikaski. Minimum-link paths revisited. *Comput. Geom.*, 47(6):651–667, 2014. doi:10.1016/j.comgeo.2013.12.005.

[28] Joseph S. B. Mitchell, Valentin Polishchuk, Mikko Sysikaski, and Haitao Wang. An optimal algorithm for minimum-link rectilinear paths in triangulated rectilinear domains. *Algorithmica*, 81(1):289–316, 2019. doi:10.1007/s00453-018-0446-1.

[29] Nabil H. Mustafa and Saurabh Ray. Improved results on geometric hitting set problems. *Discrete & Computational Geometry*, 44(4):883–895, 2010. doi:10.1007/s00454-010-9285-9.

[30] Aniket Basu Roy, Sathish Govindarajan, Rajiv Raman, and Saurabh Ray. Packing and covering with non-piercing regions. *Discrete & Computational Geometry*, 60(2):471–492, 2018. doi:10.1007/s00454-018-9983-2.

[31] Christian Scheffer. More flexible curve matching via the partial Fréchet similarity. *Int. J. Comput. Geom. Appl.*, 26:33–52, 2016. doi:10.1142/S0218195916500023.

[32] René van Oostrum and Remco C. Veltkamp. Parametric search made practical. *Comput. Geom.*, 28(2-3):75–88, 2004. doi:10.1016/j.comgeo.2004.03.006.

Discrete Planar Map Matching

Bin Fu* Robert Schweller*† Tim Wylie*†

Abstract

Route reconstruction is an important application for Geographic Information Systems (GIS) that rely heavily upon GPS data and other location data from IoT devices. Many of these techniques rely on geometric methods involving the Fréchet distance to compare curve similarity. The goal of reconstruction, or map matching, is to find the most similar path within a given graph to a given input curve, which is often approximate location data. This process can be approximated by sampling the curves and using the discrete Fréchet distance. Due to power and coverage constraints, the GPS data itself may be sparse causing improper constraints along the edges during the reconstruction if only the continuous Fréchet distance is used. Here, we look at two variations of discrete map matching: one constraining the walk length and the other limiting the number of vertices visited in the graph. We give an efficient algorithm to solve the question based on walk length showing it is in **P**. We prove the other problem is **NP**-complete and the minimization variant is **APX**-hard while also giving a parameterized algorithm to solve the problem.

1 Introduction

There are many important applications related to GIS systems due to the proliferation of GPS enabled devices and the continued development of IoT devices. Route reconstruction is the process of finding the most likely path of an object based on the GPS data and the possible pathways. For instance, GPS data may indicate a car was driving through buildings, and we want to fit the data to the road network to recreate the most likely path of the car.

Route reconstruction depends greatly on what metric is used to determine how close the reconstructed path is. The two main methodologies are those based on geometric methods and Global Weight Optimization. However, the methodologies can also be classified based on the problem definition where we have local/incremental methods, global methods, and statistical methods. These can be extended to include topological and geological conditions, current weather and traffic conditions, speed limits, and other variables that can produce more optimal routes [17, 23]. Here, we focus on global geometric methods, and assume we have all of the data as input to find an optimal solution.

One popular means of measuring this fit is the Fréchet distance. Finding a path in a graph given a polygonal curve is also referred to as map matching. Map matching with respect to the Fréchet distance was first posed by Alt et. al. [6] as follows: Let $G = (V, E)$ be an undirected connected planar graph with a given straight-line embedding in $\mathbb{R}^2$ and a polygonal line P. Find a path Q in G which minimizes the Fréchet distance between P and Q. They give an efficient algorithm which runs in $\mathcal{O}(pq \log q)$ time and $\mathcal{O}(pq)$ space where p is the number of line segments of P and q is the complexity of G. This allows for vertices and edges to be traversed multiple times. Maheshwari et al. improved the running time for the map matching problem for complete graphs [19]. The original algorithm decides it in $\mathcal{O}(pn^2 \log n)$, where n is the number of vertices in the graph, and their new algorithm solves it in $\mathcal{O}(pn^2)$. We refer to this problem (in a complete graph) as the set-chain matching problem, which was studied in more detail in [1, 2, 3, 25].

There has been work that yields better performance with certain types of curves, with dual simplification for an approximate result, with bounded simplification of one of the chains, and in graphs with certain properties, [8, 11, 12, 14]. With map matching, for the weak Fréchet distance, the bounds have been lowered further to $\mathcal{O}(pq)$ [13], and the problems can be defined with a smaller error bound [24].

All this work has focused on the continuous Fréchet distance, which assumes that every point along the curve is meaningful. In reality, all GPS data is discrete, and these approaches smooth the data. There are some methods optimized for low-sampling-rate data [17], but even these assume some maximum time between samples (less than five minutes). Our goal is to analyze data where samples may be hours apart and can not be reasonably smoothed. There are many instances where you may not have GPS data, such as power constraints (low battery), or coverage issues (no towers), or required disconnects (airline travel). In these cases, you only connect to a cell tower or satellite intermittently, and thus the resulting polygonal curve is only meaningful at the nodes.

Another application where map matching algorithms are useful are discretizations of any continuous data.

*Department of Computer Science, University of Texas - Rio Grande Valley

†This author's research was supported in part by National Science Foundation Grant CCF-1817602.

Examples include cartography applications, schematic maps, or polygon simplification [7, 16, 21].

Our Results. We introduce two additional variants of discrete map matching. We show that minimizing the resulting path is polynomial. We then show that restricting the set of vertices used from the graph is **NP**-complete and the minimization variant is **APX**-hard. We give a positive result based on a separator that yields a polynomial time algorithm under more realistic input assumptions (similar to those in [7]).

2 Preliminaries

The discrete Fréchet distance was originally defined by Eiter and Mannila in 1994 [15], and was further expanded on theoretically by Mosig et al. in 2005 [22].

Given two polygonal curves, we define the discrete Fréchet distance as follows. We use $d(a, b)$ to represent the Euclidean distance between two points a and b, but it could be replaced with other distance measures depending on the application.

Definition 1 *The discrete Fréchet distance, d_F, between two polygonal curves $f : [0, m] \to \mathbb{R}^k$ and $g : [0, n] \to \mathbb{R}^k$ is defined as:*

$$d_F(f, g) = \min_{\substack{\sigma:[1:m+n]\to[0:m],\\ \beta:[1:m+n]\to[0:n]}} \max_{s\in[1:m+n]} \left\{ d\Big(f(\sigma(s)),\, g(\beta(s))\Big) \right\}$$

where σ and β range over all discrete non-decreasing onto mappings of the form $\sigma : [1 : m + n] \to [0 : m], \beta : [1 : m + n] \to [0 : n]$.

The continuous Fréchet distance is typically explained as the relationship between a person and a dog connected by a leash walking along the two curves and trying to keep the leash as short as possible. However, for the discrete case, we only consider the nodes of these curves, and thus the man and dog must "hop" along the nodes. Figure 1 shows this relationship between the two and how with enough evenly sampled points on the two curves, the resulting discrete Fréchet distance can closely approximate the continuous Fréchet distances.

With a dynamic programming solution for finding the discrete Fréchet distance between two polygonal curves with m and n nodes, Eiter and Mannila proved that $\mathcal{O}(mn)$ was possible [15]. Recently, a slightly subquadratic algorithm was discovered by Agarwal et al. showing the discrete Fréchet distance can be computed in $\mathcal{O}(\frac{mn \log\log n}{\log n})$ time [4].

Bringmann and Mulzer [10] recently showed the there is no strongly subquadratic algorithm for the discrete Fréchet distance unless the strong exponential time hypothesis (SETH) fails [9].

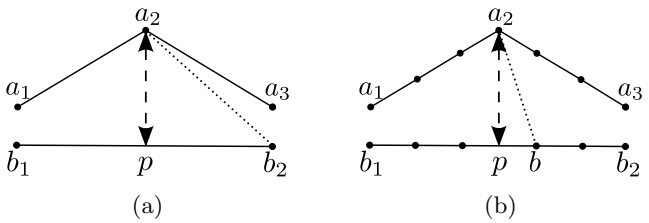


Figure 1: Figures (a) and (b) show the relationship between the discrete and continuous Fréchet distance where p is the point on the line closest to a_2 for the continuous and the dotted line represents the closest discrete distance from a_2 (using only nodes). (a) the curves have fewer nodes and a larger discrete Fréchet distance, while (b) has the same paths with more nodes, and thus provides a better approximation.

2.1 Discrete Map Matching

The definition of discrete map matching follows and we discuss two variants that we consider in this work.

Definition 2 (Discrete Map Matching)
Instance: *Given a simple connected planar graph $G = (V, E)$ embedded in $\mathbb{R}^2$, a polygonal curve P in $\mathbb{R}^d$ ($d \geq 2$), an integer $K \in \mathbb{Z}^+$, and an $\varepsilon > 0$.*
Problem: *Does there exist a walk Q in G with vertices chosen from V' where $V' \subseteq V$, such that $T \leq K$ and $d_F(P, Q) \leq \varepsilon$ where T is defined as either*

- *$T = |Q|$, where the size of the chain is being restricted, or*
- *$T = |V'|$, where the number of vertices in the graph is restricted (a vertex visited multiple times in the walk only counts as one).*

We look at the analogous variants of the set-chain matching problems (rather than a graph they find a path through a set of points) [1, 2, 3, 19, 25]: the Non-unique map matching problem constrains either the curve (NMMC-k) or the set of vertices used (NMMS-k). There is a variant where the vertices in the walk must also form a path, which is Unique map matching (UMM-k), and was shown to be **NP**-complete simultaneously in [20, 26] with extended stronger results in [16]. Note that when the vertices are unique the two decision problems ($|Q|$,$|V'|$) are equivalent. For reference, the naming convention is (U)nique/(N)on-unique (M)ap (M)atching with a k (S)ubset/(C)hain.

3 Non-unique Map Matching With Restricted Length (NMMC)

Here, we discuss discrete map matching concerned with the length of the path through the graph. As the NMMC problem restricts the length of Q, the problem is similar to the set-chain variant (NSMC) [25] and has

a similar optimal substructure. The recurrence to find the minimum size of Q (in number of vertices), is given in Equation 1. The recurrence uses a 2D table M of size $|V| \times |P|$ where the first column is initialized to one if $d(v_k, p_1) \leq \varepsilon$ where $1 \leq k \leq |V|$, and the values are set to ∞ otherwise. $N(v)$ stands for the neighborhood of vertex v, which is the set of adjacent vertices in G.

The recurrence minimizes the number of vertices used while going from p_1 to $p_{|P|}$. This is done by ensuring that for each p_i, $1 \leq i \leq |P|$, we mark all vertices v with $d(v, p_i) \leq \varepsilon$ and that v is adjacent to at least one vertex used in the walk so far, i.e., there is a v' where $d(v', p_{i-1}) \leq \varepsilon$ and there is an edge between v and v'.

$$M[i,j] = \min \begin{cases} M[i,j\text{-}1], & \text{if } d(v_i,p_j) \leq \varepsilon, M[i,j\text{-}1] \neq \infty \\ \min\limits_{v_k \in N(v_i)} M[k,j\text{-}1]+1, & \text{if } d(v_i,p_j) \leq \varepsilon \\ \infty, & \text{if } d(v_i,p_j) > \varepsilon \end{cases} \tag{1}$$

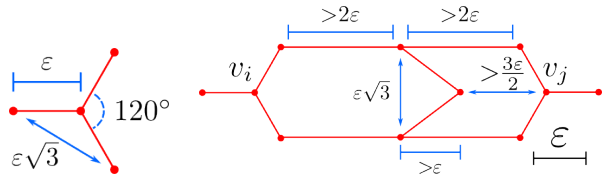
This algorithm works for any graph with the worst case being a complete graph, which is equivalent to the discrete set-chain matching variant [25] and has complexity $\mathcal{O}(|P|(|V|+|E|))$. Each vertex v only looks at its neighbor set, $N(v)$, and since a planar graph has fewer edges, the algorithm has a faster runtime. A planar graph has $|E| = 3(|V|-2) = \mathcal{O}(|V|)$, yielding a run time of $\mathcal{O}(|P||V|)$ for planar graphs.

Theorem 1 *The discrete Non-unique Map Matching (NMMC) problem restricting the number of nodes in the output polygonal curve Q (vertices in the walk) can be solved in $\mathcal{O}(|P|(|V|+|E|))$ time for general graphs, and $\mathcal{O}(|P||V|)$ time for planar graphs.*

The optimal walk can be extracted by a simple backtracking algorithm. Find the minimum value in the last column, and the index of that row is the last vertex of the walk. Then, continually look at the previous column and find either the same row (same value), or look at all neighbors of that vertex and find a row with a value that is one less than the current value.

4 Discrete Non-unique Map Matching with Restricted Set (NMMS)

Discrete map matching concerned with restricting the number of vertices of the graph that the walk uses is an interesting problem related to coverage. Imagine a route reconstruction problem looking at cellphone tower coverage. If we wanted to know whether it was possible that the driver connected to fewer than k towers, this is equivalent to NMMS. On a complete graph, this is the same as discrete unit disk cover [25], and thus NMMS is asking a DUDC question related to planar connectivity



(a) Vertex Gadget (b) Connecting Edge

Figure 2: (a) The vertex gadget replaces a vertex with 4 vertices and three connecting locations. (b) Edge Connecting two vertex gadgets. Note that there is no restriction requiring the edges to be straight, just that the distance bounds are maintained.

between the disks. Section 4.1 shows NMMS is **NP**-complete, and Section 4.2 shows the minimization variant is **APX**-hard. We give a polynomial result related to real-world application constraints in Section 4.3.

4.1 Reduction Overview

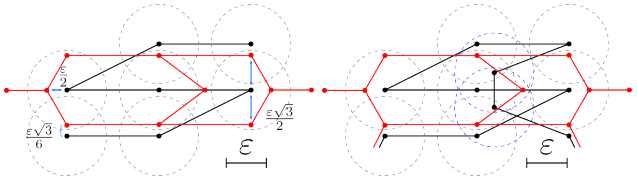
NMMS on a complete graph is equivalent to Non-Unique Set Matching with a fixed set for some k (NSMS), which is **NP**-complete [25]. On a planar graph the problem is different since our walk through the graph is limited by its neighbors and the planarity of the graph. We show that the problem is **NP**-complete via a reduction from Planar Vertex Cover with max degree three (PVC3), which was shown to be **NP**-complete in [18] and shown to be **APX**-hard in [5]. For Planar Vertex Cover we are given a planar graph $G = (V, E)$ and an integer K_{vc} as input. For this special case we know that $deg(v) \leq 3\ \forall\ v \in V$. We want to know if there is a vertex cover of G of size at most K_{vc}.

We use several gadgets to transform an instance of PVC3 into an instance of NMMS. Since we are moving from a graph to a geometry problem, we use a planar embedding of the graph. Let G_s be a planar embedding of the graph G where each edge has length greater than 5ε. This ensures our geometric gadgets work correctly for any given $\varepsilon > 0$.

The reduction is going to make a new graph and a polygonal curve that will visit each edge of G exactly twice by creating a doubly-connected edge list (Section 4.1.2). The gadgets that replace each vertex and edge of G (Section 4.1.1) ensure that this walk is possible.

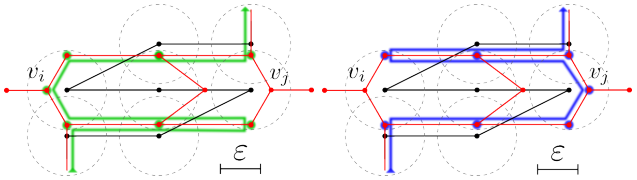
4.1.1 Gadgets

Each vertex in G_s is replaced with the vertex gadget shown in Figure 2a, which consists of four vertices and three edges connecting them. The edge lengths are exactly ε in length. The central vertex represents the original vertex while the other three will be used to connect edges. We replace each edge by three additional vertices and six edges that connect two vertex gadgets (Figure



(a) Edge Gadget (b) Edge-Cross Gadget

Figure 3: (a) An edge gadget includes the two connected vertex gadgets and the connecting edges between them, and then also has a piece of the polygonal curve. The circles represent an ϵ-ball around each node of the curve to show which vertices in the graph are within ϵ. It begins on one "side" of the edge and ends on the other "side." (b) The edge-cross gadget is an edge gadget that has the polygonal curve ending on the same side of the edge gadget that it began.



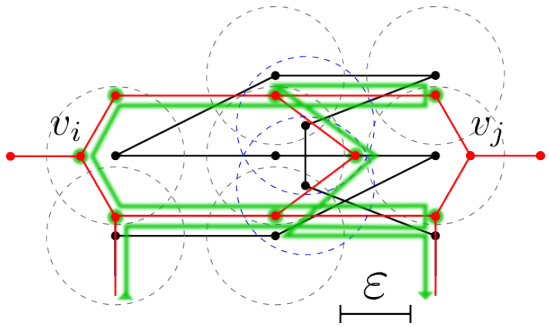
(a) Walk through v_i (b) Walk through v_j

Figure 4: The two possible walks through an edge gadget connecting vertices v_i and v_j. (a) A walk through the edge gadget that uses the center vertex of v_i. (b) A walk through the edge gadget that uses the center vertex of v_j.

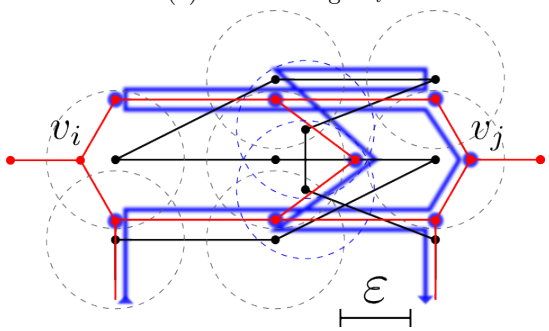
2b). Note that the new edges can be arbitrarily placed in the plane as necessary as long as the relationship between the three center vertices is maintained. Since we know the max degree of any vertex in G_s is three, we can connect up to three edge gadgets to each vertex gadget. We now have a new graph $G'_s = (V', E')$ with exactly $|V'| = 4|V| + 3|E|$, and $|E'| = 3|V| + 6|E|$.

For the full edge gadget, we add a polygonal curve to each edge, as shown in Figure 3a. We discuss connecting these into one curve later; for now we focus on a single edge. If we walk through the graph following the polygonal curve, and minimize the number of vertices used, then there are only two possible walks. Assume we begin at the lower left edge vertex (on the variable gadget). We must either follow the walk shown in Figure 4a or the one in 4b. Both walks use only 7 vertices, and this is minimal for any edge gadget. This means we must use the center vertex representing v_i or v_j for every edge e_{ij}. We could use both, but we only have to use one, and either will work. In this way, we have a vertex "covering" that edge. The same vertex could be used for all three edge gadgets.

The edge gadget ends the walk on the opposite side of the edge from where it started, which can be a problem (explained later). Thus, we finish the edge gadget with a



(a) Walk through v_i



(b) Walk through v_j

Figure 5: The two possible walks through an edge-cross gadget connecting vertices v_i and v_j. (a) A walk through the edge-cross gadget that uses the center vertex of v_i. (b) A walk through the edge-cross gadget that uses the center vertex of v_j.

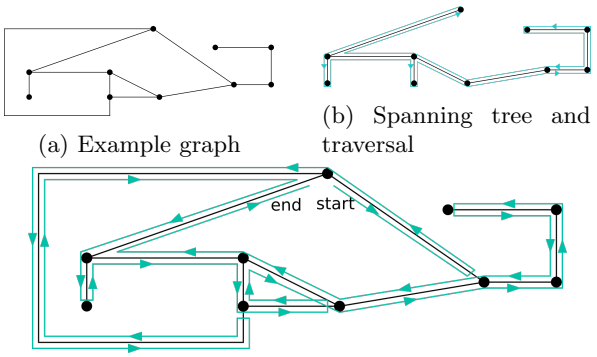
crossover that allows the walk to cross back to the other side of the edge. Figure 3b shows this construction. Figures 5a and 5b show the two possible walks through the edge. Thus, every edge gadget needs a minimum of 8 vertices for a walk following the curve.

4.1.2 Connecting Gadgets

To connect all of the small polygonal curves into a single curve, we need a walk that traverses every edge in our original graph at least once, and it may traverse an edge multiple times. The edge-traversal algorithm (Algorithm 1) ensures we go through every edge exactly twice (an example is shown in Figure 6). This generates a doubly connected edge-list (DCEL).

Algorithm 1 (Generate DCEL)
Input: Graph $G = (V, E)$
Output: Sequence of edges

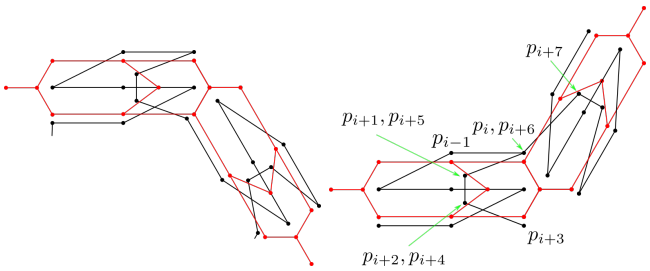
- *Compute a minimum spanning tree M of G and pick a vertex v.*
- *Create a path P around M by visiting each edge twice with the path in a counter-clockwise manner.*
- *At each vertex $v_i \in P$, add any edge $e_{\{v_i,v_j\}}$ twice to the edge sequence if it is not part of P.*



(a) Example graph (b) Spanning tree and traversal

(c) Final edge traversal

Figure 6: (a) An example planar graph with max vertex degree three. (b) A random MST with a traversal going through all vertices and going through all edges of the MST twice. (c) Adding in all other edges by following the path and inserting the missing edges into the path at each vertex.



(a) Right Turn (b) Left Turn

Figure 7: Assuming we are coming from the left-most vertex gadget. (a) Two edge-cross gadgets connected by a vertex gadget turning right. (b) Two edge-cross gadgets connect its left and thus making a left turn. After we cross the first edge, we must cross back to the other side in order to turn left, so those three nodes in the curve are repeated in reverse order: $\langle p_1, \ldots, p_i, p_{i+1}, \ldots, p_{i+6}, \ldots p_F \rangle$ s.t. the locations of p_i and p_{i+6}, p_{i+1} and p_{i+5}, and p_{i+2} and p_{i+4} are the same.

With this walk, connect the polygonal curve segments and adjust each segment whenever we cross the same edge twice (the number of times we will visit the same edge) or we need to cross to the other side of an edge (to finish on the opposite side). With the crossovers, Figure 7 shows how to make right and left turns.

4.1.3 Complexity

The new graph has $|V'| = 4|V| + 3|E|$, and the number of edges $|E'| = 3|V| + 6|E|$ with an optimal walk using $K = 7|E| - |d_2| - 3|d_3| + K_{vc}$ where $d_i = \{v_j | v_j \in V, deg(v_j) = i\}$, i.e., d_i is the set of all vertices from V with degree i. Since adjacent edges on the vertex gadget share vertices, we must subtract these for vertices of degree 2 and 3 in the original graph.

Theorem 2 *Discrete non-unique map matching with $T = |V'|$ (NMMS) is **NP**-complete.*

Proof. The graph G has a vertex cover of size K_{vc} if and only if there exists a walk Q in G'_s with P such that Q only passes through $K = 7|E| - |d_2| - 3|d_3| + K_{vc}$ vertices and $d_F(P, Q) \leq \varepsilon$.

Given the graph G, the given construction allows a walk to pass through an edge gadget with a minumum of 7 vertices. Since six of these must be used for every edge giving $7|E|-|d_2|-3|d_3|$ vertices, the crossing vertex comes from one of the two vertex gadgets, but either can be used. Thus, every edge must have one of the two center vertex gadget vertices, which can be used for the adjacent edges as well. Thus, it is a vertex cover equivalent meaning K_{vc} vertices are sufficient.

Given an instance of G'_s and P. A walk must pass through $7|E|-|d_2|-3|d_3|$ vertices. The additional vertex needed for each edge-cross gadget constitutes a single vertex associated with an adjacent edge in G. Thus, it would be a vertex cover.

For membership in NP, if given a set V' of vertices, take the induced subgraph of G of the vertices from V' and let it be G'. The problem then becomes equivalent to NMMC over G' with $k = |P|$, which is polynomial. Finally, we check the discrete Fréchet distance in polynomial time. □

4.2 APX-hardness

Here we show that the minimization variant of the problem is **APX**-hard with an L-reduction from PVC3. An L-reduction is an approximation-preserving reduction when both problems are minimization problems.

Definition 3 (L-reduction) *Let A and B be optimization problems and c_A and c_B their respective cost functions. A pair of functions f and g is an L-reduction if all of the following conditions are met:*

- *f and g are computable in polynomial time,*
- *if x is an instance of problem A, then $f(x)$ is an instance of problem B,*
- *if y' is a solution to $f(x)$, then $g(y')$ is to x,*
- *there exists a positive constant α such that $\mathrm{OPT_B}(f(x)) \leq \alpha \mathrm{OPT_A}(x)$,*
- *there exists a positive constant β such that for every solution y' to $f(x)$, $|\mathrm{OPT_A}(x) - c_A(g(y'))| \leq \beta |\mathrm{OPT_B}(f(x)) - c_B(y')|$.*

Theorem 3 *Minumum discrete non-unique map matching with $T = |V'|$ (Min NMMS) is **APX**-hard.*

Proof. We show this via an L-reduction. Let f be the L-reduction from PVC3 to NMMS using the described construction above. For every vertex cover V_c of size

K_{vc} of graph $G = (V, E)$ for PVC3, there is a vertex walk Q in our new graph $G'_s = (V', E')$ using $K' = 7|E| - |d_2| - 3|d_3| + K_{vc}$ vertices such that $d_F(P, Q) \leq \varepsilon$.

We construct our walk Q as a sequence of vertices from G'_s. Let $\{Q\}$ denote the vertices in the walk, i.e., $|\{Q\}| = K'$. Then for every vertex cover $V_c \subset V$ of G we construct the walk Q with $\{Q\} \subset V'$ of $G'_s = f(G)$ of size K'. Since G has bounded degree 3, it is clear that $3K_{vc} \geq \sum_{v \in V_{vc}} deg(v) \geq |E| \geq |V|$. We can see that $K' = 7|E| - |d_2| - 3|d_3| + K_{vc} \leq 7|E| + K_{vc} \leq 22K_{vc}$. We can replace $|E|$ with $3K_{vc}$. Thus, the first property of an L-reduction is satisfied with $\alpha = 22$. $K' \leq 22K_{vc}$.

Conversely, given a walk Q with $\{Q\} \subset V'$ of $G'_s = f(G)$ of size K', we transform it back into a vertex cover $V_{vc} \subset V$ of size K_{vc} of graph G as follows. We look at each variable gadget (any subgraph with one vertex with exactly 3 adjacent vertices of distance ε in the embedding as shown in Figure 2a). For every one of these central vertices, if the vertex is included in the walk Q, then we include it in V_{vc}. Observe this will give a vertex cover for G since the walk must include either v_i or v_j from the two variable gadgets on either side of an edge gadget. Note that $K_{vc} \leq K' - 7|E| + |d_2| + 3|d_3|$. Given any walk in the graph requires all the vertices except those that would be part of a vertex cover, we get that f is an L-reduction with $\beta = 1$. □

4.3 Positive Result Based on a Separator

Here, we develop an FPT algorithm for the NMMS problem based on a divide and conquer approach with a graph separator, which is a set of vertices that, if removed, disconnect the graph. Let $\text{ball}_d(p, \epsilon)$ be the ball in $\mathbb{R}^d$ with center $p \in \mathbb{R}^d$ and radius ϵ.[1]

Definition 4 ((ϵ, c)-local property) *A polygonal curve $P = \langle p_1, p_2, \ldots, p_n \rangle$ on a plane satisfies the (ϵ, c)-local property if for every p_i, the circle with center at p_i and radius ϵ does not contain any p_j with $|i - j| > c$.*

Definition 5 ((ϵ, c, u)-local property) *An input polygonal curve $P = \langle p_1, p_2, \ldots, p_n \rangle$ and planar graph $G = (V, E)$ satisfy the (ϵ, c, u)-local property if*

1. *The curve P satisfies the (ϵ, c)-local property, and*
2. *For every point $p \in \mathbb{R}^d$, the ball with center at p and radius ϵ contains at most u points in V.*

Theorem 4 *Assume that c and u are integer parameters. There is a $\mathcal{O}(n^{1+u \log c})$ time algorithm for the Discrete Map Matching with restricted set problem (NMMS) when the input $P = \langle p_1, p_2, \ldots, p_n \rangle$ and $G = (V, E)$ satisfy the (ϵ, c, u)-local property.*

[1] Given we are looking at planar embeddings, if we restrict the polygonal curve to be in $\mathbb{R}^2$, it is equivalent to look at the circle with radius ϵ centered at $p \in \mathbb{R}^2$.

Proof. Consider the case that the input is a polygonal curve satisfying the (ϵ, c)-local property, and an arbitrary graph. Let $S = \{p_{\lfloor \frac{n}{2} \rfloor + i} : i \in [0, c-1]\}$.

By brute force, iterate over all possible matchings between the c points in S, and at most u possible points in $V \cap \text{ball}(p_i, \epsilon)$. The number of possible matchings between the points in S and the points in V is at most c^u. For each such matching, we recurse on the two separated portions of P induced by the respective separator. This yields a divide and conquer algorithm that has time complexity $T(n) = 2c^u T(\frac{n}{2})$. Solving this recurrence equations yields a time complexity of $\mathcal{O}(n(c^{u \log n})) = \mathcal{O}(n^{1+u \log c})$. □

Corollary 1 *Assume that c and ϵ are fixed. Then there is a polynomial time algorithm if input P is a polygonal curve satisfying the (ϵ, c)-local property, and input G is a grid graph on the plane.*

Proof. Suppose the input graph is a grid graph and ϵ is fixed. Each vertex v_i in the curve P can only select at most $u = \pi(\epsilon + \sqrt{2}/2)^2$ grid points with distance ϵ to match. The result follows from Theorem 4. □

5 Conclusion and Future Work

In this work we introduce variants of discrete map matching and show that given different constraints the problem is tractable or **APX**-hard. When the length of the walk is restricted, we show the problem is decidable in $\mathcal{O}(|P||V|)$ time. For the variant restricting the number of vertices usable in the graph, after proving it is **APX**-hard, we give an FPT algorithm based on the structure of P and its relationship to the graph. Given the application of route reconstruction, this is a reasonable practical constraint. Our work leads to many open questions such as FPT algorithms based on different parameters of the input curve or graph beyond ours, and possible approximation algorithms. Is there a good constant factor approximation for minimum NMMS?

Another direction of research is to extend and generalize discrete map matching. Our problem definitions ignore all nodes in the graph outside the reach of the polygonal curve. To fully realize route reconstruction on large graphs, the problem should only ensure that at least one node is visited within the range of each node of the curve, but also accounts for and attempts to minimize paths through vertices that are not in range. This is similar to TSP with neighborhoods, except that the order of the neighborhoods is given. This is also similar to Group Steiner Tree and facility location problems.

Finally, what are the problem complexities under the continuous Fréchet distance, and are there better approximations or algorithms?

References

[1] P. Accisano and A. Üngör. Hardness results on curve/point set matching with fréchet distance. *CoRR*, abs/1211.2030, 2012.

[2] P. Accisano and A. Üngör. Approximate matching of curves to point sets. In *Proceedings of the 26th Canadian Conference on Computational Geometry*, CCCG'14, 2014.

[3] P. Accisano and A. Üngör. Finding a curve in a point set. *CoRR*, abs/1405.0762, 2014.

[4] P. K. Agarwal, R. B. Avraham, H. Kaplan, and M. Sharir. Computing the discrete fréchet distance in subquadratic time. *SIAM J. Comput.*, 43(2):429–449, 2014.

[5] P. Alimonti and V. Kann. Some apx-completeness results for cubic graphs. *Theoretical Computer Science*, 237(1):123 – 134, 2000.

[6] H. Alt, A. Efrat, G. Rote, and C. Wenk. Matching planar maps. *Journal of Algorithms*, 49(2):262–283, Nov 2003.

[7] Q. W. Bouts, I. I. Kostitsyna, M. van Kreveld, W. Meulemans, W. Sonke, and K. Verbeek. Mapping Polygons to the Grid with Small Hausdorff and Fréchet Distance. In *Proceedings of the 24th Annual European Symposium on Algorithms*, volume 57 of *ESA 2016*, pages 22:1–22:16, 2016.

[8] S. Brakatsoulas, D. Pfoser, R. Salas, and C. Wenk. On map-matching vehicle tracking data. In *Proc. of the 31st Int. Conf. on Very Large Data Bases*, VLDB'05, pages 853–864. VLDB Endowment, 2005.

[9] K. Bringmann. Why walking the dog takes time: Frechet distance has no strongly subquadratic algorithms unless SETH fails. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 661–670, 2014.

[10] K. Bringmann and W. Mulzer. Approximability of the discrete fréchet distance. *JoCG*, 7(2):46–76, 2016.

[11] K. Buchin, M. Buchin, and Y. Wang. Exact algorithms for partial curve matching via the Fréchet distance. In *Proc. of the 20th Annual ACM-SIAM Sym. on Discrete Algorithms*, SODA'09, pages 645–654, 2009.

[12] D. Chen, A. Driemel, L. J. Guibas, A. Nguyen, and C. Wenk. Approximate map matching with respect to the Fréchet distance. In *Proc. of the 13th Workshop on Algorithm Engineering and Experiments*, ALENEX'11, pages 75–83. SIAM, 2011.

[13] D. Chen, L. J. Guibas, Q. Huang, and J. Sun. A faster algorithm for matching planar maps under the weak Fréchet distance. Unpublished, December 2008.

[14] A. Driemel, S. Har-Peled, and C. Wenk. Approximating the Fréchet distance for realistic curves in near linear time. *Discrete & Comp. Geom.*, 48(1):94–127, 2012.

[15] T. Eiter and H. Mannila. Computing discrete Fréchet distance. Technical Report CD-TR 94/64, Information Systems Dept., Technical University of Vienna, 1994.

[16] M. Löffler and W. Meulemans. Discretized approaches to schematization. In *Proceedings of the 29th Canadian Conference on Computational Geometry*, CCCG'17, pages 220–225, 2017.

[17] Y. Lou, C. Zhang, Y. Zheng, X. Xie, W. Wang, and Y. Huang. Map-matching for low-sampling-rate gps trajectories. In *Proc. of the 17th ACM SIGSPATIAL Int. Conf. on Advances in G.I.S.*, GIS'09, pages 352–361, New York, NY, 2009. ACM.

[18] D. S. J. M. R. Garey. The rectilinear steiner tree problem is np-complete. *SIAM Journal on Applied Mathematics*, 32(4):826–834, 1977.

[19] A. Maheshwari, J.-R. Sack, K. Shahbaz, and H. Zarrabi-Zadeh. Staying close to a curve. In *Proc. of the 23rd Annual Canadian Conf. on Computational Geometry*, CCCG'11, 2011. August 10-12, 2011.

[20] W. Meulemans. Map matching with simplicity constraints. *CoRR*, abs/1306.2827, 2013.

[21] W. Meulemans. Discretized approaches to schematization. *CoRR*, abs/1606.06488, 2016.

[22] A. Mosig and M. Clausen. Approximately matching polygonal curves with respect to the Fréchet distance. *Comp. Geom.: Theory and Appl.*, 30(2):113–127, Feb 2005.

[23] H. Wei, Y. Wang, G. Forman, and Y. Zhu. Map matching by Fréchet distance and global weight optimization. Technical Report SJTU_CS_TR_201302001, Department of Computer Science, Shanghai Jiao Tong University, 2013.

[24] C. Wenk, R. Salas, and D. Pfoser. Addressing the need for map-matching speed: Localizing global curve-matching algorithms. In *18th Int. Conf. on Scientific and Statistical Database Management*, SSDBM'06, pages 379–388, 2006.

[25] T. Wylie and B. Zhu. Following a curve with the discrete Fréchet distance. *Theoretical Computer Science*, 556:34–44, oct 2014.

[26] T. Wylie and B. Zhu. Intermittent map matching with the discrete Fréchet distance. *CoRR*, abs/1409.2456:1–21, 2014.

ETH-Tight Algorithms for Geometric Network Problems Using Geometric Separators*

Mark de Berg†

Many well-known optimization problems on graphs, including Independent Set, Hamiltonian Cycle, and the Traveling Salesman Problem (tsp) are np-hard. Hence, we do not expect to have polynomial-time algorithms for solving these problems exactly. However, we may still be able to develop so-called *subexponential* algorithms, that is, algorithms whose running time is of the form $2^{o(n)}$, where n is the input size. This turns out to be the case for many problems (including the ones mentioned above) when the input graph is planar. In particular, Independent Set and Hamiltonian Cycle can be solved in $2^{O(\sqrt{n})}$ time on n-vertex planar graphs. The fact that many problems on planar graphs admit algorithms with $2^{O(\sqrt{n})}$ running time has been dubbed the *square-root phenomenon*. A main tool behind this phenomenon is the famous *Planar Separator Theorem*, which states that for any planar graph $\mathcal{G} = (V, E)$ there is a subset $S \subset V$ of $O(\sqrt{n})$ vertices whose removal splits $\mathcal{G}$ into connected components of size at most $2n/3$.

In the first part of my talk I will discuss some recent work [1] that extends these results to certain classes of *geometric intersection graphs*. The intersection graph induced by a set V of geometric objects is the graph $\mathcal{G} = (V, E)$ whose vertices correspond to the objects in V and where $E = \{(o, o') \in V \times V : o \cap o' \neq \emptyset\}$. In other words, there is an edge between two objects if and only if they intersect each other. *Disk graphs* and *unit-disk graphs*—here the objects in V are (unit) disks in the plane—are well know examples of geometric intersection graphs. Disk graphs generalize planar graphs, because any planar graph can be realized as the intersection graph of a set of disks. (Actually, any planar graph is the intersecting graph of a set of disks with disjoint interiors.) Disk graphs can have arbitrarily large cliques and so they do not have small separators. Still, as I will explain in the talk, there is a "clique-based" separator for disk graphs (and, more generally, for intersection graphs of so-called *fat objects*) that makes it possible to solve Independent Set on disk graphs in $2^{O(\sqrt{n})}$ time. When the disks are similar in size, then using these clique-based separators even leads to algorithms with $2^{O(\sqrt{n})}$ running time for many other classic graph problems as well. This running time is *ETH-tight*: unless the Exponential-Time Hypothesis fails—the Exponential-Time Hypothesis (ETH) states that 3-sat on n variables cannot be solved in $2^{o(n)}$ time—there can be no algorithm that solves these problems on unit-disk graphs in $2^{o(\sqrt{n})}$ time. The algorithms as well as the ETH-based lower bound generalize to $\mathbb{R}^d$, where the running times becomes $2^{O(n^{1-1/d})}$.

In the second part of the talk I will focus on Euclidean tsp, where we want to find a shortest tour visiting a given set P of n points in the plane (or in some higher-dimensional space). The celebrated Held-Karp dynamic-programming algorithm solves tsp on general weighted graphs in $O(n^2 2^n)$ time, and assuming ETH no subexponential algorithms is possible. For the Euclidean tsp, however, there are algorithms with $n^{O(\sqrt{n})} = 2^{O(\sqrt{n} \log n)}$ running time. I will explain a recent result [2] which improves this to $2^{O(\sqrt{n})}$, which is ETH-tight. In $\mathbb{R}^d$ the running time becomes $2^{O(n^{1-1/d})}$, which is also ETH-tight. The algorithm is based on a new "distance-based" separator theorem for point sets.

References

[1] M. de Berg, H.L. Bodlaender, S. Kisfaludi-Bak, D. Marx, and T. van der Zanden. A framework for ETH-tight algorithms and lower bounds in geometric intersection graphs. In: *Proc. 50th ACM Sympos. Theory Comput. (STOC 2018)*, pages 574–586, 2018.

[2] M. de Berg, H.L. Bodlaender, S. Kisfaludi-Bak, and S. Kolay. An ETH-tight exact algorithm for Euclidean TSP. In: *Proc. 59th Annual IEEE Sympos. Found. Comput. Sci. (FOCS 2018)*, pages 450–461, 2018.

*This work was supported by the NETWORKS project, funded by the Netherlands Organization for Scientific Research NWO under project no. 024.002.003

†Department of Computer Science, TU Eindhoven, M.T.d.Berg@tue.nl

Minimum Ply Covering of Points with Disks and Squares

Therese Biedl* Ahmad Biniaz† Anna Lubiw‡

Abstract

Following the seminal work of Erlebach and van Leeuwen in SODA 2008, we introduce the minimum ply covering problem. Given a set P of points and a set S of geometric objects, both in the plane, our goal is to find a subset S' of S that covers all points of P while minimizing the maximum number of objects covering any point in the plane (not only points of P). For objects that are unit squares and unit disks, this problem is NP-hard and cannot be approximated by a ratio smaller than 2. We present 2-approximation algorithms for this problem with respect to unit squares and unit disks. Our algorithms run in polynomial time when the optimum objective value is bounded by a constant.

Motivated by channel-assignment in wireless networks, we consider a variant of the problem where the selected unit disks must be *3-colorable*, i.e., colored by three colors such that all disks of the same color are pairwise disjoint. We present a polynomial-time algorithm that achieves a 2-approximate solution, i.e., a solution that is 6-colorable.

We also study the weighted version of the problem in dimension one, where P and S are points and weighted intervals on a line, respectively. We present an algorithm that solves this problem in $O(n + m + M)$-time where n is the number of points, m is the number of intervals, and M is the number of pairs of overlapping intervals. This repairs a solution claimed by Nandy, Pandit, and Roy in CCCG 2017.

1 Introduction

Motivated by interference reduction in cellular networks, Kuhn et al. [11] introduced *Minimum Membership Set Cover* (MMSC) as a combinatorial optimization problem. The input to this problem consists of a set U of elements and a collection S of subsets of U, whose union contains all elements of U. The *membership* [11] of an element $u \in U$ with respect to a subset S' of S is the number of sets in S' that contain u. The goal is to find a subset S' of S that covers all elements of U and that minimizes the maximum membership of elements in U. The MMSC problem is closely related to the well-studied *Minimum Set Cover* problem in which the goal is to find a minimum cardinality subset S' of S that covers all elements of U. By a reduction from the minimum set cover problem, Kuhn et al. [11] showed that the MMSC problem is NP-complete and cannot be approximated, in polynomial time, by a ratio less than $\ln n$ unless $NP \subset TIME\left(n^{O(\log \log n)}\right)$, where $n := |U|$ is the number of elements. They also presented an $O(\ln n)$-approximation algorithm for the MMSC problem by formulating it as a linear program.

The geometric version of the MMSC problem and its variants attracted considerable attention following the seminal work of Erlebach and van Leeuwen in SODA 2008; e.g., see [1, 2, 9, 10, 12, 15]. The input of the geometric MMSC problem consists of a set P of points and a set S of geometric objects both in the plane. The goal is to find a subset S' of S such that (i) S' covers all points of P, i.e., the membership of every point is at least 1, and (ii) S' minimizes the maximum membership of points of P. Erlebach and van Leeuwen [6] proved that the geometric MMSC problem is NP-hard for unit disks and for axis-aligned unit squares, and does not admit a polynomial-time approximation algorithm with ratio smaller than 2 unless P=NP. For unit squares, they presented a 5-approximation algorithm that takes polynomial time if the optimal objective value (i.e., the maximum membership) is bounded by a constant. To the best of our knowledge, no $O(1)$-approximation algorithm is known for unit disks.

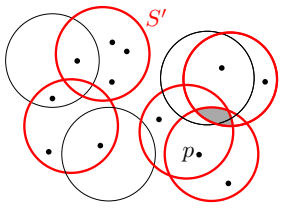


Figure 1: The ply of S' is 3.

In some applications, e.g. interference reduction in cellular networks, it is desirable to minimize the membership of every point in the plane, not only points of P. Thus we study a version of the geometric MMSC problem in which we want to find a subset S' of S that covers all points of P and minimizes the maximum membership of all points of the plane (not only points of P). We refer to this version of the problem as *minimum ply covering* (MPC). The *ply* [5] of a set S' is defined to be the maximum membership of points of the plane with respect to S'. With this definition, the MPC problem

*University of Waterloo, biedl@uwaterloo.ca
†University of Waterloo, ahmad.biniaz@gmail.com
‡University of Waterloo, alubiw@uwaterloo.ca

asks for a subset S' with minimum ply that covers P. In Figure 1 the membership of input points with respect to S' is at most 2 (see point p), while the ply of S' is 3 (see the shaded area).

By a simple modification of the hardness proof of [6], we show (in Appendix A) that the MPC problem is NP-hard, for both unit squares and unit disks, and does not admit polynomial-time approximation algorithms with ratio smaller than 2 unless P=NP. As our main result, we present 2-approximation algorithms for the MPC problem on unit squares and unit disks. Both algorithms run in polynomial time if the optimal objective value (i.e., the minimum ply) is bounded by a constant.

Motivated by channel-assignment in wireless networks, where the use of 3 channels is a standard practice, we study a variant of the MPC problem on unit disks where we want the solution to be 3-colorable, i.e., to be partitioned into three subsets such that the disks in each subset are pairwise disjoint (each subset has ply 1). See [3] for a justification of the importance of 3 channels. We present a polynomial-time 2-approximation algorithm for this version as well.

We also revisit the weighted version of the geometric MMSC problem in dimension one, where P and S are points and weighted intervals on the real line, respectively. This problem was previously claimed solved by Nandy, Pandit, and Roy [13], who referred to the problem as "minimum depth covering". We point out a mistake in their algorithm. We present an $O(n+m+M)$-time algorithm that solves this problem optimally, where n is the number of points, m is the number of intervals, and M is the number of pairs of overlapping intervals. Our algorithm can be adapted in a simple way to solve the MPC problem on weighted intervals within the same time bound.

2 Minimum Ply Covering with Unit Squares

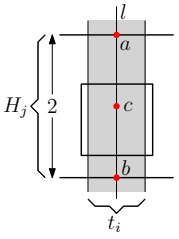
In this section we study the MPC problem on unit squares. We are given a set P of n points and a set S of m axis-aligned unit squares, both in the plane. We assume that unit squares are closed (contain their boundaries) and have side length 1. Our goal is to find a subset S' of S with minimum ply that covers all points of P. This problem cannot be approximated in polynomial-time by a ratio smaller than 2; see Appendix A. We present a 2-approximation algorithm that takes polynomial time if the minimum ply is bounded by a constant. In the rest of this section we assume that the minimum ply is bounded by ℓ.

We partition the plane into horizontal slabs of height 2; this is a standard initial step of many geometric covering algorithms. We may assume that no point of P or edge of a square in S lies on the boundary of any slab. Let $H_1, H_2, \dots$ denote the slabs from bottom to top. For $j \in \{1, 2, \dots\}$, let P_j be the points of P in H_j and let S_j be the set of squares that intersect H_j. Note that if there exists a solution S^* for the MPC problem then $S^* \cap S_j$ covers all points in P_j and has ply at most ℓ, and thus $S^* \cap S_j$ is a solution for the MPC problem on input instance P_j and S_j that has ply at most ℓ. Our approach is therefore to solve MPC for this input instance, i.e., for slab H_j. If this fails for some j, then the MPC problem on P and S has no solution with ply ℓ. If this succeeds for all j, then we set $S' = \bigcup_j S'_j$, where S'_j is the solution for slab H_j. Certainly all points of P are covered. Any square in S' belongs to solutions of at most two consecutive slabs H_j and H_{j+1}. Thus, any point in the plane is covered by squares of at most two solutions S'_j and S'_{j+1}, and hence is covered by at most 2ℓ squares of S'. Therefore, the ply of S' is at most 2ℓ.

In the rest of this section we show how to solve the MPC problem for every slab H_j. To simplify our description we assume that the left and right sides of all squares in S_j have distinct x-coordinates and no point lies on the left or right side of a square; we will describe later how to handle coinciding x-coordinates. We partition the plane into vertical strips by lines through left and right sides of all squares in S_j. Let $t_0, t_1, \dots, t_k$ denote these vertical strips, ordered from left to right. We consider every strip as an open set, i.e., the vertical line between t_{i-1} and t_i belongs to neither of them. The leftmost strip t_0 is unbounded to the left and the rightmost strip t_k is unbounded to the right. Since $|S_j| \leqslant m$, we have $k \leqslant 2m$. The following lemma is important for our strategy to solve the problem.

Lemma 1 *Let $S^*_j \subseteq S_j$ be any solution, with ply at most ℓ, for the MPC problem. The number of squares in S^*_j that intersect any strip t_i is at most 3ℓ.*

Proof. Let l be a vertical line in the interior of t_i. Let a and b be the intersection points of l with the upper and lower boundaries of the slab H_j, and let c be the midpoint of the line segment ab; see the figure to the right. Notice that $|ac| = |bc| = 1$. Because of this, and since no square has its left or right side in the interior of t_i, it follows that every square in S^*_j that intersects t_i contains at least one of the three points a, b and c. Therefore, if more than 3ℓ squares of S^*_j intersect t_i, then by the pigeonhole principle one of the three points lies in more than ℓ squares of S^*_j, a contradiction. □



Based on Lemma 1, we construct a directed acyclic graph G such that any solution S^*_j corresponds to a path from the source vertex to the sink vertex in G. Then we will find a path in G which will correspond to a solution with ply at most ℓ. Now we describe the construction of

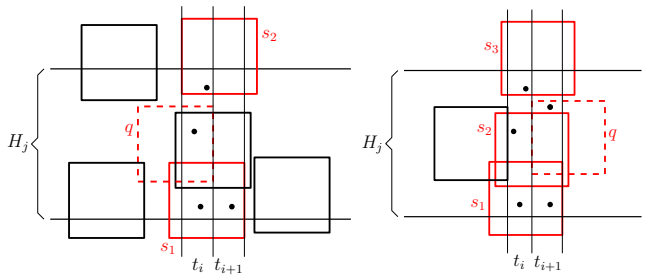
G. For every t_i, with $i \in \{0, \ldots, k\}$, we define a set V_i of vertices as follows: For every subset $Q \subseteq S_j$, containing at most 3ℓ squares that intersect t_i, we add a vertex $v_i(Q)$ to V_i if the following conditions hold:

(i) the squares in Q cover all points in $t_i \cap P_j$,
(ii) the ply of Q is at most ℓ (i.e. every point in $\mathbb{R}^2$ is in at most ℓ squares of Q).

Since no square intersects t_0 and t_k, we have $V_0 = \{v_0(\emptyset)\}$ and $V_k = \{v_k(\emptyset)\}$. The vertices $v_0(\emptyset)$ and $v_k(\emptyset)$ are the source and sink vertices of G. The vertex set of G is the union of the sets V_i. The edge set of G consists of directed edges from the vertices in V_i to the vertices in V_{i+1} defined as follows. For every $i \in \{0, \ldots, k-1\}$ and for every vertex $v_i(Q) \in V_i$ we add three directed edges from $v_i(Q)$ to the following three vertices in V_{i+1} (provided they exist):

1. the vertex $v_{i+1}(Q')$ with $Q' = Q$,
2. the vertex $v_{i+1}(Q')$ with $Q' = Q \setminus \{q\}$, where q is the square whose right side is on the left boundary of t_{i+1} (by our assumptions this is unique); see Figure 2(a),
3. the vertex $v_{i+1}(Q')$ with $Q' = Q \cup \{q\}$, where q is the square whose left side is on the left boundary of t_{i+1} (by our assumptions this is unique); see Figure 2(b).

This the end of our construction of G. Observe that in all cases sets Q and Q' differ by at most one square, and $Q \subseteq Q'$ and/or $Q \supseteq Q'$.



(a) (b)

Figure 2: Construction of G. The representation of an edge from $v_i(Q)$ to $v_{i+1}(Q')$ in two cases where in (a) $Q = \{s_1, s_2, q\}$ and $Q' = \{s_1, s_2\}$, and in (b) $Q = \{s_1, s_2, s_3\}$ and $Q' = \{s_1, s_2, s_3, q\}$.

Consider any path δ from $v_0(\emptyset)$ to $v_k(\emptyset)$ in G. Let S'_j be the union of all sets Q corresponding to the vertices of δ. Our algorithm outputs S'_j as a solution of the MPC problem on P_j and S_j. The following claim proves the correctness of our algorithm.

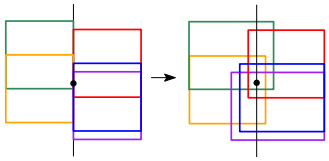
Claim 1 *If the MPC problem on P_j and S_j has a solution with ply at most ℓ, then there exists a path δ from $v_0(\emptyset)$ to $v_k(\emptyset)$ in G. If there exist such a path δ, then the set S'_j is a solution with ply at most ℓ.*

Proof. For the first direction, consider a solution S^*_j with ply at most ℓ. For $0 \leqslant i \leqslant k$, let Q^*_i be the squares of S^*_j that intersect t_i. Observe that any point of P_j that lies in t_i is covered by a square that intersects t_i. Thus Q^*_i covers all points in t_i. Since the ply of S^*_j is at most ℓ, the set Q^*_i has at most 3ℓ squares by Lemma 1. Therefore, Q^*_i satisfies conditions (i) and (ii) and thus $v_i(Q^*_i)$ is a vertex of G. Since (by our initial assumption) no two squares begin or end at the same x-coordinate, Q^*_i and Q^*_{i+1} differ by at most one square, and thus there is an edge from $v_i(Q^*_i)$ to $v_{i+1}(Q^*_{i+1})$ in G. Since this holds for every i, the solution S^*_j can be mapped to the path $v_0(\emptyset), v_1(Q^*_1), \ldots, v_{k-1}(Q^*_{k-1}), v_k(\emptyset)$ in G. Therefore, δ exists.

For the other direction, observe that the edges of G only connect the vertices of adjacent strips. Thus, δ contains exactly one vertex, say $v_i(Q_i)$, for each strip t_i, with $0 \leqslant i \leqslant k$. With this notation, we have that $S'_j = \bigcup_i Q_i$. By condition (i) the set Q_i covers all points of P_j that lie in t_i. By our assumption of no coinciding x-coordinates, every point of P_j lies in some (open) strip, and thus S'_j covers all points of P_j. Now we verify the ply of S'_j. To that end, fix an arbitrary point $p \in \mathbb{R}^2$ and notice that p can be in some strip t_i or on the boundary between two strips t_i and t_{i+1}. If p is in t_i, then by condition (ii) it has membership at most ℓ in Q_i. Therefore it also has membership at most ℓ in S'_j because by our definition of edges no square in $S'_j \setminus Q_i$ can intersect t_i. Assume now that p lies on the boundary between t_i and t_{i+1}. Any square of S'_j that contains p must belong to Q_i or Q_{i+1} (or both). Furthermore, by our construction of G, we have $Q_i \subseteq Q_{i+1}$ or $Q_i \supseteq Q_{i+1}$. Since p has membership at most ℓ in both Q_i and Q_{i+1} (by condition (ii)), the membership of p in S'_j is at most ℓ. Therefore, the ply of S'_j is at most ℓ. □

Remark 1. Our algorithm does not use the fact that elements of S are squares, but only uses that they have unit height. Therefore the algorithm extends to axis-aligned unit-height rectangles.

Remark 2. The case where sides of squares and/or input points have coinciding x-coordinates can be handled by a symbolic perturbation. At any x-coordinate X we order first all the left sides of squares at X, breaking ties by their y-ordering; then the input points at X; and finally, all the right sides of squares at X, breaking ties by their y-ordering; see the figure to the right for illustration.



For the running time to solve the problem in H_j, set $n_j = |P_j|$ and $m_j = |S_j|$. Recall that every vertex of G corresponds to a set Q of at most 3ℓ squares by our construction, and thus there are $O(m_j^{3\ell})$ such sets Q.

This and the fact that each set Q could be used repeatedly among $O(m_j)$ strips, imply that G has $O(m_j^{3\ell+1})$ vertices. Since every vertex has at most three outgoing edges, the number of edges of G is also $O(m_j^{3\ell+1})$. By an initial sorting of the points of P_j and the squares of S_j with respect to the y-axis, conditions (i) and (ii) can be verified in $O(\ell + n_j)$ time for each vertex. A path δ can be found in time linear in the size of G. Thus, the total running time to solve the MPC problem in slab H_j is $O((\ell + n_j) \cdot m_j^{3\ell+1})$.

Theorem 2 *There exists a polynomial-time algorithm that solves the problem of minimum ply covering of points in a slab of height two with unit-height rectangles, provided that the optimal objective value is constant.*

As discussed at the beginning of this section, the union of the solutions of all slabs is a 2-approximate solution for the original problem of covering n points in the plane with m unit squares. Since every point belongs to exactly one slab and every square belongs to at most two slabs, this 2-approximate solution can be computed in $\sum (\ell + n_j) \cdot m_j^{3\ell+1} = O((\ell + n) \cdot (2m)^{3\ell+1})$ time, where the sum runs over all slabs. The following theorem summarizes our result.

Theorem 3 *There exists a polynomial-time 2-approximation algorithm that solves the problem of minimum ply covering of points with unit-height rectangles, provided that the optimal objective value is constant.*

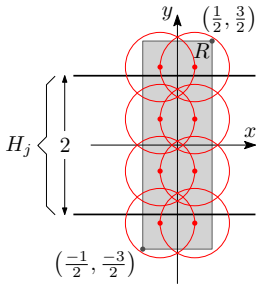
3 Minimum Ply Covering with Unit Disks

In this section we study the MPC problem on unit disks, i.e., disks with diameter 1. Given a set P of n points and a set S of m unit disks, both in the plane, the goal is to find a subset S' of S, with minimum ply, that covers all points of P. This problem cannot be approximated in polynomial-time by a ratio better than 2; see Appendix A. We present a 2-approximation algorithm that takes polynomial time if the minimum ply is bounded by a constant. In the rest of this section we assume that the minimum ply is bounded by ℓ. Our algorithm is a modification of that of unit squares. After a suitable rotation of the plane we assume that in the set consisting of points of P together with the leftmost and rightmost points of disks in S, no two points have the same x-coordinate.

As in the previous section we partition the plane into horizontal slabs of height 2. Then we solve the MPC problem in every slab H_j by constructing a directed acyclic graph G. Let P_j be the set of points in H_j, and let S_j be the set of all disks that intersect H_j. We partition the plane into vertical strips $t_0, \ldots, t_k$ by vertical lines through the leftmost and rightmost points of disks in S_j. The only major change to the algorithm of the previous section is the definition of vertices of G, because Lemma 1 does not hold for unit disks. For unit disks we have the following helper lemma.

Lemma 4 *Let $S_j^* \subseteq S_j$ be any solution, with ply at most ℓ, for the MPC problem. Then for any strip t_i at most 8ℓ disks in S_j^* intersect t_i.*

Proof. After a suitable translation assume that H_j has y-range $[-1, +1]$, and assume that the y-axis lies in t_i. Any disk $s \in S_j$ that intersects t_i must also intersect the y-axis because boundaries of vertical strips are defined by vertical lines through leftmost and rightmost points of disks in S_j. It follows that the center of s must lie within a rectangle R with corners $(\pm\frac{1}{2}, \pm\frac{3}{2})$; see the figure to the right. The rectangle R can be covered by eight unit disks $\{D_1, \ldots, D_8\}$ that are centered at eight points $\{p_1, \ldots, p_8\} = \{(\pm\frac{1}{4}, \pm\frac{3}{8}), (\pm\frac{1}{4}, \pm\frac{9}{8})\}$ respectively—the red points in the figure. Thus, the center of s lies in a disk D_i, for some $i \in \{1, \ldots, 8\}$, and hence at distance at most $\frac{1}{2}$ from p_i. This implies that $p_i \in s$. Thus, each disk in S_j that intersects t_i contains at least one of the points $\{p_1, \ldots, p_8\}$. Since S_j^* has ply at most ℓ, each point p_i lies in at most ℓ disks of S_j^*. Therefore, at most 8ℓ disks of S_j^* intersect t_i. □



For every strip t_i we introduce a set V_i that contains vertices $v_i(Q)$, for all sets Q of at most 8ℓ disks that (i) intersect t_i, (ii) cover all points of P_j in t_i, and (iii) have ply at most ℓ. Then we connect the vertices of V_i to the vertices of V_{i+1} in a similar fashion as for unit squares.

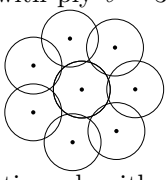
Using Lemma 4, we can claim (similar to that of unit squares) that any path from the source to the sink in G corresponds to a solution with ply at most ℓ for the MPC problem with input P_j and S_j. Therefore, we can compute in $O((\ell + n_j) \cdot m_j^{8\ell+1})$ time a solution with ply at most ℓ for every slab, and in $O((\ell + n) \cdot (2m)^{8\ell+1})$ time a 2-approximate solution for the original problem. The following theorem summarizes our result.

Theorem 5 *There exists a polynomial-time 2-approximation algorithm that solves the problem of minimum ply covering of points with unit disks, provided that the optimal objective value is constant.*

3.1 3-Colorable Unit Disk Covering

Now we study the 3-colorable unit disk covering problem. Given a set P of n points and a set S of m unit disks, both in the plane, the goal is to find a subset S' of S that covers all points of P and such that S' can be

partitioned into $\{S'_1, S'_2, S'_3\}$ where the disks in each S'_a, $a \in \{1, 2, 3\}$, are pairwise disjoint, i.e., the ply of each S'_a is 1. Although at first glance this problem seems to be a special case of the MPC problem with ply $\ell = 3$, they are different because there are input instances that have a solution with ply at most 3 but do not have any 3-colorable solution; see for example the input instance in the figure to the right.



In Appendix B we present a polynomial-time algorithm that achieves a 2-approximate solution, i.e., a solution that is 6-colorable. This algorithm is a modified version of the algorithm of Section 3.

Theorem 6 *There exists a polynomial-time 2-approximation algorithm for the 3-colorable covering problem of points in the plane with unit disks.*

4 Minimum Membership and Minimum Ply Coverings with Weighted Intervals

Nandy, Pandit, and Roy [13] studied the geometric MMSC problem in dimension one, where P is a set of n points and S is a set of m closed intervals, both on the real line. By formulating this problem as a maximum independent set problem, they solved it in $O(n + m)$ time, provided that the points of P and the endpoints of intervals are given in sorted order.

Nandy et al. [13] also studied the weighted version of the geometric MMSC problem in dimension one. In this version every interval has a weight, and the membership of a point p is defined as the sum of the weights of the intervals in S' that cover p. They claimed an $O(nm \log n)$-time algorithm that solves this version of the problem. However, as we point out in Appendix C, their algorithm does not always find the optimal solution. We present here an algorithm that solves the weighted version of the MMSC problem in dimension one in $O(n+m+M)$ time, where $M \leqslant \binom{m}{2}$ is the number of pairs of overlapping intervals, i.e., the number of edges of the interval graph formed by the intervals in S. As we will see later in Remark 3, our algorithm can easily be modified to also solve the MPC problem on weighted intervals in the same time.

Our algorithm—for the MMSC problem on weighted intervals—creates a directed acyclic graph (DAG) such that all optimal solutions correspond to directed paths from the source to the sink. More precisely, we construct a vertex-weighted DAG (because the intervals are weighted), and then search for a *bottleneck path* from the source to the sink; a bottleneck path is a path that minimizes the maximum weight along the path. Such a DAG could be obtained directly from Section 2, but we introduce a new construction with better running time.

Let P be a set of n points and let $S = \{s_1, \ldots, s_m\}$ be a set of m intervals on a line, where each s_i has weight $w(s_i)$. Assume that the points of P and the endpoints of intervals in S are given in sorted order from left to right. In the remainder of this section we show how to transform our problem, in $O(n + m + M)$ time, to an instance of the bottleneck path problem in a DAG of size $O(m + M)$. The bottleneck path problem in a DAG can be solved in linear time. For simplicity of our description we assume points of P and endpoints of intervals of S have distinct x-coordinates; this can be achieved by a symbolic perturbation similar to that of Remark 2 in Section 2.

Draw vertical lines through the endpoints of intervals in S to partition the plane into vertical strips $t_0, \ldots, t_k$ (ordered from left to right). Notice that $k \leqslant 2m$, the leftmost strip t_0 is unbounded to the left, and the rightmost strip t_k is unbounded to the right. Also t_0 and t_k have no points in them. See Figure 3. For every $i \in \{1, \ldots, k\}$, all points of P in strip t_i belong to the same set of intervals. Thus, it suffices to keep only one of the points in each strip t_i and discard the other points. This can be done in $O(n + m)$ time. Thus, we assume that $n \leqslant 2m-1$. We say that an interval s is *dominated* by an interval r if r starts before s and ends after s. The following lemma has been proved in [13].

Lemma 7 *There exists an optimal solution in which no interval is dominated by some other interval, and no vertical line intersects more than two intervals.*

Based on this lemma, it suffices to search for an optimal solution $S^* \subseteq S$ such that the interior of every strip t_i intersects at most two intervals of S^*; see Figure 3. We therefore endow our DAG, denoted by G, with the following three types of vertices, corresponding to the intersection of strips by 0, 1, or 2 intervals, respectively.

- For every strip t_i that contains no points of P, create a vertex $v_0(t_i)$ with weight 0. Choosing $v_0(t_i)$ in our path will correspond to having 0 intervals cover the interior of t_i. Vertices $v_0(t_0)$ and $v_0(t_k)$ will be the source and sink vertices of our graph.

- For every interval q and for every strip t_i that is intersected by q, create a vertex $v_1(q, t_i)$. Choosing $v_1(q, t_i)$ in our path will correspond to choosing interval q and choosing no other interval that intersects t_i. If some point of P is in t_i, then set $w(v_1(q, t_i)) = w(q)$, otherwise set $w(v_1(q, t_i)) = 0$.

- For every two overlapping intervals q and r, with q starting before r starts and also ending before r ends, create a vertex $v_2(q, r)$. Choosing $v_2(q, r)$ in our path will correspond to choosing intervals q and r. If there is a point of P in the intersection of q and r, then set $w(v_2(q, r)) = w(q) + w(r)$, otherwise set $w(v_2(q, r)) = 0$.

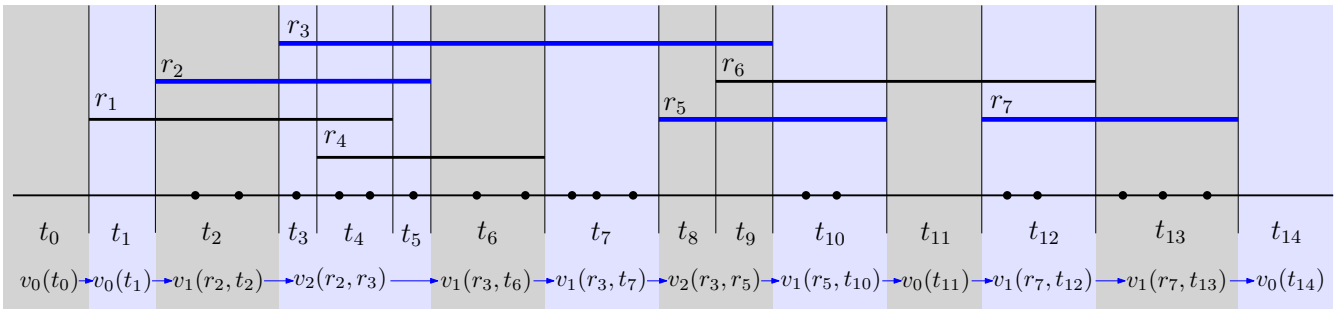
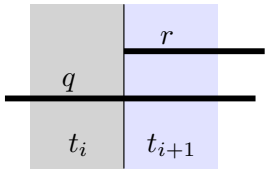


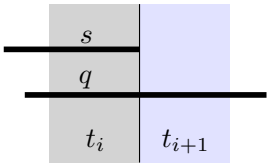
Figure 3: Correspondence between an optimal solution ($\{r_2, r_3, r_5, r_7\}$) and a bottleneck path from $v_0(t_0)$ to $v_0(t_{14})$.

We define edges as follows. Consider each $i < k$ and the strip t_i. The right boundary of t_i exists since some interval has an endpoint there. Assume first that this was a left endpoint, so there exists an interval r whose left endpoint lies on the right boundary of t_i (and r is unique by our assumption). Then we add the edges listed in Figure 4(a), adding only those where the vertices exist. If instead some interval s has its right endpoint at the right boundary of t_i, then we add the following edges listed in Figure 4(b) (again, provided the vertices listed there exist). Note that every vertex has at most two outgoing edges.



$v_0(t_i) \to v_0(t_{i+1})$
$v_0(t_i) \to v_1(r, t_{i+1})$
$v_1(q, t_i) \to v_1(q, t_{i+1})$
$v_1(q, t_i) \to v_2(q, r)$

(a)



$v_0(t_i) \to v_0(t_{i+1})$
$v_1(s, t_i) \to v_0(t_{i+1})$
$v_1(q, t_i) \to v_1(q, t_{i+1})$
$v_2(s, q) \to v_1(q, t_{i+1})$

(b)

Figure 4: The edges of G.

The construction of G implies the following claim.

Claim 2 *The solutions that satisfy the constraints of Lemma 7 are in one-to-one correspondence with the paths from $v_0(t_0)$ to $v_0(t_k)$ in G. In particular, the bottleneck path in G corresponds to an optimal solution of the minimum membership covering problem for P and S. See Figure 3 for an illustration.*

Now we analyze the running time of our algorithm. Since G is a DAG, a bottleneck path in G can be computed in linear time on the size of G.

We claim that the size of G is $O(m + M)$, where M is number of edges in the interval graph defined by S. Since every vertex has at most two outgoing edges, it suffices to show that G has $O(m + M)$ vertices. There are at most $2m + 1$ vertices of type v_0 (one per strip). Every vertex of type v_2 is uniquely defined by two overlapping intervals, so there are at most M of them. To bound the number of vertices of type v_1, observe that such a vertex is defined by an interval r and a strip t_i with $1 \leqslant i < k$. Let r' be the interval that defines the left boundary of t_i; it might be the case that $r' = r$. The number of vertices for which $r' = r$ is at most $2m-1$ because this happens (by the general position assumption) only once per strip. Assume that $r' \neq r$. Since r and r' intersect (at the left boundary of t_i), it follows that (r, r') is an edge in the interval graph. Any such edge can be charged at most 4 times, once at each of the left and the right boundaries of r and r'. Thus, the number of vertices for which $r' \neq r$ is at most $4M$. Hence G has size $O(m + M)$ and the bottleneck path can be found within the same time.

The linked-list representation of G can be obtained in $O(m+M)$ time by going through the strips from left to right. This, together with the initial pruning of points of P in $O(n + m)$ time, implies that the total running time of our algorithm is $O(n + m + M)$.

Remark 3. We can solve the MPC problem on n points and m weighted intervals in $O(n + m + M)$ time; this can be done by adjusting the weights of the vertices of G and then finding a bottleneck path in the resulting graph. We set the weight of every vertex $v_1(q, t_i)$ to $w(q)$ regardless of whether t_i contains a point of P or not, and we set the weight of every vertex $v_2(q, r)$ to $w(q) + w(r)$ regardless of whether the intersection of q and r contains a point of P or not.

Theorem 8 *Both minimum membership and minimum ply covering problems of n points with m weighted intervals on the real line can be solved in $O(n+m+M)$ time where M is the number of pairs of overlapping intervals, provided that the input points and the endpoints of the intervals are given in sorted order.*

Acknowledgement. We thank Albert Gräf for sending us his PhD thesis. Therese Biedl and Anna Lubiw are supported by NSERC. Ahmad Biniaz is supported by NSERC Postdoctoral Fellowship.

References

[1] M. Basappa, R. Acharyya, and G. K. Das. Unit disk cover problem in 2D. *Journal of Discrete Algorithms*, 33:193–201, 2015.

[2] A. Biniaz, P. Liu, A. Maheshwari, and M. H. M. Smid. Approximation algorithms for the unit disk cover problem in 2D and 3D. *Computational Geometry: Theory and Applications*, 60:8–18, 2017.

[3] P. Brass, F. Hurtado, B. J. Lafreniere, and A. Lubiw. A lower bound on the area of a 3-coloured disk packing. *International Journal of Computational Geometry & Applications*, 20(3):341–360, 2010.

[4] B. N. Clark, C. J. Colbourn, and D. S. Johnson. Unit disk graphs. *Discrete Mathematics*, 86(1-3):165–177, 1990.

[5] D. Eppstein and M. T. Goodrich. Studying (nonplanar) road networks through an algorithmic lens. In *Proceedings of the 16th ACM SIGSPATIAL International Symposium on Advances in Geographic Information Systems, ACM-GIS*, 2008.

[6] T. Erlebach and E. J. van Leeuwen. Approximating geometric coverage problems. In *Proceedings of the 19th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1267–1276, 2008.

[7] A. Gräf. *Coloring and recognizing special graph classes*. PhD thesis, Musikinformatik & Medientechnik 20/95, Johannes Gutenberg-Universität Mainz, 1995.

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

[9] T. Ito, S. Nakano, Y. Okamoto, Y. Otachi, R. Uehara, T. Uno, and Y. Uno. A 4.31-approximation for the geometric unique coverage problem on unit disks. *Theoretical Computer Science*, 544:14–31, 2014. Also in *Proceedings of ISAAC 2012*.

[10] T. Ito, S. Nakano, Y. Okamoto, Y. Otachi, R. Uehara, T. Uno, and Y. Uno. A polynomial-time approximation scheme for the geometric unique coverage problem on unit squares. *Comput. Geom.*, 51:25–39, 2016. Also in *Proceedings of SWAT 2012*.

[11] F. Kuhn, P. von Rickenbach, R. Wattenhofer, E. Welzl, and A. Zollinger. Interference in cellular networks: The minimum membership set cover problem. In *Proceedings of the 11th International Computing and Combinatorics Conference (COCOON)*, pages 188–198, 2005.

[12] N. H. Mustafa and S. Ray. Improved results on geometric hitting set problems. *Discrete & Computational Geometry*, 44(4):883–895, 2010.

[13] S. C. Nandy, S. Pandit, and S. Roy. Covering points: Minimizing the maximum depth. In *Proceedings of the 29th Canadian Conference on Computational Geometry (CCCG)*, pages 37–42, 2017.

[14] R. Peeters. On coloring j-unit sphere graphs. FEW 512. Technical report, Department of Economics, Tilburg University, 1991.

[15] E. J. van Leeuwen. *Optimization and approximation on systems of geometric objects*. PhD thesis, University of Amsterdam, June 2009.

A NP-hardness

In this section we study the hardness of the MPC problem on unit squares and unit disks. Erlebach and van Leeuwen [6] proved that the MMSC problem, on both unit disks and unit squares, is NP-hard and cannot be approximated by a ratio smaller than 2. More precisely, they proved it is NP-hard to decide whether or not a solution with membership 1 exists. They use a reduction from the NP-complete problem of deciding whether or not a planar graph G is 3-colorable. Their reduction uses a rectilinear embedding of G in the plane; see [15, Chapter 10] for details. By adjusting the gadgets in their reduction we can show that the MPC problem on unit squares and unit disks, with objective ply 1, is also NP-hard and cannot be approximated by a ratio smaller than 2. We sketch the reduction using our adjusted gadgets, and hence prove the following theorem.

Theorem 9 *It is NP-hard to decide whether or not there exists a solution with ply one, for the minimum ply cover problem on unit squares and on unit disks.*

Since ply—of a set of squares or disks—is an integer, this theorem implies that there is no polynomial-time approximation algorithm, with ratio smaller than 2, for the MPC problem on unit squares and on unit disks, unless P=NP.

In the rest of this section we prove Theorem 9. Similar to that of Erlebach and van Leeuwen [6], our proof uses a reduction from the problem of 3-coloring a planar graph G. We sketch this reduction for the MPC problem on unit sqaures; the reduction for unit disks is analogous. We need a few gadgets, their construction is described below.

Figure 5(a) shows the vertex gadget for every vertex v in G. To cover the point p_v, exactly one of the three squares containing p_v must be in the solution, and this corresponds to assigning one of the three colors to v. Depending on this choice, either 0, 1, or 2 points among the triple of points on the right will also be covered. Figure 5(b) shows a transport gadget, which transports a chosen color along a chain of squares from left to right. The reader may verify that this representation can be modified to bend around corners to represent vertical transportations. Figure 5(c) shows a gadget that duplicates a chosen color. We need one extra gadget to make sure that two adjacent vertices u and v in G will be assigned different colors. Figure 5(d) shows this gadget, assuming the color of u arrives from right and the color of v arrives from left. If u and v have the same color, then to cover a, b and c, we require overlapping squares, which contradicts the ply being 1. If u and v have different colors, then a, b, and c can be covered by three of the six dashed disks. Therefore, our instance of the MPC problem has a solution with ply 1 if and only if G is 3-colorable. Thus the hardness of the 3-colorability problem implies the hardness of the MPC problem. A similar construction of gadgets for the NP-harness of the MPC problem on unit disks is shown in Figure 6; the points b and c play the same role as in Figure 5(d), while a_1, a_2, a_3 play the role of a. Therefore, the above reduction proves Theorem 9.

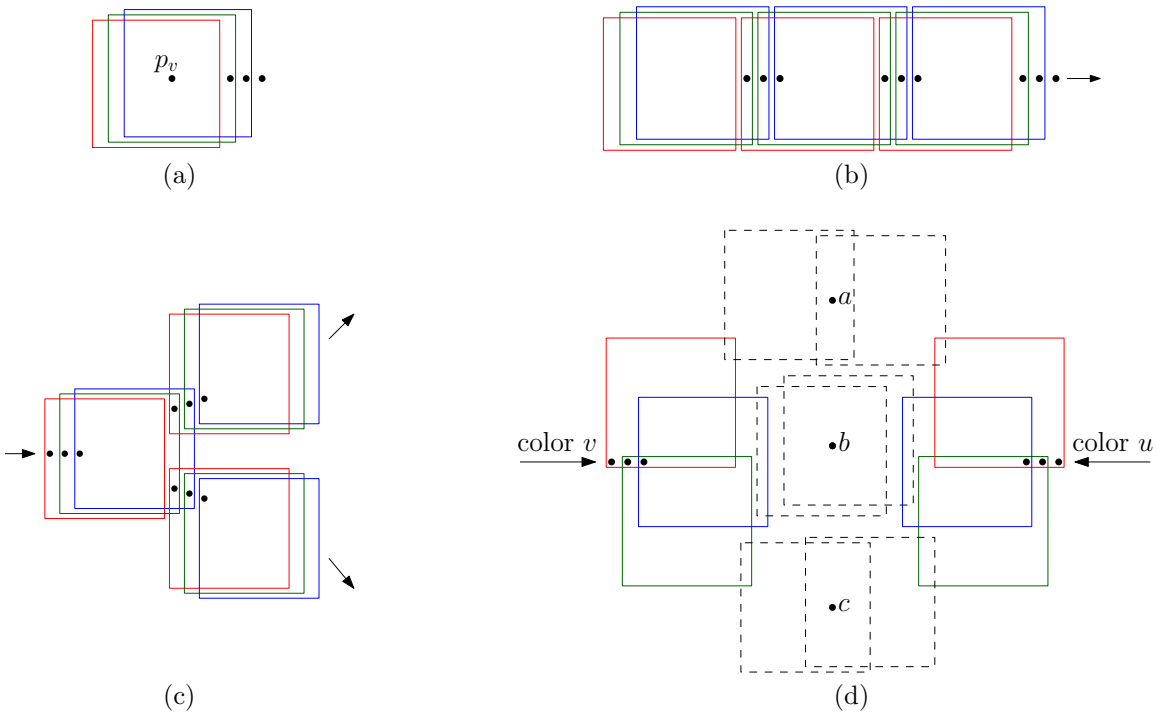


Figure 5: The NP-hardness of the MPC problem on unit squares: (a) the vertex gadget, (b) the transport gadget, (c) the duplicate gadget, and (d) the color checking gadget.

B 3-Colorable Unit Disk Covering

In this section we present a polynomial-time 2-approximation algorithm for the 3-colorable unit disk cover problem, thereby proving Theorem 6.

Before presenting our algorithm we point out a related problem that is the problem of coloring a unit disk graph (the intersection graph of a set S of unit disks in the plane) with k colors. This problem NP-hard for any $k \geqslant 3$ [4]. There are 3-approximation algorithms for this problem (see e.g. [8, 14]), and a 2-approximation algorithm when the unit disk graph has constant clique number (see [7, Chapter 4, Proposition 4.8]). We note that this problem is also different from our 3-colorable unit disk cover problem; for example if all the disks in S have a common intersection and we place our point set P in this intersection, then there exists a 1-colorable solution for our problem, while the unit disk graph is not $(|S|-1)$-colorable. The NP-hardness of 3-coloring a unit disk graph [4] immediately implies the NP-hardness of the 3-colorable unit disk cover problem: every disk in the reduction of [4, Theorem 2.1] covers a unique region of the plane, so by placing points of P in those regions we can enforce the solution S' to contain all disks of S.

Now we present our algorithm, which is a modified version of the algorithm of Section 3. Again we partition the plane into horizontal slabs of height 2. Then for every slab H_j we test the existence of a 3-colorable covering for points P_j using disks in S_j. If this test fails for some j, then there is no 3-colorable solution for P and S. If this test is successful for all j, then we assign colors 1, 2, 3 to solutions of $H_1, H_3, \ldots$ and assign colors 4, 5, 6 to solutions of $H_2, H_4, \ldots$. The union of these solutions will be a 6-colorable solution for the original problem.

After a suitable rotation we assume that in the set consisting of points of P together with the leftmost and rightmost points of disks in S, no two points have the same x-coordinate. To solve the problem for every H_j, as in Section 3, we partition the plane into vertical strips $t_0, \ldots, t_k$. Then we construct a directed acyclic graph G such that any path from the source to the sink in G corresponds to a 3-colorable solution for H_j. Consider a 3-colorable solution $S^* = S_1^* \cup S_2^* \cup S_3^*$. The disks in each S_a^* (for $a = 1, 2, 3$) are pairwise disjoint, and thus each S_a^* has ply 1. Therefore, by Lemma 4 at most 8 disks in each S_a^* intersect each strip t_i. Based on this, for every t_i we introduce a set V_i containing vertices $v_i(Q_1, Q_2, Q_3)$ for all sets Q_1, Q_2, and Q_3 that satisfy all following conditions:

(i) each of Q_1, Q_2, and Q_3 contains at most 8 disks that intersect t_i,

(ii) the disks in each of Q_1, Q_2, and Q_3 are pairwise disjoint, and

(iii) $Q_1 \cup Q_2 \cup Q_3$ covers all points of P_j that lie in t_i.

We connect a vertex $v_i(Q_1, Q_2, Q_3)$ to a vertex $v_{i+1}(Q_1', Q_2', Q_3')$ if one of the following conditions hold:

- for every index a in $\{1, 2, 3\}$ we have $Q_a' = Q_a$, or
- for exactly one index a in $\{1, 2, 3\}$ we have $Q_a' = Q_a \setminus \{d\}$, where d is the disk whose rightmost point is on the left boundary of t_{i+1}, and for every other index b we have $Q_b' = Q_b$, or
- for exactly one index a in $\{1, 2, 3\}$ we have $Q_a' = Q_a \cup \{d\}$, where d is the disk whose leftmost point is on the

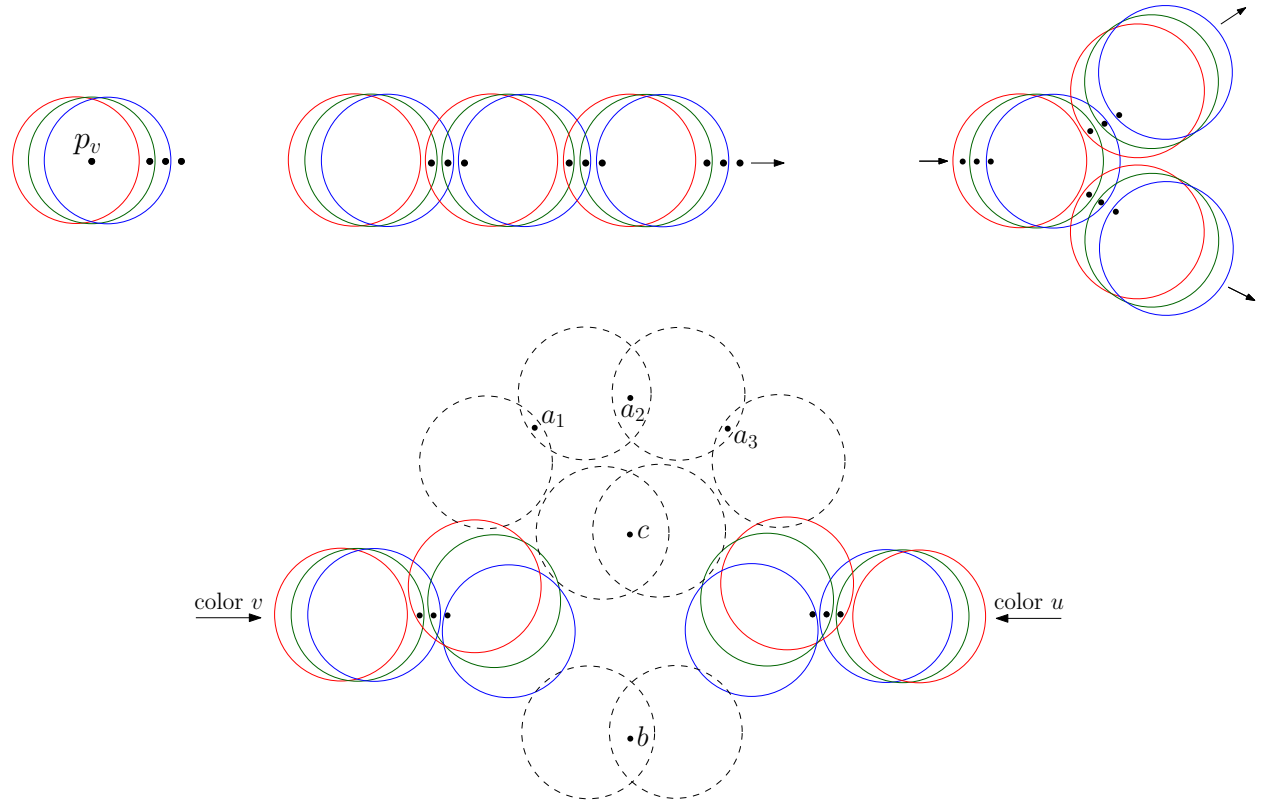


Figure 6: Gadgets for the NP-hardness proof of the MPC problem on unit disks.

left boundary of t_{i+1}, and for every other index b we have $Q'_b = Q_b$.

We briefly justify that paths from the source to the sink in G correspond to 3-colorable solutions. Fix a path δ. For $a = 1, 2, 3$, let S'_a be the union of the disks in all sets Q_a associated with vertices in δ. Condition (iii) ensures that $S'_1 \cup S'_2 \cup S'_3$ covers all points of P_j. Condition (ii) ensures that the disks in each of S'_1, S'_2, and S'_3 are pairwise disjoint. Thus, $S'_1 \cup S'_2 \cup S'_3$ is a 3-colorable solution. It remains to verify the existence of such a path δ in G. Consider a 3-colorable solution $S^* = S^*_1 \cup S^*_2 \cup S^*_3$. For any $a \in \{1, 2, 3\}$ and $i \in \{0, \ldots, k\}$ let Q^i_a be the disks in S^*_a that intersect t_i. As discussed above, we have $|Q^i_a| \leqslant 8$. Thus, $v_i(Q^i_1, Q^i_2, Q^i_3)$ is a vertex of G. By an argument similar to that of Section 2, one can verify that S^* can be mapped to a path in G. Therefore, δ exists.

The running time analysis is similar to that of Section 3, except here we consider 24 disks in every vertical strip (8 disks for every set Q_i). Therefore the total running time of our 2-approximation algorithm is $O(nm^{25})$.

C Nandy, Pandit, and Roy's Algorithm

We briefly review the algorithm of Nandy et al. [13], which uses dynamic programming and proceeds as follows. Let $p_1, p_2, \ldots, p_n$ be the points of P from left to right, and let $s_1, s_2, \ldots, s_m$ be the intervals in S sorted from left to right according to their right endpoints. (In the original presentation of this algorithm in [13], the points of P are ordered from right to left and the intervals in S are also sorted from right to left according to their left endpoints.) Let $w(s_i)$ denote the weight of s_i. For every $k \in \{1, \ldots, m\}$ let $Q_k = (P_k, S_k)$ be an instance of the problem, where $S_k = \{s_1, \ldots, s_k\}$ and P_k is the set of points of P that lie on or to the left of the right endpoint of s_k. Note that Q_m is the original problem. Now solve $Q_1, \ldots, Q_m$ in order, i.e., process interval s_k to find a solution S'_k of Q_k, using previously computed solutions $S'_1, \ldots, S'_{k-1}$ for subproblems $Q_1, \ldots, Q_{k-1}$. Nandy et al. claim that the following approach will find S'_k:

(i) If $P_k = P_{k-1}$ then set $S'_k = S'_{k-1}$.

(ii) If $P_k \neq P_{k-1}$ and some points of $P_k \setminus P_{k-1}$ are not covered by s_k, then there is no solution for Q_k, i.e., S'_k does not exist.

(iii) If $P_k \neq P_{k-1}$ and every point of $P_k \setminus P_{k-1}$ is covered by s_k, then find the index i, with $i \in \{1, \ldots, k-1\}$, such that $S'_i \cup \{s_k\}$ covers P_k and the membership of points of P_k with respect to $S'_i \cup \{s_k\}$ is minimum. Then set $S'_k = S'_i \cup \{s_k\}$.

There are two reasons why this algorithm is not correct. First, item (i) does not consider s_k, while in some cases we may need to include it in the solution, for example if s_k has the minimum weight and covers all points in P_k. This may perhaps be fixable with a minor change of formula, but there is major issue in item (iii) which somehow breaks the hope for any straightforward dynamic programming approach for this problem.

Figure 7 shows an example with five points p_1, p_2, p_3, p_4, p_5, ordered from left to right, and four

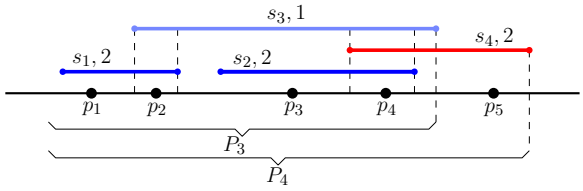


Figure 7: An instance for which the previous algorithm does not compute an optimal solution. "$s_1, 2$" indicates that interval s_1 has weight 2.

weighted intervals s_1, s_2, s_3, s_4, ordered according to their right endpoints. The optimal solutions for Q_1, Q_2, Q_3 are $S'_1 = \{s_1\}$, $S'_2 = \{s_1, s_2\}$, $S'_3 = \{s_1, s_2\}$, with maximum memberships 2, 2, 2. Notice that S'_3 should not contain s_3 because otherwise it should also contain s_1 to cover p_1, and this will make the membership of p_2 be $w(s_1) + w(s_3) = 3$. The recursive computation in item (iii) considers as solution for Q_4 the candidates $S'_1 \cup \{s_4\}$, $S'_2 \cup \{s_4\}$, $S'_3 \cup \{s_4\}$. The first one is not valid (it does not cover p_3) and the other two have maximum membership $w(s_2) + w(s_4) = 4$ (this membership is obtained by p_4 which is covered by s_2 and s_4). However, the set $S'_4 = \{s_1, s_3, s_4\}$ is a solution for Q_4 with maximum membership 3.

We are skeptical about using a straightforward dynamic programming for solving the problem, because while S'_4 is the optimum solution for Q_4, its induced solutions for Q_2 and Q_3 contain the superfluous interval s_3. To use the dynamic programming approach, a more structured recursion is needed, as described in Section 4.

Distributed Unit Clustering

Kian Mirjalali* Seyed Ali Tabatabaee* Hamid Zarrabi-Zadeh*

Abstract

Given a set of points in the plane, the unit clustering problem asks for finding a minimum-size set of unit disks that cover the whole input set. We study the unit clustering problem in a distributed setting, where input data is partitioned among several machines. We present a $(3+\varepsilon)$-approximation algorithm for the problem in the Euclidean plane, and a $(4+\varepsilon)$-approximation algorithm for the problem under general L_p metric ($p \geq 1$). We also study the capacitated version of the problem, where each cluster has a limited capacity for covering the points. We present a distributed algorithm for the capacitated version of the problem that achieves an approximation factor of $4+\varepsilon$ in the L_2 plane, and a factor of $5+\varepsilon$ in general L_p metric. We also provide some complementary lower bounds.

1 Introduction

The exponential growth of data in real-world applications and the incapability of individual computers to store and process the whole data have motivated the research in the area of distributed algorithms. In this paper, we study the distributed version of the following *unit clustering* problem. Given a set of n points in the plane, partition the points into clusters, each enclosable by a unit disk, so as to minimize the number of clusters used. An instance of the problem is illustrated in Figure 1. The problem has applications in various areas including image processing [14, 19] and wireless sensor networks [18, 20].

The unit clustering problem is known to be NP-hard in the Euclidean plane [11]. The first polynomial-time approximation scheme (PTAS) for the problem was given by Hochbaum and Maass [14]. The runtime of the PTAS was later improved by Feder and Greene to $n^{O(1/\epsilon^{d-1})}$ in any fixed d dimensions [10]. A PTAS for the capacitated version of the problem is recently given in [12]. Online variants of the problem are also studied in the literature [6, 9].

For massive datasets, where no single machine can store the whole data, distributed models such as MapReduce have been introduced and extensively used over the past decade [2, 4, 8, 13, 16]. In the *distributed unit clustering* problem, the input set S is partitioned among a set of machines, where each machine i has a subset S_i of the input, and the goal is to compute collaboratively a unit clustering of the whole set $S = \bigcup_i S_i$.

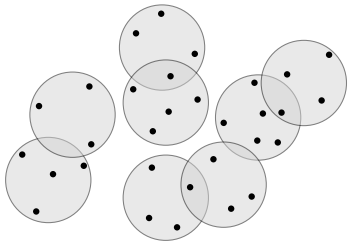


Figure 1: An instance of unit clustering.

The notion of *composable coresets* introduced in [15] has been proved to be useful in designing distributed algorithms that take $O(1)$ rounds of MapReduce. In this framework, each machine performs a computation on its portion of data, and sends a small subset of its data (called a coreset) to a central machine. The central machine then composes the coresets and finds an approximate solution based on the information carried by the coresets. This framework has been successfully used to derive approximation algorithms for several optimization problems [1, 3, 7, 17].

In this paper, inspired by the idea of composable coresets, we design distributed algorithms for the capacitated and uncapacitated versions of the unit clustering problem. For the uncapacitated version, we provide a $(3+\varepsilon)$-approximation algorithm in the Euclidean plane, and a $(4+\varepsilon)$-approximation algorithm in the plane under general L_p metric, for any real number $p \geq 1$. For the capacitated version, we provide a $(4+\varepsilon)$-approximation algorithm in the L_2 plane, and a $(5+\varepsilon)$-approximation algorithm under general L_p metric. We also prove some lower bounds on the approximation factor and communication complexity of any distributed algorithm for the problem under the composable coreset framework. In particular, we show that the unit clustering problem in the Euclidean plane admits no composable coreset with approximation factor better than 2. Moreover, we show that the communication complexity of our algorithms is optimal under this framework.

*Department of Computer Engineering, Sharif University of Technology. Email: {mirjalali,satabatabaee}@ce.sharif.edu, zarrabi@sharif.edu.

2 Preliminaries

Given a real number $p \geq 1$, and two points $a = (x_a, y_a)$ and $b = (x_b, y_b)$ in the plane, the distance of a and b under L_p metric is defined as

$$d_p(a, b) = \sqrt[p]{|x_a - x_b|^p + |y_a - y_b|^p},$$

and $d_\infty(a, b) = \max(|x_a - x_b|, |y_a - y_b|)$. We refer to the plane $\mathbb{R}^2$ in which L_p metric is the distance measure as the *L_p plane*. Whenever we state a proposition for all L_p metrics, $p \geq 1$, we implicitly assume that L_∞ is also included.

For $p \geq 1$ and $r \geq 0$, an *L_p disk of radius r* is defined as the set of points $\{a \in \mathbb{R}^2 \mid d_p(a, c) \leq r\}$, where $c \in \mathbb{R}^2$ is the *center* of the disk. An L_p disk of radius 1 is called a *unit L_p disk*. Whenever the underlying metric L_p is clear from the context, we simply use the terms disk and unit disk.

Given a set of points in the plane under an L_p metric, the *unit clustering* problem is to cover the points by congruent disks of radius r, so as to minimize the number of disks used. We refer to this problem as UC_r. Moreover, we denote by $\text{UC}_r(S)$ an optimal solution to the UC_r problem on an input set S. Whenever $r = 1$, we drop r from the notation, and simply write UC and $\text{UC}(S)$, instead.

3 Covering Disks With Smaller Ones

In this section, we present some upper bounds on the number of disks of radius $r < 1$ needed to cover a unit disk. We will use the following well-known fact as an ingredient: for any $1 \leq p \leq q$, a unit L_p disk can be covered by a unit L_q disk.

Lemma 1 *Under any L_p metric, $p \geq 1$, a unit disk can be covered by $\lceil 2/r \rceil^2$ disks of radius r, for $0 < r \leq 1$.*

Proof. Let D be a unit L_p disk, and S be a unit L_∞ disk covering D. As S is a square of side length 2, it can be covered by $\lceil 2/r \rceil^2$ squares of side length r. On the other hand, each square of side length r can be covered by an L_p disk of radius r, which completes the proof. □

According to Lemma 1, a unit disk in any L_p plane can be covered by a constant number of smaller disks, whenever the radius of the smaller disks is fixed. The next two lemmas provide tighter bounds on this constant.

Lemma 2 *Under any L_p metric, $p \geq 1$, a unit disk can be covered by four disks of radius $\sqrt{2}/2$.*

Proof. We prove the lemma in two cases:

Case 1: $1 \leq p < 2$. Let D be a unit L_p disk, and S be a unit L_2 disk covering D. As illustrated in Figure 2, S can be covered by four diamonds (L_1 disks) of diameter $\sqrt{2}$. On the other hand, each of these four diamonds can be covered by an L_p disk of radius $\sqrt{2}/2$. Hence, four L_p disks of radius $\sqrt{2}/2$ can cover a unit L_p disk in this case.

Case 2: $p \geq 2$. Let D be a unit L_p disk, and S be a square of side length 2 enclosing D. As illustrated in Figure 3, S can be covered by four L_2 disks of radius $\sqrt{2}/2$. On the other hand, each of these four L_2 disks can be covered by an L_p disk of the same radius. Therefore, four L_p disks of radius $\sqrt{2}/2$ can cover a unit L_p disk in this case, which completes the proof. □

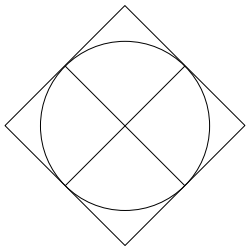


Figure 2: Four diamonds covering a disk.

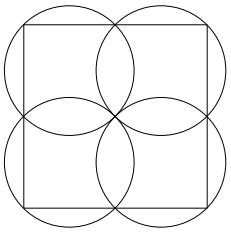


Figure 3: Four disks covering a square.

It is worth noting that a unit L_1 disk cannot be covered by less than four smaller L_1 disks. Moreover, in L_2 metric, four disks of radius $r < \sqrt{2}/2$ cannot cover a unit disk. Hence, in general L_p metric, both our bounds of 4 and $\sqrt{2}/2$ are essentially tight. Nevertheless, for the special case of L_2 metric, it is possible to cover a unit disk by a fewer number of smaller disks.

Lemma 3 *In the L_2 plane, a unit disk can be covered by three disks of radius $\sqrt{3}/2$.*

Proof. The proof is illustrated in Figure 4. □

4 Distributed Unit Clustering

In this section, we present a distributed approximation algorithm for the unit clustering problem under any L_p metric, $p \geq 1$. The pseudo-code is presented in Algorithm 1. The algorithm runs in two phases. In the first

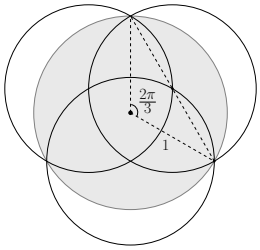


Figure 4: Three disks of radius $\frac{\sqrt{3}}{2}$ covering a unit disk.

phase, the i-th local machine ($1 \leq i \leq m$), processes its input data S_i and sends a subset T_i as a coreset to the central machine. In the second phase, the central machine combines the coresets obtained form local machines into a single set T, and computes a disk cover C of T, which after a proper adjustment can cover the whole input set.

Algorithm 1 Distributed Unit Clustering

1: Let $r = \sqrt{3}/2$ and $\delta = (1-r)/2$.
2: **on** each machine i ($1 \leq i \leq m$) in parallel **do**
3: Find an $O(1)$-approximation C_i to $\mathrm{UC}_\delta(S_i)$.
4: For each disk $D \in C_i$, pick an arbitrary point in $S_i \cap D$, and add it to a set T_i.
5: Send T_i to the central machine.
6: **on** the central machine **do**
7: Let $T = \bigcup_{i=1}^{m} T_i$.
8: Find a $(1+\varepsilon)$-approximation C to $\mathrm{UC}_r(T)$.
9: Increase the radii of disks in C from r to 1.
10: **return** C.

Theorem 4 *Algorithm 1 is a $(3+\varepsilon)$-approximation algorithm for the unit clustering problem in the L_2 plane, and a $(4+\varepsilon)$-approximation algorithm for the problem under general L_p metric, $p \geq 1$. The runtime of the algorithm is $O(n \log n) + (mk)^{O(1/\varepsilon)}$, and its communication complexity is $O(mk)$, where n is the total number of points, m is the number of machines, and k is the size of an optimal solution.*

Proof. Let $S = \bigcup_{i=1}^{m} S_i$ be the input set in the plane, under a given L_p metric, $p \geq 1$. We first prove that the output of the algorithm, C, is a feasible solution, i.e., each point in S is covered by a disk in C. Fix a point $q \in S_i \subseteq S$. By our algorithm, q is covered by a disk of radius δ in C_i. As we add one point from each disk in C_i to T_i, there is point $t \in T_i$ which is within distance 2δ to q. On the other hand, each point of T is covered by a disk of radius r in C. Let D be the disk in C covering t. Therefore, the distance of t to the center of D is at most r. As such, the distance of q to the center of D is at most $r + 2\delta = r + (1-r) = 1$. Therefore, q is covered by D after its radius is increased to one. Hence, C is a feasible solution.

Now, we prove the approximation factor of the algorithm. Let C^* be an optimal solution to $\mathrm{UC}(S)$, and C' be an optimal solution to $\mathrm{UC}_r(T)$. By Lemma 2, each disk in C^* can be covered by four disks of radius $r = \sqrt{3}/2 > \sqrt{2}/2$. Therefore, there is a set of $4|C^*|$ disks of radius r covering S. Since $T \subseteq S$, we have $|C'| \leq 4|C^*|$. Moreover, the set C computed by the algorithm is a $(1+\varepsilon)$-approximation to C', and therefore we have $|C| \leq (1+\varepsilon)|C'| \leq (4+4\varepsilon)|C^*|$. By re-adjusting ε properly (e.g., by running the algorithm with $\varepsilon' = \varepsilon/4$), we get an approximation factor of $4+\varepsilon$ for the problem, for any $\varepsilon > 0$. In the special case of L_2 metric, Lemma 3 states that each disk in C^* can be covered by three disks of radius $r = \sqrt{3}/2$, and hence, the approximation factor of the algorithm is $3+\varepsilon$ in this case.

The communication complexity of the algorithm corresponds to the size of $T = \bigcup_{i=1}^{m} T_i$. For $1 \leq i \leq m$, let C_i^* and C_i' be optimal solutions to $\mathrm{UC}(S_i)$ and $\mathrm{UC}_\delta(S_i)$, respectively. Since $S_i \subseteq S$, we have $|C_i^*| \leq |C^*|$. Moreover, by Lemma 1, each unit disk in C_i^* can be covered by a constant number of disks of radius δ, and hence, $|C_i'| \leq c \cdot |C_i^*|$, for some constant $c \geq 1$. On the other hand, each C_i is an α-factor approximation to C_i', for some constant $\alpha \geq 1$, and thus, $|C_i| \leq \alpha|C_i'| \leq \alpha c|C_i^*| \leq \alpha c|C^*|$. As $|T_i| = |C_i|$, we have $|T| = \bigcup_{i=1}^{m} |T_i| \leq m \cdot \alpha c|C^*|$. Since $|C^*| = k$, the communication complexity of the algorithm is $O(mk)$.

For the runtime, we note that a $(1+\varepsilon)$-approximation to UC can be computed in $n^{O(1/\varepsilon)}$ time [10], and a constant-factor approximation to UC can be obtained in $O(n \log n)$ time [5]. The runtime of our algorithm on the i-th machine is therefore $O(|S_i| \log |S_i|)$, which sums to $O(|S| \log |S|) = O(n \log n)$ on all local machines, and amounts to $|T|^{O(1/\varepsilon)} = (mk)^{O(1/\varepsilon)}$ on the central machine. □

5 Capacitated Unit Clustering

In this section, we consider the capacitated version of the unit clustering problem, where each disk has a fixed capacity L. We present a distributed approximation algorithm for this version of the problem under any L_p metric, $p \geq 1$. The algorithm is presented in Algorithm 2. The first phase of the algorithm is similar to that of Algorithm 1, except that here, each point $t \in T_i$ is assigned a weight $w(t)$ which specifies the number of points t is representative for. These weights are then used in the second phase to properly limit the number of points assigned to each computed unit disk.

Algorithm 2 Capacitated Unit Clustering

Let $r = \sqrt{3}/2$ and $\delta = (1-r)/2$.
on each machine i $(1 \le i \le m)$ in parallel **do**
  Find an $O(1)$-approximation C_i to $\mathrm{UC}_\delta(S_i)$.
  Assign each point of S_i to one of its covering disks in C_i, with ties broken arbitrarily.
  For each disk $D \in C_i$, pick an arbitrary point $t \in S_i \cap D$, set its weight $w(t)$ to the number of points assigned to D, and add t to T_i.
  Send T_i to the central machine.
on the central machine **do**
  Let $T = \bigcup_{i=1}^{m} T_i$.
  Find a $(1+\varepsilon)$-approximation C_0 to $\mathrm{UC}_r(T)$.
  Assign each point of T to one of its covering disks in C_0, with ties broken arbitrarily.
  For each disk $D \in C_0$, add $\lceil w(D)/L \rceil$ copies of D to a set C, where $w(D)$ is the total weight of points assigned to D.
  Distribute point weights among their covering disks in C, so that each disk receives weight $\le L$. (A point weight may be split among two disks.)
  Increase the radii of disks in C from r to 1.
  return C.

Theorem 5 *Algorithm 2 is a $(4+\varepsilon)$-approximation algorithm for the capacitated unit clustering problem in the L_2 plane, and a $(5+\varepsilon)$-approximation algorithm for the problem under general L_p metric, $p \ge 1$. The runtime of the algorithm is $O(n \log n) + (mk)^{O(1/\varepsilon)}$, and its communication complexity is $O(mk)$, where n is the total number of points, m is the number of machines, and k is the size of an optimal solution.*

Proof. Let $S = \bigcup_{i=1}^{m} S_i$ be the input set in the plane, under a given L_p metric, $p \ge 1$. First, notice that the output of the algorithm, C, is a feasible solution. This is because each point in S is within distance $r + 2\delta = 1$ to the center of one of the disks in C, by an argument similar to what we used in Algorithm 1. Moreover, by our distribution of the weights among disks, no disk in C receives more than L points. Therefore, C is a feasible solution. The runtime and communication complexity of the algorithm are also implied by the same arguments used in the proof of Algorithm 1.

It only remains to prove the approximation factor of the algorithm. Let C^* be an optimal solution to the capacitated unit clustering problem on the set S, and let C' be an optimal solution to (uncapacitated) $\mathrm{UC}(S)$. Note that $|C'| \le |C^*|$. Moreover, $|C^*| \ge n/L$, because all n points in S are covered by $|C^*|$ disks of capacity L. According to the algorithm,

$$\begin{aligned} |C| &= \sum_{D \in C_0} \lceil w(D)/L \rceil \\ &\le \sum_{D \in C_0} (1 + w(D)/L) \\ &= |C_0| + n/L \\ &\le |C_0| + |C^*|. \end{aligned}$$

Moreover, according to the proof of Theorem 4, C_0 is a $(4+\varepsilon)$-approximation to C' under general L_p metric, and a $(3+\varepsilon)$-approximation to C' under L_2 metric. Therefore, $|C| \le (5+\varepsilon)|C^*|$ in general L_p metric, and $|C| \le (4+\varepsilon)|C^*|$ in the L_2 plane, which completes the proof. □

6 Lower Bounds

In this section, we provide lower bounds on the approximation factor of any distributed algorithm for the unit clustering problem in the L_2 plane under the composable coreset framework. We also prove a lower bound on the communication complexity of the distributed algorithms for the problem under this framework.

A *coreset algorithm* receives as input a sequence S of points, and returns as output a subset of S, called a coreset. We call a coreset algorithm *rotation-invariant* if for a fixed sequence S of points, it always returns the same coreset, even if the input is rotated in the plane.

Theorem 6 *The unit clustering problem in the L_2 plane admits no composable coreset with approximation factor better than 2. If the underlying coreset algorithm is rotation-invariant, the problem admits no α-composable coreset, for any $\alpha < 3$.*

Proof. Let $\mathcal{A}$ be the coreset algorithm used by local machines. Let S be a sequence of points evenly placed on a circle of radius $1/2$. We can pick S sufficiently large so that $|\mathcal{A}(S)| < |S|$. Then, by the pigeonhole principle, there exist two distinct subsequences T_1 and T_2 of S such that $\mathcal{A}(T_1) = \mathcal{A}(T_2)$. Assume w.l.o.g. that a point $v \in S$ is in T_1 but not in T_2. Since $\mathcal{A}(T_1) = \mathcal{A}(T_2)$, we have $v \notin \mathcal{A}(T_1)$. Let C_1 and C_2 be two concentric circles of radius 1 and $1+\varepsilon$, respectively, for some $\varepsilon > 0$, such that v is on the boundary of C_2, while other points lie inside C_1 (see Figure 5). Let u be the point on the boundary of C_1 furthest away from v.

Consider an instance with two partitions S_1 and S_2 (on two separate machines), where $S_1 = \{u\}$ and S_2 is either T_1 or T_2. If $S_2 = T_1$, at least two unit disks are needed to cover all the points as $d(u, v) > 2$. On the other hand, if $S_2 = T_2$, the whole input can be covered by a single unit disk, C_1. When $\mathcal{A}(S_2)$ is sent to the central machine, it cannot distinguish whether the original set has been T_1 or T_2. Therefore, any solution

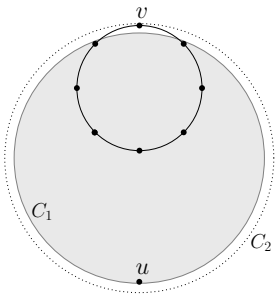


Figure 5: A lower bound example with two partitions.

returned by it must have at least two unit disks to ensure feasibility, causing an approximation factor of at least 2.

If $\mathcal{A}$ is rotation-invariant, we can obtain a stronger lower bound as follows. Define m copies of S rotated and evenly placed on the perimeter of C_2, as shown in Figure 6. Consider an input consisting of m partitions $S_1, \ldots, S_m$ where each partition corresponds to one of these copies, and can be a rotated copy of either T_1 or T_2. If all S_i's are of type T_1 and m is sufficiently large, then at least three unit disks are needed to cover all the points, in particular, those on the perimeter of C_2. On the other hand, if all S_i's are of type T_2, the whole input can be covered by a single unit disk, C_1. In both cases, the composable coresets received by the central machine are the same, since $\mathcal{A}(T_1) = \mathcal{A}(T_2)$ in each copy. Thus, the number of unit disks returned must be at least 3 to make sure the output is feasible. Therefore, the approximation factor cannot be better than 3. □

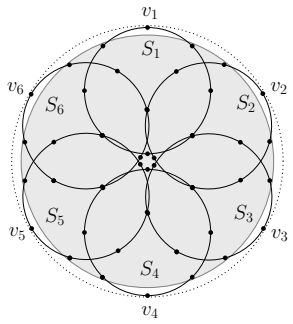


Figure 6: Data partitions on six machines.

The algorithms provided in this paper both have $O(mk)$ communication complexity. The following theorem shows that the communication complexity of our algorithms is indeed optimal.

Theorem 7 *Any distributed algorithm for the unit clustering problem under the composable coreset framework requires $\Omega(mk)$ communication, where m is the number of machines, and k is the size of an optimal solution.*

Proof. Let S_i be the set of points in the i-th machine ($1 \leq i \leq m$), and let k_i be the size of an optimal unit clustering for S_i. Suppose that all S_i's are far from each other, so that no disk covering a point in S_i can cover a point in S_j, for $j \neq i$. If the coreset sent by the i-th machine contains less than k_i points, the central machine receives no enough information to cover all the points in S_i, and hence, the final solution will not be feasible. Therefore, the number of points sent by the i-th machine must be at least k_i.

Now, consider the case where all machines have the same set of points, and hence, $k_i = k$ for all $1 \leq i \leq m$. By the argument provided above, each machine, independently from the others, must send at least k points to the central machine, and hence, the central machine receives at least mk points in this case. □

7 Conclusions

In this paper, we studied the unit clustering problem in a distributed settings, and presented approximation algorithms for both capacitated and uncapacitated versions of the problem in general L_p metric, $p \geq 1$. Our algorithms can be implemented in $O(1)$ rounds of MapReduce. Moreover, the composable coresets provided in this paper naturally lead to algorithms in the one-pass streaming model. In higher dimensions, our algorithms can be extended in a natural way to obtain constant factor approximations in any fixed d dimensions. It is interesting to see if the approximation factors of our algorithms can be improved, in particular, in the capacitated version.

References

[1] S. Aghamolaei, M. Farhadi, and H. Zarrabi-Zadeh. Diversity maximization via composable coresets. In *Proc. 27th Canad. Conf. Computat. Geom.*, page 43, 2015.

[2] A. Andoni, A. Nikolov, K. Onak, and G. Yaroslavtsev. Parallel algorithms for geometric graph problems. In *Proc. 46th Annu. ACM Sympos. Theory Comput.*, pages 574–583, 2014.

[3] M. Bateni, A. Bhaskara, S. Lattanzi, and V. Mirrokni. Distributed balanced clustering via mapping coresets. In *Proc. 27th Annu. Conf. Neural Info. Process. Sys.*, pages 2591–2599, 2014.

[4] P. Beame, P. Koutris, and D. Suciu. Communication steps for parallel query processing. In *Proc. 32nd ACM Sympos. Principles of Database Systems*, pages 273–284, 2013.

[5] A. Biniaz, P. Liu, A. Maheshwari, and M. Smid. Approximation algorithms for the unit disk cover problem in 2D and 3D. *Comput. Geom. Theory Appl.*, 60:8–18, 2017.

[6] T. M. Chan and H. Zarrabi-Zadeh. A randomized algorithm for online unit clustering. *Theory Comput. Sys.*, 45(3):486–496, 2009.

[7] K. Chen. On coresets for k-median and k-means clustering in metric and euclidean spaces and their applications. *SIAM J. Comput.*, 39(3):923–947, 2009.

[8] J. Dean and S. Ghemawat. MapReduce: simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113, 2008.

[9] L. Epstein, A. Levin, and R. van Stee. Online unit clustering: Variations on a theme. *Theoret. Comput. Sci.*, 407(1-3):85–96, 2008.

[10] T. Feder and D. Greene. Optimal algorithms for approximate clustering. In *Proc. 20th Annu. ACM Sympos. Theory Comput.*, pages 434–444, 1988.

[11] R. J. Fowler, M. S. Paterson, and S. L. Tanimoto. Optimal packing and covering in the plane are NP-complete. *Inform. Process. Lett.*, 12(3):133–137, 1981.

[12] T. Ghasemi and M. Razzazi. A PTAS for the cardinality constrained covering with unit balls. *Theoret. Comput. Sci.*, 527:50–60, 2014.

[13] M. T. Goodrich, N. Sitchinava, and Q. Zhang. Sorting, searching, and simulation in the MapReduce framework. In *Proc. 22nd Annu. Internat. Sympos. Algorithms Comput.*, pages 374–383, 2011.

[14] 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.

[15] P. Indyk, S. Mahabadi, M. Mahdian, and V. S. Mirrokni. Composable core-sets for diversity and coverage maximization. In *Proc. 33rd ACM Sympos. Principles of Database Systems*, pages 100–108, 2014.

[16] H. Karloff, S. Suri, and S. Vassilvitskii. A model of computation for MapReduce. In *Proc. 21st ACM-SIAM Sympos. Discrete Algorithms*, pages 938–948, 2010.

[17] V. Mirrokni and M. Zadimoghaddam. Randomized composable core-sets for distributed submodular maximization. In *Proc. 47th Annu. ACM Sympos. Theory Comput.*, STOC '15, pages 153–162, 2015.

[18] J. Tang, B. Hao, and A. Sen. Relay node placement in large scale wireless sensor networks. *Comput. Commun.*, 29(4):490–501, 2006.

[19] S. L. Tanimoto and R. J. Fowler. Covering image subsets with patches. In *Proc. 5th Internat. Conf. Pattern Recognition*, pages 835–839, 1980.

[20] Y. Wang. Topology control for wireless sensor networks. In *Wireless Sensor Networks and Applications*, pages 113–147. Springer, 2008.

Local Search for Geometric Partial Covering Problems

Tanmay Inamdar*

Abstract

Partial Set Cover problem is a generalization of Set Cover, where we are required to cover at least k out of n elements in the given set system, using the minimum number of sets. In geometric versions of Partial Set Cover, the sets are induced by geometric objects, and elements are points in $\mathbb{R}^d$. For many such versions, the previously best known approximation guarantees were $\Omega(1)$. On the other hand, the Local Search technique has been successful in obtaining PTASs [1] for related problems such as Geometric Set Cover and Geometric Maximum Coverage. Extending the analysis of Chaplick et al. [5] for the latter problem, we show that Local Search also gives a PTAS for various geometric versions of Partial Set Cover.

1 Introduction

We start with the unweighted Partial Set Cover (PSC) problem. The input to PSC is a tuple $(X, \mathcal{R}, k)$: X is a set of n elements, $\mathcal{R}$ is a collection of subsets of X such that $\bigcup \mathcal{R} = X$ [2], and $1 \leq k \leq n$ is the coverage requirement. The goal is to find a minimum-size collection $\mathcal{R}' \subseteq \mathcal{R}$, such that $|\bigcup \mathcal{R}'| \geq k$. Notice that the unweighted Set Cover (SC) problem is a special case of PSC with $k = n$, i.e., we are required to cover all elements in X in SC. Therefore, PSC, like SC, is NP-hard to approximate even within a factor of $o(\log n)$ ([8]). On the positive side, the approximation guarantees of $O(\log n)$ ([17, 23]), or f ([11]), for PSC match those for SC ([24]). Here, f denotes the maximum frequency of any element, i.e., the maximum number of sets an element is contained in.

The Maximum Coverage (MC) problem is closely related to PSC. The input to MC is $(X, \mathcal{R}, D)$. Here, $1 \leq D \leq |\mathcal{R}|$ denotes the budget on the size of the solution. The goal is to select a solution $\mathcal{R}' \subseteq \mathcal{R}$, where $|\mathcal{R}'| \leq D$, that covers the maximum number of elements. It is easy to see that the exact versions of PSC and MC are reducible to each other; however, these reductions are not approximation preserving, in general. Improving upon the classical $(1 - 1/e)$-approximation for MC is known to be NP-hard ([18]).

An interesting special case of SC is known as Geometric Set Cover (GSC). Here, the elements are usually points in $\mathbb{R}^d$ for a constant d, and the sets are induced by geometric objects in $\mathbb{R}^d$. Many interesting versions of GSC have been studied. A detailed discussion of the literature and techniques can be found in [15].

Now, we turn to the Geometric Partial Set Cover (GPSC) problem. Gandhi et al. [11] give a PTAS for partially covering points in $\mathbb{R}^2$ with unit disks, extending the "shifting technique" of Hochbaum and Maass [14], combined with a dynamic program. However, the collection of sets, $\mathcal{R}$, is not given explicitly – it consists of *all* unit disks in the plane. Chan and Hu [4] give a PTAS for the version of GPSC, where the sets are induced by a given set of unit squares in $\mathbb{R}^2$. This PTAS uses a clever dynamic programming based approach. However, their technique does not seem to extend to a more general class of objects – even for arbitrary squares. Inamdar and Varadarajan [15], show that the standard Linear Program for PSC can be rounded within a factor of $2\beta+2$. Here β is the approximation guarantee of an "LP-based" algorithm for the corresponding SC instance, which they use as a black box. Using this result, they obtain approximation guarantees that match the respective SC guarantees up to a constant factor for many instances of GPSC. However, an intrinsic drawback of this approach is that they cannot use an arbitrary approximation algorithm for SC (such as a Local Search algorithm) as a black box.

1.1 Local Search

The Local Search technique was first analyzed by Mustafa and Ray [21] in the context of GSC (also by Chan and Har-Peled [3] for Geometric Maximum Independent Set). Since then, it has been successful for obtaining PTASs for a variety of geometric problems. Before we discuss the details of this technique, we introduce the notion of sublinear separators for "sparse" graphs, which is central to the analysis of Local Search. Much of the following terminology is borrowed from [5].

*Department of Computer Science, The University of Iowa, tanmay-inamdar@uiowa.edu.

[1] PTAS stands for Polynomial Time Approximation Scheme, i.e., a $(1+\epsilon)$-approximation that runs in polynomial time for any constant $\epsilon > 0$.

[2] For any $\mathcal{R}' \subseteq \mathcal{R}$, we use the notational shorthand $\bigcup \mathcal{R}' \coloneqq \bigcup_{S \in \mathcal{R}'} S$.

Graph Separators

Let $G = (V, E)$ be an m vertex graph, and let $\alpha \in [\frac{1}{2}, 1)$ be a constant. A set $S \subseteq V$ is said to be an α-balanced separator for G, if $V \setminus S$ can be partitioned into $V_1 \cup V_2$, where (i) $|V_1|, |V_2| \leq \alpha \cdot m$, and (ii) there is no edge between a vertex in V_1 and a vertex in V_2. Let $f(m) = O(m^{1-\delta})$ be a non-decreasing sublinear function, with $0 < \delta < 1$. We say that a graph $G = (V, E)$ is f-separable, if for every subgraph $G' = (U', E')$ of G, there exists an α-balanced separator of size $f(|U'|)$, for some constant $\alpha \in [\frac{1}{2}, 1)$.

From a celebrated result of Lipton and Tarjan [20], it is known that the planar graphs are $\sqrt{m}$-separable. More generally, it is known that any minor-closed graph class is $\Theta(\sqrt{m})$-separable [1]. The following theorem is obtained by a recursive application of the sublinear separator theorem [10, 13] (as stated in [5]), the variants of which are often critical to the analyses of Local Search algorithms.

Theorem 1 ([10, 13]) *Fix a sublinear function f. There exist constants c_1, c_2 such that, for any f-separable m-vertex graph $G = (V, E)$ the following holds.*
For any integer $r \geq 1$, there is an integer $t = \Theta\left(\frac{m}{r}\right)$, such that V can be partitioned into disjoint sets $(B, V_1, V_2, \ldots, V_t)$ with the following properties.

1. *$N(V_i) \cap V_j = \emptyset$ for each $i \neq j$ and $B = \bigcup_i N(V_i)$* [3]*,*
2. *$|V_i \cup N(V_i)| \leq r$ for all i,*
3. *$|N(V_i)| \leq c_1 \cdot f(r)$ for each i, which also implies $|B| \leq \sum_{i=1}^{t} |N(V_i)| \leq \frac{c_2 m \cdot f(r)}{r}$.*

Standard Local Search Framework

Now, we give a brief overview of the standard framework for the design and analysis of the Local Search technique. We first define a key operation in any Local Search algorithm for SC.

Definition 1 (b-swap) *Let $\mathcal{R}' \subseteq \mathcal{R}$ be a feasible solution for the given SC instance $(X, \mathcal{R})$. For a given positive integer b, a b-swap is an operation of replacing $\mathcal{R}'$ with $(\mathcal{R}' \setminus \mathcal{R}_1) \cup \mathcal{R}_2$, for any collections $\mathcal{R}_1, \mathcal{R}_2$ of sets such that:*
1. $\mathcal{R}_1 \subseteq \mathcal{R}'$ with $|\mathcal{R}_1| \leq b$,
2. $\mathcal{R}_2 \subseteq \mathcal{R}$ with $|\mathcal{R}_2| \leq |\mathcal{R}_1| - 1$.
Furthermore, if the resulting solution $(\mathcal{R}' \setminus \mathcal{R}_1) \cup \mathcal{R}_2$ is feasible, the b-swap is said to be feasible.

Note that a feasible b-swap is a "local operation" that decreases the size of the solution, while still covering X. Starting from an arbitrary feasible solution $\mathcal{R}' \subseteq \mathcal{R}$, in each iteration, we repeatedly check for the existence of a feasible b-swap, and perform it, if it is possible to do so. If no such local improvement is possible, the resulting solution is called a b-locally optimal solution. For a constant b, each iteration takes polynomial time, and the number of iterations is bounded by $|\mathcal{R}|$. Therefore, the overall running time of the algorithm is polynomial in the input size.

Let $\mathcal{O}$ be an optimal solution, and $\mathcal{A}$ be a candidate feasible solution. A key idea in the analysis of any Local Search algorithm for SC is the concept of an exchange graph, which relates the sets covering each element of X in the two solutions. It is known that the class of exchange graphs defined by many geometric set systems admit sublinear separators. Suppose $|\mathcal{A}|$ is "much larger" than $|\mathcal{O}|$, then, by using Theorem 1 on the exchange graph, one can show the existence of a feasible small swap, i.e., that $\mathcal{A}$ is not a locally optimal solution.

Given the success of Local Search in the context of GSC and its variants (e.g., [12, 21, 22]), it is natural to wonder whether it can be used to obtain PTASs for versions of GPSC. Note that the notions of b-swap, feasible b-swap, and b-local optimality can be naturally generalized in the context of (G)PSC, recalling that a solution is feasible if it covers at least k elements. However, observe that in GPSC, the solutions $\mathcal{O}$ and $\mathcal{A}$ may cover different sets of at least k elements. Therefore, the standard analysis of the Local Search algorithm as described above, does not work for GPSC. Note that a similar issue also exists in the context of MC. Nevertheless, Chaplick et al. [5] were able to show that a version of the Local Search algorithm gives a PTAS for a variety of Geometric MC problems. We adapt their analysis in the context of GPSC. We first discuss our result and its applications, before we delve into a detailed comparison of our work with theirs in Section 1.4.

1.2 Result

Now we describe the general idea of our algorithm for GPSC and its analysis. We use the Local Search algorithm described in the previous section, the only difference being the definition of feasibility of a solution – a collection $\mathcal{R}' \subseteq \mathcal{R}$ is feasible if it covers at least k elements of X.

Now, let $\mathcal{A}$ be a candidate feasible solution such that $|\mathcal{A}| > (1 + \epsilon) \cdot |\mathcal{O}|$, where $\mathcal{O}$ is an optimal solution, and $\epsilon > 0$ is a constant. We define an exchange graph with respect to these two solutions as follows.

Definition 2 (Exchange Graph) *An exchange graph is a graph $G = (V, E)$, such that*
(1) $V = \mathcal{O} \cup \mathcal{A}$, and
(2) For any element $e \in X$ that is covered by both solutions $\mathcal{O}$ and $\mathcal{A}$, there is an edge $\{P, Q\} \in E$, such that

[3] For any vertex $w \in V$, let $N(w)$ denote the set of neighbors of w. For a set $W \subseteq V$, the "outer neighborhood" of W, $N(W)$, is defined as $\left(\bigcup_{w \in W} N(w)\right) \setminus W$. When we use this notation in our analysis, it will always be the case that $N(W) \subseteq B$.

$P \in \mathcal{A}, Q \in \mathcal{O}$ *and* $e \in P \cap Q$.

In the analysis, we construct such an exchange graph G. We assume that G is f-separable for some sublinear function f. Then, we use Theorem 1 on an exchange graph, where r is chosen carefully in terms of ϵ. Then, using the properties of this exchange graph, we prove the existence of a local swap of a small size. We state this in the following (slightly informal) theorem, which is proved in Section 2.

Theorem 2 *Consider a* PSC *instance* $(X, \mathcal{R}, k)$. *Let* $\mathcal{O}$ *be an optimal solution and* $\mathcal{A}$ *be a feasible solution for this instance, such that* $|\mathcal{A}| > (1+\epsilon) \cdot |\mathcal{O}|$ *for some constant* $\epsilon > 0$. *Furthermore, suppose that the corresponding exchange graph* G *is* f*-separable for some sublinear function* f. *Then, for some positive constant* $b = b(f, \epsilon)$, *there exists a feasible* b*-swap from* $\mathcal{A}$. *That is,* $\mathcal{A}$ *is not a* b*-locally optimal solution.*

We note that, if the exchange graph is f-separable for $f(t) = O(t^{1-\delta})$, where $0 < \delta < 1$ is a constant, then the running time of the PTAS is $m^{O(\epsilon^{-2/\delta})}$, where m is the number of sets in $\mathcal{R}$. We contrast this with $m^{O(\epsilon^{-1/\delta})}$ running time of the Local Search algorithm for GSC, which is known to be tight for various set systems ([16]). It is an interesting question whether the exponent in the running time of our PTAS can be improved.

1.3 Applications

Before we discuss the consequences of Theorem 2, we define some variants of GPSC that are also amenable to the Local Search algorithm.

Partial Hitting Set Let $\mathcal{S}$ be a set of geometric objects in $\mathbb{R}^d$ and P be a set of points in $\mathbb{R}^d$. A point $p \in P$ is said to *hit* an object $S \in \mathcal{S}$ if $p \in S$. The instance $(\mathcal{S}, P, k)$ of the Partial Hitting Set problem asks to find a minimum-size set $P' \subseteq P$ of points that hits at least k objects from $\mathcal{S}$.

Partial Terrain Guarding Let T be a 1.5D terrain, i.e., an x-monotone polygonal chain in $\mathbb{R}^2$. Let $X, Y \subseteq T$ be finite sets on the terrain T. For any two points $x, y \in T$, x is said to *see* y if no point on the line segment $\overline{xy}$ lies below the terrain T. The instance (X, Y, k) of the Partial Terrain Guarding problem asks to find a minimum-size set $Y' \subseteq Y$ of points that sees at least k out of n points from X.

We borrow the following definition of r-admissible regions from [5]. A set of regions (each bounded by a closed Jordan curve) in $\mathbb{R}^2$ is called r-admissible, if for any two regions q_1, q_2 in the set, the curves bounding them cross at most $s \leq r$ times, and the regions $q_1 \setminus q_2$, and $q_2 \setminus q_1$ are connected. Here, $s \leq r$ are positive even integers. This class of objects includes translates of convex objects. In the special case where $r = 2$, the objects are also known as pseudo-disks (which includes arbitrary disks, squares, and axis-parallel rectangles of same height).

As a consequence of Theorem 2, we obtain new PTASs for various versions of GPSC and its variants, via Local Search. The f-separability of the exchange graphs for GPSC, follows from that in the corresponding SC versions, already known in the literature. We give the relevant references next to each result, which show the f-separability of the corresponding exchange graph in the full covering version.

Theorem 3 *Local Search gives a PTAS for the following geometric partial covering problems:*

1. *Partially covering points by halfspaces in* $\mathbb{R}^3$ *([21]).*
2. *Partially covering points by* r*-admissible regions in* $\mathbb{R}^2$ *([22]).*
3. *Partially hitting* r*-admissible regions by points in* $\mathbb{R}^2$ *([21]).*
4. *Partially hitting halfspaces by points in* $\mathbb{R}^3$ *([21]).*
5. *Partial Terrain Guarding ([19]).*

1.4 Comparison to the work of Chaplick et al. [5]

Here, we compare and contrast our result with that of [5] for MC. Recall that, in MC, the goal is to cover the maximum number of elements using at most D sets. First, we sketch the Local Search algorithm used in [5] for MC. Here, we start with an arbitrary collection $\mathcal{R}' \subseteq \mathcal{R}$ of size exactly D. In each iteration, we check for the existence of a local swap that improves the number of elements covered. More precisely, for a positive integer b, we check whether there exists some $\mathcal{R}'' := (\mathcal{R}' \setminus \mathcal{R}_1) \cup \mathcal{R}_2$, where $|\mathcal{R}_1| = |\mathcal{R}_2| \leq b$, and $|\bigcup \mathcal{R}''| > |\bigcup \mathcal{R}'|$. We iteratively perform such b-swaps while it is possible to do so.

Note that in PSC, a b-swap reduces the number of sets in the solution, while covering at least k elements. On the other hand, in MC, the corresponding operation increases the number of elements covered, while keeping the number of sets at most D. Consequently, the respective analyses of the Local Search algorithms are different, which we informally discuss next.

Chaplick et al. [5] also use a decomposition $(B, V_1, V_2, \ldots, V_t)$ of the exchange graph in their analysis. Each *part* V_i corresponds to a possible swap in the analysis (we also use a similar notion, which is formalized in the proof). Notice that during such a swap, we may lose the coverage of some elements, while we may win

coverage of some elements [4]. Using a careful counting argument, they show that in at least one of these swaps, the number of elements "lost" is much smaller as compared to the number elements "won". However, this swap may not be feasible, i.e., the number of sets in the resulting solution may be slightly larger than D. This is because, $|\mathcal{R}_2| > |\mathcal{R}_1|$ using the notation from above. Nevertheless, they show that for a carefully chosen subset $\mathcal{R}'_2 \subseteq \mathcal{R}_2$, the corresponding swap has the desired properties.

In our analysis, also, each V_i is associated with a possible swap. However, some swaps may not decrease the size of the solution, or they may result in an infeasible solution. We then show (cf. Lemma 5) that the parts V_i can be combined into slightly larger parts U_i, such that all the associated swaps are of small size, and they result in a solution of smaller size. However, not all swaps may be feasible, i.e., cover at least k elements. Then, we use a version of the counting argument from [5] (referenced above), to show in Lemma 6 that at least one of the swaps is feasible, and hence $\mathcal{A}$ is not locally optimal.

We note that Chaplick et al. [5] prove a "color-balanced" version of the separator theorem. Informally speaking, this provides an additional guarantee that in each part V_i, the "local ratio" of the number of sets in $\mathcal{O} \cap V_i$ to $\mathcal{A} \cap V_i$ is close to the "global ratio" of $\mathcal{O}$ to $\mathcal{A}$, up to a small error. Even though this small error does not pose a problem in their analysis for MC, it is an issue in the context of PSC. In particular, the small error may correspond to an increase in the size of the solution after the swap. Therefore, we cannot use the color-balanced separator theorem in our analysis. At a high level, the Combination Lemma (Lemma 5) plays a similar role in our analysis, but the details are different.

Thus, the kinds of challenges faced in both proofs are different. Despite this, we are able to extend the ideas from [5] for PSC. Interestingly, in Section 3, we show how to obtain a PTAS for MC, using a PTAS for the corresponding version of PSC as a black box, which can be seen as an alternative proof for the results in [5]. However, we are not aware of a similar result in the opposite direction.

Chaplick et al. [5] observe that their algorithm also gives a PTAS for "Maximum Coverage" versions of Vertex Cover and Dominating Set problems for the graphs that admit sublinear separators – this includes the intersection graphs of homothetic copies of convex objects, e.g., arbitrary squares, regular k-gons etc. Indeed, our proof also shows that Local Search is a PTAS for the corresponding partial covering versions. However, it is already known from the previous results ([9, 11]) that these problems admit a PTAS on certain "sparse" graph classes, such as planar graphs. Therefore, these results about geometric partial covering problems were already implied by the constructions of sparse graphs for the respective full coverage problems ([12, 22]).

[4] Note that the elements "won" by adding $\mathcal{R}_2$ may include a subset of elements that were "lost" as a result of the removal of $\mathcal{R}_1$.

2 Analysis of Local Search

In this section, we prove Theorem 2.

Let $\mathcal{A}$ be an arbitrary feasible solution and $\mathcal{O}$ be an optimal solution, such that $|\mathcal{A}| = \alpha' \cdot |\mathcal{O}|$, for some $\alpha' > 1$. Notice that both $|\mathcal{A}|$ and $|\mathcal{O}|$ cover at least k elements. We assume that the solution $\mathcal{A}$ is *minimal* w.r.t. removal of sets, i.e., no set in $\mathcal{A}$ can be removed while still covering at least k elements. Note that $\mathcal{O}$ is also minimal, since it is an optimal solution.

First, we assume that the number of elements covered by the both solutions is exactly k, i.e., $|\bigcup \mathcal{A}| = |\bigcup \mathcal{O}| = k$. This can be done by deleting arbitrary elements from $\bigcup \mathcal{A}$ and $\bigcup \mathcal{O}$, if necessary. Note that this process does not change the sizes of the two solutions, since this would contradict the minimality of the solutions $\mathcal{A}$ and $\mathcal{O}$. Next, it is easy to see that following inequality holds, given the assumption on the sizes of the two solutions: $|\mathcal{A} \setminus \mathcal{O}| = \alpha \cdot |\mathcal{O} \setminus \mathcal{A}|$, where $\alpha \geq \alpha'$. Furthermore, observe that, $|\bigcup(\mathcal{A} \setminus \mathcal{O})| = |\bigcup(\mathcal{O} \setminus \mathcal{A})| = k' \leq k$, and that the solutions $\mathcal{A} \setminus \mathcal{O}$ and $\mathcal{O} \setminus \mathcal{A}$ are also minimal for k'. Therefore, by renaming $\mathcal{A}, \mathcal{O}, k$, if necessary, we assume that $\mathcal{A}$ and $\mathcal{O}$ are disjoint, minimal, and each solution covers exactly k elements.

Now, we construct an exchange graph $G = (V, E)$ with respect to the two solutions $\mathcal{A}$, and $\mathcal{O}$, as defined above. By assumption, G is f-separable for some sublinear function f. Let $m = |\mathcal{A}| + |\mathcal{O}|$ denote the number of vertices in the exchange graph. Since the vertices in the exchange graph correspond to sets, we will use the terms 'set' and 'vertex' interchangeably. Let r be a positive integer constant to be fixed later. We use Theorem 1 to obtain a decomposition $(B, V_1, \ldots, V_t)$ with this r.

Let c_3 be an absolute constant such that $t \leq \frac{c_3 \cdot m}{r}$, and let $q(r) \coloneqq 2c_2 f(r) + 2c_3 + \frac{r(r+1)}{m}$. Assuming $m \gg r$ and since $f(r)$ is a sublinear function in r, we assume that r is large enough so that $r - q(r) \geq r/2$.

Recall that $|\mathcal{A}| = \alpha \cdot |\mathcal{O}|$, and now suppose that $\alpha > 1 + \frac{4q(r)}{r}$. We will show that $\mathcal{A}$ is not a b-locally optimal solution, by showing the existence of a b-swap, where $b = 2r(r+1)$. First, we show the following technical lemma, which, informally speaking, shows that the "discrepancy" in the number of sets in $\mathcal{A}$ as compared to $\mathcal{O}$, is large enough.

Lemma 4

$$\sum_{i \in [t]} (|\mathcal{A} \cap V_i| - |\mathcal{O} \cap V_i| - |\mathcal{O} \cap N(V_i)|) \geq 2t + r + 1.$$

Proof. Since $(B, V_1, V_2, \ldots, V_t)$ is a partition of $V =$

$\mathcal{A} \cup \mathcal{O}$,

$$|\mathcal{A} \cap B| + \sum_{i=1}^{t} |\mathcal{A} \cap V_i| = \alpha \cdot |\mathcal{O} \cap B| + \alpha \cdot \sum_{i=1}^{t} |\mathcal{O} \cap V_i|.$$

Rearranging this, we get,

$$\begin{aligned}
&\sum_{i=1}^{t} (|\mathcal{A} \cap V_i| - |\mathcal{O} \cap V_i|) \\
&= (\alpha - 1) \cdot |\mathcal{O}| + |\mathcal{O} \cap B| - |\mathcal{A} \cap B| \\
&\geq (\alpha - 1) \cdot \frac{m}{\alpha + 1} - |B| \\
&\qquad (\because |\mathcal{A}| = \alpha |\mathcal{O}| \text{ and } |\mathcal{A}| + |\mathcal{O}| = m)
\end{aligned}$$

Therefore,

$$\begin{aligned}
&\sum_{i=1}^{t} (|\mathcal{A} \cap V_i| - |\mathcal{O} \cap V_i| - |\mathcal{O} \cap N(V_i)|) \\
&\geq m \cdot \frac{\alpha - 1}{\alpha + 1} - m \cdot \frac{2c_2 f(r)}{r} \qquad (1)
\end{aligned}$$

Where, the last step follows from Property 3 of Theorem 1, which implies that, $|B| \leq \frac{c_2 m f(r)}{r}$, and that $\sum_i |N(V_i) \cap \mathcal{O}| \leq \sum_i |N(V_i)| \leq \frac{c_2 m f(r)}{r}$.

It is sufficient to prove that the right hand side of (1) is at least $\frac{2c_3 m}{r} + r + 1 \geq 2t + r + 1$. Equivalently, we want to prove that,

$$\frac{\alpha - 1}{\alpha + 1} \geq \frac{2c_2 f(r) + 2c_3 + r(r+1)/m}{r} = \frac{q(r)}{r}.$$

Since $r - q(r) \geq r/2$, we have $\alpha - 1 > \frac{4q(r)}{r} \geq \frac{2q(r)}{r - q(r)}$. Therefore, $\frac{r}{q(r)} > 1 + \frac{2}{\alpha - 1} = \frac{\alpha + 1}{\alpha - 1}$. □

Let $(P_1, P_2, \ldots, P_l)$ be any partition of $V \setminus B$. For every part P_i, we define the *discrepancy* of P_i as $\mu(P_i) \coloneqq |P_i \cap \mathcal{A}| - |P_i \cap \mathcal{O}| - |N(P_i) \cap \mathcal{O}|$. Furthermore, let $\lambda(P_i) \coloneqq \mu(P_i) - 1$. Using this notation, the result of Lemma 4 can be restated as $\sum_{i \in [t]} \mu(V_i) \geq 2t + r + 1$, which also implies that $\sum_{i \in [t]} \lambda(V_i) \geq t + r + 1$. Now in the following lemma, we show that the parts V_i can be combined into slightly larger parts U_i, such that the discrepancy of each U_i is strictly positive.

Lemma 5 *Suppose we are given a partition $(B, V_1, V_2, \ldots, V_t)$ of the vertex set $V = \mathcal{A} \cup \mathcal{O}$ that satisfies the properties of Theorem 1. Furthermore, suppose $\sum_i \mu(V_i) \geq 2t + r + 1$. Then, there exists a partition $(B, U_1, U_2, \ldots, U_s)$ that satisfies the following properties.*

1. *$|U_i \cup N(U_i)| \leq 2r(r+1)$ for all $i \in [s]$,*
2. *$\mu(U_i) > 0$ for all $i \in [s]$.*

Proof. In order to prove this lemma, we need the following claim, which is a straightforward adaptation of an analogous lemma in [2] (also [6, 7]). For the sake of completeness, we prove it in the appendix.

Claim 1 *Let $N = \{n_1, n_2, \ldots, n_t\}$ be a set of t integers, where $|n_i| \leq r+1$ for any $n_i \in N$, and $\sum_{i \in t} n_i \geq t+r+1$. Then, there exists a partition $\mathcal{P} = \{P_1, P_2, \ldots, P_s\}$ of the index set $[t]$, satisfying the following properties:*

1. *For any $i \in [s]$, $1 \leq |P_i| \leq 2(r+1)$,*
2. *For any $i \in [s]$, $\sum_{j \in P_i} n_j \geq 0$.*

Recall that for any $i \in [t]$, $\lambda(V_i) = \mu(V_i) - 1 \leq |V_i| \leq r$, using property 2 of Theorem 1. Therefore, $|\lambda(V_i)| \leq r+1$. Also, recall that $\sum_{i \in [t]} \lambda(V_i) \geq t + r + 1$. Therefore, setting $n_i \coloneqq \lambda(V_i)$, and $N = \{\lambda(V_1), \lambda(V_2), \ldots, \lambda(V_t)\}$ satisfies the properties required to apply Claim 1. We then obtain a partition $\mathcal{P}$ of the index set $[t]$ with the properties guaranteed in Claim 1.

Now, for any $P_i \in \mathcal{P}$, let $U_i = \bigcup_{j \in P_i} V_j$. From property 1, we know that each U_j is obtained by taking a union of at least one and at most $r + 1$ V_j's from the original partition. That is, the original partition $(V_1, V_2, \ldots, V_t)$ of the set $V \setminus B$, is a *refinement* of $(U_1, U_2, \ldots, U_s)$. Furthermore, since, for any $j \in [t]$ $|V_i| \leq r$, the first property of the lemma is satisfied.

For any $i \in [s]$, $N(U_i) \cap \mathcal{O} = \bigcup_{j \in P_i} (N(V_j) \cap \mathcal{O})$. Therefore, $|N(U_i) \cap \mathcal{O}| \leq \sum_{j \in P_i} |N(V_i) \cap \mathcal{O}|$. Now, recalling the definition of $\mu(\cdot)$, it holds that, $\mu(U_i) \geq \sum_{j \in P_i} \mu(V_j)$. Recall that $\lambda(\cdot) = \mu(\cdot) - 1$. Therefore, $\lambda(U_i) \geq \sum_{j \in P_i} \lambda(V_j)$.

Now, using the second property of Claim 1, we have that $\sum_{j \in P_i} \lambda(V_j) \geq 0$. Therefore, $\lambda(U_i) = \mu(U_i) - 1 \geq 0$, which proves the second property. □

The goal of the rest of the proof is to show the existence of a feasible swap of a small size. This part of the proof is very similar to that in [5], adapted in the context of PSC.

First, we define some notation. Let $\mathcal{A}_i = U_i \cap \mathcal{A}$, and $\mathcal{O}_i = U_i \cap \mathcal{O}$ denote the sets in the part $U_i : i \in [s]$, from the solutions $\mathcal{A}$ and $\mathcal{O}$ respectively. Furthermore, let $\overline{\mathcal{O}}_i = \mathcal{O}_i \cup (N(U_i) \cap \mathcal{O})$ be the optimal vertices in U_i or on the boundary of U_i. Notice that $\mathcal{A}_i, \mathcal{O}_i, \overline{\mathcal{O}}_i$ are collections of sets in $\mathcal{A} \cup \mathcal{O}$. Furthermore, we will restrict our attention to the following candidate swaps: $(\mathcal{A}_i, \overline{\mathcal{O}}_i)$ for different $i \in [s]$, i.e., replacing $\mathcal{A}$ with $(\mathcal{A} \setminus \mathcal{A}_i) \cup \overline{\mathcal{O}}_i$.

Now, we define some subsets of elements in $\bigcup(\mathcal{A} \cup \mathcal{O})$. For any $i \in [s]$, let $Z_i \coloneqq \bigcup (\mathcal{A} \setminus \mathcal{A}_i)$ denote the elements that remain covered even if all sets in $\mathcal{A}_i$ are removed from $\mathcal{A}$. Let $Z \coloneqq \bigcap_{i \in [s]} Z_i$ denote the elements that are covered by $\mathcal{A}$ but not exclusively by any particular $\mathcal{A}_i$. Observe that $\bigcup(B \cap \mathcal{A}) \subseteq Z$. Let $L_i \coloneqq (\bigcup \mathcal{A}_i) \setminus Z$ be the elements that are "lost" while swapping out sets in $\mathcal{A}_i$. Finally, let $W_i \coloneqq (\bigcup \overline{\mathcal{O}}_i) \setminus Z$ denote the elements that are "won" after swapping in sets in $\overline{\mathcal{O}}_i$ for the sets in $\mathcal{A}_i$. We now prove the following two claims.

Claim 2

$$0 < \sum_{i\in[s]} |L_i| \le \left|\bigcup \mathcal{A}\right| - |Z|.$$

Proof. Notice that $\mu(U_i) > 0$ for all $i \in [s]$, therefore, $\mathcal{A}_i \neq \emptyset$ for all $i \in [s]$. We claim that $L_i \neq \emptyset$ for all $i \in [s]$. Suppose we have $L_i = (\bigcup \mathcal{A}_i) \setminus Z = \emptyset$, which means that $\bigcup \mathcal{A}_i \subseteq Z$, i.e., the elements in $\bigcup \mathcal{A}_i$ are also covered by $\bigcup(\mathcal{A} \setminus \mathcal{A}_i)$. Therefore, we can remove the sets in $\mathcal{A}_i$, which decreases the size of $\mathcal{A}$ by at least 1 (since $\mathcal{A}_i \neq \emptyset$), without decreasing the number of elements covered. This contradicts the minimality of the solution $\mathcal{A}$. This proves the first inequality.

Now we prove the second inequality. Note that $Z \subseteq \bigcup \mathcal{A}$. Furthermore, the sets L_i's are pairwise disjoint. Therefore, an element $e \in \bigcup_i L_i$ is not contained in Z. The left hand side is equal to the number of such elements, whereas each such element contributes exactly 1 to the right hand side. □

Claim 3

$$\sum_{i\in[s]} |W_i| \ge \left|\bigcup \mathcal{O}\right| - |Z|.$$

Proof. We consider different cases for an element $e \in (\bigcup \mathcal{O}) \cup Z$. Note that these are the only elements that play a role in this inequality.

1. $e \in Z$.

An element in Z cannot belong to any W_i by definition, and therefore contributes 0 to the LHS. If $e \in \bigcup \mathcal{O}$, it contributes 0 to the RHS. Otherwise, it contributes -1.

2. $e \in \bigcup \mathcal{O} \setminus \bigcup \mathcal{A}$.

Such an element belongs to at least one W_i (since every set in $\mathcal{O}$ belongs to at least one $\overline{\mathcal{O}}_i$), and therefore contributes at least 1 to the LHS. Since $e \notin Z$, it contributes exactly 1 to the RHS.

3. $e \in \bigcup \mathcal{O} \cap \bigcup \mathcal{A}$, but $e \notin Z$.

Since $e \notin Z$, it cannot be covered by a set in $B \cap \mathcal{A}$. Suppose $e \in S$, where $S \in \mathcal{A}_i$ for some i. Note that all sets in $\mathcal{A}$ covering e must belong to $\mathcal{A}_i$ – for otherwise $e \in Z$, which is a contradiction. Now, since e is covered by both solutions, there exists an edge $\{S', T\}$ in the exchange graph, where $S' \in \mathcal{A}$, $T \in \mathcal{O}$, and $e \in S' \cap T$. Therefore, $T \in \overline{\mathcal{O}}_i$, which implies that $e \in W_i$, contributing at least 1 to the LHS. On the other hand, $e \in \bigcup \mathcal{O} \setminus Z$, so it contributes exactly 1 to the RHS. □

In the following Lemma, we show that, for some $i \in [s]$, replacing $\mathcal{A}$ with $(\mathcal{A}\setminus\mathcal{A}_i)\cup\overline{\mathcal{O}}_i$ results in a feasible solution of smaller cost.

Lemma 6 *There exists an index $i \in [s]$ such that:*

1. $\left|(\mathcal{A} \setminus \mathcal{A}_i) \cup \overline{\mathcal{O}}_i\right| < |\mathcal{A}|$, *and*
2. $\left|\bigcup\left((\mathcal{A} \setminus \mathcal{A}_i) \cup \overline{\mathcal{O}}_i\right)\right| \ge k$.

Proof. For any $i \in [s]$, we have that $\mu(U_i) = |\mathcal{A}_i| - |\overline{\mathcal{O}}_i| > 0$. Therefore, the first property holds for all $i \in [s]$. Now we show that there exists some $i \in [s]$ for which the second property holds.

From Claim 2 and Claim 3, we have the following.

$$\max_{i\in[s]} \frac{|W_i|}{|L_i|} \ge \frac{\sum_{i\in[s]} |W_i|}{\sum_{i\in[s]} |L_i|} \ge \frac{|\bigcup \mathcal{O}| - |Z|}{|\bigcup \mathcal{A}| - |Z|} = \frac{k - |Z|}{k - |Z|} = 1.$$

Therefore, for some $i \in [s]$, $|W_i| \ge |L_i|$, which implies the second property. □

Since $|\mathcal{A}_i|, |\overline{\mathcal{O}}_i| \le 2r(r+1)$, this lemma shows the existence of a feasible $2r(r+1)$-swap, assuming that $|\mathcal{A}| > (1 + \frac{4q(r)}{r}) \cdot |\mathcal{O}|$. This concludes the proof of Theorem 2.

3 A PTAS for Maximum Coverage Problem

In this section, we show how to obtain a $(1-\delta)$-approximate solution for MC, using the PTAS for the corresponding version of PSC. Here, $\delta \in (0,1)$ is assumed to be a constant. Let $(X, \mathcal{R}, D)$ be the given MC instance. Recall that the goal is to cover the maximum number of elements from X, using at most D sets from $\mathcal{R}$. By trying all possible values, suppose we know the value of OPT — the number of elements covered by an optimal solution — for the MC instance $(X, \mathcal{R}, D)$. Now, we use the PTAS from the previous section for the PSC instance $(X, \mathcal{R}, OPT)$ for some ϵ (to be fixed later), and obtain a feasible solution $\mathcal{R}' \subseteq \mathcal{R}$.

We first claim that $|\mathcal{R}'| \le (1+\epsilon) \cdot D$. This is because, an optimal solution for the MC instance covers OPT points using at most D sets, therefore it is a feasible solution for the PSC instance $(X, \mathcal{R}, OPT)$. Now, if $|\mathcal{R}'| \le D$, then $\mathcal{R}'$ is an optimal solution for MC, and we are done. Otherwise, $|\mathcal{R}'| > D$. We arbitrarily partition $\mathcal{R}'$ into $t = \lceil 1+\frac{1}{\epsilon} \rceil$ non-empty collections $\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_t$, such that $0 < |\mathcal{R}_1| \le |\mathcal{R}_2| = \ldots |\mathcal{R}_t| = \lceil \epsilon D \rceil$.

Let $i \in [t]$ be an index such that removal of $\mathcal{R}_i$ results in the least loss in the coverage. More formally, for any $j \in [t]$, let $\ell_j := |\bigcup \mathcal{R}'| - |\bigcup(\mathcal{R}' \setminus \mathcal{R}_j)|$ be the number of elements lost by removal of $\mathcal{R}_i$, and let $i = \arg\min_{j\in[t]} \ell_j$. By an averaging argument, we must have that $\ell_i \le OPT/t$. Therefore, $\mathcal{R} \setminus \mathcal{R}_i$ has at most D sets, and it covers at least $OPT \cdot (1 - \frac{1}{t}) \ge OPT \cdot (1 - \frac{\epsilon}{1+\epsilon})$ elements. Therefore, it suffices to choose ϵ such that $\frac{\epsilon}{1+\epsilon} = \delta$.

Acknowledgment I thank my advisor Prof. Kasturi Varadarajan for several valuable discussions and suggestions, and for pointing out the connection to the Maximum Coverage problem in Section 3. I also thank the reviewers for their comments.

References

[1] Noga Alon, Paul Seymour, and Robin Thomas. A separator theorem for nonplanar graphs. *Journal of the American Mathematical Society*, 3(4):801–808, 1990.

[2] Sayan Bandyapadhyay and Kasturi R. Varadarajan. On Variants of k-means Clustering. In *32nd International Symposium on Computational Geometry, SoCG 2016, June 14-18, 2016, Boston, MA, USA*, pages 14:1–14:15, 2016.

[3] Timothy M. Chan and Sariel Har-Peled. Approximation Algorithms for Maximum Independent Set of Pseudo-Disks. *Discrete & Computational Geometry*, 48(2):373–392, 2012.

[4] Timothy M. Chan and Nan Hu. Geometric red-blue set cover for unit squares and related problems. *Comput. Geom.*, 48(5):380–385, 2015.

[5] Steven Chaplick, Minati De, Alexander Ravsky, and Joachim Spoerhase. Approximation Schemes for Geometric Coverage Problems. In *26th Annual European Symposium on Algorithms, ESA 2018, August 20-22, 2018, Helsinki, Finland*, pages 17:1–17:15, 2018.

[6] Vincent Cohen-Addad and Claire Mathieu. Effectiveness of local search for geometric optimization. In *31st International Symposium on Computational Geometry (SoCG 2015)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2015.

[7] Vincent Cohen-Addad, Philip N Klein, and Claire Mathieu. Local search yields approximation schemes for k-means and k-median in euclidean and minor-free metrics. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 353–364. IEEE, 2016.

[8] Irit Dinur and David Steurer. Analytical Approach to Parallel Repetition. In *Proceedings of the Forty-sixth Annual ACM Symposium on Theory of Computing*, STOC '14, pages 624–633, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2710-7.

[9] Fedorr V Fomin, Daniel Lokshtanov, and Saket Saurabh. Excluded grid minors and efficient polynomial-time approximation schemes. *Journal of the ACM (JACM)*, 65(2):10, 2018.

[10] Greg N. Frederickson. Fast Algorithms for Shortest Paths in Planar Graphs, with Applications. *SIAM J. Comput.*, 16(6):1004–1022, 1987.

[11] Rajiv Gandhi, Samir Khuller, and Srinivasan Aravind. Approximation algorithms for partial covering problems. *J. Algorithms*, 53(1):55–84, 2004.

[12] Matt Gibson and Imran A. Pirwani. Algorithms for Dominating Set in Disk Graphs: Breaking the log$n$ Barrier - (Extended Abstract). In *Algorithms - ESA 2010, 18th Annual European Symposium, Liverpool, UK, September 6-8, 2010. Proceedings, Part I*, pages 243–254, 2010.

[13] Monika Rauch Henzinger, Philip N. Klein, Satish Rao, and Sairam Subramanian. Faster Shortest-Path Algorithms for Planar Graphs. *J. Comput. Syst. Sci.*, 55(1):3–23, 1997.

[14] Dorit S. Hochbaum and Wolfgang Maass. Approximation Schemes for Covering and Packing Problems in Image Processing and VLSI. *J. ACM*, 32(1):130–136, January 1985.

[15] Tanmay Inamdar and Kasturi R. Varadarajan. On Partial Covering For Geometric Set Systems. In *34th International Symposium on Computational Geometry, SoCG 2018, June 11-14, 2018, Budapest, Hungary*, pages 47:1–47:14, 2018.

[16] Bruno Jartoux and Nabil Mustafa. Optimality of Geometric Local Search. In *34th International Symposium on Computational Geometry (SoCG 2018)*, 2018.

[17] Michael J. Kearns. *Computational Complexity of Machine Learning*. MIT Press, Cambridge, MA, USA, 1990. ISBN 0262111527.

[18] Samir Khuller, Anna Moss, and Joseph Naor. The Budgeted Maximum Coverage Problem. *Inf. Process. Lett.*, 70(1):39–45, 1999.

[19] Erik Krohn, Matt Gibson, Gaurav Kanade, and Kasturi R. Varadarajan. Guarding Terrains via Local Search. *JoCG*, 5(1):168–178, 2014.

[20] Richard J Lipton and Robert Endre Tarjan. A separator theorem for planar graphs. *SIAM Journal on Applied Mathematics*, 36(2):177–189, 1979.

[21] Nabil H. Mustafa and Saurabh Ray. Improved Results on Geometric Hitting Set Problems. *Discrete & Computational Geometry*, 44(4):883–895, 2010.

[22] Aniket Basu Roy, Sathish Govindarajan, Rajiv Raman, and Saurabh Ray. Packing and Covering with Non-Piercing Regions. *Discrete & Computational Geometry*, 60(2):471–492, 2018.

[23] Petr Slavík. Improved Performance of the Greedy Algorithm for Partial Cover. *Inf. Process. Lett.*, 64 (5):251–254, December 1997.

[24] Vijay V. Vazirani. *Approximation Algorithms*. Springer-Verlag New York, Inc., New York, NY, USA, 2001. ISBN 3-540-65367-8.

A Proof of Claim 1

Claim 1 *Let $N = \{n_1, n_2, \ldots, n_t\}$ be a set of t integers, where $|n_i| \leq r+1$ for any $n_i \in N$, and $\sum_{i \in t} n_i \geq t+r+1$. Then, there exists a partition $\mathcal{P} = \{P_1, P_2, \ldots, P_s\}$ of the index set $[t]$, satisfying the following properties:*

1. *For any $i \in [s]$, $1 \leq |P_i| \leq 2(r+1)$,*
2. *For any $i \in [s]$, $\sum_{j \in P_i} n_j \geq 0$.*

Proof. For the simplicity of the exposition, we will work with a partition of N instead of a partition of the index set $[t]$, as in the statement of the claim. Note that there is a one-to-one correspondence between these two kinds of partitions.

We initialize $\mathcal{P}$ to be an empty collection, and let N' be a copy of N, which is modified over the course of the following algorithm, while iteratively building $\mathcal{P}$. For any subset $M \subseteq N'$, we use the notation $\sigma(M) \coloneqq \sum_{n_i \in M} n_i$.

First, if there are any n_i's with $n_i = 0$, we add them to $\mathcal{P}$ as singleton sets. Note that, so far, each P_i satisfies the required properties. We remove all such n_i's from N'. Now, for remaining $n_i \in N'$, we have that $1 \leq |n_i| \leq r+1$.

Algorithm 1

```
1: while |N'| > 2(r+1) do
2:     P' ← {n_i}, where n_i ∈ N' with n_i > 0
3:     for i = 1 to r+1 do
4:         if σ(P') ≥ 0 then
5:             if n_i > 0 for all n_i ∈ N' then
6:                 P ← P ∪ N' ∪ {P},    N' ← ∅
7:                 return
8:             else
9:                 Pick an arbitrary n_i ∈ N' with n_i < 0.
10:                P' ← P' ∪ {n_i},   N' ← N' \ {n_i}
11:            end if
12:        else
13:            Pick an arbitrary n_i ∈ N' with n_i > 0.
14:            P' ← P' ∪ {n_i},   N' ← N' \ {n_i}
15:        end if
16:    end for
17:    P ← P'
18:    while σ(P) < 0 do
19:        Pick an arbitrary n_i ∈ N' with n_i > 0.
20:        P' ← P' ∪ {n_i},   N' ← N' \ {n_i}
21:    end while
22:    P ← P ∪ {P}
23: end while
24: P ← P ∪ N'
```

Consider any iteration of the outer while loop, where we are iteratively building a set P'. In each iteration, P' is initialized to a single positive integer from N' – the existence of such an integer follows an inductive claim proved later. Assuming this is true, at the start of the for loop, $\sigma(P') > 0$. Now, consider an iteration of the for loop. If $\sigma(P') \geq 0$, add to it a negative integer n_i, if such n_i exists in N'. If all $n_i \in N'$ are positive, then we add the each of them as a singleton set in $\mathcal{P}$. We also add the current U' to $\mathcal{P}$, and terminate. Note that this satisfies the required properties.

Otherwise, if $\sigma(P') < 0$, we add a positive n_i to it. We will prove later that $\sigma(N') \geq 0$ at any point. Assuming this holds, such $n_i \in N'$ must exist. Furthermore, since $1 \leq |n_i| \leq r+1$ for any $n_i \in N$. Therefore, at the end of the for loop (line 16), it holds that $|\sigma(P')| \leq r+1$. Therefore, in lines 18-20, we need to add at most $r+1$ positive n_i's so as to make $\sigma(P) \geq 0$. Therefore, at line 21, $|P| \leq 2(r+1)$.

Note that in each iteration except possibly the last, the size of N' decreases by at least $r+1$. Also, $|N'| \leq t$ at the beginning. If I denotes the number of iterations of the outer while loop, then $(I-1) \cdot (r+1) \leq t$, which implies that, $I \leq \frac{t}{r+1} + 1$. We prove by induction that after iteration j of the while loop, $\sigma(N') \geq (r+1) \cdot \left(\frac{t}{r+1} + 1 - j\right)$. This implies that for any $0 \leq j \leq I$, $\sigma(N') \geq 0$ after iteration j. As for the base case, note that $\sigma(N) = \sum_{n_i \in N} n_i \geq t + r + 1$.

Suppose the claim is true after iteration $j-1$ of the while loop. At this point, if the condition in line 5 holds, then we add all sets in N' as well as the current P' to $\mathcal{P}$, and the algorithm terminates, trivially proving the claim, since $N' = \emptyset$. On the other hand, if we add P to $\mathcal{P}$ in line 22, then $\sigma(N') \geq (r+1) \cdot \left(\frac{t}{r+1} + 2 - j\right) - \sigma(P) \geq (r+1) \cdot (\frac{t}{r+1} - 1 + j)$, using the induction hypothesis and the fact that $\sigma(P) \leq r+1$.

Thus, $\mathcal{P}$ satisfies the required properties. Now, we simply redefine $\mathcal{P}$ to be the corresponding partition of the index set $[t]$, which finishes the proof. □

Affine invariant triangulations*

Prosenjit Bose† Pilar Cano‡ Rodrigo I. Silveira§

Abstract

We study affine invariant 2D triangulation methods. That is, methods that produce the same triangulation for a point set S for any (unknown) affine transformation of S. Our work is based on a method by Nielson [A characterization of an affine invariant triangulation. *Geom. mod*, 191-210. Springer, 1993] that uses the inverse of the covariance matrix of S to define an affine invariant norm, denoted A_S, and an affine invariant triangulation, denoted $DT_{A_S}[S]$. We revisit the A_S-norm from a geometric perspective, and show that $DT_{A_S}[S]$ can be seen as a standard Delaunay triangulation of a transformed point set based on S. We prove that it retains all of its well-known properties. In addition, we provide different affine invariant order methods of a point set S and of the vertices of a polygon P that can be combined with well-known algorithms in order to obtain other affine invariant triangulation methods of S and P.

1 Introduction

A *triangulation* of a point set S in the plane is a geometric graph such that its vertices are the points of S and all of its faces (except possibly the exterior face) are triangles. Triangulations of point sets are of great interest in different areas such as approximation theory, computational geometry, computer aided geometric design, among others [4, 5, 7]. In particular, the computation of triangulations that are optimal with respect to certain criteria has been widely studied. One of the most popular triangulations is the *Delaunay triangulation*, denoted $DT[S]$, defined by having a triangle between any three points in S if their circumcircle contains no other point of S. This triangulation has the property that it maximizes the minimum angle of all the angles of the triangles in the triangulation, which allows it to avoid narrow triangles. Other properties include the containment of the minimum spanning tree, closest pair of points, and being a constant spanner. For a survey see [2]. Another famous triangulation is the *minimum weight triangulation*, denoted MWT, which minimizes the sum of the length of its edges. The Delaunay triangulation may fail to be a minimum weight triangulation by a factor of $\Theta(n)$ where n is the size of S [15].

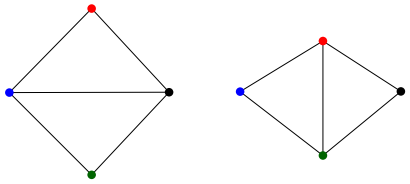


Figure 1: The points on the left correspond to an affine transformation of the points on the right, where each point is mapped to the point with the same color. The DT and MWT (equal in this case) differ between the left and right points, hence they are not affine invariant.

A property that is very important in areas like graphics and computer aided geometric design is affine invariance. In the context of triangulations, consider a triangulation algorithm T, which given a point set S computes a triangulation $T(S)$. We say that T is *affine invariant* if and only if for any invertible *affine transformation* α (see Section 2 for a formal definition), the triangulations $\alpha(T(S))$ and $T(\alpha(S))$ are the same; i.e., triangle $\triangle(pqr)$ is in $T(S)$ if and only if triangle $\triangle(\alpha(p)\alpha(q)\alpha(r))$ is in $T(\alpha(S))$. Note that α is not known to the triangulation procedure.

It is easy to see that neither the Delaunay nor the minimum weight triangulation is affine invariant in general (see Fig. 1). This is because non-uniform stretching can make a point previously outside of a circumcircle become inside, or increase edge lengths non-uniformly.

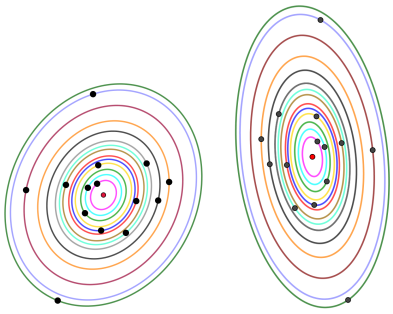
Affine invariance is also important in the analysis and visualization of data, to guarantee for instance that different units of measurement do not influence the triangulation computed. For this reason, Nielson [17] defined an affine invariant normed metric A_S of a point set S, denoted A_S-norm, where for each point $v \in S$ and any affine transformation α, $A_S(x) = A_{\alpha(S)}(\alpha(x))$. The A_S-norm produces ellipses (see Fig. 2) as the boundary of

*P.B. was partially supported by NSERC. P.C. was supported by CONACyT. R.S. was supported by MINECO through the Ramón y Cajal program. P.C. and R.S. were also supported by projects MINECO MTM2015-63791-R and Gen. Cat. 2017SGR1640. This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 734922.

†School of Computer Science, Carleton University, jit@scs.carleton.ca

‡Department de Mathemàtiques, Universitat Politècnica de Catalunya. School of Computer Science, Carleton University, m.pilar.cano@upc.edu

§Department de Mathemàtiques, Universitat Politècnica de Catalunya, rodrigo.silveira@upc.edu



(a) Point set S (b) Point set S'

Figure 2: Point set S' is an affine transformation of S. Each color of the ellipses represents the corresponding boundary of the A-disk centered at the red point (mean) and containing the corresponding point of each transformation.

the A_S-norm disk and using this notion Nielson [18] defined an A_S-Delaunay triangulation that is affine invariant. Nielson's approach does not distinguish if the point set is rotated or reflected. While this is not an issue to obtain an affine invariant Delaunay triangulation, it makes the method unsuitable to construct other triangulations or geometric objects, like the ones discussed in Sections 4 and 5. Surprisingly, it seems that this variant of the Delaunay triangulation, and the whole topic of affine invariant geometric constructions, has gone virtually unnoticed in the computational geometry literature.

Our work. We revisit the A_S-norm and explain the geometry behind it in order to understand how the A_S-Delaunay triangulation behaves. We show that such triangulations have a spanning ratio related to the spanning ratio of standard Delaunay triangulations, and that the hierarchy subgraphs of the A_S-Delaunay triangulation is affine invariant. In addition, we show how to use the A_S-norm to compute different affine invariant orderings of a point set (radial order, sweep line ordering, and a polygon traversal ordering). Using these affine invariant orderings as subroutines, we can adapt standard geometric algorithms for computing a triangulation of a point set or a polygon to become affine invariant. Due to space constraints we omitted some proofs.

2 Preliminaries

A *norm* of a vector space X is a nonnegative function $\rho : X \to \mathbb{R}^+$ with the properties that for all $\lambda \in \mathbb{R}^+$ and $u, v \in X$: (a)$\rho(u+v) \leq \rho(u) + \rho(v)$, (b) $\rho(\lambda v) = \lambda\rho(v)$ and, (c) if $\rho(v) = 0$ then $v = 0$ is the zero vector. A *metric* is a nonnegative function $d : X \times X \to \mathbb{R}^+$ where X is a set such that for all $u, v, w \in X$ the following properties hold: (a) $d(u, v) = 0 \iff u = v$, (b) $d(u, v) = d(v, u)$ and, (c) $d(u, w) \leq d(u, v) + d(v, w)$.

When a metric defines a norm, then it is called a normed metric. Let N be a normed metric, then the *N-disk* D_N centered at $c \in X$ with radius r is the set of points in X within N-distance r from c, i.e., $D_N = \{x : x \in X$ and $N(x-c) \leq r\}$. When the radius is 1 then we call it a *unit N-disk*. An *affine transformation* $\alpha : X \to Y$ is of the form $\alpha(x) = Mx + b$ where X is an affine space mapped to another affine space Y, denoted $Y = \alpha(X)$, and M is a linear transformation on each vector in X plus a translation by vector b in $\alpha(X)$. In this paper we will work in $\mathbb{R}^2$, i.e., $X = Y = \mathbb{R}^2$, M is a matrix in $\mathbb{R}^2 \times \mathbb{R}^2$ and b is a vector in $\mathbb{R}^2$. For the rest of this paper we will not distinguish a point from a vector unless notation is confusing and we will assume that α is invertible, i.e., it is a non degenerate function and $det(M) \neq 0$. The following proposition states some well known properties of affine transformations.

Proposition 1 *[6] Let $\alpha(x) = Mx + b$ be an affine invertible transformation on the affine space X and let S be a point set in X. Then the function α*

1. *maps lines (line segments) to lines (line segments),*
2. *preserves parallelism between lines and segments,*
3. *maps a simple n-gon to a simple n-gon,*
4. *preserves the ratio of lengths of two parallel segments, and*
5. *maps the mean of S to the mean of $\alpha(S)$.*

Using similar arguments as the ones for showing Property 4 in Proposition 1, it can be shown that order types are preserved up to a change of sign. This is shown by checking for each triple $u, v, w \in S$, the signed area of the triangles $\triangle(uvw)$ and $\triangle(\alpha(u)\alpha(v)\alpha(w))$ given by the cross product in the following equation $\alpha(u - v) \times \alpha(w - v) = det(M)((u - v) \times (w - v))$.

Let $S = \{v_1, v_2, \ldots, v_n\}$ be an n-point set in the plane where the coordinates of each point $v_i \in S$ are denoted by (x_i, y_i).

Nielson [17] defines an affine invariant normed metric, that we call A_S-norm, in the following fashion.

Let

$$\mu_x = \frac{\sum_{i=1}^{n} x_i}{n}, \quad \mu_y = \frac{\sum_{i=1}^{n} y_i}{n}, \quad \sigma_x^2 = \frac{\sum_{i=1}^{n} (x_i - \mu_x)^2}{n}$$

$$\sigma_y^2 = \frac{\sum_{i=1}^{n} (y_i - \mu_y)^2}{n}, \quad \sigma_{xy} = \frac{\sum_{i=1}^{n} (x_i - \mu_x)(y_i - \mu_y)}{n}.$$

Note that the mean $\mu = (\mu_x, \mu_y)$ is the barycenter of S. The *covariance matrix* of a point set S is defined as $\Sigma = \begin{pmatrix} \sigma_x^2 & \sigma_{xy} \\ \sigma_{xy} & \sigma_y^2 \end{pmatrix}$ and the matrix $\Sigma^{-1} = \frac{1}{\sigma_x^2\sigma_y^2 - \sigma_{xy}^2} \cdot \begin{pmatrix} \sigma_y^2 & -\sigma_{xy} \\ -\sigma_{xy} & \sigma_x^2 \end{pmatrix}$ is the inverse of Σ.

Then, the A_S-norm metric is defined as, $A_S(x,y)=(x\ y)\Sigma^{-1}\begin{pmatrix}x\\y\end{pmatrix}$.

The matrix Σ^{-1} is also known as the *concentration matrix*, which defines a norm with respect to the normal distribution defined by S. The eigenvectors of Σ are the same of Σ^{-1} and such vectors define the principal orthogonal directions of how spread is the point set with respect to its mean (barycenter) μ. In other words, if we compute the Gaussian manifold defined by the bivariate normal distribution given by the point set S and then cut the Gaussian manifold with a plane parallel to the plane $z = 0$, then we obtain an ellipse. See Figure 2. Such an ellipse has principal orthogonal axes defined by the eigenvectors of Σ^{-1}. Thus, the boundary of an A_S-disk will be defined by a homothet of the resulting ellipse where the boundary of the *unit A_S-disk* will be represented by the ellipse with principal axes being parallel to the eigenvectors of Σ^{-1} and the magnitude of each principal axis will be given by the square root of the eigenvalue of the corresponding unit eigenvector.

The *N-Delaunay triangulation* of S, denoted $DT_N[S]$, is defined in the following fashion. For any three points p_i, p_j, p_k in S add the triangle $\triangle(p_ip_jp_k)$ if there exists an N-disk containing the three points on its boundary and there is no other point of S in its interior. So, the L_2-Delaunay triangulation is the standard Delaunay triangulation, simply denoted $DT[S]$. We say that a point set S is in *general position* if no three points are collinear and all points in S are at different A_S-norm distances from the mean μ.

Using that the boundary of the A_S-disk is an ellipse, Nielson uses the A_S-Delaunay triangulation and shows the following.

Theorem 2 (Nielson [18]) *The triangulation represented by $DT_{A_S}[S]$ is affine invariant.*

3 The A_S-Delaunay triangulation revisited

In this section we discuss the connection between the standard Delaunay triangulation and the A_S-Delaunay triangulation.

Consider the $2\times n$ matrix N such that for each point v in S there is one column in N represented by the vector $v-\mu$. Then, one can check that $\Sigma = \frac{1}{n}NN^T$. So, if a point set $S' = \alpha(S)$ and $\alpha(v) = Mv + b$ with $v \in S$, is an affine transformation of the point set S, then its mean is given by $\alpha(\mu)$ and the covariance matrix Σ' of S' is given by $\Sigma' = M\Sigma M^T$.

Since S is in general position, $det(\Sigma) \neq 0$. Thus, Σ is invertible. Moreover, since Σ is a square symmetric matrix, $\Sigma = Q\Lambda Q^T$ where Q is the matrix of eigenvectors of Σ, Λ is the diagonal matrix of eigenvalues and $Q^{-1}= Q^T$. Therefore, we can also rewrite the covariance matrix as $\Sigma = (Q\Lambda^{\frac{1}{2}})(Q\Lambda^{\frac{1}{2}})^T$. Moreover, Q represents a rotation matrix and $\Lambda^{\frac{1}{2}}$ the scaling factor of a point set with covariance matrix as the identity matrix $\mathbb{I}$. Looking carefully at this representation of Σ and Σ' above we obtain that $(Q\Lambda^{\frac{1}{2}})^{-1}S$ is an affine transformation of S with $\mathbb{I}$ as its covariance matrix. Thus, the unit $A_{(Q\Lambda^{\frac{1}{2}})^{-1}S}$-disk is given by the Euclidean unit disk. This implies that the $A_{(Q\Lambda^{\frac{1}{2}})^{-1}(S)}$-Delaunay triangulation of $(Q\Lambda^{\frac{1}{2}})^{-1}S$ is given by the L_2-Delaunay triangulation of $(Q\Lambda^{\frac{1}{2}})^{-1}S$, which together with Theorem 2 proves the following proposition.

Proposition 3 *Let S be a point set in general position in $\mathbb{R}^2$ and let $\Sigma = Q\Lambda Q^T$ be its covariance matrix. Then $DT[(Q\Lambda^{\frac{1}{2}})^{-1}S] = DT_{A_S}[S]$.*

A nice implication of Proposition 3 is that the A_S-Delaunay triangulation behaves in many ways like a standard Delaunay triangulation. For instance, straightforward properties are that the $DT_{A_S}[S]$ contains a perfect matching when $|S|$ is even, and that the graph is 1-tough, we refer to Dillencourt [8].

Given a weighted graph $G = (V, E)$ and real number $t \geq 1$, a *t-spanner* of G is a spanning subgraph G' such that for every edge xy in G, there exists a path from x to y in G' whose weight is no more than t times the weight of the edge xy in G. When G' is a t-spanner of the complete graph and each edge is weighted with its Euclidean length, we denote $t = sr(G')$ and we simply say that G' is a spanner if $sr(G')$ is finite. It is known that the standard Delaunay triangulation is a spanner, see [9, 14, 19]. The following theorem shows that $DT_{A_S}[S]$ is a spanner.

Theorem 4 *Let S be a point set in general position and let $\Sigma = Q\Lambda Q^T$ be the covariance matrix of S. Let $\lambda_{\max}$ and $\lambda_{\min}$ be the maximum and minimum eigenvalues of Σ, respectively. Then, $sr(DT_{A_S}[S])\leq\left(\frac{\lambda_{\max}}{\lambda_{\min}}\right)^{\frac{1}{2}}\cdot sr(DT[(Q\Lambda^{\frac{1}{2}})^{-1}S])$.*

Proof. Let $S' = (Q\Lambda^{\frac{1}{2}})^{-1}S$. $DT(S')$ is a standard Delaunay triangulation. For any pair of points $u, v \in S'$ let δ_{uv} be a shortest path from u to v contained in $DT(S')$. Thus, $\frac{\sum_{(p_i,p_j)\in\delta_{uv}} d(p_i,p_j)}{d(u,v)} \leq sr(DT[S'])$.

Note that the only thing that changes the spanning ratio is when the graph $DT(S')$ is stretched with different scaling factors in the x- and y-coordinates. As discussed previously, such scaling is defined by the square root of the eigenvalues of Σ given in the diagonal matrix Λ. Hence, for any $u, v \in S'$ we have $d(\Lambda^{\frac{1}{2}}u, \Lambda^{\frac{1}{2}}v) \leq \lambda_{\max}^{\frac{1}{2}}\cdot d(u, v)$ and $d(\Lambda^{\frac{1}{2}}u, \Lambda^{\frac{1}{2}}v)\geq\lambda_{\min}^{\frac{1}{2}}\cdot d(u, v)$. Therefore, $sr(DT_{A_S}[S])\leq(\frac{\lambda_{\max}}{\lambda_{\min}})^{\frac{1}{2}}\cdot sr(DT[S'])$. □

3.1 Hierarchy of affine invariant subgraphs

Let N be a normed metric and S be a point set in general position. The *N-Gabriel graph of S*, denoted

$GG_N[S]$, is defined with S as vertex set and for each pair of points $u, v \in S$, the edge uv is in $GG_N[S]$ iff there exists an N-disk of radius $N(u-v)/2$ containing u and v on its boundary and no other point of S. In other words, the edge uv is in $GG_N[S]$ if a smallest N-disk containing u and v contains no other point of S. If the boundary of the N-disk is defined by a smooth convex shape, then the N-disk of radius $N(u-v)/2$ containing u and v is unique. The *N-relative neighborhood graph of S*, denoted $RNG_N[S]$, is the graph with vertex set S and for each pair of points $u, v \in S$, the edge uv is in $RNG_N[S]$ iff $N(u-v) \leq \max\{N(u-w), N(v-w)\}$ for any $w \in S$. Consider the complete weighted graph G with vertex set S such that the weight of each edge is the N-distance between its endpoints. An *N-minimum spanning tree of S*, denoted $MST_N[S]$, is a spanning tree of G with minimum total edge length. When the boundary of the N-disk defines a convex shape in the plane, Aurenhammer et. al. [3] proved that $MST_N[S] \subseteq RNG_N[S] \subseteq GG_N[S] \subseteq DT_N[S]$. The following result shows that the subgraphs of the A_S-Delaunay triangulation are affine invariant.

Theorem 5 *The $MST_{A_S}[S]$, $RNG_{A_S}[S]$ and $GG_{A_S}[S]$ are affine invariant graphs.*

Proof. Let $\alpha(S)$ be an affine transformation of S. Let us show first that $MST_{A_S}[S]$ is preserved under affine transformations. Let $W(MST_{A_S}[S])$ denote the weight of $MST_{A_S}[S]$ and let ST be an arbitrary spanning tree of S. Thus, $\sum_{uv \in MST_{A_S}[S]} A_S(u-v) = W(MST_{A_S}[S]) \leq W(ST) = \sum_{xy \in ST} A_S(x-y)$. From Property 1 of Proposition 1, $\alpha(u)\alpha(v) = \alpha(uv) \in \alpha(MST_{A_S}[S])$ for any edge uv in $MST_{A_S}[S]$. Since the A_S-norm is affine invariant, $W(\alpha(MST_{A_S}[S])) = \sum_{\alpha(u)\alpha(v) \in \alpha(MST_{A_S}[S])} A_{\alpha(S)}(\alpha(u)-\alpha(v)) = \sum_{uv \in MST_{A_S}[S]} A_S(u-v) = W(MST_{A_S}[S])$. Let ST' be a spanning tree of $\alpha(S)$. Let xy be an edge in ST', from Property 1 of Proposition 1, $\alpha^{-1}(x)\alpha^{-1}(y) = \alpha^{-1}(xy) \in \alpha^{-1}(ST')$. Then, $W(\alpha(MST_{A_S}[S])) = W(MST_{A_S}[S]) \leq W(\alpha^{-1}(ST')) = \sum_{\alpha^{-1}(x)\alpha^{-1}(y) \in \alpha^{-1}(ST')} A_S(\alpha^{-1}(x)-\alpha^{-1}(y)) = \sum_{xy \in ST'} A_\alpha(S)(x-y) = W(ST')$. Therefore, $\alpha(MST_{A_S}[S])$ is a minimum spanning tree of $\alpha(S)$.

Now, we show that $RNG_{A_S}[S]$ is affine invariant. Every edge uv is in $RNG_{A_S}[S]$ if and only if $A_S(u-v) \leq \max\{A_S(u-w), A_S(v-w)\}$ for all $w \in S$. Since the A_S-norm is affine invariant, $A_{\alpha(S)}(\alpha(u)-\alpha(v)) \leq \max\{A_{\alpha(S)}(\alpha(u)-\alpha(w)), A_{\alpha(S)}(\alpha(v)-\alpha(w))\}$ for any $w \in S$ if and only if the edge $\alpha(u)\alpha(v)$ is in $RNG_{A_{\alpha(S)}}[\alpha(S)]$. Thus, $RNG_{A_S}[S] = RNG_{A_{\alpha(S)}}[\alpha(S)]$.

It remains to show that $GG_{A_S}[S]$ is affine invariant. Let uv be an edge in $GG_{A_S}[S]$, thus the A_S-disk of radius $A_S(u-v)/2$ containing u and v contains no other point of S. Now, assume for the sake of a contradiction that the edge $\alpha(u)\alpha(v)$ is not in $GG_{A_{\alpha(S)}}[\alpha(S)]$. Hence, there exists a point $w \in \alpha(S)$ inside the $A_{\alpha(S)}$-disk of radius $A_{\alpha(S)}(\alpha(u)-\alpha(v))/2$ containing $\alpha(u)$ and $\alpha(v)$. Let c be the center of such $A_{\alpha(S)}$-disk. Then, $A_{\alpha(S)}(c-w) \leq A_{\alpha(S)}(\alpha(u)-\alpha(v))/2$. Since the A_S-norm is affine invariant, the A_S-disk of radius $A_S(u-v)/2$ containing u and v is centered at $\alpha^{-1}(c)$. In addition, $A_S(\alpha^{-1}(c)-\alpha^{-1}(w)) \leq A_S(u-v)/2$. Therefore, $\alpha^{-1}(w)$ is in the A_S-disk of radius $A_S(u-v)/2$ containing u and v, which contradicts that uv is an edge in $GG_{A_S}[S]$. With similar arguments, it is shown that if $u'v'$ is an edge in $GG_{A_{\alpha(S)}}[\alpha(S)]$ then $\alpha^{-1}(u')\alpha^{-1}(v')$ is in $GG_{A_S}[S]$. □

4 Other affine invariant triangulations of point sets

In this section we present two affine invariant triangulations of a point set in general position S.

4.1 An affine invariant Graham triangulation

One of the most popular algorithms for computing the convex hull of a point set S is Graham's scan [13]. A nice property of this algorithm is that a modification of the algorithm can also produce a triangulation, sometimes called the *Graham triangulation* [11]. In this section we present an affine invariant version of Graham's scan using the A_S-norm metric. Our method is based on the algorithm for Graham triangulation by Fabila-Monroy and Urrutia [11]. Their method consists of choosing a point p and adding edges from p to the rest of the points, then visiting in radial order from p each remaining point v, edge vu is added if u is visible from v, for each u different from p and v.

The Graham triangulation can be computed in linear time when S is radially ordered. Moreover, since the edges of the triangulation are added according to the radial ordered of S, it follows that if the radial order is affine invariant, then the triangulation is affine invariant. Notice that mapping S to the Euclidean distance in the A_S-norm and then radially sorting is not enough since for any affine transformation α the resulting point sets of the affine transformations $(Q\Lambda^{\frac{1}{2}})^{-1}S$ and $(Q'\Lambda'^{\frac{1}{2}})^{-1}\alpha(S)$ are the same up to rotations and reflections, where $\Sigma = Q\Lambda Q^T$ and $\Sigma' = Q'\Lambda' Q'^T$ are the covariance matrices of S and $\alpha(S)$, respectively.

We will say that a point u is the *A_S-closest* point to μ if u minimizes the A_S distance to μ.

Observation 6 *Let S be a point set in general position with mean μ and let $\alpha(S)$ be an affine transformation of S with mean μ'. The k-th A_S-closest point to μ is mapped to the k-th $A_{\alpha(S)}$-closest point to μ'.*

Using Observation 6, we show that the following radial order is affine invariant.

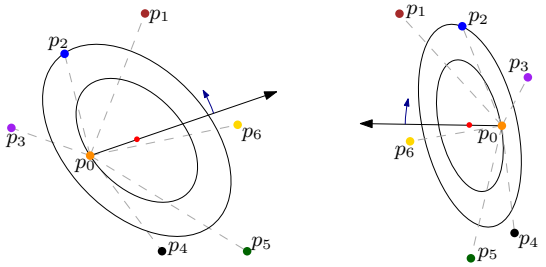


Figure 3: An affine invariant ordering of the point sets S (*left*) and its affine transformation $S'{=}\alpha(S)$ (*right*).

RadiallyOrder(S): Let μ be the mean of S, p be the A_S-closest point to μ. Let q be the second A_S-closest point to μ if it is not on the line defined by μ and p. Otherwise, let q be the third A_S-closest point to μ. If q is on the right of $\overrightarrow{p\mu}$, then sort S clockwise radially from $\overrightarrow{p\mu}$ and $p_0 = p$. Otherwise, sort S counterclockwise radially from $\overrightarrow{p\mu}$ and $p_0 = p$ [See Fig. 3].

Theorem 7 RadiallyOrder *computes an affine invariant radial order of any point set S in general position.*

Proof. [sketch] Let S be a point set in general position and let μ and p be the mean and A_S-closest point to μ, respectively. Let q be the second A_S-closest point to μ if it is not on the line defined by μ and p. Otherwise, let q be the third A_S-closest point to μ. Let $p_0 = p, p_1, \ldots, p_{n-1}$ be the order of S given by RadiallyOrder(S). Let α be an affine transformation. Let μ' be the mean of $\alpha(S)$. By Property 5 from Proposition 1, $\alpha(\mu) = \mu'$ is the mean of $\alpha(S)$. Let p' be the $A_{\alpha(S)}$-closest point to μ'. Let q' be the second $A_{\alpha(S)}$-closest point to μ' if it is not on the line defined by μ' and p'. Otherwise, let q' be the third $A_{\alpha(S)}$-closest point to μ'. By Observation 6, $\alpha(p) = p'$ and $\alpha(q) = q'$. Let $p'_0, p'_1, \ldots, p'_{n-1}$ be the order of $\alpha(S)$ given by RadiallyOrder($\alpha(S)$). We know from before $p'_0 = p' = \alpha(p = p_0)$. It remains to show that $p'_i = \alpha(p_i)$ for all $1 \leq i \leq n-1$. If α contains an even number of reflections then the directions (clockwise/counterclockwise) are preserved. The order types are preserved, so q' is on the same side $\overrightarrow{p'\mu'}$ as q is with respect to $\overrightarrow{p\mu}$. Thus, the radial order is the same. When α contains an odd number of reflections the order types are inverted. Hence, q' is on different side of $\overrightarrow{p'\mu'}$ as q from $\overrightarrow{p\mu}$. Therefore, the direction will be inverted for all points in the algorithm and the order will be preserved. □

The following result is an implication of Theorem 7.

Corollary 8 *There exists an affine invariant Graham triangulation for any point set S in general position.*

4.2 An affine invariant Hamiltonian triangulation

When the point set S has at least one point in the interior of its convex hull, then S can be triangulated by the *insertion method*, which consists of computing the convex hull of S and then inserting points from the interior in arbitrary order. Every time a point v is inserted, the edges connecting v with the points defining the face that contains v are added.

Note that since the convex hull of a point set is preserved under affine transformations, follows that if the points that are in the interior of the convex hull are inserted in an affine invariant order, then the insertion method computes an affine invariant triangulation of S. Hence, applying the insertion method to S such that the interior points in the convex hull of S are inserted in the order given by RadiallyOrder computes an affine invariant triangulation.

These triangulations are Hamiltonian, i.e., their duals [1] contain a Hamiltonian path, and are of interest for fast rendering in computer vision [1, 11].

5 Affine invariant triangulations of polygons

In this section we present two affine invariant triangulations of any simple polygon P.

5.1 An affine invariant triangulation by ear clipping

An *ear of a polygon* is a triangle formed by three consecutive vertices p_1, p_2 and p_3 such that the line segment p_1p_3 is a *diagonal*[2] of the polygon. It is a well-known fact that every simple polygon contains two ears (see Meisters [16]). By recursively locating and chopping an ear, one can triangulate any simple polygon. This method is known as *ear clipping*.

It follows from Properties 1 and 3 of Proposition 1 that the diagonals of a simple polygon are preserved under affine transformations. Thus, the ears of a simple polygon are also preserved. Hence, if the ear clipping procedure locates an ear by traversing P in an affine invariant order, then such procedure computes an affine invariant triangulation of P. The traversal of P in an affine invariant order depends only on finding an affine invariant starting point, and on deciding correctly whether to traverse it clockwise or counterclockwise. The latter depends only on whether the affine transformation contains an odd number of reflections.

Let P have vertex sequence $S := \{v_1, \ldots, v_n\}$ such that S is in general position. Consider the following traversal of P. Traversal (P): Let μ be the mean of S and let p_0 be the A_S-closest point to μ. Let q be the

[1] The *dual* of a graph G has as vertices the faces of G and an edge between vertices is added if the two faces in G share an edge.

[2] A *diagonal* of a polygon is a line segment between two non-consecutive vertices that is totally contained inside the polygon.

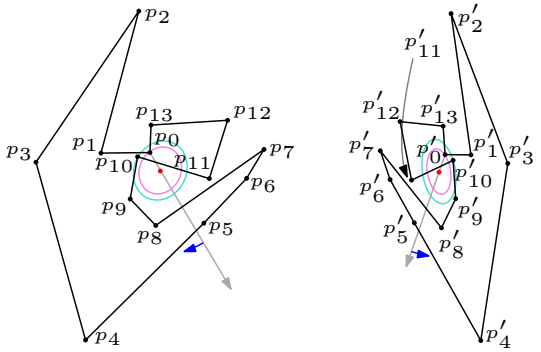


Figure 4: An affine invariant traverse ordering of two simple polygons with point set S and $\alpha(S)$ from Fig. 2.

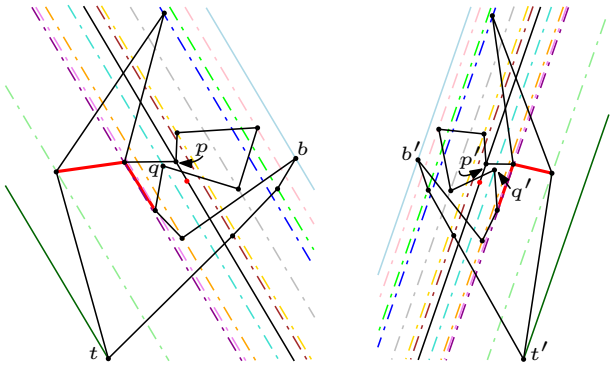


Figure 5: Each colored line is a line parallel to the black line containing the corresponding A-closest point and mean. The red diagonals partition the simple polygon into monotone polygons. Each diagonal contains a cusp with respect to the black line. Each color corresponds to the same line in the transformation.

second A_S-closest point to μ. If q is on the left of $\overrightarrow{\mu p_0}$, then order S by traversing P from p_0 in counterclockwise order. Otherwise, order S by traversing P from p_0 in clockwise order. See Fig. 4.

Using arguments similar to the ones used to prove Theorem 7, we show that TRAVERSAL is affine invariant.

Theorem 9 *Let P be a simple polygon such that its vertices are in general position.* TRAVERSAL*(P) is affine invariant.*

The following result is an implication of Theorem 9.

Corollary 10 *Let P be a simple polygon such that its vertices are in general position. There exists an ear clipping triangulation of P that is affine invariant.*

5.2 An affine invariant triangulation by sweep-line

A simple polygon P is *monotone with respect to a line* ℓ if for any line $\ell^\perp$ perpendicular to ℓ, the intersection of P with $\ell^\perp$ is connected. A line ℓ divides the plane into two half planes, and we say that two points *lie on the same side of* ℓ if they lie on the same half plane. Let v be a vertex of P and ℓ_v be the line containing v that is parallel to ℓ. We say that v is an *ℓ-cusp* if v is a reflex vertex of P and its neighbors in P lie on the same side of ℓ_v. A characterization of a monotone polygon is that a polygon is ℓ-monotone if and only if it does not contain an $\ell^\perp$-cusp (see Edelsbrunner et. al. [10]).

An ℓ-monotone polygon can be triangulated by sweeping its vertices with line $\ell^\perp$ (see, e.g., Garey et. al. [12]). If P is not ℓ-monotone, one can first split it into ℓ-monotone subpolygons by adding diagonals to break all $\ell^\perp$-cusps. Then each resulting ℓ-monotone polygon can be triangulated independently. See Figure 5. It can be shown that if two vertices of P lie on the same side with respect to $\ell^\perp$, then the corresponding vertices in $\alpha(P)$ lie again on the same side with respect to $\alpha(\ell^\perp)$. Using this fact, we show that $\ell^\perp$-cusps are preserved under affine transformations. Therefore, if P is ℓ-monotone and $\ell^\perp$ is perpendicular to ℓ, then $\alpha(P)$ is ℓ^*-monotone where ℓ^* is perpendicular to $\alpha(\ell^\perp)$.

Thus in order to compute an affine invariant triangulation of P it remains to give an affine invariant sweep line order of its vertices. In order to define an affine invariant sweep line order of a point set in general point set S, we use the line $\ell^\perp$ defined by the mean of S and its A_S-closest point. Using the second A_S-closest point to the mean we determine the direction of the sweep with $\ell^\perp$ (see Figure 5). The following theorem shows that such ordering is affine invariant.

Theorem 11 *There exists an affine invariant sweep line ordering of any point set S in general position.*

6 Conclusions

In this paper we initiated the study of affine invariant geometric algorithms, a topic absent in the computational geometry literature. We revisited Nielson's affine invariant norm, which geometrically, the A_S-disk represents how spread is the point set with respect to its mean. We also proposed affine invariant point sorting methods, which are necessary for other affine invariant geometric constructions. Our methods heavily rely on being able to distinguish three points. To this end, we used the A_S-norm. However, for this we had to require that all the points are at different A_S-distances from the mean. Otherwise, the point set becomes highly symmetric, which introduces a problem of indistinguishability. An interesting open question is to what extent such restriction can be removed, while still being able to distinguish rotations and reflections. In addition, finding affine invariant methods to construct other geometric objects is a promising direction for future research.

References

[1] E. M Arkin, M. Held, J. SB Mitchell, and S. S. Skiena. Hamiltonian triangulations for fast rendering. *Vis. Comput.*, 12:429–444, 1996.

[2] F. Aurenhammer, R. Klein, and D. Lee. *Voronoi diagrams and Delaunay triangulations*. World Scientific Publishing Company, 2013.

[3] F. Aurenhammer and G. Paulini. On shape Delaunay tessellations. *Inform. Process. Lett.*, 114(10):535–541, 2014.

[4] F. Aurenhammer and Y. Xu. *Optimal triangulations*. Springer, 2009.

[5] M. Bern and D. Eppstein. Mesh generation and optimal triangulation. In *Computing in Euclidean geometry*, pages 47–123. World Scientific, 1995.

[6] O. Byer, F. Lazebnik, and D. L. Smeltzer. *Methods for Euclidean geometry*, volume 37. MAA, 2010.

[7] L. De Floriani. Surface representations based on triangular grids. *Vis. Comput.*, 3:27–50, 1987.

[8] M. B. Dillencourt. Toughness and Delaunay triangulations. *Discrete Comput. Geom.*, 5(6):575–601, 1990.

[9] D. P. Dobkin, S. J. Friedman, and K. J. Supowit. Delaunay graphs are almost as good as complete graphs. *Discrete Comput. Geom.*, 5(4):399–407, 1990.

[10] H. Edelsbrunner, L. J. Guibas, and J. Stolfi. Optimal point location in a monotone subdivision. *SIAM J. Comput.*, 15(2):317–340, 1986.

[11] R. Fabila-Monroy and J. Urrutia. Graham triangulations and triangulations with a center are Hamiltonian. *Inform. Process. Lett.*, 93:295–299, 2005.

[12] M. R. Garey, D. S. Johnson, F. P. Preparata, and R. E. Tarjan. Triangulating a simple polygon. *Inform. Process. Lett.*, 7(4):175–179, 1978.

[13] R. L. Graham. An efficient algorithm for determining the convex hull of a finite planar set. *Inform. Process. Lett.*, 1:132–133, 1972.

[14] J. M. Keil and C. A Gutwin. The Delaunay triangulation closely approximates the complete Euclidean graph. In *1st WADS proc.*, pages 47–56. Springer, 1989.

[15] D. G. Kirkpatrick. A note on Delaunay and optimal triangulations. *Inform. Process. Lett.*, 10:127–128, 1980.

[16] G. H. Meisters. Polygons have ears. *AMM*, 82:648–651, 1975.

[17] G. M. Nielson. Coordinate free scattered data interpolation. In *Topics in multivariate approximation*, pages 175–184. Elsevier, 1987.

[18] G. M. Nielson. A characterization of an affine invariant triangulation. In *Geom. Mod.*, pages 191–210. Springer, 1993.

[19] Ge X. The stretch factor of the Delaunay triangulation is less than 1.998. *SIAM J. Comput.*, 42:1620–1659, 2013.

Flipping in Spirals

Sander Verdonschot*

Abstract

We study the number of edge flips required to transform any triangulation of a spiral polygon into any other. We improve the upper bound from $4n-6$ to $3n-9$ flips and show a lower bound of $2n-8$ flips. Instead of using a single canonical triangulation as the intermediate point in the transformation between two triangulations, we use a family of closely-connected triangulations.

1 Introduction

This paper deals with edge flips in triangulated spiral polygons. In a triangulated polygon, an *edge flip* (or just *flip*) replaces an internal edge whose incident triangles form a convex quadrilateral with the other diagonal of that quadrilateral (see Figure 1). The *flip distance* between two triangulations is the minimum number of flips required to transform one into the other. Since their introduction by Wagner in 1936 [11], flips have been studied extensively [1]. One of the seminal results in the area is a tight bound on the flip distance between two triangulations of an n-vertex convex polygon. Sleator, Tarjan, and Thurston showed that $2n-10$ flips suffice and are sometimes necessary [10]. This is one of the only settings for which matching upper and lower bounds are known.

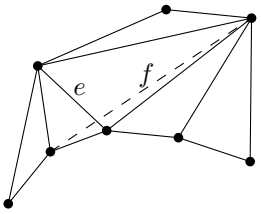


Figure 1: A triangulated spiral polygon where edge e can be flipped to f.

Spiral polygons are polygons with a single contiguous group of reflex vertices. They can be considered the next step up in complexity from convex polygons. Hanke [5] showed that $2(n+n_C-2)$ flips suffice to transform any triangulation of a spiral polygon with n_C convex vertices into any other. Her proof follows a typical pattern: first define a *canonical triangulation* of your polygon, then show that any other triangulation can be transformed into the canonical triangulation using x flips. Since flips are reversible (if a flip replaces an edge e with f, flipping f in the resulting triangulation gives us back e), this gives an upper bound of $2x$ flips between any pair of triangulations, T_1 and T_2, by combining the flip sequence from T_1 to the canonical triangulation with the reversed flip sequence from T_2 to the canonical triangulation. Hanke's canonical form makes certain convex vertices incident to all vertices they can see. In Section 3.1 we show that this requires at least $2n-O(1)$ flips, which means that this technique can never give a worst-case upper bound below $4n-O(1)$. Our proof also gives a lower bound of $2n-8$ on the worst-case flip distance between two triangulated spiral polygons.

In Section 3.2, we show how to break the $4n$ barrier. Using a family of canonical triangulations instead of a single triangulation, we prove an upper bound of $3n-9$ flips. To the best of our knowledge, this is one of the first upper bounds on flip distance that avoids the use of a single intermediary triangulation. The only other such result we know of is by Hanke, Ottmann, and Schuierer [6], who show that the flip distance between two triangulations of a set of points in the plane is bounded by the number of intersections of their edges.

1.1 Related work

Hurtado, Noy, and Urrutia [7] use Hanke's linear bound on the flip distance between triangulations of a spiral polygon[1] to prove an $O(n+n_R^2)$ bound on the flip distance between triangulations of general polygons with n_R reflex vertices. They also prove an $\Omega(n^2)$ lower bound on the flip distance between two triangulations of a polygon with two reflex chains. This suggests that spiral polygons could hold a special position as the largest natural class of polygons with a linear flip distance.

Spiral polygons have also been studied in the setting where each edge has a unique label and a flip transfers the label of the flipped edge to the new edge. In this setting, it is no longer possible to transform any labelled triangulation into any other, since there may not exist any sequence of flips that moves one edge to its counterpart with the same label, for example if there is an unflippable edge that separates the two. However,

*Shopify, sander.verdonschot@gmail.com

[1]The conference version of their paper contains an incorrect proof of a $2n-6$ upper bound on the flip distance between triangulations of a spiral polygon, but they refer to Hanke's result in the journal version.

Bose et al. [2] show that this necessary condition is also sufficient: if each edge individually can be flipped to its matched counterpart, the entire triangulation can be transformed into the target triangulation. Whether the same was true for more complex triangulations was unknown, until Lubiw, Masárová, and Wagner [8] recently proved that it is also true for triangulations of sets of points in the plane.

2 Preliminaries

Before we can get into the meat of the proofs, we need some definitions.

Let P be a spiral polygon. The n vertices of P form one convex chain $C = c_1, c_2, \ldots, c_{|C|}$ and one reflex chain $R = r_1, r_2, \ldots, r_{|R|}$ such that $n = |C| + |R|$, and c_1 and $c_{|C|}$ are adjacent to r_1 and $r_{|R|}$, respectively (see Figure 2).

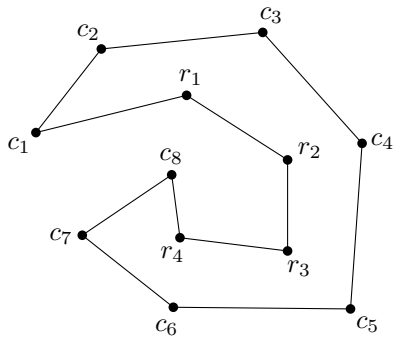


Figure 2: A spiral polygon.

One of the most useful properties of convex polygons is that splitting them along a chord results in two smaller convex polygons. The same holds for spiral polygons.

Lemma 1 *Inserting any chord into a spiral polygon P splits P into two smaller spiral polygons.*

Proof. It is well-known that inserting a chord into a polygon splits it into two smaller polygons, so we only need to show that both parts (say P_1 and P_2) are still spiral polygons. This is determined by the interior angles of their vertices, which are the same as in P, except for the endpoints of the inserted chord. At each endpoint, the interior angle is divided between P_1 and P_2, which means they can only get smaller. In particular, any reflex vertex of P_1 or P_2 is also a reflex vertex of P. Thus, if the chord connects two convex vertices of P, P_1 is a convex polygon, while P_2 is a spiral polygon with the same reflex chain as P. On the other hand, if the chord connects a convex vertex of P to a reflex vertex r, it splits the reflex chain so that part of it goes to P_1 and part to P_2. But since r is the only reflex vertex whose angle was changed, and it is now adjacent to a convex vertex in both pieces, there will still be at most one reflex chain in each piece, regardless of whether r is reflex or convex in P_1 and P_2. Thus, both P_1 and P_2 are spiral polygons. □

We now turn our attention to visibility. We say that two vertices of P *see* each other if the line segment connecting them does not intersect the exterior of P. The *visibility graph* of P is the graph whose vertices are the vertices of P, with an edge between two vertices if and only if they see each other. Our interest in visibility graphs stems from the observation that the triangulations of a polygon P are the maximal plane subgraphs of the visibility graph of P.

Everett and Corneil [4] classified the visibility graphs of spiral polygons. For our purposes, their most important findings are the following.

1. Every reflex vertex sees at least one convex vertex, and vice versa.
2. For each vertex v, the set of convex vertices seen by v is contiguous along C. Likewise, the set of reflex vertices seen by v is contiguous along R.
3. Every two vertices a and b that do not see each other have a *blocking reflex vertex*: a reflex vertex r such that no two vertices x and y with $x \in [a, \ldots, r)$ and $y \in (r, \ldots, b]$ (along the boundary of the polygon) see each other.
4. The visibility graph does not contain a chordless cycle of length 4 or greater.

We use these properties to prove two useful lemmas.

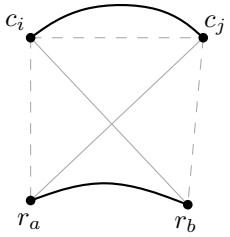


Figure 3: If c_i sees r_b and c_j sees r_a, then c_i sees c_j, c_i sees r_a, and c_j sees r_b.

Lemma 2 *If a convex vertex c_i sees a reflex vertex r_b and a later convex vertex c_j sees an earlier reflex vertex r_a, then c_i sees c_j, c_i sees r_a, and c_j sees r_b (see Figure 3).*

Proof. Suppose, for a contradiction, that c_i does not see c_j. Then they have a blocking reflex vertex x. If x lies before r_b, it is not a blocking vertex since (c_i, r_b) connects the chain $[c_i, x)$ to the chain $(x, c_j]$. On the

other hand, if x is r_b itself or lies after r_b, (c_j, r_a) connects the chain $[c_i, x)$ to the chain $(x, c_j]$. Thus, no such blocking vertex can exist and c_i and c_j must see each other.

To prove that c_i sees r_a, imagine inserting the chord (c_j, r_a). By Lemma 1, this subdivides the polygon into two smaller spiral polygons, so consider the piece containing c_i. This is either a convex polygon – in which case c_i definitely sees r_a – or another spiral polygon. Since c_i is still a convex vertex, it must see some reflex vertex r_x that is either r_a itself, or comes before r_a. Note that all reflex vertices in the smaller polygon were reflex vertices in the original polygon, and if c_i sees r_x in the smaller polygon, it also sees r_x in the original polygon. Thus, c_i either sees r_a, or it sees a reflex vertex before r_a. But we know that it also sees r_b and that the set of reflex vertices it sees is contiguous. Thus, c_i must see r_a. An analogous argument shows that c_j sees r_b. □

Lemma 3 *Let c_i and c_j be two convex vertices that see each other, with $i < j$. Let r be the last reflex vertex c_i sees. Then c_j sees r too.*

Proof. Let r' be the first reflex vertex that c_j sees. If r' lies before r, then c_j sees r by Lemma 2. On the other hand, if r' lies after r, then $c_i r \dots r' c_j$ forms a chordless cycle of length 4 or greater in the visibility graph, which is impossible by Everett and Corneil's result. Therefore c_j must see r. □

3 Flip distance

With the preliminaries out of the way, we are ready to bound the number of flips needed to transform any triangulation of a spiral polygon into any other. We begin with a simple lower bound.

3.1 Lower bound

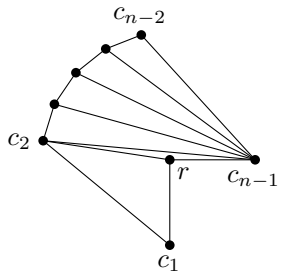


Figure 4: A spiral polygon where most edges need to be flipped twice to become incident to c_1.

Theorem 4 *For any $n > 4$, there is an n-vertex triangulated spiral polygon where at least $2n - 8$ flips are required to connect c_1 to all vertices it sees.*

Proof. Consider a spiral polygon with a single reflex vertex r, such that all vertices other than the two neighbours of r are visible to all vertices (see Figure 4). In the initial triangulation, c_{n-1} is connected to all convex vertices except for c_1, and c_2 is connected to r. In the final triangulation, c_1 is connected to all convex vertices except for c_{n-1}, and c_{n-2} is connected to r.

No edge $c_i c_{n-1}$ can be flipped directly to an edge $c_1 c_j$, since any quadrilateral $c_1 c_i c_j c_{n-1}$ contains r. Thus, each internal edge needs to be flipped at least twice, save for $c_2 r$ and one edge that ends up incident to r. This gives us the desired bound of $2(n-3) - 2 = 2n - 8$. □

By adding additional vertices near c_1, this argument can be adapted to show a similar lower bound for strategies that connect the first reflex vertex or the i-th convex vertex to all vertices it sees. It also gives a general lower bound on the worst-case number of flips required.

Corollary 5 *For any $n > 4$, there is an n-vertex spiral polygon with two triangulations T_1 and T_2 such that transforming T_1 into T_2 requires at least $2n - 8$ flips.*

3.2 Upper bound

Next we turn our attention to the upper bound. As with many such proofs [3], we first transform each triangulation into a form that is easier to work with, called a *canonical form*. One significant difference between our proof and most others is that our canonical form consists of a family of triangulations, instead of a single triangulation.

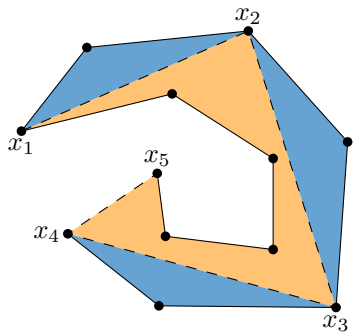


Figure 5: A spiral polygon with its spine (dashed), fins (darker) and body (lighter). Note that the spine includes part of the polygon boundary.

3.3 The canonical form

Let P be a spiral polygon. The *spine* of P starts at c_1, which we label x_1, and continues to the last convex vertex c_i that x_1 sees. We refer to c_i as x_2 and if $x_2 = c_{|C|}$, we are done. Otherwise, we continue to the last convex vertex c_j that x_2 sees and call it x_3, and so on (see Figure 5). Thus, the spine of P is a path

$X = x_1, x_2, \ldots, x_{|X|}$ such that for each i, x_{i+1} is the last convex vertex visible from x_i. The spine partitions P into several convex polygons on one side, which we call *fins*, and one spiral polygon on the other side, which we call the *body*.

We say that a triangulation of a spiral polygon P is in *canonical form* if all spine edges are present and every internal edge is incident on a spine vertex[2]. Two triangulations in canonical form are "close" in terms of flip distance. We show how to transform one triangulation in canonical form into another by flipping each edge at most once. First, we introduce one more useful lemma.

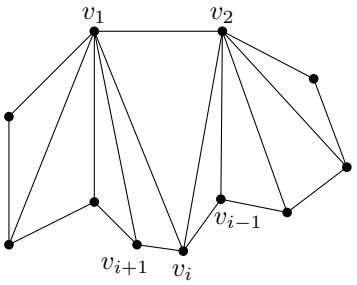


Figure 6: A binary star. All vertices see both v_1 and v_2 and all internal edges are incident on one of these two vertices.

Lemma 6 *Let P be a polygon with a boundary edge (v_1, v_2) such that all other vertices of P see both v_1 and v_2 (see Figure 6). Then any triangulation T_1 of P where all edges are incident to v_1 or v_2 can be transformed into any other such triangulation T_2 by flipping each edge in T_1 that is not in T_2 exactly once.*

Proof. The triangulations T_1 and T_2 are characterized by the triangle incident on edge (v_1, v_2): fixing this triangle fixes all other edges. Thus, we can uniquely label each triangulation with the third vertex v_i of this triangle. Suppose that (v_1, v_i) is an internal edge. Then it must be flippable, since all vertices see v_2. After flipping (v_1, v_i), we get triangulation v_{i+1}. Similarly, if (v_2, v_i) is an internal edge, flipping it yields triangulation v_{i-1}. Thus, if T_1 has label v_a and T_2 has label v_b with $a \leq b$, it takes exactly $b-a$ flips to transform T_1 into T_2. Since each flipped edge was incident on v_1 before the flip and is not after the flip (or vice versa), no edge is flipped twice. Thus, we flip each edge that differs between T_1 and T_2 exactly once. □

We refer to a triangulated polygon P with a boundary edge (v_1, v_2) such that all other vertices of P see both v_1 and v_2, and all internal edges are incident to either v_1 or v_2, as a *binary star*.

[2]The second condition actually implies the first, but it is more convenient for us to include both in the definition explicitly.

Lemma 7 *In a spiral polygon, we can transform any triangulation in canonical form into any other such triangulation by flipping each non-spine edge at most once.*

Proof. Recall that the spine partitions the polygon into a number of convex polygons on one side (the fins) and one spiral polygon on the other (the body). Since both triangulations contain all spine edges, we can transform each fin and the body independently. Each fin is a convex polygon with a single spine edge on its boundary and all internal edges are incident to a spine vertex. Thus, each fin is a binary star and, by Lemma 6, we can transform all fins by flipping each edge inside them at most once. Now all that is left is transforming the body. We show that this part, too, is composed of binary stars.

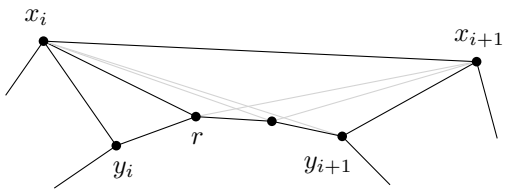


Figure 7: A spine edge (x_i, x_{i+1}) with relevant reflex vertices: y_i is the last reflex vertex that x_{i-1} sees, y_{i+1} is the same for x_i, and r is the first reflex vertex that x_{i+1} sees. The black edges are present in all triangulations in canonical form. Note that $x_i, r, \ldots, y_{i+1}, x_{i+1}$ forms a binary star.

Consider a spine edge (x_i, x_{i+1}). Recall that x_{i+1} is the last convex vertex that x_i sees. Let y_{i+1} be the last reflex vertex that x_i sees. By Lemma 3, x_{i+1} sees y_{i+1} as well. We show that the edge $e = (x_{i+1}, y_{i+1})$ is not intersected by any other chord and must therefore be part of all canonical triangulations.

Suppose, for a contradiction, that a chord e' intersects e. Note that all chords in the body connect a reflex vertex to a convex vertex. If e' connects a reflex vertex before y_{i+1} to a convex vertex after x_{i+1}, we can apply Lemma 2 to e' and $x_i y_{i+1}$ to conclude that x_i sees the convex vertex after x_{i+1}: a contradiction. Similarly, if e' connects a reflex vertex after y_{i+1} to a convex vertex before x_{i+1}, x_i would see the reflex vertex after y_{i+1}, yielding another contradiction. Since triangulations are maximal plane graphs, (x_{i+1}, y_{i+1}) is part of every canonical triangulation.

Let y_i be the last reflex vertex seen by x_{i-1} (or r_1 if $i = 1$) and consider the polygon $P_i = x_i, y_i, \ldots, y_{i+1}, x_{i+1}$. If x_{i+1} sees y_i, all vertices on $y_i, \ldots, y_{i+1}$ see both x_i and x_{i+1}, making it a binary star. If x_{i+1} does not see y_i, let r be the first reflex vertex that x_{i+1} sees. In this case, $x_i, r, \ldots, y_{i+1}, x_{i+1}$ forms a binary star, and all reflex vertices between y_i and r must be connected to x_i in all canonical triangulations. Thus, each P_i can be independently transformed

by flipping each internal edge at most once. Since the subpolygons P_i form a partition of the body, we can transform the entire body by flipping each internal edge at most once. □

3.4 Canonicalization

Now that we know how to transform triangulations in canonical form into one another, we consider how to transform arbitrary triangulations into canonical form. The next lemma allows us to introduce the spine edges.

Lemma 8 *Given a triangulated spiral polygon, we can introduce any chord $e = (a, b)$ between two convex vertices by flipping each edge crossing e exactly once. Moreover, all new edges are incident to an endpoint of e.*

Proof. We prove this by induction on the number of edges crossing e. If there are none, e must already be present, since triangulations are maximal plane graphs. This settles the base case. Otherwise, let $e' = (u, v)$ be the edge whose intersection with e is closest to a. Let x be the third vertex of the triangle incident on e' opposite a (see Figure 8).

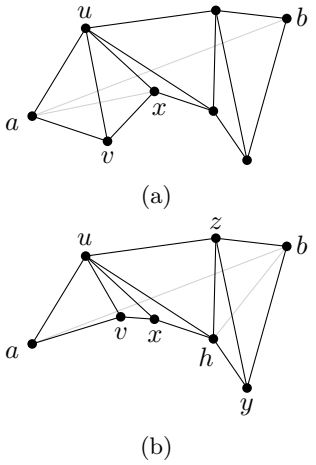


Figure 8: (a) If a sees x, we flip (u, v). (b) If a does not see x, b must see h, so we flip (y, z).

If a sees x, $auxv$ forms a convex quadrilateral and we can flip uv to ax. This reduces the number of edges crossing e by one, using exactly one flip. Since the new edge is incident on a, the result follows by induction.

If a does not see x, $auxv$ forms a non-convex quadrilateral with u or v as reflex vertex. Suppose, without loss of generality, that v is the reflex vertex. Now consider the edge $e'' = (y, z)$ whose intersection with e is closest to b, and let h be the third vertex of the triangle incident on e'' opposite b (see Figure 8b). If b sees h, we are done by the same argument as before, so suppose b does not see h. We show that this is impossible.

Consider the polygon P formed by the union of all triangles crossing e. Lemma 1 tells us that P is a spiral polygon. In P, v is the first vertex of the reflex chain. Suppose, without loss of generality, that y is the reflex vertex in $byhz$. Then y is the last vertex along the reflex chain. But a is a neighbour of v and b is a neighbour of y, and the neighbours of the first and last reflex vertex cannot see each other. Since a and b see each other by the construction of P, this situation cannot arise and we can always flip either e' or e''. □

The next lemma helps us show that edges flipped to introduce one spine edge do not need to be flipped again to introduce later ones.

Lemma 9 *An edge e incident on a spine vertex x_i cannot cross a spine edge (x_j, x_{j+1}) with $j \geq i$.*

Proof. It is clear that e cannot cross (x_i, x_{i+1}), since they already intersect in x_i and two line segments intersect in at most one point. Now suppose that e crosses some spine edge (x_j, x_{j+1}) with $j > i$. Then its other endpoint must lie on the part of the convex chain strictly between x_j and x_{j+1}. But that means e connects x_i to a convex vertex after x_{i+1}. This is a contradiction, since by definition, x_{i+1} is the last convex vertex that x_i sees. □

This final lemma helps us transform the fins into canonical form. It is a common tool for convex polygons [10], included here for completeness.

Lemma 10 *In a convex polygon P, we can connect any vertex v to all other vertices by flipping each internal edge not incident to v exactly once.*

Proof. We prove this by induction on the number of vertices not connected to v. If this number is 0, all internal edges are incident on v, so we are done and need no flips. If there is at least one such vertex x, consider the chord (v, x). This chord intersects at least one edge not incident on v, otherwise (v, x) would be part of the triangulation, connecting x to v. Of those edges, let (a, b) be the one whose intersection with (v, x) is closest to v. Flipping (a, b) creates an edge (v, y), where y is a vertex that was not yet connected to v (y and x may be the same). Thus we reduced the number of vertices not connected to v by one, by flipping exactly one edge not incident on v. The lemma follows by induction. □

Now we have all the tools we need to transform triangulations into canonical form.

Lemma 11 *We can transform any triangulation of a spiral polygon into canonical form by flipping each edge at most once.*

Proof. Recall that the canonical form requires that all spine edges are present and that all edges are incident to a spine vertex. We begin with the first requirement.

We consider each spine edge in sequence, starting with (x_1, x_2). If it is already present, we move on to the next spine edge. Otherwise, we introduce it using the technique from Lemma 8. This guarantees that we flip each edge crossing a spine edge exactly once, since all resulting edges are incident to a spine vertex and by Lemma 9, these edges cannot cross any later spine edges.

Once all spine edges are present, the body is already in canonical form, since no two reflex vertices see each other and the only convex vertices of the body are spine vertices. The fins, on the other hand, may contain edges that are not incident to a spine vertex. Any such edge has not been flipped yet, since all flipped edges are now incident to a spine vertex.

Consider a fin F with spine edge (x_i, x_{i+1}) where not all edges are adjacent to a spine vertex. The triangle incident on (x_i, x_{i+1}) splits the fin into two convex polygons, F_i and F_{i+1}, each containing the corresponding spine vertex. Now Lemma 10 lets us connect all vertices in F_i to x_i and all vertices in F_{i+1} to x_{i+1} by flipping each edge not incident to one of the two spine vertices exactly once. □

Putting everything together, we obtain the following bound on the maximum flip distance.

Theorem 12 *We can transform any triangulation of a spiral polygon into any other by flipping each internal edge at most thrice and each internal edge that becomes part of the spine at most twice.*

Proof. Given two triangulations T_1 and T_2, we can transform each into the canonical form using the technique from Lemma 11, flipping each edge at most twice. This leaves us with two triangulations T_1' and T_2' in canonical form, so we can transform T_1' into T_2' by flipping each non-spine edge at most once using Lemma 7. To obtain a full flip sequence that transforms T_1 into T_2, we follow these steps to transform T_1 into T_2' and then reverse the flip sequence that transformed T_2 into T_2', to obtain T_2. □

Since any triangulation of a polygon has exactly $n-3$ internal edges, this gives us the following upper bound.

Corollary 13 *We can transform any triangulation of an n-vertex spiral polygon into any other with no more than $3n - 9$ flips.*

Similar to convex polygons [9], Theorem 12 also gives us an approximation on the flip distance.

Corollary 14 *Given two triangulations T_1 and T_2 of a spiral polygon P, we can transform T_1 into T_2 with at most $3d(T_1, T_2)$ flips, where $d(T_1, T_2)$ is the flip distance between T_1 and T_2.*

Proof. Let S be the set of edges that are common between T_1 and T_2. By Lemma 1, inserting all of these edges into P splits it into a number of smaller spiral polygons. Of these subpolygons, let P_1 through P_m be the ones that are not triangles.

Each subpolygon P_i is present in both T_1 and T_2 with different triangulations that have no edges in common. Thus, each internal edge of P_i in T_1 needs to be flipped at least once to obtain T_2. This gives us a lower bound $d(T_1, T_2) \geq \sum_{i=1}^{m} |P_i|$, where $|P_i|$ is the number of internal edges in P_i.

On the other hand, by Theorem 12, we can transform the triangulation for T_1 in each P_i into the triangulation for T_2 by flipping at most $3|P_i|$ edges. Therefore the total number of flips is bounded by $3\sum_{i=1}^{m} |P_i| \leq 3d(T_1, T_2)$. □

References

[1] P. Bose and F. Hurtado. Flips in planar graphs. *Computational Geometry*, 42(1):60–80, 2009.

[2] P. Bose, A. Lubiw, V. Pathak, and S. Verdonschot. Flipping edge-labelled triangulations. *Computational Geometry: Theory and Applications*, 68:309–326, 2018. Special Issue in Memory of Ferran Hurtado.

[3] P. Bose and S. Verdonschot. A history of flips in combinatorial triangulations. In *Proceedings of the XIV Spanish Meeting on Computational Geometry (EGC 2011)*, volume 7579 of *Lecture Notes in Computer Science*, pages 29–44. 2012.

[4] H. Everett and D. G. Corneil. Recognizing visibility graphs of spiral polygons. *Journal of Algorithms*, 11(1):1–26, 1990.

[5] S. Hanke. Ebene triangulationen. Master's thesis, Institut für Informatik, Albert-Ludwigs-Universität Freiburg, 1994.

[6] S. Hanke, T. Ottmann, and S. Schuierer. The edge-flipping distance of triangulations. *Journal of Universal Computer Science*, 2(8):570–579, 1996.

[7] F. Hurtado, M. Noy, and J. Urrutia. Flipping edges in triangulations. *Discrete & Computational Geometry*, 22(3):333–346, 1999.

[8] A. Lubiw, Z. Masárová, and U. Wagner. A proof of the orbit conjecture for flipping edge-labelled triangulations. *Discrete & Computational Geometry*, 61(4):880–898, 2019.

[9] D. D. Sleator, R. E. Tarjan, and W. P. Thurston. Rotation distance. In T. M. Cover and B. Gopinath, editors, *Open Problems in Communication and Computation*, pages 130–137. 1987.

[10] D. D. Sleator, R. E. Tarjan, and W. P. Thurston. Rotation distance, triangulations, and hyperbolic geometry. *Journal of the American Mathematical Society*, 1(3):647–681, 1988.

[11] K. Wagner. Bemerkungen zum Vierfarbenproblem. *Jahresbericht der Deutschen Mathematiker-Vereinigung*, 46:26–32, 1936.

Graph Realization on a Random Embedding

Saad Quader* Alexander Russell†

Abstract

We consider the power of selecting a Euclidean metric in order to realize an undirected graph of interest as a unit disc graph after random embedding into the hypercube. Specifically, let $G = (V, E)$ be an n-vertex graph. We say that a map $h : V \to \mathbb{R}^d$ *realizes* G if $\{u, v\} \in E$ if and only if $\|h(u) - h(v)\|_2 < 1$. We study the probability that a random embedding $f : V \to \{0, 1\}^d$ can be rectified to realize G by merely scaling the coordinates.

This yields an interesting relationship between the structure of G and the dimension d at which this event becomes likely. Our basic quantity of interest, denoted $R(G, d)$, is the probability that—for a random embedding $f : V \to \{0, 1\}^d$—there is an axis-parallel scaling $\hat{W} : \mathbb{R}^d \to \mathbb{R}^d$ for which $h = \hat{W} \circ f$ realizes G.

We show that for a function $d(n) = O(n \log n)$, the probability $R(T, d)$ is $1 - o(1)$ for any tree T on n vertices. For general graphs, we show that a function $d(n) = O(na^2 \log n)$ is sufficient to guarantee $R(G, d) = 1 - o(1)$ for G with n vertices and arboricity a. These results apply to both ℓ_2 and ℓ_1 norms.

To complement these results, we show that $d = \Omega(n^2)$ is necessary to realize an Erdős-Rényi random graph even if we allow *arbitrary* realization maps h. We also prove a probabilistic analog of Radon's theorem; this may be of independent interest.

1 Introduction

In an embedding or a realization problem, in general, we are given a graph $G = (V, E)$ with n vertices and a target metric space $(Y, \|\cdot\|)$. The goal is to find a map $h : V \to Y$ so that some desired properties of V are preserved in the mapped points $h(V)$ under the metric $\|\cdot\|$. It is typically desirable that Y should have a low dimension. Sometimes, in addition, h must satisfy some structural constraints. Prime examples of such problems include Euclidean minimum spanning tree (EMST) realization [9], metric embedding [15, Ch. 15], low-dimensional structure-preserving embedding [17], and Feige's volume-respecting embedding [5]. These problems and their solutions have impressive algorithmic applications.

*Department of Computer Science, University of Connecticut, saad.quader@uconn.edu

†Department of Computer Science, University of Connecticut, acr@cse.uconn.edu. This research was partially supported by NSF Grant 1717432.

We adopt a new perspective and consider the power of selecting the Euclidean metric in order to realize an undirected graph of interest as a unit disc graph after random embedding into the hypercube. Specifically, let $G = (V, E)$ be an n-vertex graph. We say that a map $h : V \to \mathbb{R}^d$ *realizes* G if

$$\forall u, v, \in V, \quad \{u, v\} \in E \iff \|h(u) - h(v)\|_2 < 1\,. \quad (1)$$

In particular, h has to bring the neighbors close while pushing the non-neighbors far apart:

$$\max_{\{u,v\}\in E} \|h(u) - h(v)\|_2 < \min_{\{u',v'\}\notin E} \|h(u') - h(v')\|_2\,. \quad (2)$$

We study the probability that a random embedding $f : V \to \{0, 1\}^d$ can be rectified to realize G by merely scaling the coordinates. This yields an interesting relationship between the structure of G and the dimension d at which this event becomes likely. Our basic quantity of interest, denoted $R(G, d)$, is the probability that—for a random embedding $f : V \to \{0, 1\}^d$—there is an axis-parallel scaling $\hat{W} : \mathbb{R}^d \to \mathbb{R}^d$ for which $h = \hat{W} \circ f$ realizes G. We express the scaling $\hat{W}$ as the diagonal matrix $\mathbf{diag}(\hat{w})$ for a tuple of non-negative values $\hat{w} \in \mathbb{R}^d$. Here, $\circ$ denotes function composition.

Algorithmically, we can treat this as the following problem.

Problem 1. *Let $G = (V, E)$ be an undirected graph; let $d \in \mathbb{N}$. Let $f : V \to \{0, 1\}^d$ be a uniformly random function. Given f and G, the goal is to find non-negative weights $\hat{w} \in \mathbb{R}^d$—which may depend on f and G—such that the map $h : V \to \mathbb{R}^d, h = \mathbf{diag}(\hat{w}) \circ f$ satisfies* (1). ◇

Observe that f is non-injective with probability at most $\binom{|V|}{2}/2^d$. If $d \geq 3 \log_2 |V|$, this probability is at most $1/|V|$. The *weighted ℓ_2^d norm* given by a non-negative weight vector $w \in \mathbb{R}^d$ is $\|x\|_w \triangleq (\sum_i w_i x_i^2)^{1/2}$. Define

$$w \triangleq \left(\hat{w}_1^2, \cdots, \hat{w}_d^2\right). \quad (3)$$

Since the function $x \mapsto x^2$ is monotonically increasing on non-negative reals x, the constraints (1) are equivalent to

$$\forall\, u, v \in V, \quad \{u, v\} \in E \iff \|f(u) - f(v)\|_w^2 < 1 \quad (4)$$

where we use $h = \mathbf{diag}(\hat{w}) \circ f$. Understood in this way, we say "w realizes G" to mean "h realizes G."

As mentioned above, analytically we study the probability $R(G, d)$ for various graphs and dimensions d (which typically scale with the size or structural features of the graph).

We cast this in an algorithmic framework, both for expository purposes and to emphasize the simple constructions we provide for w.

1.1 The literature

You must have noticed that Problem 1 is reminiscent of many well-known problems such as metric embedding, graph/tree realization, and metric optimization. However, the randomness in f sets Problem 1 apart. Specifically, in Problem 1, we wish to find a "good metric" $\|\cdot\|_w$ with respect to a *random* embedding f; in the usual metric embedding and realization problems, the metric (such as ℓ_2) is fixed, and we look for a "good embedding" h. We delve into these connections in Appendix A.

Why do we study a uniformly random embedding? Without any assumption about the structure of f, it is difficult to answer questions such as how large a d is necessary and/or sufficient for the realization. Studying a uniformly random f gives us an elegant mathematical structure to analyze. Although this structure is simple, it is nevertheless a first step towards analyzing the embeddings given by more general stochastic processes, e.g., with a known covariance matrix.

The inspiration for Problem 1 came from Ghadie et. al [6]. They used a linear program to realize a tree T via a map $h = \mathbf{diag}(\hat{w}) \circ f$. The embedding $f : V \to \{0,1\}^d$, however, was extracted from a gene-expression dataset. They asked: *do the points $f(V)$ explain the tree T?* The existence of a realizing map h was taken as the evidence that the embedding indeed explains the tree; cf. Appendix A.3. However, the interplay between d and the size/structure of G remains unclear in this approach. Our paper sheds light on this interplay by assuming an additional structure (i.e., the randomness) in f.

1.2 Results

Notice that for a fixed f, one can attempt to realize G via a linear program (as in [6]) since the constraints (4) are linear in w. In contrast, we are interested in realization algorithms that harness the randomness in f.

We present two realization algorithms. For some dimension $d = O(n \log n)$, Algorithm 1 realizes any n-vertex tree with high probability; cf. Theorem 1. In addition, for some dimension $d = O(na^2 \log n)$, Algorithm 2 realizes an n-vertex graph $G = (V, E)$ with high probability; cf. Theorem 2. Here, a is the arboricity of G (cf. Definition 4.1). It is known that a is at most $\lceil \sqrt{|E|/2} \rceil$ [2]. Both algorithms use only Boolean weights. Moreover, we outline how the effective *resistances* of the edges impact this bound; cf. the comments at the end of Section 4

To complement the above results, we show that for large n, it is impossible to realize an Erdős-Rényi random graph on n vertices if d is less than $\binom{n}{2}/6$; cf. Theorem 4. In addition, for large n, it is impossible to realize a random spanning tree of the complete graph K_n if d is less than $n/2$; cf. Theorem 5. In particular, these statements holds even if (i.) the embedding f is arbitrary and (ii.) the weights w_i are allowed to be negative although, in that case, w would represent "flips" as well as scaling. Note that the necessary and sufficient bounds on d are tight up to a $\log n$ factor for realizing trees.

Realization in ℓ_1 and generalizations of f. Let $\mathcal{B}$ be the Boolean hypercube, and let $\mathcal{C} = s\mathcal{B} + t$ where $s, t \in \mathbb{R}$. As it turns out, our results hold for embeddings $f : V \to \mathcal{C}$ and both ℓ_1 and ℓ_2 metrics. We outline these generalizations at the end of Section 3. However, to keep our exposition lean, we treat only Boolean embeddings and the ℓ_2 metric.

Application: is a given realization non-trivial? Let $G = (V, E)$ be an n-vertex graph and f be a d-dimensional random embedding as in Problem 1. Suppose we are presented with weights $w \in \mathbb{R}^d$ that realize G. As we outlined in Section 1.1, one can interpret this realization as the evidence that the points $f(V)$ *explain* the input graph G.

Suppose d is at least as large as the bound given in Theorem 2. Then, although the weights w realize G on the points $f(V)$, the embedding dimension d is so large that the points $f(V)$ would explain *every* n-vertex graph. (This, in fact, is guaranteed by Theorem 2.) Hence, the "explanation" is meaningless. If d is asymptotically smaller, we could not so easily have ruled out the possibility that $f(V)$ indeed explains G.

2 An overview of main ideas

For any map $f : V \to \{0,1\}^d$ and non-negative weights $w = (w_1, \ldots, w_d) \in \mathbb{R}^d$, we define the (parameterized) *squared* Euclidean distance

$$D(u, v) \triangleq \|f(u) - f(v)\|_w^2 = \sum_{i=1}^{d} D_i(u, v)\,, \qquad (5)$$

where $D_i(u, v) \triangleq w_i(f(u)_i - f(v)_i)^2$. We also write $D(e)$ to mean $D(u, v)$ when $e = \{u, v\}$.

Observe that according to (4) and (2), $D(u, v)$ must be small if $\{u, v\} \in E$ and large otherwise. How do we enforce this behavior? To start, let us try to enforce this *in expectation*. Notice that if the weights w_i are selected independently for each i, the $\binom{n}{2}$ random variables $\{D_i(e)\}_{e \in \binom{V}{2}}$ are independent and identically distributed Bernoulli random variables. Fix two vertex-pairs $e \in E$ and $e' \in \overline{E}$. *Suppose* that the weights $\{w_i\}$ are selected in such a way that for each i, $D_i(e') - D_i(e)$ is large in expectation. Then a Chernoff-type bound would imply that with high probability, $D(e) < D(e')$.

Realizing trees. Suppose we want to realize a tree $T = (V, E)$. A reasonable way to select $w_i \in \{0, 1\}$ is to select coordinates (i.e., set $w_i = 1$) as follows: we may count how

many of the tree-edges $\{u, v\} \in E$ satisfy $f(u)_i = f(v)_i$. If this fraction is larger than some carefully-computed threshold then most of the tree-edges $\{u, v\}$ already have a small value for the ith component of $D(u, v)$; this suits our purpose and hence we set $w_i = 1$. Otherwise, we set $w_i = 0$. This strategy, dubbed the *census strategy*, is employed in Algorithm 1. Its analysis leads to the bound $d = O(n \log n)$ which is only a $\log n$ factor away from the linear lower bound of Theorem 5.

Realizing graphs. When realizing a general graph G, the edges on a cycle are dependent in a non-trivial way. Observe that the census strategy "touches" multiple edges. If two of them are on a cycle, a crucial argument breaks down in the proof of Claim 3.1. How to overcome this obstacle? A reasonable idea is to *select a single edge* $r = \{u, v\} \in E$ uniformly at random, and then select $w_i \in \{0, 1\}$ as $w_i = 1$ if and only if $f(u)_i = f(v)_i$. We call this the *random sample strategy*. Admittedly, the above w_i would steer only one tree-edge to the right direction, i.e., make $D_i(u, v) = 0$. However, this strategy overcomes the barrier posed by cycles at the cost of a weak bound on d.[1]

Can we do better? Can we combine the ideas in the census strategy and the random sample strategy? The answer is, yes: we can use our tree-realization algorithm to realize a *random acyclic approximation* of G. Specifically, we may look at a family $\mathcal{A}$ of acyclic subgraphs of G and invoke Algorithm 1 on a random member A from this family. (Selection of a random edge, thus, is replaced by the selection of a random acyclic subgraph.) The bound on d that we get in this process depends on the probability that a given edge is contained in the sampled subgraph A. Thus we require that for every edge $e \in E$, there is some $A \in \mathcal{A}$ such that e appears in A.

The best result comes when every edge belongs to exactly one member of $\mathcal{A}$. In that case, $\mathcal{A}$ has to be a collection of edge-disjoint forests. This gives rise to the bound containing $a(G)$, the arboricity of G (cf. Definition 4.1). The arboricity can be interpreted as a measure of how sparse G is. Since the arboricity of a tree is 1, this bound implies the bound for trees. If we take $\mathcal{A}$ to be the set of all spanning trees of G, the bound on d is proportional to $1/r^2$ where r is the smallest effective resistance among all the edges.

A geometric interpretation of Problem 1. The graph realization problem can be reduced to a hyperplane separation problem. Informally speaking, the constraints in (1) specify that $\sum w_i (f(u)_i - f(v)_i)^2$ be small if $\{u, v\} \in E$, and large otherwise. Observe that this quantity is the inner product of the vector $w = (w_1, \cdots, w_d)$ with the vector $g(u, v) \in \{0, 1\}^d$,

$$g(u, v)_i \triangleq \left((f(u)_i - f(v)_i\right)^2 , \tag{6}$$

for $i \in [d]$. Thus, G is realizable if there is a vector w such that for all $u, v \in V$, $\langle w, g(u, v)\rangle < 1$ if and only if $\{u, v\} \in E$. For $e = \{u, v\} \in \binom{V}{2}$, we use the shorthand $g(e)$ for $g(u, v)$.

In Problem 1, we first fix a random map f. Then we locate the $\binom{n}{2}$ points $g(u, v) \in \mathbb{R}^d$ for $\{u, v\} \in \binom{V}{2}$. Observe that the graph G colors these points in red or blue as follows: $g(u, v)$ is colored red if $\{u, v\} \in E$, and blue otherwise. In Problem 1, we have to find a hyperplane h_w, given by weights $w \in \mathbb{R}^d, w_i = \hat{w}_i^2$, so that h_w perfectly separates the red points from the blue points.

Consider the two convex hulls pertaining to the sets of red and blue points. If they intersect, no hyperplane could realize G. This observation is at the heart of our exploration in Section 5 where we investigate whether a random red–blue coloring is separable.

3 Realizing a tree

Observe that the constraints (1) can be equivalently stated as follows: there exists a positive real θ (which may depend on f and w) such that

$$\forall\, u, v \in V\,, \{u, v\} \in E \iff D(u, v) < \theta\,. \tag{7}$$

This can be seen by scaling the weights $\hat{w}$ in Problem 1 by $\sqrt{\theta}$. Hence, if some $w \in \mathbb{R}^d$ and θ satisfy (7), $h = \mathbf{diag}(\hat{w}) \circ f$ satisfies (1) where $\hat{w} \triangleq \sqrt{w/\theta}$.

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

Theorem 1 (Realizing a tree)**.** *Let $n \geq 25$, T be a tree on n vertices, and f be the random variable as in Problem 1. Let $w \in \{0, 1\}^d$ be the output of Algorithm 1 invoked with $\alpha = 1/4$. For a function $d = O(n \log n)$, w satisfies the constraints* (4) *with probability $1 - 1/n$. Thus $R(T, d) \geq 1 - 1/n$.* ◇

Remark. The uncertainty in the claim comes entirely from f as Algorithm 1 is deterministic. Let w be as in Theorem 1 and define $\hat{w}_i = \sqrt{w_i/\theta}$ where θ is defined in (11). Then $h = \mathbf{diag}(\hat{w}) \circ f$ satisfies the constraints (1) with probability $1 - 1/n$. The algorithm runs in time nd.

Suppose we want to realize a tree T using only Boolean weights. Fix any $e = \{u, v\} \in E$ and $e' = \{u', v'\} \in \overline{E}$. How can we enforce the desired behavior *in expectation*? A clue is that only the coordinates i with weight $w_i = 1$ contribute in $D(e)$ or $D(e')$. Thus, we want to select the coordinates (i.e., select which of the w_is are non-zero) such that in expectation, $D_i(e)$ is smaller than $D_i(e')$. To make this precise, we introduce the notion of *gap*.

Definition 3.1 (Gap and total gap)**.** Let $T = (V, E)$ be a tree. Let $e = \{u, v\} \in E$ and $e' = \{u', v'\} \in \overline{E}$. Let

$$\delta_i(e, e') \triangleq \mathop{\mathbb{E}}_{f} \left[D_i(e') - D_i(e)\right], \tag{8}$$

$$\Delta(e, e') \triangleq \sum_i \delta_i(e, e')\,. \tag{9}$$

[1] In fact, if we use the random sample strategy, we can realize for some $d = O(n^2 \log n)$ for trees and some $d = O(n^4 \log n)$ for graphs. We omit further details.

δ_i is the *gap* between e and e' at coordinate i and δ is the *total gap* between e and e'. ◇

Thus, we want to enforce a large gap $\delta_i(e, e')$ at each coordinate i. By the linearity of expectation, the total gap $\Delta(e, e')$ would be large. This deterministic strategy is formalized in Algorithm 1 below.

Algorithm 1 RealizeTree(T, f, α)

Input: $d \in \mathbb{N}$, $T = (V, E)$ a tree, $\alpha \in (0, 1/2)$, and f as defined in Problem 1
Output: $w \in \{0, 1\}^d$

```
for i ∈ [d] independently do
    Set w_i ← 0
    Let p_i be the fraction of edges {u, v} ∈ E
        such that f(u)_i = f(v)_i
    if 1/2 + α/√n ≤ p_i ≤ 3/4 then w_i ← 1
```

We devote the rest of this section analyzing Algorithm 1.

Definition 3.2 (PrAgree)**.** The *agreement probability* for two vertices $u, v \in V$ at coordinate i is $\mathsf{PrAgree}(u, v, i) \triangleq \Pr_f [f(u)_i = f(v)_i \mid w_i = 1]$. ◇

When speaking about a vertex-pair $e = \{u, v\}$, sometimes we write $\mathsf{PrAgree}(e, i)$ to mean $\mathsf{PrAgree}(u, v, i)$.

Definition 3.3 (q)**.** For Algorithm 1, define the *weight selection probability* $q \triangleq \Pr_f[w_i \text{ is set to } 1]$. ◇

When $e \in E, e' \in \overline{E}$ are identified, we can expand Equation 8 to show that

$$\delta_i(e, e') = q\,(\mathsf{PrAgree}(e, i) - \mathsf{PrAgree}(e', i))\,. \tag{10}$$

A bad event occurs when there exist two vertex-pairs $e \in E, e' \in \overline{E}$ with $D(e') \le D(e)$. Our argument for proving Theorem 1 has two steps. In the first step, we prove that for any fixed vertex-pairs a bad event does not occur in expectation. This is equivalent to showing that the total gap $\Delta(e, e')$ is large. The second step has two phases. First, we bound the "bad probability" for a given vertex-pair $\{u, v\} \in \binom{V}{2}$ via a Chernoff bound. Finally, we bound the total bad probability by applying a union bound over all vertex-pairs. Requiring that this probability be $1 - 1/n$ gives a bound on d.

Step one: proving that the total gap δ is large. Fix two vertex-pairs $e \in E$ and $e' \in \overline{E}$. The quantity $Z = \sum_i D_i(e') - D_i(e) = D(e') - D(e)$ is the sum of d independent (but not identically distributed) Bernoulli random variables since $\{w_i\}$ are independent. We proceed by showing that the expectation of the ith component of this sum—i.e., δ_i—is "large." This implies that $D(e')$ is larger than $D(e)$ in expectation. Next, a Chernoff bound on Z would reveal that Z is unlikely to be "too small" compared to its expectation $\mathbb{E}\, Z = \Delta(e, e')$. Equivalently, with "large" probability, the length of the edge e will be strictly shorter than the length of the non-edge e'. This satisfies the constraints on the lengths of e, e' imposed by (7).

Suppose Algorithm 1 assigns $w_i = 1$. We want a lower bound on the gap $\delta_i \triangleq \delta_i(e, e')$, or more appropriately, on the quantity $\mathsf{PrAgree}(e, i) - \mathsf{PrAgree}(e', i)$. Since $w_i = 1$, we have seen exactly $p_i|E|$ edges of T to have the same values at both endpoints. For any two vertices $a, b \in V$, how does $\mathsf{PrAgree}((a, b), i)$ depend on p_i? The answer is given by the following claim.

Claim 3.1 (Decaying correlation)**.** *Let $T = (V, E)$ be a tree and fix a coordinate i. Let $p_i \in (1/2, 1]$ such that $p_i|E|$ is an integer. Let the random variable f be as in Problem 1 and condition on the event that exactly $p_i|E|$ edges $\{u, v\} \in E$ satisfy $f(u)_i = f(v)_i$. Let $\{u, v\} \in \binom{V}{2}$ be an arbitrary vertex-pair. Then, $\mathsf{PrAgree}(u, v, i) = (1 + (2p_i - 1)^t)/2$ where t is the length of the path in T from u to v.* ◇

We remark that the proof of the above claim is the only portion of our analysis which requires T to be a tree. Claim 3.1 implies that $\mathsf{PrAgree}(e, i) - \mathsf{PrAgree}(e', i) = ((2p_i - 1) - (2p_i - 1)^t)/2$ is at least $((2p_i - 1) - (2p_i - 1)^2)/2 = (2p_i - 1)(1 - p_i)$ since $t \ge 2$ for $e' \notin T$ and $(2p_i - 1) \le 1$. It follows that $\delta_i \ge q(2p_i - 1)(1 - p_i)$. However, we want an expression for the right-hand side which does not depend on i. Then the sum $\sum \delta_i$, in turn, would not depend on i as well. Can we lowerbound δ_i in terms of $p \triangleq 1/2 + \alpha/\sqrt{n}$ as opposed to p_i? Indeed we can, but we have to work for it. This is the following claim whose proof is deferred till Appendix B.

Claim 3.2. *$q \ge 1/2 - \alpha - 2^{(H(1/4)-1)n}$ where H is the binary entropy function. In particular, $q \ge 1/6$ when $n \ge 20$ and $\alpha = 1/4$. In addition, for any $i \in [d]$, $\delta_i \ge q(2\alpha/\sqrt{n})(1/2 - \alpha/\sqrt{n})$. In particular, for any $\beta \in (0, 1)$ and $n \ge 4\alpha^2/\beta^2$, $\delta_i \ge (1 - \beta)q\alpha/\sqrt{n}$.* ◇

Step two: bounding the bad probability via Chernoff/union bound. We have already seen that for two fixed vertex-pairs $e \in E$ and $e' \in \overline{E}$, the gap between their respective expectations, i.e., $\Delta(e, e')$, is large. Let $\theta \triangleq \Delta(e, e')/2$ be the midpoint of this gap. A *bad event* occurs when either $D(e) > \theta$ or $D(e') < \theta$. The probability of an individual bad event can be obtained via the Chernoff-Hoeffding bound. Note that there can be at most $\binom{n}{2}$ bad events. The probability that no bad event occurs can be found via a union bound. By setting this probability to at most $1 - 1/n$, we get a bound on d. This is recorded in the following lemma; we defer its proof to Appendix B.

Lemma 3.3 (Bounding d from gap δ_i)**.** *Let the random variable f be as in Problem 1. Let $w_1, \cdots, w_d$ be the weights from Algorithm 1 invoked on the tree $T = (V, E)$ and the embedding f. Define $\delta \triangleq \inf \delta_i(e, e')$ where the infimum is taken over all $i \in [d], e \in E$, and $e' \in \overline{E}$. If $d \ge (6 \log n)/\delta^2$, the constraints (7) are satisfied with probability $1 - 1/n$ (over the random choice of f) with $\theta = d\delta/2$.* ◇

Proof of Theorem 1. Let $w \in \{0,1\}^d$ be the weights generated by Algorithm 1 with $\alpha = 1/4$. Using Claim 3.2, $\delta \geq (1-\beta)q\alpha/\sqrt{n} \geq 9/10 \cdot 1/24\sqrt{n} = 3/80\sqrt{n}$ by taking $q \geq 1/6$, $\alpha = 1/4$, and $\beta = 0.1$. This means $n \geq 4\alpha^2/\beta^2 = 4/4^2 \times 10^2 = 25$. Thus, it *suffices* to use

$$\theta = d\delta/2 = 3d/160\sqrt{n}\,. \tag{11}$$

The bound on d from Lemma 3.3 states that it suffices to take $d \geq 6\log n/\delta^2 = Cn\log n$, where $C = 6/(q\alpha(1-\beta))^2 \leq 6 \times (80/3)^2$, so that w and θ would satisfy (7) with probability $1-1/n$. Thus, as discussed after (7), $\hat{w} \triangleq \sqrt{w/\theta}$ satisfies (1) with probability $1-1/n$. □

Realizing tree-complements and a closer look at the bound. If we modify Algorithm 1 to tally edge-disagreements instead of edge-agreements, we would realize the *complement* of T. The factor $n = (\sqrt{n})^2$ in the bound $d = O(n\log n)$ in Theorem 1 comes from the bias $p = 1/2 + O(1/\sqrt{n})$ in Algorithm 1. The $\log n$ factor in the bound is an artifact of the $1 - 1/\mathrm{poly}(n)$ probability required from the Chernoff bound in the proof of Lemma 3.3, and that there are $\mathrm{poly}(n)$ vertex-pairs in the union bound.

Non-Boolean weights, ℓ_1 realization, and a transformed hypercube. Suppose we use weights $w_i \in \{0,a\}$, $a > 0$ and a uniformly random map $f : V \to \{x,y\}^d$ for arbitrary distinct reals x, y. Let $s \triangleq |x-y|$. Then $D_i(u,v) \in \{0, as^2\}$ in (5). This change will cascade into scaling δ_i and the quantity c in (15) by as^2. Therefore, the ratio δ^2/c^2 would stay the same. The expression $\theta = d\delta/2$ in Lemma 3.3 would be scaled by as^2, however.

A weighted ℓ_1 distance between two points $f(u), f(v)$ is defined as $\sum_i w_i|f(u)_i - f(v)_i|$. It is not hard to see that this would affect δ_i, d, and θ in the same way as above. We omit further details.

4 Realizing a graph

The analysis of the census strategy in the proof of Claim 3.1 requires that the graph being realized is indeed a tree. In particular, it breaks down if G contains a cycle.

To see why, fix a coordinate i and define $\sigma_(u,v) \in \{\pm 1\}$, $\sigma_i(u,v) \triangleq +1$ if and only if $f(u)_i = f(v)_i$. If G contains a triangle (u,v,w) and $\sigma_i(u,v) = \sigma_i(v,w)$, then $\sigma_i(u,w)$ must be 1. In general, $\prod_{e\in C}\sigma_i(e) = 1$ for every cycle C. Due to this correlation in coordinate values along a cycle, a uniform distribution on the coordinate values $\{f(u)_i\}_{u\in V}$ does not imply a uniform distribution on the ensemble $\{\sigma_i(u,v)\}_{\{u,v\}\in E} \cup \{f(r)_i\}$ where $r \in V$ is arbitrary. Thus, the proof in Claim 3.1 (i.e., the census strategy) breaks down when G contains a cycle.

The random sample strategy mentioned in Section 2, however, is immune to any correlation since it samples a single edge. In fact, it is oblivious to any structure in the graph.

Realizing an acyclic approximation of G. Here is an idea which unifies the random edge-sampling and the ease of realizing acyclic graphs. *What if we use an acyclic subgraph as an approximation of G?* Let $\mathcal{A}$ be a collection of acyclic subgraphs of G. We would sample a member A from $\mathcal{A}$ uniformly at random and run the tree-realization algorithm on A. This eliminates all cycles from our view, but it is not obvious that the resulting weights would satisfy the constraints for the edges *not* in A. It turns out that the gap between the two kinds of inner products (i.e., edges vs. non-edges) depends on the probability that the edge belongs to A. This is why $\mathcal{A}$ must cover every edge of G. This strategy is applied by the following algorithm.

Algorithm 2 RealizeGraph$(G, f, \mathcal{A}, \alpha)$

Input: G, an undirected, unweighted, simple graph; f as in Problem 1; $\alpha \in (0, 1/2)$; $\mathcal{A}$, a family of acyclic subgraphs of G such that every edge of G belongs to at least one member of $\mathcal{A}$

Output: $w \in \{0,1\}^d$

1: Sample an element A uniformly at random from $\mathcal{A}$
2: Set $w \leftarrow$ RealizeTree(A, f, α)

$\mathcal{A}$ can simultaneously contain different kinds of acyclic subgraphs such as a single edge, a non-spanning subtree, a spanning tree, a forest, a matching, etc. We record the following lemma and defer its proof till Appendix B.

Lemma 4.1. *Let $n \geq 25$, $\mathcal{A}$ be as described above, f the random variable in Problem 1, and $r \triangleq \min_{\{u,v\}\in E}\Pr_{A\sim\mathcal{A}}[\{u,v\} \in A]$. For a function $d = O\left((n\log n)/r^2\right)$, Algorithm 2 invoked with $\alpha = 1/4$ outputs a non-negative $w \in \{0,1\}^d$ which realizes G with probability $1-1/n$.* ◇

Definition 4.1 (Arboricity)**.** The *arboricity* a of an undirected graph $G = (V,E)$ is the minimum number of forests $F_1, F_2, \ldots, F_a$ so that E is the disjoint union of $F_1, F_2, \ldots, F_a$. ◇

Theorem 2 (Realizing a graph)**.** *Let G be a graph on n vertices with arboricity a, and the random variable f be as in Problem 1. Let $w \in \{0,1\}^d$ be the output of Algorithm 1 invoked with $\alpha = 1/4$. For a function $d = O(na^2\log n)$, w satisfies the constraints* (4) *with probability $1-1/n$. Thus $R(G,d) \geq 1-1/n$.* ◇

Proof. Let a be the arboricity of G. Let $\mathcal{A} = \{\phi_i\}_{i=1}^a$ be the set of all edge-disjoint forests of G so that $E = \phi_1 \sqcup \ldots \sqcup \phi_a$. The edge-disjointedness implies that every edge belongs to a unique forest ϕ_i, and hence $r(e) = 1/a$. Recalling Lemma 4.1, it follows that $d \geq Cna^2\log n$ suffices for realizing G with probability $1-1/n$ where the constant C is from the proof of Theorem 1. In the worst case, $d \geq Cn|E|\log n$ since according to [2], a is at most $\lceil\sqrt{|E|/2}\,\rceil$. □

Theorem 1 is a special case of Theorem 2 because in the former, $\mathcal{A}$ contains only one member: the tree T itself.

Effective resistance and dense graphs. In Appendix B, we show how one can get a weaker bound on d by taking $\mathcal{A}$ as the set of all spanning trees of G. In that case, $r(u, v)$ is in fact the effective resistance (cf. Definition B.1) between the vertices u and v. If a graph G is dense, e.g., if $|E| = \binom{n}{2} - O(n)$, its complement $\overline{G}$ is sparse; consequently, we can realize $\overline{G}$ with a smaller d and then reverse the edge/non-edge labels to recover G. Thus taking some $d = O(na^2 \log n)$ suffices where $a = \min(a(G), a(\overline{G}))$.

5 Realizing random graphs and trees

Recall that for three sets A, B, and C, $C = A \sqcup B$ means C is a disjoint union of A and B, or equivalently, C is partitioned into A and B. A *uniformly random partition* of a set B yields two disjoint sets B_0 and B_1 such that (i.) $B = B_0 \sqcup B_1$ and (ii.) for all $b \in B$, $\Pr[b \in B_0] = \Pr[b \in B_1] = 1/2$.

Informally, Proposition 5.1 below states that if a point-set is linearly separable, then some separating hyperplane passes through exactly d points. Likewise, Proposition 5.2 below (informally) states that a uniformly random partition of a "large" point-set cannot be linearly separated by a hyperplane supported on only "a few" points. We defer the proofs to Appendix C.

Proposition 5.1. *Let S be the affine subspace spanned by a point-set B. Let $B_0 \sqcup B_1$ be a partition of B such that the convex hulls of B_0 and B_1 do not intersect. If $d \triangleq \dim(S) \geq 2$, there exists a hyperplane h which separates B_0, B_1 and is supported on exactly d points of B.* ◇

Proposition 5.2. *Let $M, d \in \mathbb{N}, d \geq 3, M \geq 6d$. Let B be an arbitrary set of M points in $\mathbb{R}^d$. Let $B = B_0 \sqcup B_1$ be a uniformly random partition of B. Then with probability $1 - 1/d$, the convex hulls of B_0 and B_1 cannot be separated by a hyperplane supported on any d points of B.* ◇

A probabilistic analog of Radon's theorem. Recall the geometric interpretation of the realization problem from Section 2. Suppose G is an Erdős-Rényi random graph $G \sim \mathcal{G}(n, 1/2)$. Since the edges in G are sampled independently, the red/blue assignments of the points $\{g(u, v)\}_{u,v \in V}$ will be uniformly random. *Is there a hyperplane which separates the red points from the blue points?* The following theorem says that it is unlikely for small d.

Theorem 3. *Let $d, n \in \mathbb{N}$ and let B be an arbitrary subset of $\{0, 1\}^d$ of size at least $6d$. In a uniformly random partition $B_0 \sqcup B_1 = B$, the convex hulls of B_0 and B_1 intersect with probability at least $1 - 1/d$.* ◇

Proof. Proposition 5.2 states that for a uniformly random partition $B_0 \sqcup B_1 = B \subset \{0, 1\}^d$, with high probability, there is no d-supported hyperplane. The contrapositive of Proposition 5.1 states that if there is no d-supported separating hyperplane, there is no separating hyperplane as well. Therefore, if d is at most $|B|/6$, a uniformly random partition of a subset of $\{0, 1\}^d$ is nonseparable with probability $1 - 1/d$. □

Theorem 3 can be considered a probabilistic analog of Radon's Theorem in convex geometry (cf. Theorem 6). See Appendix C for a discussion. Since the points B can be arbitrary, we can take $f : V \to \{0, 1\}^d$ to be arbitrary, set $B = g\binom{V}{2}$ where g is from (6), and allow the weights w_i to be arbitrary reals (i.e., arbitrary hyperplanes). This leads to the main theorem of this section.

Theorem 4 (Realizing random graphs)**.** *Let $n, d \in \mathbb{N}, n \geq 7, d \leq \binom{n}{2}/6$, and let $G \sim \mathcal{G}(n, 1/2)$ be an Erdős-Rényi random graph. With probability at least $1 - 1/d$ in the randomness of G, G is not realizable under any $f : V \to \{0, 1\}^d$ and any $w \in \mathbb{R}^d$. Thus $R(G, d) = 0$.* ◇

Proof. Sample an Erdős-Rényi random graph $G = (V, E) \sim \mathcal{G}(n, 1/2)$. Also, let $f : V \to \{0, 1\}^d$ be an arbitrary embedding with $d \leq \binom{n}{2}/6$. Let g be as in (6). Since E is a uniformly random subset of $\binom{V}{2}$, we can invoke Theorem 3 to show that with high probability, the uniformly random partition $g(E) \sqcup g(\overline{E})$ of the mapped vertex-pairs $B = g\binom{V}{2}$ is not linearly separable. (However, this would require that $\binom{n}{2} \geq 6d \geq 18$ since we require $d \geq 3$ as well. Thus it suffices to have $n \geq 7$.) Consequently, there exists no hyperplane (indicated by some $w \in \mathbb{R}^d$) which separates $g(E)$ from $g(\overline{E})$. Recall that our definition of linear separability has inequality constraints. If these constraints cannot be satisfied by any hyperplane, it follows that the strict inequality constraints in (1) cannot be satisfied either. Therefore, the random graph G is not realizable by any w under any embedding f. Note that the randomness in this argument comes from G. Hence, $R(G, d) = 0$. □

A similar theorem holds for random spanning trees; we defer the proof to Appendix C.

Theorem 5 (Realizing random spanning trees)**.** *Let $n \geq 17, d \leq n/2$, and $\mathcal{T}$ be the uniform distribution on the spanning trees of the complete graph K_n. Sample a tree $T = (V, E)$ according to $\mathcal{T}$. With probability at least $1 - 1/n$ (in the randomness of T), T is not realizable under any map $f : V \to \{0, 1\}^d$ and any weights $w \in \mathbb{R}^d$. Thus $R(T, d) = 0$.* ◇

Acknowledgments

We thank Ion Măndoiu for helpful discussions and in particular, for introducing to us the realization problem in [6]. We also thank Benjamin Fuller and Donald Sheehy for many discussions and comments; these have greatly improved the quality of the manuscript. At last but not the least, S.Q. would like to thank Dilruba Mukti for her comments.

References

[1] J. Bourgain. On lipschitz embedding of finite metric spaces in hilbert space. *Israel Journal of Mathematics*, 52(1):46–52, 1985.

[2] A. M. Dean, J. P. Hutchinson, and E. R. Scheinerman. On the thickness and arboricity of a graph. *Journal of Combinatorial Theory, Series B*, 52(1):147–151, 1991.

[3] P. Eades and S. Whitesides. The realization problem for euclidean minimum spanning trees is np-hard. *Algorithmica*, 16(1):60–82, 1996.

[4] W. Ellens, F. Spieksma, P. Van Mieghem, A. Jamakovic, and R. Kooij. Effective graph resistance. *Linear algebra and its applications*, 435(10):2491–2506, 2011.

[5] U. Feige. Approximating the bandwidth via volume respecting embeddings. *Journal of Computer and System Sciences*, 60(3):510–539, 2000.

[6] M. A. Ghadie, N. Japkowicz, and T. J. Perkins. Gene selection for the reconstruction of stem cell differentiation trees: a linear programming approach. *Bioinformatics*, 31(16):2676–2682, 2015.

[7] X. He, D. Cai, S. Yan, and H.-J. Zhang. Neighborhood preserving embedding. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 2, pages 1208–1213. IEEE, 2005.

[8] B. Hendrickson. Conditions for unique graph realizations. *SIAM journal on computing*, 21(1):65–84, 1992.

[9] J. A. King. Realization of degree 10 minimum spanning trees in 3-space. In *CCCG*, 2006.

[10] L. Liberti and C. Lavor. On a relationship between graph realizability and distance matrix completion. In *Optimization theory, decision making, and operations research applications*, pages 39–48. Springer, 2013.

[11] N. Linial, E. London, and Y. Rabinovich. The geometry of graphs and some of its algorithmic applications. *Combinatorica*, 15(2):215–245, 1995.

[12] N. Linial and M. Saks. The euclidean distortion of complete binary trees. *Discrete and Computational Geometry*, 29(1):19–22, 2003.

[13] G. Liotta and G. Di Battista. Computing proximity drawings of trees in the 3-dimensional space. In *Workshop on Algorithms and Data Structures*, pages 239–250. Springer, 1995.

[14] J. Matoušek. On embedding expanders into ℓ_p spaces. *Israel Journal of Mathematics*, 102(1):189–197, 1997.

[15] J. Matoušek. *Lectures on discrete geometry*, volume 212. Springer Science & Business Media, 2002.

[16] M. Schultz and T. Joachims. Learning a distance metric from relative comparisons. In *NIPS*, volume 1, page 2, 2003.

[17] B. Shaw and T. Jebara. Structure preserving embedding. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 937–944. ACM, 2009.

[18] S. Yan, D. Xu, B. Zhang, H.-J. Zhang, Q. Yang, and S. Lin. Graph embedding and extensions: A general framework for dimensionality reduction. *IEEE transactions on pattern analysis and machine intelligence*, 29(1), 2007.

A Related work

The connections between Problem 1 and the related problems in the literature fall into two broad categories: (1) finding the best map under some constraints (e.g., constrained embedding and metric embedding), and (2) finding the best metric for a given map (i.e., metric optimization). We delve into these connections below. In the end, we would see that we cannot use existing results to solve Problem 1. is

A.1 Constrained embedding

Euclidean minimum spanning tree realization. The EMST realization problem has important applications in graph drawing and VLSI circuit design [3, 9].

Definition A.1 (Euclidean minimum spanning tree (EMST) realization)**.** Let $d, n \in \mathbb{N}$; let $T = (V, E)$ be a tree on n vertices. Let $h : \binom{V}{2} \to \mathbb{R}^d$ and let $T' = (h(V), E')$ be the geometric spanning tree on the points $h(V)$ under ℓ_2 with the smallest total edge-length. We say h *EMST-realizes* T if for all $u, v \in V$, $\{u, v\} \in E$ if and only if $\{h(u), h(v)\} \in E'$. ◇

A restricted formulation of the EMST-realization problem is, in fact, equivalent to Problem 1 for trees.

Claim A.1. *Suppose $w \in \mathbb{R}^d$ realizes a tree $T = (V, E)$ on a fixed random map f. Then $h = \mathbf{diag}(\hat{w}) \circ f$ EMST-realizes T where the weights are $\hat{w}_i = \sqrt{w_i}, i \leq d$. On the other hand, if $h = \mathbf{diag}(\hat{w}) \circ f$ EMST-realizes T, then there is some positive real θ so that the weights $w \in \mathbb{R}^d$, defined as $w_i = \hat{w}_i^2/\theta$, realizes T on f.* ◇

Proof. If w realizes T then according to (1),

$$\|h(u) - h(v)\|_2 < 1 \quad \text{for all} \quad \{u, v\} \in E\,, \text{ and}$$
$$\|h(u) - h(v)\|_2 > 1 \quad \text{for all} \quad \{u, v\} \notin E\,.$$

Clearly, a spanning tree algorithm using the ℓ_2 metric will detect this gap and the resulting tree T' will be identical to T.

On the other hand, suppose $h = \mathbf{diag}(\hat{w}) \circ f$ EMST-realizes T, and let T' be the resulting unique geometric minimum spanning tree on $h(V)$ under the ℓ_2 metric. Consider the T'-edges incident on a point $h(u)$. Clearly, the lengths of these edges are strictly smaller than $\|h(u) - h(u')\|_2$ for any point $h(u')$ that is not a T'-neighbor of $h(u)$. By assumption, T' is identical to T. Accounting for the neighborhoods of all points $h(u), u \in V$, we conclude that there must be some positive real θ so that

$$\|h(u) - h(v)\|_2 < \theta \qquad \text{if and only if} \qquad \{u, v\} \in T\,. \quad (12)$$

Let $\hat{W}, W$ and F be the matrices associated with the liner maps $\hat{w}, w$ and f. Let us also write the matrix $W = \hat{W}^2$ and $f(u) = F\mathbf{u}$ where $\mathbf{u}$ is the uth standard basis vector of $\mathbb{R}^d$. Let the matrix W' associated with the linear map w' be defined as $W' \triangleq \hat{W}/\theta$; define the linear map $h' \triangleq w' \circ f$. Then

$$\begin{aligned}\left\|h'(u) - h'(v)\right\|_2^2 = \left\|W'F(\mathbf{u} - \mathbf{v})\right\|_2^2 &= \left\|\hat{W}F(\mathbf{u} - \mathbf{v})\right\|_2^2/\theta^2 \\ &= \|h(u) - h(v)\|_2^2/\theta^2\,. \end{aligned} \quad (13)$$

By appealing to (12), it follows that the weights $w = (w_1, \ldots, w_d)$, defined as $w_i = w_i'^2$ would satisfy (1) and thus realize T on f. □

The contrast between the EMST realization problem and Problem 1 lies in the role of the embedding f, i.e., whether it is an input or an output. In connection with 6, suppose a map $h = \mathbf{diag}(\hat{w}) \circ f$ solves the EMST-realization problem. Then h is equivalent to the following: First, imagine that the tree $T = (V, E)$ colors the $\binom{n}{2}$ elements of $\binom{V}{2}$ as red (edges) or blue (non-edges). Then, we fix a hyperplane h_w given by its normal vector $w = (1, \ldots, 1) \in \mathbb{R}^d$. Finally, we choose an embedding $f : V \to \{0,1\}^d$ such that $g(u, v)$ from (6) maps all red elements in one side of the hyperplane h_w and all blue points in the other side of h_w. Thus, we are free to choose f whereas in Problem 1, f is a given random embedding.

Not all trees can be EMST-realized in low dimensions. Let Δ be the largest vertex degree in the tree. An $\mathbb{R}^2$-realization is always possible if $\Delta \leq 5$ but impossible if $\Delta \geq 7$. In addition, deciding whether an $\mathbb{R}^2$-realization is possible is NP-hard if $\Delta \leq 6$ [3]. Likewise, an $\mathbb{R}^3$-realization is always possible if $\Delta \leq 10$ [9], while it is impossible if $\Delta \geq 12$ [13].

Structure preserving embedding. A *structure preserving map* (SPE) of a graph G into ℓ_2^d preserves some global or local topological structure of a set of high-dimensional data points P while projecting them into a space of lower dimension [17, 7, 18]. As mentioned in the beginning, this problem is a restricted embedding problem and thus has a notional similarity with Problem 1. In these problems, however, one has to infer the "structure" from P itself, whereas in Problem 1, the structure is given as the adjacency matrix. Moreover, it is hard to relate the data points P to the vertices V or the mapping $f(V)$ in Problem 1; in particular, V does not live in a metric space and the map f is random.

A.2 Metric embedding

Recall the metric embedding problem from [15, Ch. 15.1].

Definition A.2 (D-embedding of metric spaces)**.** A mapping $f : X \to Y$, where X is a metric space with a metric ρ and Y is a metric space with a metric σ, is called a D-embedding, where $D \geq 1$ is a real number, if there exists a number $r > 0$ such that for all $x, y \in X$,

$$r \cdot \rho(x, y) \leq \sigma(f(x), f(y)) \leq D \cdot r \cdot \rho(x, y)\,.$$

The infimum of the numbers D such that f is a D-embedding is called the distortion of f. We express a D-embedding as $(X, \rho) \overset{D}{\hookrightarrow} (Y, \sigma)$. ◇

Problem 1 is different from metric embedding, as follows. In a metric embedding problem, we want to preserve all pairwise distances. However, in Problem 1, the map h has to preserve only the adjacency relation A (given by G) on the mapped points. In particular, let $A : \binom{V}{2} \to \{0, 1\}$ be defined as $A(u, v) = 1$ if and only if $\{u, v\} \in E$ and $A_h : \binom{h(V)}{2} \to \{0, 1\}$ be defined as $A_h(u, v) = 1$ if and only if $\|h(u) - h(v)\| < 1$. Then h has to satisfy $A(u, v) = A_h(u, v)$ for all $u, v \in V$. In addition, A is not a metric since it does not satisfy the triangle inequality.

However, a solution to Problem 1 is implied by a constant-distortion metric embedding into a subset of ℓ_2. However, one cannot embed a tree or a graph into ℓ_2 with a constant distortion; a $\Omega(\log n)$ distortion is necessary for graphs and a $\Omega(\sqrt{\log \log n})$ distortion is necessary for trees [14, 1, 11, 12]. It follows that we cannot solve Problem 1 via metric embedding.

Claim A.2. *Let ρ be the shortest-path metric for the unweighted, undirected graph $G = (V, E)$ and let $\epsilon \in [0, 1)$. Let $d \in \mathbb{N}$ and $f : V \to \{0,1\}^d$ be arbitrary. If $(V, \rho) \overset{1+\epsilon}{\hookrightarrow} (\{0,1\}^d, \ell_2)$ then* (4) *is satisfied by the weights $w \triangleq \omega/4$.* ◇

Proof. First, we claim that $\{u, v\} \in E$ iff $\|f(u) - f(v)\|_\omega^2 < 4$. To see why, observe that since G is unweighted, $\rho(e) = 1$ if $e \in E$ and $\rho(e') \geq 2$ if $e' \in \overline{E}$. For any $e = \{u, v\} \in E$ and $e' = \{u', v'\} \in \overline{E}$, we have $\|f(u) - f(v)\|_\omega \leq (1 + \epsilon)\rho(u, v) < 2$, and $\|f(u') - f(v')\|_\omega \geq \rho(e') \geq 2$.

Consider the scaling w given by the weights $w \in \mathbb{R}^d, w_i = \omega_i/4$. Since $\|\cdot\|_w^2 = \|\cdot\|_\omega^2/4$, the weights w would satisfy (4).

Observe that the metric space $(f(V), \|\cdot\|_\omega)$ is in fact isomorphic to $(\phi(V), \ell_2)$ for some $\phi : V \to \mathbb{R}^d$ which depends only on f and w. This completes the proof. □

A.3 Optimizing a weighted ℓ_2 metric

In the *Euclidean distance matrix realization* problem [10], we are given a set V and a matrix $\hat{D}$ containing the desired pairwise distances for the elements in V. To realize $\hat{D}$ in $\mathbb{R}^d$, we have to find a map $h : V \to \mathbb{R}^d$ such that for all $u, v \in V$, $\|h(u) - h(v)\|_2$ in fact equals $D(u, v)$. In [8], Hendrickson studied the conditions under which a graph has a unique realization in this sense. Although Problem 1 can be seen as a thresholded version of this distance matrix realization problem—the adjacency matrix plays the role of the distance matrix $\hat{D}$. Unfortunately, the adjacency matrix does not give a metric. (For example, it does not give a triangle inequality.) Hence, the results concerning this problem do not directly apply to our problem.

Metric learning. Under a suitable formulation, the *supervised metric learning problem* requires one to learn a weighted ℓ_2 metric on a given point-set P where the adjacencies are also given as input [16]. In fact, in that study, one finds a metric $d_{A,W}$ given by the positive semidefinite matrix AWA^T where W is diagonal and A is arbitrary. The goal is to make $d_{A,W}$ as close to the unweighted Euclidean metric as possible. Although this formulation becomes identical to Problem 1 if we set A as the identity matrix, an important issue is the random map f in Problem 1; in fact, the data points in a learning task are not random. (Otherwise, how could we learn anything about these points?)

Realizing a tree on biological data. A 2015 study [6] studies a realization problem for a tree $T = (V, E)$ on an embedding $f(V) \subset \{0,1\}^d$ in ℓ_2. It studies the problem of realizing a tree on gene-expression data. Specifically, the given rooted tree T is a *stem-cell differentiation tree* for $n = 38$ *cell-types*. The vertices of T are different cell-types and an edge indicates whether two cell-types u and v are related. The matrix F associated with the embedding f is a $d \times n$ Boolean matrix where $d = 22,515$. Each row of F corresponds to a different gene; there are d genes in the study. The entry $f(u)_i$ indicates whether the gene $i \in [d]$ is *expressed* in the sample corresponding to the cell-type u. Thus, the embedding of the cell-type $u \in V$ is the uth column of F.

In this study, the authors ask *whether the data points $f(V)$ can explain the given tree T*. Here, *explanation* means that there should exist a map $h = \mathbf{diag}(\hat{w}) \circ f$ capturing the underlying biological phenomenon, i.e., which genes were important in generating the given tree T. They formulated a linear program to find a feasible set of non-negative weights $w \in \mathbb{R}^d$ while keeping the number

of non-zero weights as small as possible. (This is reminiscent of the Semidefinite Programming (SDP) formulation of the celebrated paper by Linial, London, and Rabinovich [11].)

A difference between the problem in [6] and Problem 1 is that in their case, the embedding f was given by a biological mechanism (i.e., gene expression) whereas in Problem 1, the embedding is random.

B Proofs omitted in Sections 3 and 4

Claim 3.1

Proof. Fix a coordinate i. Below, we describe a stochastic process for selecting the values $\{f(u)_i\}_{u\in V}$; we call this the *uniform edge-sign process*. It is as follows: (i.) Associate with each edge $e \in E$ a random variable $\sigma(e) \in \{0,1\}$. (ii.) Independently for each edge $e \in E$, flip a fair two-sided coin; set $\sigma(e) = 1$ if and only if it turns up head. (iii.) Select an arbitrary vertex $u \in V$ and flip a fair two-sided coin; set $f(u)_i = 1$ if and only if it turns up head. (iv.) Finally, for each edge $e = \{a, b\} \in T$ such that $f(a)_i$ is assigned but $f(b)_i$ is unassigned, set $f(b)_i = f(a)_i$ if $\sigma(e) = 1$; otherwise, set $f(b)_i = 1 - f(a)_i$.

We claim that the outcome of the uniform edge-sign process is isomorphic to the outcome of the original process described in Problem 1. In particular, the variables $\{f(u)_i\}_{u\in V}$ are i.i.d. Bernoulli random variables with parameter $1/2$. In addition, in both cases, we used n independent coin flips to assign n bits of information.

Now let us describe a biased version of the above process. Specifically, let f be as in Problem 1 and as in the statement of Claim 3.1, let us condition on the event that there are exactly $p_i|E|$ edges $\{a, b\} \in E$ such that $f(a)_i = f(b)_i$. As before, we can define an analogous edge-sign stochastic process. We need only modify the step (ii.) of the uniform edge-sign process. We call the resulting process the *biased edge-sign process*.

Specifically, let $m = |E|$, $m' = p_i|E|$, and let $\binom{E}{m'}$ be the collection of all size-m' subsets of E. The step (ii.) of the uniform edge-sign process is modified as follows: (ii.) Uniformly select an element $A \in \binom{E}{m'}$. For each $e \in E$, set $\sigma(e) = 1$ if $e \in A$; otherwise, set $\sigma(e) = 0$.

Notice that the distribution of $\{f(u)_i\}$ generated by the biased edge-sign process is identical to the *observed* distribution of $\{f(u)_i\}$, i.e., exactly $p_i|E|$ of the $\sigma(e)$s are 1. Specifically, since A is a random element of $\binom{E}{m'}$, the "edge signs" $\{\sigma(e)\}_{e\in E}$ have i.i.d. Bernoulli distribution with parameter p_i.

Let P be the unique path from u to v along T, whose length is t. Let $S_t = \sum_{e\in P} \sigma(e)$ and notice that S_t has a binomial distribution since $\sigma(e)$ are i.i.d. Bernoulli random variables. Define $c(t) \triangleq \Pr[S_t \text{ is even}]$. Since S_t has a binomial distribution with parameters (t, p_i), it is not hard to show that $c(t) = \left(1 + (2p_i - 1)^t\right)/2$. Note that $c(t)$ also equals $\mathsf{PrAgree}(u, v, i)$ conditioned on $d_T(u, v) = t$ where d_T is the shortest-path metric on T. The claim follows. □

Proposition B.1 (Anti-concentration)**.** *Let $n \geq 3$. Let Z be a random variable with the binomial distribution $B(n-1, 1/2)$. Suppose $\alpha \in (0, 1/2)$. Then*

$$(1/2 - \alpha) < \Pr\left[Z \geq \mathbb{E}\, Z + \alpha\sqrt{n}\right] < 1/2\,.$$

◇

Proof. It is easy to see that $\Pr[Z > \mathbb{E}\, Z + \alpha\sqrt{n}]$ is less than $1/2$ since the volume of a "proper" tail cannot exceed $1/2$.

Note that the peak of a binomial distribution remains relatively flat for small deviations from the mean. The area under the pmf curve in that region can be closely overestimated by a (slightly larger) rectangle. This rectangle will have width $\alpha\sqrt{n}$ and height $\binom{n}{n/2}$ where $\sigma^2 = n/4$ is the variance of a binomial distribution $B(n, 1/2)$ and α is a small positive constant. We want to show that the mass in the tail beyond $n/2 + \alpha\sqrt{n}$ is *larger* than a constant. Let $N = n - 1$.

$$\begin{aligned}
q = \Pr\left[S_N \geq N/2 + \alpha\sqrt{n}\right] &= 1/2 - \sum_{k=N/2}^{N/2+\alpha\sqrt{n}} \frac{\binom{N}{k}}{2^N} \\
&> 1/2 - (\alpha\sqrt{n})\binom{N}{N/2}2^{-N} \\
&\approx 1/2 - (\alpha\sqrt{n})\left[\frac{\sqrt{2}}{\sqrt{\pi}}\frac{2^N}{\sqrt{N}}\right]2^{-N} \qquad \text{(Stirling)} \\
&= 1/2 - \alpha\sqrt{2/\pi}\sqrt{n/N} \\
&> 1/2 - \alpha \qquad \text{for} \quad n \geq 3\,.
\end{aligned}$$

□

Claim 3.2

Proof. Fix coordinate i. Let $p \triangleq 1/2 + \alpha\sqrt{n}$ and $\epsilon \triangleq \epsilon_i = p_i - p$ where p_i is the fraction of agreeing edges at coordinate i. Recall that $\delta_i \geq q(2p_i - 1)(1 - p_i)$. Substituting $p_i = p + \epsilon$, we get $\delta_i \geq q(2p + 2\epsilon_i - 1)(1 - p - \epsilon_i) = q(2p - 1)(1 - p) + \lambda(p, \epsilon)$ where $\lambda(p, \epsilon) = q\epsilon(3 - 4p - 2\epsilon)$. It follows that

$$(2p_i - 1)(1 - p_i) \geq (2\alpha/\sqrt{n})(1/2 - \alpha/\sqrt{n})\,, \tag{14}$$

or $\delta_i \geq q(2p - 1)(1 - p)$, as long as $\lambda(p, \epsilon) \geq 0$. Since both q is strictly positive, this inequality gives us $\epsilon \leq 1/2 - 2\alpha/\sqrt{n}$. This condition is equivalent to requiring $p_i \leq 1 - \alpha/\sqrt{n}$. Recall that in Algorithm 1, we have put a stronger requirement that p_i must fall within the interval $[p, 3/4]$ for w_i to be 1. Therefore, if $w_i = 1$ then $\lambda(p, \epsilon) \geq 0$. Since $\delta_i = 0$ if $w_i = 0$, it follows that $\delta_i \geq q(2p-1)(1-p) = q(2\alpha/\sqrt{n})(1/2-\alpha/\sqrt{n}) = q\alpha/\sqrt{n}-2q\alpha^2/n$.

Let $\beta \in (0, 1)$. The above quantity will be at least $(1-\beta)q\alpha/\sqrt{n}$ if $2q\alpha^2/n$ is at most $\beta q\alpha/\sqrt{n}$; this is equivalent to requiring $n \geq 4\alpha^2/\beta^2$. It remains to find a lowerbound for $q = \Pr[w_i = 1] = \Pr[p \leq p_i \leq 3/4]$ so that we can use it to lowerbound δ_i.

Let Z be a random variable with a binomial distribution $B(n - 1, 1/2)$. Let $a \in [0, 1]$, and define $\mathsf{Tail}(a) \triangleq \Pr[Z \geq (n - 1)a]$. According to Proposition B.1, $\mathsf{Tail}(p) > 1/2 - \alpha$. However, $\mathsf{Tail}(p) = q + \mathsf{Tail}(3/4)$, which implies $q \geq (1/2-\alpha) - \mathsf{Tail}(3/4)$.

Fact B.2. *For any positive integer n and $\beta \in [0, 1/2]$ such that βn is an integer,*

$$\sum_{k=0}^{\beta n}\binom{n}{k} \leq 2^{H(\beta)n}\,,$$

where $H(\beta)$ is the binary entropy function defined as $H(x) = -x\log_2 x - (1 - x)\log_2(1 - x)$ for $x \in [0, 1]$. ◇

Therefore, $\mathsf{Tail}(3/4) = \sum_{k=3n/4}^{n}\binom{n}{k} = \sum_{k=0}^{n/4}\binom{n}{k} \leq 2^{(H(1/4)-1)n} \leq 2^{-0.18n}$ since $H(1/4) \leq 0.82$. Consequently, $q \geq (1/2 - \alpha) - 2^{-0.18n}$.

□

We remark that if we use the uniformly random embedding $f : V \to \{x, y\}^d$ for arbitrary $x, y \in \mathbb{R}$, then the δ_i in the above claim is at least $qs^2(2\alpha/\sqrt{n})(1-\alpha/\sqrt{n})$ where $s \triangleq |x-y|$.

Lemma 3.3

Proof. Let $X = D(u, v)$ and $Y = D(u', v')$. The random variables X and Y are sums of d independent sub-Gaussian components, each component taking values in the interval $[0, 1]$ of width $c \triangleq 1$.

First, we want to show that $Y - X > 0$ with high probability. Since (9) tells us $\mathbb{E}(Y - X) = \sum \delta_i \geq d\delta$, it suffices to show that $(Y - \mathbb{E}\, Y) > \theta > (X + \mathbb{E}\, X)$ where $\theta = d\delta/2$.

Let $\mathcal{H}_{u,v}$ be the event that for an arbitrary edge $\{u, v\} \in E$, X is "too small" compared to its expectation. Then, by Hoeffding's tail inequality, we have

$$\Pr[\mathcal{H}_{u,v}] = \Pr\{\mathbb{E}\, X - X > \theta\} < \exp\left(-\frac{2\theta^2}{\sum_{i \leq d} c^2}\right) = e^{-2\theta^2/c^2 d} = e^{-d\delta^2/2}. \quad (15)$$

Similarly, let $\mathcal{H}_{u,u'}$ be the event that for an arbitrary non-edge $\{u, u'\} \in \overline{E}$, Y is "too large" compared to its expectation. In this case, we get $\Pr \mathcal{H}_{u,u'} = \Pr\{Y - \mathbb{E}\, Y > \theta\} < e^{-d\delta^2/2}$.

Now, a bad event $\mathcal{B}$ is one of the above two events for some vertex-pair in $\binom{V}{2}$. We want to show that the probability of this event is at most an inverse polynomial in n. Using a union bound over the $(n-1)$ tree edges and the remaining non-tree edges, we get $\Pr \mathcal{B} \leq n^2 e^{-d\delta^2/2} = \exp\left(2\log n - d\delta^2/2\right)$.

This probability will be at most $1/n = e^{-\log n}$ if $-2\log n + d\delta^2/2 \geq \log n$, giving us $d \geq 6(\log n)/\delta^2$. □

Lemma 4.1

Proof. In Algorithm 1's context, let $p \triangleq 1/2 + 1/4\sqrt{n}$, and $q = \Pr[w_i = 1]$. For every edge $e = \{u, v\} \in G$, let

$$r(e) \triangleq \Pr_A[e \in A],$$

and t be the length of the unique u-v path in A if it exists, and ∞ otherwise. Notice that $t = 1$ if $\{u, v\} \in A$, and $t_e \geq 2$ otherwise.

Fix a coordinate i and condition on the event that $w_i = 1$. Recall the quantity $\mathsf{PrAgree}(e, i)$ (cf. Definition 3.2) for an arbitrary edge $e = \{u, v\} \in G$.

$$\begin{aligned}
&\mathsf{PrAgree}(e, i | e \in G) \\
&\quad = r(e)\mathsf{PrAgree}(e, i | e \in A) + (1 - r(e))\mathsf{PrAgree}(e, i | e \notin A) \\
&\quad = r(e)\left(\mathsf{PrAgree}(e, i | e \in A) - \mathsf{PrAgree}(e, i | e \notin A)\right) \\
&\qquad + \mathsf{PrAgree}(e, i | e \notin A)
\end{aligned}$$

This implies, the gap δ_i is q times

$$\begin{aligned}
&\mathsf{PrAgree}(e, i | e \in G) - \mathsf{PrAgree}(e, i | e \notin G) \\
&\quad = r(e)\left(\mathsf{PrAgree}(e, i | e \in A) - \mathsf{PrAgree}(e, i | e \notin A)\right) \\
&\qquad + \mathsf{PrAgree}(e, i | e \notin A) - \mathsf{PrAgree}(e, i | e \notin G) \\
&\quad = r(e)\left(\mathsf{PrAgree}(e, i | e \in A) - \mathsf{PrAgree}(e, i | e \notin A)\right)
\end{aligned}$$

since conditioned on any A, the last two terms are the same. Continuing, the δ_i is q times

$$\begin{aligned}
&r(e)\left([1 + (2p_i - 1)]/2 - [1 + (2p_i - 1)^t]/2\right) \quad \text{using Claim 3.1} \\
&\quad \geq r(e)\left((2p_i - 1) - (2p_i - 1)^2\right)/2 \quad \text{since} \quad t \geq 2 \\
&\quad = r(e)\,(2p_i - 1)(1 - p_i)\,.
\end{aligned}$$

Writing $r \triangleq \min_e r(e)$ and applying the reasoning from the proof of Claim 3.2, it follows that $\delta_i \geq r(1-\beta)q\alpha/\sqrt{n}$ if $n \geq 4\alpha^2/\beta$. Retracing our footsteps in the proof of Theorem 1, we see that taking $d \geq C \cdot (n/r^2)\log n$—where C is the constant in the proof of Theorem 1—would suffice for realizing G with probability at least $1 - 1/n$. Using $\beta = 0.1, \alpha = 1/4$, and $q \geq 1/6$ implies $n \geq 25$. □

Effective resistance and Theorem 2.

Definition B.1 (Effective resistance)**.** Let $G = (V, E)$ be an undirected graph corresponding to an electrical network where each edge contains a unit resistance. For every vertex $u \in [n]$, let $\mathbf{u}$ be the uth standard basis vector of $\mathbb{R}^n$ i.e., $\mathbf{u}_u = 1, \mathbf{u}_v = 0$ for all $v \neq u$. Let A be the adjacency matrix of G and let D be a diagonal *degree matrix* of G defined as $D(u, u) = \deg(u)$. Then the matrix $L = D - A$ is called the *Laplacian matrix* of G. Let L^+ be the Moore-Penrose pseudoinverse of L. ($\mathrm{rank}(L) = n - c$ where c is the number of connected components of G.) The *effective resistance* between two vertices u, v is given by

$$\mathrm{R}_{\mathrm{eff}}(u, v) = (\mathbf{u} - \mathbf{v})^T L^+ (\mathbf{u} - \mathbf{v})\,.$$

◇

The effective resistance is intimately linked with many combinatorial properties of a graph. (Cf. [4] for further reading.) We recall the following well-known fact.

Fact B.3. *Let $\mathcal{T}$ be the set of all spanning trees of the undirected graph $G = (V, E)$. Sample a tree T uniformly at random from $\mathcal{T}$. Then $\Pr_{T \sim \mathcal{T}}[e \in T] = \mathrm{R}_{\mathrm{eff}}(e)$. Moreover, $\mathrm{R}_{\mathrm{eff}}(e) \geq 2/n$ for any $e \in E$.* ◇

Suppose we take $\mathcal{A}$ as the set of all spanning trees of G in the proof of Theorem 2. Using Fact B.3, we can see that $r(u, v) = \mathrm{R}_{\mathrm{eff}}(u, v)$. This gives $d = O\left((n \log n)/\left(\min_e \mathrm{R}_{\mathrm{eff}}(e)\right)^2\right)$. Since $\mathrm{R}_{\mathrm{eff}}(e) \geq 2/n$, $d = O(n^3 \log n)$ in the worst case. This bound is weaker than what we get if we use the arboricity in the proof of Corollary 2.

C Proofs omitted in Section 5

Let us make concrete the notion of "linear separability" which is at the center of our argument.

Definition C.1 (Linear separability)**.** Two point-sets $A, B \in \mathbb{R}^d$ are *linearly separable* (or *separable* in short) if there exists a hyperplane with a normal vector w such that $\langle a, w\rangle \leq \langle b, w\rangle$ for all $a \in A, b \in B$. ◇

Radon's Theorem and Theorem 3 A well-known theorem in convex geometry is Radon's theorem. It relates the linear separability of point-sets with the ambient dimension. The theorem states that it is always possible to label any collection of at least $d + 2$ points in $\mathbb{R}^d$ into two subsets which are not linearly separable.

Theorem 6 (Radon's Theorem). *If B is a set of M points in $\mathbb{R}^d$ with $M \geq d+2$, there exists a partition $B_0 \sqcup B_1 = B$ such that the convex hulls of B_0 and B_1 have a non-empty intersection. Consequently, there can be no hyperplane separating B_0 from B_1.* ◇

In our context, Theorem 6 says "for every point-set $B = g\binom{V}{2}$, there exists a nonseparable partition of B." However, we want to show that "there exists a graph $G = (V, E)$ such that the partition $g(E) \sqcup g(\overline{(E)}) = B$ are nonseparable for every f." This requires a change in the order of the two quantifiers (the "for every" and the "there exists") in the statement of Radon's theorem. Fortunately, it turns out that a random partition just works: it effectively lets us exchange the said quantifiers. Moreover, a uniformly random partition of $\binom{V}{2}$ means G is an Erdős-Rényi random graph $G \sim \mathcal{G}(n, 1/2)$. This notion is captured in Theorem 3 which is somewhat more expensive than Radon's theorem. Specifically, the number of points in B needs to be at least $6d$ instead of just $d+2$. Additionally, the claim of Theorem 3 holds only with a high probability.

Proposition 5.1

Proof. Let $B \subset \mathbb{R}^d$. Let $\langle x, y\rangle = y^T x = \sum_i x_i y_i$ for every $x, y \in \mathbb{R}^d$.

Since the convex hulls of B_0 and B_1 do not intersect, the separating hyperplane theorem implies that there exists a hyperplane h such that

$$\langle b, h\rangle \geq 1 \text{ for all } b \in B_0$$
$$\langle c, h\rangle \leq 1 \text{ for all } c \in B_1$$

Let $\mathcal{L}$ be the above feasible linear system. We make the following claim.

Claim. *The feasibility polytope P of the above linear system does not contain an affine linear subspace of dimension* 1. ◇

If the claim is true, P will have a vertex h^* that meets d constraints, each a $d-1$ dimensional facet of P. This vertex corresponds to a separating hyperplane which satisfies d linear constraints of $\mathcal{L}$ with equality. Since each constraint is given by a point of B, h^* is supported by d points in B.

It remains to prove the claim. For the sake of contradiction, assume that P contains an affine subspace $H = \{h\}$ of dimension 1 defined by the equation $h = h_x + \lambda h_y$ for some $h_x, h_y \in P$ and all $\lambda \in \mathbb{R}$.

Suppose there exists a point $b \in B$ that is not orthogonal to the (separating) hyperplane h_y i.e., $\langle b, h_y\rangle \neq 0$. Such a point b will always exist because otherwise, all points of B would lie on the same line (normal to h_y) and d would be one, violating the condition that $d = \dim(\text{span}(B)) \geq 2$. Without loss of generality, assume that $b \in B_0$.

Since $h \in H \subseteq P$, it implies that for all $\lambda \in \mathbb{R}$,

$$\langle b, x + \lambda y\rangle \geq 1 \text{ for all } b \in B_0$$
$$\langle c, x + \lambda y\rangle \leq 1 \text{ for all } c \in B_1$$

Thus we can freely choose $\lambda_1, \lambda_2 \in \mathbb{R}$ and write $h_1 = h_x + \lambda_1 h_y, h_2 = h_x + \lambda_2 h_y$ such that $h_1, h_2 \in H \subseteq P$ and $\langle b, h_1\rangle \leq 1 \leq \langle b, h_2\rangle$. Intuitively speaking, we have translated a separating hyperplane h_1 to a new separating hyperplane h_2 along the direction h_y. However, there is now a point $b \in B$ which "satisfies" only one of the hyperplanes h_1, h_2 but not both. This is a contradiction, since both $h_1, h_2 \in H \subseteq P$ are two feasible solutions of $\mathcal{L}$. Therefore, the claim must be true. □

Proposition 5.2

Proof. Let H_B be the set of hyperplanes that pass through exactly d points of B. Suppose some hyperplane $h \in H_B$ separates B_0 from B_1 Fix h. We claim that the number of distinct binary labelings $c : B \to \{B_0, B_1\}$ that h can separate is $2 \cdot 2^d = 2^{d+1}$, as follows. We have two choices for the symmetry of B_0 and B_1 with respect to h: one gets the label B_0 and the other gets the label B_1. We also have 2^d choices for the classification of the d points which support the hyperplane. What is the number of distinct random partitions of the points of B that can be separated by *some* hyperplane (not just h)? By a union bound over all hyperplanes, this number is at most $2^{d+1}\,|H_B| = 2^{d+1}\binom{M}{d}$.

However, the total number of labelings $c : B \to \{B_0, B_1\}$ is 2^M since one can encode such a labaling using an M dimensional Boolean vector. Each labeling c induces a partition $B_0(c) \sqcup B_1(c)$ on B. Consider the partition induced by a *random* labeling $r \in \{0,1\}^M$. The probability that this partition is separated by *some* hyperplane $h \in H_B$ equals

$$\Pr_{r\sim\{0,1\}^M}[\text{some } h \in H_B \text{ separates the partition } B_0(r) \sqcup B_1(r)\,]$$
$$\leq \frac{2^{d+1}\binom{M}{d}}{2^M} \leq \frac{\left(\frac{Me}{d}\right)^d}{2^{M-d-1}} = \frac{(\alpha e)^d}{2^{d(\alpha-1)-1}}\,,$$

where $\alpha \triangleq M/d, \alpha > 1$. This quantity will be at most $1/d$ for all $d \geq 3$ if we set $\alpha \geq 6$.

Therefore, the probability that no d-supported hyperplane $h \in H_B$ separates the uniformly random partition $B_0 \sqcup B_1$ is at least $1 - 1/d$ when $d \geq 3$ and $M \geq 6d$.

□

Theorem 5 While Theorem 3 applies to random graphs, it is possible to modify Proposition 5.2 so that a similar statement applies to random graphs with $(n-1)$ edges.

Proof. One can make an argument similar to that in the proof of Theorem 4. The only place to change would be the proof of Proposition 5.2. Let n_t be the number of colorings (trees) that are separable (realizable) by some d-supported hyperplane. Although we do not know an exact estimate on n_t, it is certainly smaller than the number of all colorings separable by some hyperplane passing through d points. From the proof of Proposition 5.2, we know that this number is $\binom{M}{d}2^{d+1}$. Hence,

$$n_t \leq \binom{M}{d}2^{d+1} \leq \left(\frac{Me}{d}\right)^d 2^{d+1}$$
$$\leq \left(\frac{(n^2/2)e}{d}\right)^d 2^{d+1} = n^{2d}(e/2d)^d 2^{d+1}\,,$$

since $M = \binom{n}{2} \leq n^2/2$. By Cayley's formula, the number of labeled trees on n vertices is n^{n-2}. Thus, the probability p that a coloring, chosen uniformly at random from the colorings corresponding to random spanning trees, is

$$p = \frac{n_t}{n^{n-2}} \leq \frac{(e/2d)^d 2^{d+1}}{n^{n-2-2d}}.$$

p will be less than $1/n$ if

$$\begin{aligned}
&\frac{(e/2d)^d 2^{d+1}}{n^{n-2-2d}} < \frac{1}{n} \\
\text{or,}\quad & (e/2d)^d 2^{d+1} < n^{n-3-2d} \\
\text{or,}\quad & d\log(e/2d) + (d+1)\log 2 < (n-3-2d)\log n \\
\text{or,}\quad & d - d\log(2d) + (d+1)\log 2 < (n-3-2d)\log n \\
\text{or,}\quad & d + \log 2 + (2d+3-n)\log n < d\log(d).
\end{aligned}$$

By setting $d \leq n/2$, the left-hand side is at most $n/2 + \log 2 + 3\log n$, which is strictly smaller than the right-hand side $(n/2)\log(n/2)$ when $n \geq 17$.

Therefore, with probability $1 - 1/n$, a random spanning tree of the complete graph K_n is not realizable by any real weights $w \in \mathbb{R}^d$ and any map $f : V \to \{0,1\}^d$ when $d \leq n/2$ and $n \geq 17$. □

The extrinsic nature of the Hausdorff distance of optimal triangulations of manifolds

Gert Vegter * Mathijs Wintraecken †

Abstract

Fejes Tóth [5] and Schneider [9] studied approximations of smooth convex hypersurfaces in Euclidean space by piecewise flat triangular meshes with a given number of vertices on the hypersurface that are optimal with respect to Hausdorff distance. They proved that this Hausdorff distance decreases inversely proportional with $m^{2/(d-1)}$, where m is the number of vertices and d is the dimension of Euclidean space. Moreover the proportionality constant can be expressed in terms of the Gaussian curvature, an intrinsic quantity. In this short note, we prove the extrinsic nature of this constant for manifolds of sufficiently high codimension. We do so by constructing an family of isometric embeddings of the flat torus in Euclidean space.

1 Introduction

In [5] Fejes Tóth introduced inscribed triangulations approximating convex surfaces in $\mathbb{R}^3$ optimally and the 'Approximierbarkeit' (approximation parameter A_2). By a triangulation we shall mean a geometric realization of a simplicial complex in Euclidean space homeomorphic to the surface, that is piecewise linear in ambient space. From now on we take a simplicial complex to mean the geometric realization.

Optimal triangulations T_m with m vertices are triangulations which minimize the Hausdorff distance between the surface and the simplicial complex when this simplicial complex ranges over the space of triangulations with m vertices. We always assume that the vertices lie on the surface.

The Hausdorff distance between two subsets X and Y in a Euclidean space of arbitrary but fixed dimension d is defined as:

$$d_H(X,Y) = \max\{\sup_{x\in X}\inf_{y\in Y}|x-y|, \sup_{y\in Y}\inf_{x\in X}|x-y|\},$$

where $|x-y|$ denotes the standard Euclidean distance of x and y. The one-sided Hausdorff distance from X to $Y$is given by

$$d_H^o(X,Y) = \sup_{x\in X}\inf_{y\in Y}|x-y|.$$

The inverse of the asymptotic value of the product of the number of vertices and the Hausdorff distance is referred to as the *Approximierbarkeit* A_2.

Let K be the Gaussian curvature of the surface $\Sigma \subset \mathbb{R}^3$, and let dVol denote the volume (area) form on the surface. Fejes Tóth [5] gave the expression

$$\frac{1}{A_2} = \lim_{m\to\infty} d_H(\Sigma, T_m)m \geq \frac{1}{\sqrt{27}}\int_\Sigma \sqrt{K}\mathrm{dVol}, \qquad (1)$$

for the Approximierbarkeit for convex surfaces in three dimensional Euclidean space.

Schneider [9] generalized the discussion of Fejes Tóth to convex hypersurfaces (Σ) in Euclidean space of arbitrary dimension. The formula for A_{d-1}, derived by Schneider reads

$$\begin{aligned}\frac{1}{A_{d-1}} &= \lim_{m\to\infty} m^{2/(d-1)} d_H(\Sigma, T_m)\\ &= \frac{1}{2}\left(\frac{\theta_{d-1}}{\kappa_{d-1}}\int_\Sigma \sqrt{K(x)}\mathrm{dVol}\right)^{2/(d-1)}, \qquad (2)\end{aligned}$$

where $\kappa_d = \pi^{d/2}/\Gamma(1+d/2)$ is the volume of the d-dimensional unit ball, θ_d the covering density of the ball in d-dimensional space, dVol the volume form and K the Gaussian curvature. The covering density is defined as the infimum of the density over all coverings of, in this case, Euclidean space by the Euclidean unit ball, see for example [8]. The density of a cover $U = \{U_i\}$ of a compact measurable space with volume form dVol by a finite number of sets U_i is defined as follows: Let f_U be the integer valued function whose value $f_U(x)$ at a point x is the number of sets U_i such that $x \in U_i$. The density for the covering U is

$$\frac{\int f_U \,\mathrm{dVol}}{\int \mathrm{dVol}}.$$

Formula (2) is *intrinsic* in nature, because the Gaussian curvature is intrinsic. Generally, we call a quantity intrinsic if it depends only on the geometry of the surface or manifold itself. On the other hand a quantity is called extrinsic if depends on the embedding in the ambient space. If we for example consider a topological circle or loop in the plane, the length of the circle is intrinsic, while the curvature is not. This is because one can deform the loop without changing the distances

*Johann Bernoulli Institute for Mathematics and Computer Science, Rijksuniversiteit Groningen, g.vegter@rug.nl

†IST Austria, mathijs.wintraecken@ist.ac.at

between the points on the loop (as measured along the loop).

To make our statement concerning (2) more precise we note that the Gaussian curvature is invariant under isometry if $d-1$ is even and invariant up to sign if $d-1$ is odd, see [11, Chapter 7, Proposition 24]. It is clear that here the positive sign is the relevant one, because expression and thus the root $\sqrt{K}$ must be real.[1]

We will show that the intrinsic nature of the approximation parameter is particular to low co-dimension, by giving a sequence of isometric embeddings $E_k(\Sigma)$ of a surface Σ such that $\frac{1}{A_2}(E_k(\Sigma))$ tends to infinity with k. This means that there is no intrinsic quantity that can bound $\frac{1}{A_2}(E_k(\Sigma))$.

This makes heuristically some sense because the rigidity of a manifold disappears if the codimension of the embedding is sufficiently high, as was noted by Nash in [6]. In the setting of Nash, rigidity concerns metric preserving perturbations of the embedding. Nash proved that a compact n-manifold with a C^k Riemannian metric has a C^k isometric embedding in any small volume of Euclidean $(n/2)(3n+11)$-space, provided $3 < k \leq \infty$. So roughly speaking, one can squash a manifold in a small volume without affecting the intrinsic metric, but this would lead to wrinkles (and a build-up of extrinsic curvature).

This is the complete opposite of manifolds embedded in Euclidean space of codimension one, where manifold with non-zero curvature are embedded rigidly. Rigidity here means that an isometric embedding is unique up to Euclidean motions. Euclidean motions are generated by rotations and translations. We refer to Spivak [10, 12] for an overview of results on rigidity.

Our interest in the extrinsic nature was raised by the upper bounds on

$$\lim_{m\to\infty} d_H(M, T_m^o) m^{n/2},$$

where M is an n-dimensional manifold embedded in Euclidean space and T_m^o denotes an optimal triangulation of M. These bounds have been discussed in the Master's thesis of David de Laat [3]. Similar upper bounds were the topic of, among others[2], Chen, Sun and Xu[2]. These authors studied the L_p norm of the difference between a function and a linear approximation of this function. The bounds in [3] and [2] are defined in terms of the Hessian and thus extrinsic in nature. Our result below, gives us that the extrinsic nature of the bounds is unavoidable:

Theorem 1 *Let M be a Riemannian surface, then there is generally no function $f(g, \partial g, \ldots)$ which depends only on the metric and all its derivatives and a constant $\tilde{c}$ such that*

$$\lim_{m\to\infty} d_H(T_m, E(M)) m \leq \tilde{c} \int_M f \,\mathrm{d}\, Vol,$$

where $E(M)$ denotes the embedding of the manifold in Euclidean space and d*Vol the volume form.*

We prove the theorem by constructing an explicit example of a family of embeddings, which we'll describe in detail in the next section.

2 The construction of a sequence of embeddings

We consider a family of isometric embeddings $E : S^1 \times S^1 \to \mathbb{R}^n$ of the flat torus, whose members are discriminated by the index $k \in \mathbb{Z}_{\geq 1}$. We write

$$\lim_{m\to\infty} d_H(E_k, T_m) m = c_{E_k},$$

where E_k indicates a member of the family of isometric embeddings of $S^1 \times S^1$, T_m is an optimal triangulation with m vertices that lie on E_k. c_{E_k} is a real number depending on E_k. For the family of embeddings, we construct we have

$$\lim_{k\to\infty} c_{E_k} = \infty.$$

To simplify the calculations we focus[3] on embeddings in $\mathbb{R}^8$. We shall study the family of embeddings of the flat torus E_k parameterized by $k \in \mathbb{Z}_{\geq 1}$:

$$\begin{aligned}\mathcal{E}_k(\theta,\varphi) = (\,&\cos(\theta), \sin(\theta), \cos(k\theta)/k, \sin(k\theta)/k,\\ &\cos(\varphi), \sin(\varphi), \cos(k\varphi)/k, \sin(k\varphi)/k)\\ E_k = \{&E_k(\theta,\varphi) \mid \theta \in [0, 2\pi], \varphi \in [0, 2\pi]\}.\end{aligned}$$

Note that $\psi \mapsto (\cos(\psi), \sin(\psi), \cos(k\psi)/k, \sin(k\psi)/k)$, is an embedding of the circle in $\mathbb{R}^4$. This makes E_k an embedded flat torus.

Because E_k contains no straight line segments, we see that

Lemma 2 *For each (fixed) k the edge length of each edge in a triangulation T_m tends to zero as $d_H(T_m, E_k)$ tends to zero.*

Proof. Suppose that there is a subsequence $T_{m(l)}$ for which the length of edges $e_{m(l)} \in T_{m(l)}$ does not tend to zero. Without loss of generality we can assume (by chosing a convergent subsequence) that $e_{m(l)}$ converges to a limit line element, whose length by assumption is not zero. Because we assume that the Hausdorff distance $d_H(T_m(l), E_k)$ tends to zero, this line element lies within E_k, which contradicts the fact that E_k contains no straight lines. □

[1] We know that for a large class of negatively curved surfaces in $\mathbb{R}^3$ that $\frac{1}{A_2}$ is proportional to $\int \sqrt{|K|}\mathrm{d}A$, see [1, 7, 13].

[2] The introduction of [2] offers an extensive literature overview.

[3] It should be possible to prove the result for embeddings in $\mathbb{R}^4$, where the flat torus is not rigid, but the calculations would be significantly more difficult.

Because of Lemma 2 we may locally approximate the surface. In particular the tangent plane of the surface in the neighbourhood of a triangle is asymptotically well defined, because the triangle becomes small. Secondly, we may employ the natural group action of $(\mathrm{SO}(2))^4$ on the ambient space $\mathbb{R}^8$ to shift a given point on the torus to the origin. This means that we can approximate the surface, parametrized by $\mathcal{E}_k$, locally by

$$(1-\theta^2/2, \theta, 1-\theta^2/(2k), \theta, 1-\varphi^2/2, \varphi, 1-\varphi^2/(2k), \varphi),$$

where (θ, φ) are near the origin, and thus through a translation by

$$\Sigma_k(\theta,\varphi) \simeq (-\theta^2/2, \theta, -\theta^2/(2k), \theta, -\varphi^2/2, \varphi, -\varphi^2/(2k), \varphi).$$

Furthermore we may assume that the vertices of a triangle are $\Sigma_k(0,0) = 0 \in \mathbb{R}^8$, $\Sigma_k(\theta_1,\varphi_1)$, $\Sigma_k(\theta_2,\varphi_2)$.

We shall employ techniques similar to the ones employed by Fejes Tóth [5] to find a lower bound for $\frac{1}{A_2}$. To be precise we fix the one-sided Hausdorff distance for a family of triangles and search for the triangle in the family with the largest area. The area of surface divided by the area of the largest triangle in the family will give a bound on the number of triangles needed in a triangulation that attains the fixed Hausdorff distance.

Because the number of triangles ($\tilde{m}$), edges (e) and vertices (m) are related by the fact that every triangle has three edges and each edge is shared by two triangles as well as the formula for the Euler characteristic $\chi = \tilde{m} - e + m$ a bound on $\frac{1}{A_2}$ follows. Here we are only interested in (rough) lower bounds, so it suffices to fix some lower bound on the Hausdorff distance and then determine some upper bound on the area of the triangles in a triangulation satisfying this bound.

To be able to fix a bound on the Hausdorff distance we calculate the following:

Lemma 3 *The one-sided Hausdorff distance d_H^o of a triangle on the surface parametrized by $\Sigma_k(\theta,\varphi)$ with vertices $(0,0)$, (θ_1,φ_1) and (θ_2,φ_2) satisfies:*

$$d_H^o \geq \eta = \frac{1}{8}\sqrt{1+k^2}\max\left\{\sqrt{\theta_1^4+\varphi_1^4},\ \sqrt{\theta_2^4+\varphi_2^4}\right\} + \mathcal{O}(|(\theta,\varphi)|^2).$$

Proof. A point p on the triangle with vertices $(0,0)$, (θ_1,φ_1) and (θ_2,φ_2) will be given as $p = \Sigma_k(\theta_1,\varphi_1)\lambda_1 + \Sigma_k(\theta_2,\varphi_2)\lambda_2$, with $\lambda_1 \in [0,1]$ and $\lambda_2 \in [0, 1-\lambda_1]$. We now want to find the point on the surface $\Sigma_k(\theta_c,\varphi_c)$ which is closest to p. This point is determined my the following equations:

$$\partial_\theta |p - \Sigma_k(\theta,\varphi)|^2 = 0$$
$$\partial_\varphi |p - \Sigma_k(\theta,\varphi)|^2 = 0.$$

It is not difficult to verify that $\theta_c \simeq \theta_1\lambda_1 + \theta_2\lambda_2$ and $\varphi_c \simeq \varphi_1\lambda_1 + \varphi_2\lambda_2$, where $\simeq$ denotes equality up to linear order in θ_i, φ_i. This means that the distance between a point on the triangle and the surface is approximately given by

$$\begin{aligned}
&\|\Sigma_k(\theta_1,\varphi_1)\lambda_1 + \Sigma_k(\theta_2,\varphi_2)\lambda_2 \\
&\qquad - \Sigma_k(\theta_1\lambda_1+\theta_2\lambda_2, \varphi_1\lambda_1+\varphi_2\lambda_2)\| \\
&= \Bigg|\Bigg((\theta_1\lambda_1+\theta_2\lambda_2)^2/2 - \frac{\theta_1^2}{2}\lambda_1 - \frac{\theta_2^2}{2}\lambda_2, 0, \\
&\quad k(\theta_1\lambda_1+\theta_2\lambda_2)^2/2 - k\frac{\theta_1^2}{2}\lambda_1 - k\frac{\theta_2^2}{2}\lambda_2, 0, \\
&\quad (\varphi_1\lambda_1+\varphi_2\lambda_2)^2/2 - \frac{\varphi_1^2}{2}\lambda_1 - \frac{\varphi_2^2}{2}\lambda_2, 0, \\
&\quad k(\varphi_1\lambda_1+\varphi_2\lambda_2)^2/2 - k\frac{\varphi_1^2}{2}\lambda_1 - k\frac{\varphi_2^2}{2}\lambda_2, 0\Bigg)\Bigg|.
\end{aligned}$$

For the choice $\lambda_1 = 1/2$, $\lambda_2 = 0$ and $\lambda_1 = 0$, $\lambda_2 = 1/2$ this yields $\sqrt{1+k^2}\sqrt{\theta_1^4+\varphi_1^4}/8$ and $\sqrt{1+k^2}\sqrt{\theta_2^4+\varphi_2^4}/8$ respectively, so

$$d_H \geq \eta = \frac{1}{8}\sqrt{1+k^2}\max\left\{\sqrt{\theta_1^4+\varphi_1^4},\ \sqrt{\theta_2^4+\varphi_2^4}\right\} + \mathcal{O}(|(\theta,\varphi)|^2).$$

□

From Lemma 3 we can conclude that

$$\frac{8}{\sqrt{1+k^2}}\eta \geq \theta_1^2, \theta_2^2, \varphi_1^2, \varphi_2^2.$$

On the other hand the area of the triangle is approximately equal to

$$|\varphi_1\theta_2 - \theta_1\varphi_2|/2.$$

So the area of a triangle is bounded from above by

$$\frac{4}{\sqrt{1+k^2}}\eta. \tag{3}$$

Let us denote by π_{E_k} the closest point projection onto E_k. Although it seems intuitively clear that the projection π_{E_k} of T_m to E_k is surjective for sufficiently small Hausdorff distance, it is not so easy to prove. We will use that any point in E_k must be at most a distance $d_H(T_m, E_k)$ from a (projected) triangle $\pi_{E_k}(t)$ in $\pi_{E_k}(T_m)$. We furthermore recall that Theorem 4.8(8) of [4] gives us that the closest point projection on a set of positive reach $\mathcal{S}$ is a Lipschitz map with Lipschitz constant

$$\frac{\mathrm{rch}(\mathcal{S})}{\mathrm{rch}(\mathcal{S}) - \delta},$$

where rch denotes the reach and δ an upper bound on the distance of the points to the set, that is the distance between the medial axis and the set itself. As the Hausdorff distance tends to zero the Lipschitz constant

of the projection tends to 1. Using this we see that the area of the d_H-neighbourhood of a triangle $\pi_{E_k}(t)$ equals the area of the triangle t itself plus the length of the boundary times d_H plus higher order terms. Using the estimates in Lemma 3 we find that the area of the neighbourhood is $\mathrm{Area}(t) + \mathcal{O}(\mathrm{Area}(t)^{1/2} d_H)$ or equivalently $\mathrm{Area}(t) + \mathcal{O}(d_H^{3/2})$. So up to leading order the bound of (3) on the area holds even after projecting and with a safe margin.

As we already noted above, the number of triangles $\tilde{m}$ in a triangulation is bounded from below by

$$\mathrm{Area}(t_{\max})\tilde{m} \gtrsim \mathrm{Area}(E_k),$$

where $\gtrsim$ is used to suppress terms that are not of leading order in the Hausdorff distance, E_k denotes the embedding of the surface and $t_{\max}$ denotes the biggest triangle in the triangulation. These considerations give us

$$\begin{aligned} d_H m &\gtrsim \eta\, m \\ &\gtrsim \eta \frac{\mathrm{Area}(E_k)}{\mathrm{Area}(\triangle)} \\ &\gtrsim \eta \frac{(4\pi)^2}{4\eta/\sqrt{1+k^2}} \\ &= 4\pi^2\sqrt{1+k^2} \end{aligned}$$

This implies that

$$\lim_{m\to\infty} d_H(T_m, E_k) m \geq 4\pi^2\sqrt{1+k^2}.$$

The result is summarized in the main theorem

Theorem 1 *Let M be a Riemannian surface, then there is generally no function $f(g, \partial g, \ldots)$ which depends only on the metric and all its derivatives and a constant $\tilde{c}$ such that*

$$\lim_{m\to\infty} d_H(T_m, E(M)) m \leq \tilde{c} \int_M f \,\mathrm{d}\, Vol,$$

where $E(M)$ denotes the embedding of the manifold in Euclidean space and $\mathrm{d}\,Vol$ *the volume form.*

In this theorem we could have absorbed $\tilde{c}$ in $f(g)$. However, we have chosen this form to mimic the traditional form of the result of Fejes Tóth [5] and Schneider [9]. The generalization of the above theorem to manifolds of arbitrary dimension is straightforward, because one can take cross product of our example with any other manifold and find the same result.

3 Open question: rigidity

If the embedding of a manifold M is rigid then

$$\lim_{m\to\infty} m d_H(M, T_m)^{(n-1)/2},$$

where again T_m is optimal, must only depend on intrinsic quantities. This is because any two embeddings are the same up to Euclidean motions and thus the intrinsic geometry determines the geometry of the embedding completely.

What the converse statement should be is not so clear. For example the cylinder with boundaries $S^1 \times [a, b]$ is non-rigid, while the limit of $m d_H(M, T_m)$ is independent of embedding, albeit zero. It would be interesting to understand the exact relation between rigidity and the asymptotic behavior with respect to m of $d_H(M, T_m)$ better. Results in this direction could also provide a different perspective on combinatorial rigidity.

Acknowledgment

The authors thank David Cohen-Steiner, Ramsay Dyer, David de Laat and Rien van de Weijgaert for discussion. This work has been supported in part by the European Union's Seventh Framework Programme for Research of the European Commission, under FET-Open grant number 255827 (CGL Computational Geometry Learning) and ERC Grant Agreement number 339025 GUDHI (Algorithmic Foundations of Geometry Understanding in Higher Dimensions) and the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 754411.

References

[1] D. Atariah. *Parameterizations in the Configuration Space and Approximations of Related Surfaces.* PhD thesis, Freie Universität Berlin, 2014.

[2] L. Chen, P. Sun, and J. Xu. Optimal anisotropic meshes for minimizing interpolation errors in l_p-norm. *Mathematics of computation*, 76(257):179–204, 2007.

[3] D. de Laat. Upper bounds on the optimal meshing error of manifolds in higher codimension. Master's thesis, Rijksuniversiteit Groningen, 2011.

[4] H. Federer. Curvature measures. *Transactions of the America mathematical Society*, 93:418–491, 1959.

[5] L. Fejes Tóth. *Lagerungen in der Ebene, auf der Kugel und im Raum.* Berlin, Göttingen, Heidelberg: Springer, 1953.

[6] J. Nash. The imbedding problem for riemannian manifolds. *The Annals of Mathematics*, 63(1):pp. 20–63, 1956.

[7] H. Pottmann, R. Krasauskas, B. Hamann, K. Joy, and W. Seibold. On piecewise linear approximation of quadratic functions. *Journal for Geometry and Graphics*, 4(1):9–31, 2000.

[8] C. Rogers. *Packing and Covering.* Cambridge: University Press, 1964.

[9] R. Schneider. Zur optimalen approximation konvexer hyperflächen durch polyeder. *Mathematische Annalen*, 256:289–301, 1981.

[10] M. Spivak. *A comprehensive introduction to differential geometry: Volume III*. Publish or Perish, 1999.

[11] M. Spivak. *A comprehensive introduction to differential geometry: Volume IV*. Publish or Perish, 1999.

[12] M. Spivak. *A comprehensive introduction to differential geometry: Volume V*. Publish or Perish, 1999.

[13] M. Wintraecken. *Ambient and intrinsic triangulations and topological methods in cosmology*. PhD thesis, Rijksuniversiteit Groningen, 2015.

Author Index