Booklet of Abstracts: 32nd Annual Fall Workshop on Computational Geometry FWCG 25

The 32nd Annual Fall Workshop on Computational Geometry (FWCG 2025) was hosted by the Department of Computer Science at Queens College of the City University of New York, in Flushing, NYC, on November 7-8, 2025.

Program Committee Members

Brittany Fasy(Montana State University)
Christiane Schmidt (Linköpings University)
Greg Aloupis(Northeastern University)
Hugo Akitaya(UMass Lowell)
Hu Ding(University of Science and Technology of China)
Hubert Wagner (University of Florida)
Haitao Wang (University of Utah)
Jeff Philips(University of Utah)
Mayank Goswami(Chair, Queens College CUNY)
Sharath Raghvendra (North Carolina State University)

Organizing Committee

Mayank Goswami(Queens College CUNY)
Joseph S.B. Mitchell(Stony Brook University)

Local Organizers

Gaurish Telang, GiBeom Park, Xinying Chyn, Jordan Passarella, Xiuyi Huang (tech support), Paul Cesaretti, Xinglong Zhou.

An Efficient Algorithm for Computing Diverse c-Maximum Independent Sets in Planar Graphs*

Waldo Gálvez[1] Mayank Goswami[2] Arturo Merino[3]
GiBeom Park[4] Meng-Tsung Tsai[5] Victor Verdugo[6]

Abstract

Given a classical optimization problem Π with input size n and an integer $k \geq 1$, the goal is to generate a collection of k maximally diverse solutions to Π. For problems Π in P (such as spanning tree and minimum cut), there are efficient poly(n, k) approximation algorithms available for the diverse variants. In contrast, only FPT algorithms are known for NP-hard problems such as vertex covers and independent sets, but in the worst case, these algorithms run in time $\exp((kn)^c)$ for some $c > 0$. In this work, we address this gap and give $f(k)$poly(n) time approximation algorithms for diversification variants of maximum weight independent sets and minimum weight vertex covers in planar graphs.

1 Introduction

Computing a collection of diverse solutions to a given optimization problem has gained a lot of attention recently [1, 2, 5, 9, 11, 14, 16, 19, 20, 25, 26]. While classical algorithms are tailored to produce one solution, the task here is to output a collection of $k \geq 1$ solutions that are *maximally dispersed* in the solution space. In general, one is given a diversity measure on the space of k-tuples of solutions to a problem, and the goal is to output the set of k solutions that maximize this measure.

When solutions can be represented as a subset of the input, the metric for diversity measure is the size of the symmetric difference between two sets: $d(X, Y) = |X \Delta Y|$, where X and Y are two feasible solutions of a given optimization problem. This is extended to a k-tuple of solutions by considering either the average, or the minimum pairwise distance between all pairs of solutions. For example, if $\mathcal{F}$ is the family of all minimum spanning trees of a given graph G, then the task is to find k spanning trees whose average (or, minimum) of pairwise distances is maximized. The weighted setting is also of interest, e.g., $\mathcal{F}$ is the family of all *minimum* spanning trees of G. However, if G has a unique minimum spanning tree, then the problem of returning k diverse minimum spanning trees becomes uninteresting. The natural approach here is to enlarge the set of solutions by allowing approximations. We call such approximately optimal solutions ***nice***. In fact, if we replace the minimum spanning tree above by an NP-hard problem, say, maximum weight independent set (MWIS), and we want polynomial time algorithms, then *allowing approximations becomes necessary*,[7] as otherwise we cannot even solve the problem for $k = 1$.

Thus, we consider the setting where we have a quality function $\sigma : \mathcal{F} \to \mathbb{R}_{\geq 0}$ assigning a value to each feasible solution, and a niceness factor $c \in (0, 1)$. For maximization (or, minimization) problems, we say that a solution is ***c-optimal*** if its objective is at least $c \cdot \max_{S \in \mathcal{F}} \sigma(S)$ (or, at most $(1/c) \cdot \min_{S \in \mathcal{F}} \sigma(S)$). Now, we give the formal definition for ***diverse and nice optimization*** as follows.

Definition 1 (Diverse and Nice Optimization)**.** *The input is a four-tuple (I, k, σ, c), where I is a ground set with $n := |I|$, $k \geq 1$ is an integer, $\sigma : 2^I \to \mathbb{R}_{\geq 0}$ is a quality function, and $c \in (0, 1)$ is a niceness factor. Let $\mathcal{F}_c = \{S \subseteq I : S \text{ is } c\text{-optimal}\}$. The* diverse and nice optimization *problem asks to find a collection $\mathcal{S} = \{S_1, \ldots, S_k\} \subseteq \mathcal{F}_c$ of size k so as to maximize $\sum_{i<j} |S_i \Delta S_j|$.*

Remark. Note that, in order to exploit the quality-diversity tradeoff, c is input by the user, unlike in approximation algorithms where one would like c to be as close to 1 as possible.

Despite considerable research, there are *no polynomial time algorithms*, even with approximation guarantees, known for obtaining diverse solutions to *any NP-hard optimization problem*. All existing results either give polynomial (in n and k) time approximations for problems in P, or FPT algorithms

*A full version of this paper is available at https://arxiv.org/abs/2501.12261

[1]Universidad de Concepción, Chile,
[2]Queens College, CUNY, USA
[3]Universidad de O'Higgins, Chile
[4]CUNY Graduate Center, USA
[5]Academia Sinica, Taiwan
[6]Pontificia Universidad Católica de Chile

[7]In fact, for any constant $\varepsilon > 0$ approximating maximum independent set to within a factor of $n^{1-\varepsilon}$ for n-node graphs remains NP-hard [29]. This is one of the reasons why in this paper we focus on planar graphs.

(that are exponential in n the worst case) for NP-hard problems.

A natural class of NP-hard problems arise from packing and covering. In this paper, we give the first $f(k)\text{poly}(n)$ time approximation algorithms for MWIS and minimum weight vertex covers (MWVC) in planar graphs.

Definition 2 (DMWIS-PG and DMWVC-PG)**.** *Let $G = (V, E)$ be a vertex-weighted planar graph, let $k \geq 1$ be an integer, and let $\mathcal{F}_c = \{S \subseteq V : S \text{ is independent and } c\text{-optimal}\}$. The* Diverse c-Maximum Weight Independent Sets *(DMWIS-PG) problem asks to find a collection $\mathcal{S} = \{S_1, \ldots, S_k\} \subseteq \mathcal{F}_c$ of size k (distinct whenever $|\mathcal{F}_c| \geq k$, otherwise as a multiset) so as to maximize $\sum_{i<j} |S_i \Delta S_j|$. The* Diverse c-Minimum Weight Vertex Covers *(DMWVC-PG) is defined analogously.*

Related Work. Finding diverse MVCs has received considerable attention with several FPT results. This line of work focuses on $c = 1$, i.e., algorithms that return *optimal* solutions and maximize the diversity *exactly*. While it is clear that such algorithms cannot be polynomial in n and k for NP-complete problems like MWISs in (planar) graphs [17], even problems such as finding a diverse pair of maximum matchings is hard [28]. Thus the work in this area focuses on fixed-parameter-tractable algorithms that avoid an exponential dependence on the input size n; see, e.g., [2, 4, 5, 10–13, 23, 25, 27].

The works on the Diverse Vertex Cover problem are the most relevant to us. The algorithm in [5] runs in time $2^{k\psi} n^{O(1)}$ where ψ denotes the *size of each solution* (e.g., the size of a minimum vertex cover or a maximum independent set), and the algorithm in [4] runs in time $2^{\omega k} \psi^{O(k)} n^{O(1)}$, where ω represents the *treewidth* of the input graph. While the $2^{k\psi} n^{O(1)}$ or $2^{\omega k} \psi^{O(k)} n^{O(1)}$ result is impressive and important in the FPT context, in our setting there is a limitation: planar graphs can have large treewidth and large independent sets or vertex covers, i.e., ψ or ω could be $n^{\Omega(1)}$. Even if $k = 2$, this translates to a runtime of $2^{n^{\Omega(1)}}$, which could be prohibitive[8] for many applications.

Two related frameworks for diverse solutions to problems in P were presented in [16, 18]. The first framework [18] only allows to compute diverse solutions in the space of *optimal* solutions (i.e., c cannot be input by the user) but guarantees distinct solutions. The second work [16] allows the user to specify a value of c, but may return a multiset of k solutions instead of a set, i.e., may repeat solutions.[9] Note that not only do we want the best of both worlds (c as input, and a set of k distinct solutions as output), but, more importantly, we also want our framework to apply to NP-hard problems.

[8] [22] shows that the size of a minimum dominating set in a sensor network deployed on a 600m X 600m square goes from 15 to 35 as the number of nodes increases from 100 to 1000 (Figure 5). Assuming a runtime of $2^{k\psi}$ where ψ is the size of a dominating set, the computational task for generating $k = 4$ solutions when each dominating set has a size of 15 will take at least 5 years on a 5GHz computer.

2 Our Results

We begin with defining the notions of approximation and resource augmentation.

Definition 3. *An algorithm is a* **β-approximation with α-resource augmentation for c-optimal solutions** *(abbreviated as* **β-APX. with α-RA.***) for the diverse and nice optimization problem if for every integer $k \geq 1$ it computes k many (αc)-optimal solutions $S_1, \ldots, S_k$ such that $\sum_{i<j} |S_i \Delta S_j| \geq \beta \sum_{i<j} |S'_i \Delta S'_j|$ for any c-optimal solutions $S'_1, \ldots, S'_k$.*

We remark that whenever one of α and β is equal to one, we omit the qualifier from the statement.

Theorem 4. [DMWIS-PG and DMWVC-PG] *For DMWIS-PG, there exists a $2^{O(k\delta^{-1}\varepsilon^{-1})} n^{O(\varepsilon^{-1})}$-time $(1-\varepsilon)$-APX. algorithm with $(1-\delta)$-RA. When $k = O(\log n)$, this is a PTAS (Polynomial Time Approximation Scheme). The same statement holds for the DMWVC-PG.*

Remark. 1. The running time does not depend on c. It will turn out that the factor c will be dominated by $n^{1/\varepsilon}$. 2. The above result is the first example of an approximation algorithm for the diverse solutions version of *any strongly* NP*-complete problem* that is *fixed parameter tractable using only k as a parameter*. As mentioned, the dependence on k allows us to obtain a PTAS up to $k = O(\log n)$. This was not possible with existing work even for $k = 2$ due to the exponential dependence on other parameters such as the treewidth or the size of an MWIS, as the focus was on exact algorithms (for both diversity and quality). The tradeoff is that we lose the small factors of ε in diversity and δ in the quality.

Other Applications. Our framework extends to other problems, including, Diverse Knapsack, Diverse Rectangle Packings, and Diverse TSP problem. See the full version [15] for more details.

3 Our Algorithm

For brevity, we henceforth write $A - a + b := (A - \{a\}) \cup \{b\}$ for any set A.

One of main goals of our algorithm is to efficiently implement the local search algorithm by Cevallos et

[9] A recent work [9] on finding diverse minimum s-t cuts also guarantees a multiset of k diverse cuts.

al. [7]. This local search algorithm starts with an arbitrary set $X = \{x_1, \ldots, x_k\}$ of k elements in the search space $\mathcal{X}$, and then finds a pair of elements $x_{\text{in}} \in X$ and $x_{\text{out}} \in \mathcal{X} - X$ such that swapping these two maximizes the diversity in X, i.e., $(x_{\text{in}}, x_{\text{out}}) = \text{argmax}_{(x',x'')} \sum_{x_i,x_j \in (X-x'+x'')} |x_i \Delta x_j|$, where $x' \in X$ and $x'' \in \mathcal{X} - X$. This local search algorithm is guaranteed to return a solution with diversity at least $\max\{\frac{1}{2}, \frac{k-1}{k+1}\}$ of optimal within $O(k \log k)$ swaps. Note, however, that this algorithm might run in time $\exp(n)$ if the search space is given implicitly (e.g., spanning tree).

Hanaka et al. [18] showed a strategy to overcome the aforementioned issue. Their approach is the following. Given $\mathcal{S} = \{S_1, \ldots, S_k\}$, define an objective function $r(\cdot)$ as $r(e) = \sum_{i \in [k]} (\mathbb{1}(e \notin S_i) - \mathbb{1}(e \in S_i))$ and $r(S) = \sum_{e \in S} r(e)$. If $(S_{\text{in}}, S_{\text{out}})$ is the best swapping pair, then S_{out} is the ***farthest*** point from $\mathcal{S} - S_{\text{in}}$, i.e.,

$$S_{\text{out}} = \underset{S \in \mathcal{F} - \mathcal{S}}{\text{argmax}} \sum_{S' \in \mathcal{S} - S_{\text{in}}} |S_i \Delta S_j|, \qquad (1)$$

where $\mathcal{F}$ denotes the collection of all feasible solutions. In other words, the diverse optimization problem can be approximated via maximization problem. Therefore, with correctly designed farthest insertion algorithm, by making at most k guesses for S_{in} we may compute the best swapping pair $(S_{\text{in}}, S_{\text{out}})$. Hence, by executing farthest insertion $O(k^2 \log k)$ times, we may compute a collection of solutions with diversity at least $\max\{\frac{1}{2}, \frac{k-1}{k+1}\}$ of optimal. Also, note that solving maximization problem w.r.t $r(\cdot)$ above might return a solution S that is already in $\mathcal{S}$. To circumvent this, they used the k-best enumeration w.r.t. $r(\cdot)$, where $S'_1, S'_2, \ldots, S'_k$ are said to be the k-best enumeration if they are all distinct and $r(S'_1) \geq r(S'_2) \geq \ldots \geq r(S'_k) \geq r(S')$ for any $S' \in \mathcal{S} - \{S_1, \ldots, S_k\}$. Once a guessed solution S_{in} is kicked out, there are only $k-1$ remaining solutions, thus a distinct farthest solution from the remaining ones is guaranteed. Hence, $\max\{\frac{1}{2}, \frac{k-1}{k+1}\}$ factor for diversity may be achieved by running k-best enumeration $O(k^2 \log k)$ times.

However, how do we run the k-best enumeration w.r.t. $r(\cdot)$ in the space of c-optimal solutions $\mathcal{F}_c$? Moreover, if the underlying optimization problem is already NP-complete, is it possible to design an efficient algorithm for this task?

Baker's Technique. Finding one maximum independent set (MIS) in planar graphs is NP-complete [17], and an influential work by Baker [3] provided a PTAS for this problem.

A planar graph $G = (V, E)$ can be embedded in the plane and the ***layers*** of vertices can be computed in linear time [21, 24]. A vertex is said to in the 1st layer if it is on the exterior face. In general, a vertex on the exterior face after the first $i-1$ layers have been removed is in the ith layer.

Given an approximation factor $(1-\gamma)$, let $\ell = (1/\gamma) - 1$, and let $p \in \{0, \ldots, \ell\}$. The ***$p$-th strata of G***, denoted L^p, is the set of all vertices in G that are at layers congruent to p modulo $\ell + 1$, i.e., the collection of every $(\ell+1)$-st layer from the p-th layer; see Figure 1 for an illustration. A planar

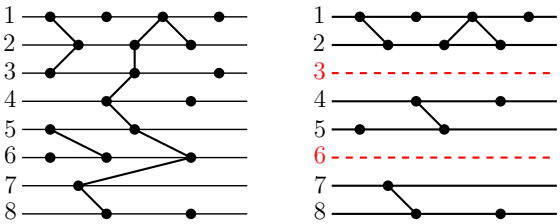


Figure 1: An Illustration of decomposition of G. Here, G consists of 8 layers and $\ell = 2$. The left one denotes a part of G, and the right one is a collection of ℓ-outerplanar graphs constructed by removing the 3rd strata from G.

graph is said to be ℓ-outerplanar if it has at most ℓ layers. Baker's technique proceeds in two steps. First, Baker shows that there exists a MIS S' and a $p \in \{0, \ldots, \ell\}$ such that removing the pth strata from S' results in an independent set S such that $|S| \geq (1-\gamma)|S'|$. That is, ignoring the vertices in the strata does not decrease the size of a MIS by a factor more than $(1-\gamma)$. Second, removal of such a pth strata now decomposes G into a collection of several ℓ-outerplanar graphs $G_1, \cdots, G_s$, for $\ell = (1/\gamma) - 1$. Baker then provides a dynamic programming based algorithm for independent sets in ℓ-outerplanar graphs running in time $O(2^{3\ell} n)$. The final solution is the union of all the MISs for the ℓ-outerplanar graphs $G_1, \cdots, G_s$, which by above is at least $(1-\gamma)$ of optimal.

Using our notation with $c = 1 - \gamma$, if we let $\mathcal{A}_{(1-\gamma)}$ denote Baker's algorithm above, then the solutions output by $\mathcal{A}_{(1-\gamma)}$ belong in a smaller class $\mathcal{F}'_{1-\gamma} \subseteq \mathcal{F}_{(1-\gamma)}$ defined as $\mathcal{F}'_{1-\gamma} := \{S \subseteq V : \exists\, p \in \{0, \ldots, \ell\}$ s.t. $S \cap L^p = \emptyset$, and S is a MIS in $G[V - L^p]\}$, where $G[A]$ denotes the graph induced by A for some subset $A \subseteq V$. In other words, the space of solutions is restricted in that the independent sets returned by Baker's algorithms will be missing vertices from a certain strata. By ignoring a strata we lose some c-optimal solutions from $\mathcal{F}_c$. Thus any generalization of Baker's algorithm is likely to be insufficient for solving DMWIS-PG over $\mathcal{F}_c$.

Note that one could consider the following obvious approach: Given $G_1, \ldots, G_s$ as in Baker's analysis, find (approximately) diverse maximum independent sets $\{S_i^j\}_{j=1}^k$ for every $1 \leq i \leq r$ and then combine the k sets from every G_i to obtain k diverse independent sets in G. This is not what our algorithm does, for two reasons. First, it may that be in an optimally diverse collection of MIS, the missing strata in Baker's analysis contributes more than an ε fraction

to the total diversity. This would therefore be lost in the above algorithm. Second, it is not clear that an optimally diverse collection of k MISs in G, when restricted to a particular G_i, give k MISs in G_i; i.e., the distribution of the optimal collection may be very non-uniform across the G_is.

To overcome this issue, we boost Baker's analysis to show that there exists strata, called ***marginal strata***, such that removing it does not decrease the size of *any* of the maximally-diverse c-optimal solutions by too much, *and* the removed vertices do not decrease the diversity of the c-optimal solutions by too much.

Lemma 5. [Existence of Marginal Strata] *Given a planar graph $G = (V, E)$, $c, \delta, \varepsilon \in (0, 1)$ and an integer $k \geq 1$, let $\ell \geq 2k\delta^{-1} + 2\varepsilon^{-1} - 1$. Then, for any c-maximum independent sets $S_1, \ldots, S_k$ of V, there exists some $p \in [0, \ell]$ such that the following conditions simultaneously hold: (i) $|S_h \cap L^p| \leq (\delta/2)|S_h|$ for all $h \in [k]$, and (ii) $\sum_{i \neq j} |(S_i \cap L^p) \Delta (S_j \cap L^p)| \leq \frac{\varepsilon}{2} \sum_{i \neq j} |S_i \Delta S_j|$.*

Since every ℓ-outerplanar graph has a treewidth of at most $3\ell - 1$ [6], using the algorithms in Lemma 7.4 and Theorem 7.18 of [8], any ℓ-outerplanar graph can be transformed in time $2^{O(\ell)}n^2$ into a tree decomposition with a treewidth of $O(\ell)$ and $O(\ell n)$ nodes such that each node has at most two children[10]. *Therefore, we assume hereafter that every ℓ-outerplanar graph is given by its tree decomposition with treewidth $O(\ell)$ and $O(\ell n)$ nodes, where each node has at most two children.*

We now formally define ***budget-constrained k-best enumeration***.

Definition 6 (Budget-Constrained k-Best Enumeration)**.** *Let (I, k, σ, c) be an input to a diverse and nice optimization problem, and let $r : 2^I \to \mathbb{R}$ be an objective function. The* Budget-Constrained k-Best Enumeration *problem (abbreviated k-*BCBE*, or simply* BCBE*) asks to find k* distinct *subsets, if they exist, $S_1, \ldots, S_k \subseteq I$ such that a) S_i is c-optimal w.r.t. σ for all $i \in [k]$ and b) $r(S_1) \geq r(S_2) \geq \cdots \geq r(S_k) \geq r(S)$ for every $S \subseteq I$ that is c-optimal. If such subsets do not exist, the answer should be* **no**.

Theorem 7 (k-BCBE in ℓ-outerplanar graphs)**.** *Let G be an ℓ-outerplanar graph. Consider the budget-constrained k-best c-maximum independent sets problem in G with objective function $r : 2^V \to \mathbb{Z}$, where $r(S) \in [-R, R]$ for all $S \subseteq V$, and weight function w. Then the k-BCBE problem can be solved in time $2^{O(\ell)}k^2R^2n$.*

Given any collection of k independent sets $\mathcal{S} = \{S_1, \ldots, S_k\}$, define an objective function $r(\cdot)$ as earlier, i.e., $r(e) = \sum_{i \in [k]} (\mathbb{1}(e \notin S_i) - \mathbb{1}(e \in S_i))$ and $r(S) = \sum_{e \in S} r(e)$. Since each vertex can appear at most once in each of S_i, it follows $r(S) \in [-nk, nk]$. Recall that $\max\{\frac{1}{2}, \frac{k-1}{k+1}\}$ factor for diversity is achieved by running k-BCBE $O(k^2 \log k)$ times. Therefore, we have the following corollary.

Corollary 8 ($\max\{\frac{1}{2}, \frac{k-1}{k+1}\}$-apx. for c-Maximum Independent Sets)**.** *Let G be an ℓ-outerplanar graph. Then, there is an $2^{O(\ell)}k^4n^3$-time algorithm that generates k c-maximum independent sets with diversity at least $\max\{\frac{1}{2}, \frac{k-1}{k+1}\}$ of optimal.*

We are now in position to prove Theorem 4. We prove here the unweighted version. For all other missing details, see the full version.

Algorithm. Although we do not know in advance which $p \in \{0, \ldots, \ell\}$ yields the marginal strata, we may check every $p \in \{0, \ldots, \ell\}$ without affecting the overall time bound. Hence, without loss of generality, we assume that L^p denotes the marginal strata guaranteed by Lemma 5.

First, we delete the marginal strata in Lemma 5 to decompose G into $G_1, \ldots, G_q$, where $G[V - L^p] = G_1 \cup \cdots \cup G_q$ and each G_i is ℓ outerplanar for the value of ℓ in Lemma 5. Next, we create a tree decomposition T_i of G_i for every $1 \leq i \leq q$, and connect all the roots of these q trees to a new common root. This creates a tree whose root has q subtrees. Note that this resulting tree has width $O(\ell)$ and $O(\ell n)$ nodes. We then transform this tree into a tree with the same width and the number of total nodes, where each none leaf node has at most two children.

When $k < \frac{4}{\varepsilon}$, first run the Baker's algorithm to find a $(1 - \frac{\delta}{2})$-optimal independent set $\tilde{S}$. Then, using the existing result by Baste et al. [4] that runs in time $2^{O(\ell k)}n^k$, where ℓ is the treewidth and in our case equals $2k\delta^{-1} + 2\varepsilon^{-1}$ from Lemma 5, find maximally diverse independent sets each of which has size at least $(1 - \frac{\delta}{2})c|\tilde{S}|$. Since the optimal diversity may be at most nk^2, we may find our guaranteed solutions by running the algorithm of [4] at most $O(nk^2)$ times. Thus, this process ends in time $2^{O(\delta^{-1}\varepsilon^{-1})}n^{O(\varepsilon^{-1})}$. Note also that the size of each of obtained solutions will be at least $c(1 - \delta)$ of the size of a maximum independent set of G.

When $k \geq \frac{4}{\varepsilon}$, we apply the algorithm of Corollary 8 to $G[V - L^p]$. By Lemma 5 and Corollary 8, then the diversity of the generated solutions will be at least $\left(\frac{k-1}{k+1}\right)(1 - \frac{\varepsilon}{2})$ of optimal diversity of c-maximum independent sets of G. Since $k \geq \frac{4}{\varepsilon}$, $\frac{k-1}{k+1}$ is no less than $(1 - \frac{\varepsilon}{2})$, and this give us the desired approximation factor. Also, $\ell = 2k\delta^{-1} + 2\varepsilon^{-1} + 1$, the running time of this algorithm is $2^{O(k\delta^{-1} + \varepsilon^{-1})}n^3$. This together with the running time for the case $k \leq \frac{4}{\varepsilon}$ gives the overall running time $2^{O(k\delta^{-1}\varepsilon^{-1})}n^{O(\varepsilon^{-1})}$.

[10] See the full version or [8] for the precise definition for a tree decomposition of a graph.

References

[1] Amir Abboud, Vincent Cohen-Addad, Euiwoong Lee, and Pasin Manurangsi. Improved approximation algorithms and lower bounds for search-diversification problems. In *49th International Colloquium on Automata, Languages, and Programming (ICALP 2022)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2022.

[2] Per Austrin, Ioana O Bercea, Mayank Goswami, Nutan Limaye, and Adarsh Srinivasan. Algorithms for the diverse-k-SAT problem: the geometry of satisfying assignments. In *ICALP*, 2025.

[3] Brenda S Baker. Approximation algorithms for NP-complete problems on planar graphs. *Journal of the ACM (JACM)*, 41(1):153–180, 1994.

[4] Julien Baste, Michael R Fellows, Lars Jaffke, Tomáš Masařík, Mateus de Oliveira Oliveira, Geevarghese Philip, and Frances A Rosamond. Diversity of solutions: An exploration through the lens of fixed-parameter tractability theory. *Artificial Intelligence*, 303:103644, 2022.

[5] Julien Baste, Lars Jaffke, Tomáš Masařík, Geevarghese Philip, and Günter Rote. FPT algorithms for diverse collections of hitting sets. *Algorithms*, 12(12):254, 2019.

[6] Hans L. Bodlaender. Planar graphs with bounded treewidth. Technical Report RUU-CS-88-14, Department of Computer Science, Utrecht University, the Netherlands, 1988.

[7] Alfonso Cevallos, Friedrich Eisenbrand, and Rico Zenklusen. An improved analysis of local search for MAX-SUM diversification. *Mathematics of Operations Research*, 44(4):1494–1509, 2019.

[8] Marek Cygan, Fedor V Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized algorithms*. Springer, 2015.

[9] Mark de Berg, Andrés López Martínez, and Frits Spieksma. Finding diverse minimum s-t cuts. In *34th International Symposium on Algorithms and Computation*, 2023.

[10] Eduard Eiben, Tomohiro Koana, and Magnus Wahlström. Determinantal sieving. In *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 377–423. SIAM, 2024.

[11] Fedor V Fomin, Petr A Golovach, Lars Jaffke, Geevarghese Philip, and Danil Sagunov. Diverse pairs of matchings. In *31st International Symposium on Algorithms and Computation (ISAAC 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.

[12] Fedor V Fomin, Petr A Golovach, Fahad Panolan, Geevarghese Philip, and Saket Saurabh. Diverse collections in matroids and graphs. *Mathematical Programming*, pages 1–33, 2023.

[13] Ryo Funayama, Yasuaki Kobayashi, and Takeaki Uno. Parameterized complexity of finding dissimilar shortest paths. *arXiv preprint arXiv:2402.14376*, 2024.

[14] Waldo Gálvez, Mayank Goswami, Arturo Merino, GiBeom Park, and Meng-Tsung Tsai. Computing diverse and nice triangulations. In *International Symposium on Fundamentals of Computation Theory*, pages 180–193. Springer, 2025.

[15] Waldo Gálvez, Mayank Goswami, Arturo Merino, GiBeom Park, Meng-Tsung Tsai, and Victor Verdugo. A framework for the design of efficient diversification algorithms to np-hard problems. *arXiv preprint arXiv:2501.12261*, 2025.

[16] Jie Gao, Mayank Goswami, CS Karthik, Meng-Tsung Tsai, Shih-Yu Tsai, and Hao-Tsung Yang. Obtaining approximately optimal and diverse solutions via dispersion. In *Latin American Symposium on Theoretical Informatics*, pages 222–239. Springer, 2022.

[17] Michael R Garey and David S Johnson. *Computers and intractability*, volume 174. freeman San Francisco, 1979.

[18] Tesshu Hanaka, Masashi Kiyomi, Yasuaki Kobayashi, Yusuke Kobayashi, Kazuhiro Kurita, and Yota Otachi. A framework to design approximation algorithms for finding diverse solutions in combinatorial problems. In Brian Williams, Yiling Chen, and Jennifer Neville, editors, *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI*, pages 3968–3976. AAAI Press, 2023.

[19] Tesshu Hanaka, Yasuaki Kobayashi, Kazuhiro Kurita, See Woo Lee, and Yota Otachi. Computing diverse shortest paths efficiently: A theoretical and experimental study. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI*, pages 3758–3766. AAAI Press, 2022.

[20] Tesshu Hanaka, Yasuaki Kobayashi, Kazuhiro Kurita, and Yota Otachi. Finding diverse trees, paths, and more. In *Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI)*, pages 3778–3786. AAAI Press, 2021.

[21] John Hopcroft and Robert Tarjan. Efficient planarity testing. *Journal of the ACM (JACM)*, 21(4):549–568, 1974.

[22] Kamrul Islam, Selim G Akl, and Henk Meijer. Maximizing the lifetime of wireless sensor networks through domatic partition. In *2009 IEEE 34th Conference on Local Computer Networks*, pages 436–442. IEEE, 2009.

[23] Soh Kumabe. Max-distance sparsification for diversification and clustering. *arXiv preprint arXiv:2411.02845*, 2024.

[24] Richard J Lipton and Robert Endre Tarjan. A separator theorem for planar graphs. *SIAM Journal on Applied Mathematics*, 36(2):177–189, 1979.

[25] Neeldhara Misra, Harshil Mittal, and Ashutosh Rai. On the parameterized complexity of diverse sat. In *35th International Symposium on Algorithms and Computation*, 2024.

[26] Noah Schulhof, Pattara Sukprasert, Eytan Ruppin, Samir Khuller, and Alejandro A Schäffer. Finding multiple optimal solutions to an integer linear program by random perturbations of its objective function. *Algorithms*, 18(3):140, 2025.

[27] Yuto Shida, Giulia Punzi, Yasuaki Kobayashi, Takeaki Uno, and Hiroki Arimura. Finding diverse strings and longest common subsequences in a graph. In *35th Annual Symposium on Combinatorial Pattern Matching*, 2024.

[28] Jukka Suomela. Complexity of two perfect matchings with minimum shared edges? URL: https://cstheory.stackexchange.com/questions/1278/complexity-of-two-perfect-matchings-with-minimum-shared-edges.

[29] David Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. *Theory Comput.*, 3(1):103–128, 2007.

Visibility Optimization on Imprecise Terrains

Nathan Baker * Bradley McCoy * Binhai Zhu †

Abstract

The Earth's surface is often modeled using triangulated terrains. However, geographic and engineering measurements are inherently imprecise. An imprecise terrain is a triangulation in which each vertex has a fixed planar location but its elevation is only known to lie within a specified interval. In the 1.5D setting, an imprecise terrain is represented by an x-monotone polyline with fixed x-coordinates and interval constraints on the y-coordinates. In the 2.5D case, an imprecise terrain is a triangulated surface with fixed x and y-coordinates, and intervals constraining the z-coordinates. In this work, we study visibility problems on imprecise terrains with n vertices. We present an $O(n)$ time algorithm to compute the minimum length realization of a 1.5D terrain that is fully visible from a given imprecise interval. For imprecise 2.5D terrains with k viewpoints, we show that finding a realization that minimizes the maximum area seen by a viewpoint is strongly NP-hard. Finally, we give an $O(n \log n)$ time algorithm to construct a realization of a 1.5D terrain that maximizes the minimum height of a watchtower.

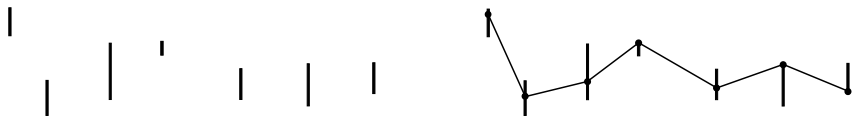
1 Introduction

Terrains are frequently studied in computational geometry, where applications include avalanche prediction [23], hydrology [7, 8], and defense [10]. Terrain data consists of measurements that will be uncertain due to noise, sampling sparsity, and environmental variability. Small variations in elevation can change visibility graphs, drainage basins, and optimal paths. In an imprecise terrain, the elevation of each vertex is not given by a number and is instead known to be in a vertical interval. In this model, we consider finding algorithms that guarantee correctness over all realizations. We consider finding relations with the best case (optimistic) and worst case (pessimistic) scenarios.

The investigation of imprecise terrains was initiated by Gray and Evans [12], where it is shown that computing an optimistic shortest path on a 2.5D imprecise terrain is NP-hard. Since then, there has been a growing interest in developing algorithms on imprecise terrains [8, 13, 14, 20].

Visibility problems on precise terrains are well studied [1, 2, 4, 5, 17, 19, 25–27, 29]. Applications of visibility on terrains include the construction of communication towers [24], urban planning [6], and the placement of observers [9]. Visibility on imprecise terrains has received less attention. The optimistic shortest watchtower problem was studied in 1.5 and 2.5D by McCoy and Zhu [21]. A variation of the watchtower problem on an imprecise terrain where the top of the

*James Madison University
†Montana State University



(a) (b)

Figure 1: (a) Imprecision segments that make up a 1.5D terrain. (b) A possible realization of the segments in (a).

watchtower is represented as a horizontal line segment was studied by McCoy, Zhu, and Dutt [22]. Another variation, where the position of the viewpoint is imprecise in the sense that it can be placed anywhere on a given edge, and the objective is to maximize the visible region on a 1.5D terrain, was studied by Keikha et al. [18].

2 Preliminaries

An *imprecise 1.5D terrain* is a 1.5D terrain with a closed y-interval at each x-coordinate rather than a fixed y-coordinate. We denote the n vertical intervals $\ell_1, \ell_2, \ldots, \ell_n$ (Figure 1a). Let t_i and b_i denote the top of and bottom of ℓ_i, respectively. A *realization* of an imprecise 1.5D terrain is a sequence of y-coordinates, specifying a point on each of the n intervals (Figure 1b). We refer to the points where a realization intersects the intervals as *vertices* and the segments between vertices as *edges*. A vertex is *left-turning* when the polyline, seen from left to right, turns to the left (upward), and *right-turning* if it turns to the right (downward), and *straight* if it does not turn. For vertex v let $x(v)$ and $y(v)$ denote the x and $y-$coordinates of v. Given an edge $e_i = (v_i, v_{i+1})$ of a terrain T, the *edge extension* of e_i, denoted by $E(e_i)$, is the line obtained by extending line segment e_i.

In the *imprecise 2.5D terrain model*, we are given n precise x, y-coordinates with an interval of z coordinates, and a triangulation defined when the vertices are projected onto the xy plane. See Figure 2 for an example. A realization of a 2.5D terrain is a choice of z-coordinate for each of the n intervals.

The shortest watchtower problem asks for the location and height of a vertical line segment $\overline{uv}$, where u is a point on a terrain T and each point $p \in T$ is visible to v.

3 Minimum length 1.5D visible realization

When guarding, there is often an operating cost for monitoring a portion of a terrain. For example, sensors or cameras may have a fixed range. This motivates the problem of finding realizations of minimum length or area. In this section, we consider an imprecise 1.5D terrain T together with an interval ℓ_v containing a viewpoint. We ask whether there exists a point $v \in \ell_v$ and a realization R

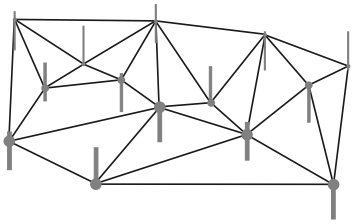


Figure 2: A possible realization of a 2.5D imprecise terrain.

such that v sees all of R. If so, we give an $O(n)$ time algorithm to compute a minimum-length realization entirely visible from a point on ℓ_v.

Theorem 1. *Given a 1.5D imprecise terrain with n intervals and an interval ℓ_v, the minimum-length realization entirely visible from a point on ℓ_v can be computed in $O(n)$ time.*

A complete proof is included in A.

4 2.5D Minimizing the Maximum Area

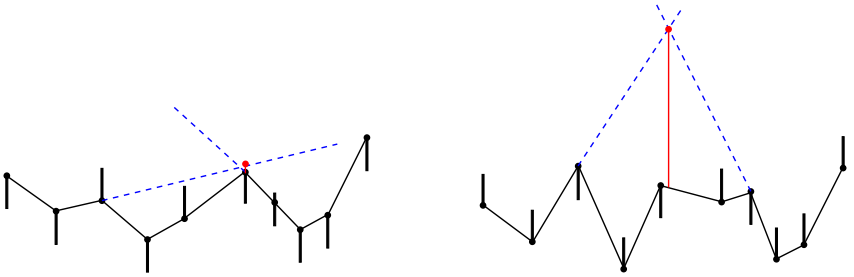
In this section, we consider the problem of guarding an imprecise 2.5D terrain with multiple viewpoints. We show that it is strongly NP-hard to compute a realization that minimizes the maximum area seen by any single viewpoint, subject to the condition that every point of the terrain is visible from at least one viewpoint.

Theorem 2. *Given multiple viewpoints on an imprecise 2.5D terrain, computing a realization that minimizes the maximum area visible to a viewpoint is strongly NP-hard.*

We reduce from the *Partition problem* with rational numbers, which is known to be strongly NP-hard [11,28], and, thus, no pseudo-polynomial algorithm can exist for solving this problem unless P=NP. In the Partition problem, we are given a multiset $S = \{x_1, x_2, \ldots, x_n\}$ of n positive rational numbers and must decide whether S can be partitioned into two disjoint subsets S_1 and S_2 such that

$$\sum_{x \in S_1} x = \sum_{x \in S_2} x \quad \text{and} \quad S_1 \cup S_2 = S.$$

Given an instance of the Partition problem, we construct an imprecise terrain with two viewpoints such that a solution to the Partition instance exists if and only if the maximum visibility area of the terrain is minimized. The proof is included in B.



(a) (b)

Figure 3: Realizations for an (a) optimistic and (b) pessimistic watchtower on same 1.5D imprecise terrain. We can see that the edge extensions that determine the minimum height of the tower shift with the terrain.

5 Pessimistic Watchtower

A linear-time algorithm has been given for the optimistic 1.5D imprecise watchtower problem [21]. In this section, we present an $O(n \log n)$ algorithm for the 1.5D *pessimistic* watchtower problem.

Theorem 3. *The 1.5D pessimistic watchtower problem can be computed in* $O(n \log n)$ *time.*

A complete proof is given in C. The bottleneck in the runtime is to compute the upper envelope of the $2n - 2$ lines; all other steps are linear. Whether a linear-time algorithm exists remains an interesting open problem.

6 Discussion

In this paper, we presented a linear-time algorithm for finding the minimum-length realization of a 1.5D terrain visible from a fixed interval. We proved that minimizing the maximum area visible from any single viewpoint on a 2.5D terrain with multiple viewpoints is strongly NP-hard. Finally, we gave an $O(n \log n)$ algorithm for the pessimistic shortest 1.5D watchtower problem.

Several related questions remain open. First, the optimistic and pessimistic imprecise 2.5D shortest watchtower problems are still unresolved. Second, in 1.5D, for a fixed viewpoint, optimizing the ratio of visible to invisible length offers an interesting variant of the problem studied here. Third, in the multiple-viewpoint setting of the 1.5D visibility problem, minimizing the maximum visible area remains an open challenge.

References

[1] Pankaj K. Agarwal, Sergey Bereg, Ovidiu Daescu, Haim Kaplan, Simeon Ntafos, Micha Sharir, and Binhai Zhu. Guarding a terrain by two watch-

towers. *Algorithmica*, 58(2):352–390, 2010.

[2] Boaz Ben-Moshe, Matthew J. Katz, and Joseph S. B. Mitchell. A constant-factor approximation algorithm for optimal 1.5D terrain guarding. *SIAM Journal on Computing*, 36(6):1631–1647, 2007.

[3] Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark Overmars. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, 3rd ed. edition, 2008.

[4] Richard Cole and Micha Sharir. Visibility problems for polyhedral terrains. *J. Symb. Comput.*, 7(1):11–30, 1989.

[5] Ovidiu Daescu, Stephan Friedrichs, Hemant Malik, Valentin Polishchuk, and Christiane Schmidt. Altitude terrain guarding and guarding uni-monotone polygons. *Computational Geometry*, 84:22–35, 2019.

[6] Leila De Floriani, Paola Magillo, and Enrico Puppo. Visibility on terrains: a survey. *Environment and Planning B: Planning and Design*, 21(3):329–344, 1994.

[7] Bruce A. DeVantier and Arlen D. Feldman. Review of gis applications in hydrologic modeling. *Journal of Water Resources Planning and Management*, 119(2):246–261, 1993.

[8] Anne Driemel, Herman Haverkort, Maarten Löffler, and Rodrigo I. Silveira. Flow computations on imprecise terrains. In *Algorithms and Data Structures (WADS)*, pages 350–361. Springer, 2011.

[9] W. Randolph Franklin. Siting observers on terrain. In *Advances in Spatial Data Handling: 10th International Symposium on Spatial Data Handling*, pages 109–120. Springer-Verlag, Berlin, Heidelberg, 2002.

[10] W. Randolph Franklin, Clark K. Ray, and Shashank Mehta. Geometric algorithms for siting of air defense missile batteries. Technical Report Contract DAAL03-86-D-0001, Delivery Order No. 2756, US Army Topographic Engineering Center, Battelle, Columbus Division, 1994.

[11] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, San Francisco, 1979.

[12] Chris Gray and William Evans. Optimistic shortest paths on uncertain terrains. In *Proceedings of the 16th Canadian Conference on Computational Geometry, (CCCG)*, pages 68–71, 2004.

[13] Chris Gray, Frank Kammer, Maarten Löffler, and Rodrigo I. Silveira. Removing local extrema from imprecise terrains. *Computational Geometry*, 45(7):334–349, 2012.

[14] Chris Gray, Maarten Löffler, and Rodrigo I. Silveira. Smoothing imprecise 1.5D terrains. *International Journal of Computational Geometry & Applications*, 20(04):381–414, 2010.

[15] John Hershberger. Finding the upper envelope of n line segments in O(n *log* n) time. *Information Processing Letters*, 33(4):169–174, 1989.

[16] Ferran Hurtado, Maarten Löffler, Inês Matos, Vera Sacristán, Maria Saumell, Rodrigo I. Silveira, and Frank Staals. Terrain Visibility with Multiple Viewpoints. In *Algorithms and Computation*, pages 317–327. Springer, 2013.

[17] Byeonguk Kang, Hwi Kim, and Hee-Kap Ahn. Guarding terrains with guards on a line. In *Combinatorial Algorithms (IWOCA 2025)*, pages 31–43. Springer Nature Switzerland, 2025.

[18] Vahideh Keikha, Maarten Löffler, Maria Saumell, and Pavel Valtr. Guarding a 1.5D Terrain with Imprecise Viewpoints. In *Combinatorial Algorithms (IWOCA 2025)*, pages 3–16. Springer Nature Switzerland, 2025.

[19] James King and Erik Krohn. Terrain guarding is NP-hard. *SIAM Journal on Computing*, 40(5):1316–1339, 2011.

[20] Anna Lubiw and Graeme Stroud. Computing realistic terrains from imprecise elevations. *Computing in Geometry and Topology*, 2(2):3:1–3:18, 2023.

[21] Bradley McCoy and Binhai Zhu. Optimistic imprecise shortest watchtower in 1.5D and 2.5D. *Submitted*, 2025.

[22] Bradley McCoy, Binhai Zhu, and Aakash Dutt. Guarding precise and imprecise polyhedral terrains with segments. In *Combinatorial Optimization and Applications (COCOA)*, pages 323–336. Springer Nature Switzerland, 2024.

[23] A. Miller, P. Sirguey, S. Morris, P. Bartelt, N. Cullen, T. Redpath, K. Thompson, and Y. Bühler. The impact of terrain model source and resolution on snow avalanche modeling. *Natural Hazards and Earth System Sciences*, 22(8):2673–2701, 2022.

[24] Md Mahbubur Rahman, Franklin Abodo, Leonardo Bobadilla, and Brian Rapp. Optimal placement and patrolling of autonomous vehicles in visibility-based robot networks, 2018. `arXiv:1710.07725`.

[25] Ritesh Seth, Anil Maheshwari, and Subhas C. Nandy. Acrophobic guard watchtower problem. *Computational Geometry*, 109:101918, February 2023.

[26] Micha Sharir. The shortest watchtower and related problems for polyhedral terrains. *Information Processing Letters*, 29(5):265–270, 1988.

[27] Nitesh Tripathi, Manjish Pal, Minati De, Gautam Das, and Subhas C. Nandy. Guarding polyhedral terrain by k-watchtowers. In *Frontiers in Algorithmics*, volume 10823, pages 112–125. Springer International Publishing, Cham, 2018.

[28] Dominik Wojtczak. On strong np-completeness of rational problems. In *Computer Science - Theory and Applications - 13th International Computer Science Symposium in Russia*, volume 10846 of *Lecture Notes in Computer Science*, pages 308–320. Springer, 2018.

[29] Binhai Zhu. Computing the shortest watchtower of a polyhedral terrain in $O(n \log n)$ time. *Computational Geometry*, 8(4):181–193, 1997.

A Appendix

In this appendix, we include omitted proofs.

Theorem 1. *Given a 1.5D imprecise terrain with n intervals and an interval ℓ_v, the minimum-length realization entirely visible from a point on ℓ_v can be computed in $O(n)$ time.*

To begin, we compute a shortest-path realization π and determine whether the point $\pi_v = \pi \cap \ell_v$ sees all of π in linear time [16, 20]. If π_v sees all of π, then π is the minimum-length realization visible from a point on ℓ_v. If π_v does not see all of π, we determine whether the minimum watchtower height of T at ℓ_v is zero [21]. If the minimum watchtower height is nonzero, no realization is entirely visible from a point on ℓ_v. Otherwise, if the minimum watchtower height at ℓ_v is zero, then such a realization exists.

We next establish a lower bound on the y-coordinate of a viewpoint that sees an entire realization. Let t_l denote the vertex of π at the top of an interval with minimum x-value, and let t_r denote the vertex of π at the top of an interval with maximum x-value less than the x-coordinate of ℓ_v. See Figure 4a for an example. Let π' be the subpath of π from t_l to t_r, together with its symmetric subpath to the right of the viewpoint. Define

$$v_m = \max\{\, y(E(e_i) \cap \ell_v) \mid e_i \in \pi' \,\},$$

where $E(e_i)$ is the supporting line of edge e_i, and let π_m be the edge whose extension, $E(\pi_e)$, determines to v_m.

Lemma 1. *For any optimal placement $v' \in \ell_v$, we have $y(v') \geq y(v_m)$.*

Proof. We consider the subpath of π' to the left of ℓ_v, the right side is symmetric. Since the leftmost point of π' is at the top of an interval, the edge π_m has positive slope (or is a subedge of a straight line of positive slope) and begins at the top of an interval and ends at the bottom of an interval. Any realization of this edge has slope at least that of π_m, so the intersection $E(\pi_m) \cap \ell_v$ has y-coordinate at least $y(v_m)$.

□

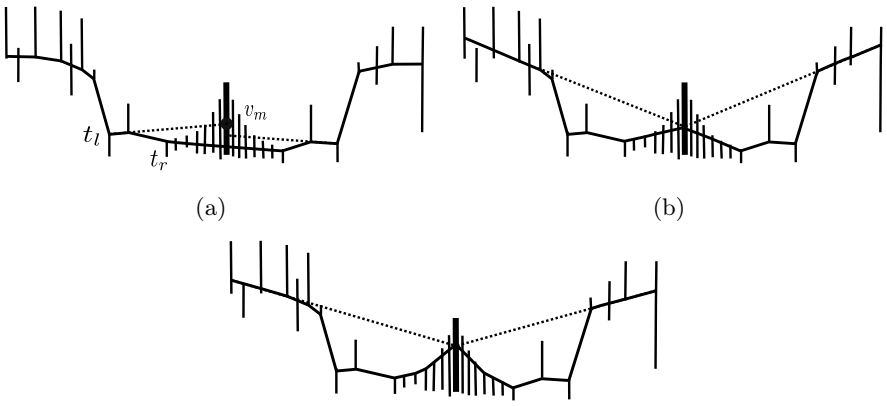


Figure 4: (a) The shortest path realization π in a terrain. The interval ℓ_v is in bold. The tops of the intervals between t_r and v_m are convex and the bottoms of the intervals between v_1 and v_l are concave. The entire terrain is not visible from π_v. (b) A point $v \in \ell_v$ from which the top of the second interval from the left is not visible. (c) A minimum length realization visible to ℓ_v.

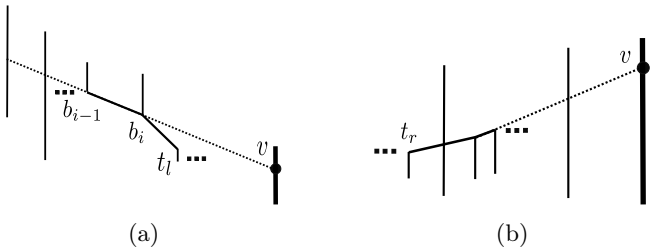
Let $\hat{\pi}$ be the union of the shortest paths from v_m to ℓ_1 and ℓ_n. If $v_1, v_n \in \hat{\pi}$ lie at the tops of their intervals, then by Lemma 1 we have found the shortest realization visible from v_m. Otherwise, π has no vertex at the top of an interval with $x < x(t_l)$ and must be monotonically decreasing and concave down from v_1 to t_l. An optimal solution balances raising the viewpoint and raising v_1, v_n until the entire realization becomes visible from ℓ_v.

Theorem 1. *Given a 1.5D imprecise terrain with n intervals and an interval ℓ_v, the minimum-length realization entirely visible from a point on ℓ_v can be computed in $O(n)$ time.*

Proof. Define a piecewise function $f : \ell_v \to \mathbb{R}$ such that for each $v \in \ell_v$, $f(v)$ is the length of the shortest realization entirely visible from v, if such a realization exists. By Lemma 1, we consider $y(v)$ increasing from v_m up to the top of ℓ_v.

The equation for the piecewise function f changes formulas at certain transition points. The first type of transition arises from intervals between $x(\ell_1)$ and $x(t_l)$. Let L be the concave decreasing portion of $\hat{\pi}$ between $x(\ell_1)$ and $x(t_l)$. If the supporting line of two bottoms of intervals in L intersects ℓ_v above v_m, then this intersection defines a transition point of $f(v)$. Thus, there is one transition for each edge of the upper convex hull of the bottoms of the intervals between ℓ_1 and t_l (see Figure 5a). Since L is decreasing and concave, these edges are the edges of $\hat{\pi}$ for x-values less than $x(t_l)$.

The second type of transition arises from intervals between $x(t_r)$ and $x(\ell_v)$. If the supporting line of two tops of intervals in this range intersects ℓ_v above



(a) (b)

Figure 5: (a) A transition of f in between ℓ_1 and t_l. (b) A transition of f in between t_r and ℓ_v.

v_m, then this intersection defines another transition point of $f(v)$. Hence there is one transition point for each edge of the lower convex hull of the tops of the intervals between t_r and ℓ_v (see Figure 5b). Since the intervals are already sorted by x-coordinate, the convex hull can be computed in $O(n)$ time [3]. Thus we obtain $O(n)$ transition points in total, which we denote by the set V.

For each $v \in V$, we test whether v can see all of L. If not, there exist vertices $b_i, v_{i-1} \in L$ such that v sees b_i but not b_{i-1}. We then check whether it is possible to raise the realization to the left of b_i so that it becomes visible from v. If any line $\overline{vb_i}$ passes below the top of an interval to the left of b_i, then no realization is visible from v, and we move to the next point in V (see Figure 4b). Otherwise, v sees all of L, and we compute the length of a straight line from $\ell_1 \cap \overline{vb_i}$ union the shortest path from b_i to v and store this value with v. When we consider the tops and bottoms of the intervals from right to left and all lengths are computed in $O(n)$. This is a minimum length realization visible to v because any shortest path from v_1 to v contains t_l and t_r.

Between transition points, $f(v)$ is convex because it is the sum of convex functions. Moreover, at each transition point, the slope of $f(v)$ increases, so $f(v)$ is strictly convex and has a unique minimum. By checking the transition points and their neighboring intervals, we can find the overall minimum in $O(n)$ time.

□

B Appendix

We next include a proof of the following.

Theorem 2. *Given multiple viewpoints on an imprecise 2.5D terrain, computing a realization that minimizes the maximum area visible to a viewpoint is strongly NP-hard.*

We reduce from the *Partition problem* with rational numbers, which is known to be strongly NP-hard [11,28]. In the Partition problem, we are given a multiset

$S = \{x_1, x_2, \ldots, x_n\}$ of n positive rational numbers and must decide whether S can be partitioned into two disjoint subsets S_1 and S_2 such that

$$\sum_{x \in S_1} x = \sum_{x \in S_2} x \quad \text{and} \quad S_1 \cup S_2 = S.$$

Construction. Given an instance of the Partition problem, we construct an imprecise terrain with two viewpoints such that a solution to the Partition instance exists if and only if the maximum visibility area of the terrain is minimized. Without loss of generality, assume x_1 is the largest element of S. We construct a square of area x_1 with two imprecise vertices on the x-axis and two precise vertices on the y-axis. Two viewpoints, v_l and v_r, are placed on the x-axis far to the left and right of the imprecise vertices, respectively.

We then add precise triangles above and below the quadrilateral on the y-axis. We choose the z-coordinate large enough to prevent its viewshed from interfering with those of other quadrilaterals. Next, we construct a quadrilateral of area x_2 above the square of area x_1, again using two precise vertices on the y-axis and two imprecise vertices off the axis. There is a unique y-coordinate ensuring the quadrilateral has area x_2 and is visible to both viewpoints. We continue this process, alternating above and below the base square. The quadrilateral corresponding to x_i is denoted q_i. An overview of the terrain is illustrated in Figure 6.

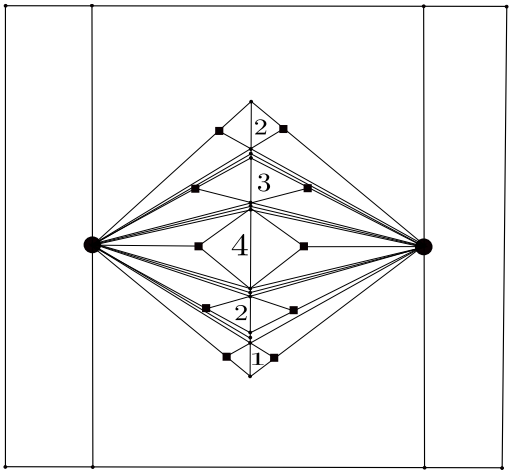


Figure 6: An overview of the imprecise terrain corresponding to the set $\{4, 3, 2, 2, 1\}$. The large circles are the viewpoints and the squares indicate imprecise points. The elements of S correspond to the area of the quadrilaterals.

The bottoms of the imprecise intervals are placed at rational height ϵ_b above the xy plane so that each viewpoint looks upward. See Figure 7 for a side view. The tops of the imprecise intervals are ϵ_t above the line formed by the

viewpoint and bottom of the imprecise interval on the opposite side of the y-axis. To enforce a partition structure, we add precise 'backstops' behind each viewpoint so that if neither imprecise point of q_i is at the top of an interval the area visible to each viewpoint is not minimized (Figure 6).

Lemma 2. *For each imprecise q_i if neither of the imprecise points are at the top of their intervals, then the viewpoints do not see the minimum possible area.*

Proof. If neither imprecise interval of q_i is maximized, the slope of the backstop can be chosen so that each viewpoint sees over the other and an arbitrarily large area is visible.

□

We next ensure that one of the imprecise points is near the bottom of its interval. For each q_i, consider the line formed by a viewpoint and the top of imprecise interval on the opposite side of the y-axis. Let ϵ_d denote the the z-coordinate where this line intersects the other imprecise interval (Figure 7).

Lemma 3. *For each q_i, if at least one imprecise vertex has z-coordinate greater than $\epsilon_b + \epsilon_d$, then the terrain is not fully visible.*

Proof. If the z-coordinate is greater than or equal to $\epsilon_b + \epsilon_d$, the terrain is concave downward at the imprecise point and q_i is not visible. □

When the imprecise vertices are adjusted, the areas of incident triangles change. Each imprecise vertex is incident to at most four triangles. By choosing ϵ_b and ϵ_t sufficiently small, the change in the area of any such triangle is bounded above by $\frac{x_{\min}}{8n+1}$, where $x_{\min}$ is the minimum positive rational number in S.

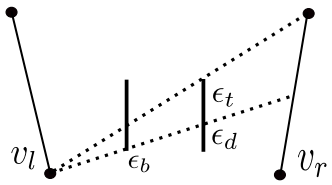


Figure 7: A side view of two imprecise intervals and their relation to the backstop.

Theorem 2. *Given multiple viewpoints on an imprecise 2.5D terrain, computing a realization that minimizes the maximum area visible to a viewpoint is strongly NP-hard.*

Proof. Given a Partition instance S, construct the imprecise terrain as described. Compute the realization that minimizes the maximum visibility area. For this terrain, let $A(v_\ell)$ and $A(v_r)$ denote the areas of the visibility maps of v_l and v_r, respectively, computed using a polynomial time algorithm [16].

If there exists a valid partition of S, then the difference $|A(v_\ell) - A(v_r)|$ arises only from small distortions due to imprecise vertices. Each q_i contributes

at most two imprecise vertices, each incident to at most four triangles, so the total number of affected triangles is at most $8n$. With our choice of ϵ_b, ϵ_t, the total error is bounded by $8n \cdot \frac{x_{\min}}{8n+1} < x_{\min}$. Hence,

$$|A(v_l) - A(v_r)| < x_{\min}.$$

If no valid partition exists, then by Lemmas 2 and 3, each quadrilateral has one imprecise vertex maximized and the other near its minimum. In this case, the difference in areas seen by the two viewpoints is at least the minimum imbalance among subset sums of S, which is strictly greater than $x_{\min}$. Thus,

$$|A(v_\ell) - A(v_r)| > x_{\min}.$$

Therefore, there is a partition if and only if $|A(v_l) - A(v_r)| < x_{\min}$. The terrain size is polynomial, visibility areas are computable in polynomial time, the coordinates of the points of the terrain are rational, and hence minimizing the maximum visibility area is strongly NP-hard.

□

C Appendix

Finally, we consider the pessimistic watchtower problem in 1.5D.

Theorem 3. *The 1.5D pessimistic watchtower problem can be computed in $O(n \log n)$ time.*

There are two natural variations of the shortest watchtower problem on terrains: the *discrete* version, where the base of the watchtower must be located at a vertex, and the *continuous* version, where the base may also be placed in the interior of an edge. We address both variations simultaneously. The following lemmas characterize solutions to the pessimistic watchtower problem.

Lemma 4. *The edges of the realization whose extensions determine the top of the watchtower in a solution to the 1.5D imprecise pessimistic problem are as steep as possible.*

Proof. If the edges whose extensions determine the top of the watchtower are not as steep as possible, making them steeper increases the height of the watchtower. Notice that this is still true when the base of the watchtower is on the edge adjacent to the edges whose extensions determine the top of the watchtower. An example is shown in Figure 8. □

Lemma 5. *Let $e = (v_i, v_{i+1})$ denote an edge containing the base of a watchtower in a solution to the 1.5D imprecise pessimistic problem. Then v_i and v_{i+1} are placed at the top or bottom of their respective intervals.*

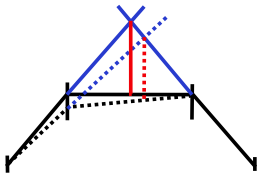


Figure 8: If the top of a watchtower is determined by the extension of the edges adjacent to the edge that contains the base, then the height of the watchtower is maximized when the steepness of the extensions is maximized. Here, the dashed watchtower is shorter than the solid watchtower.

Proof. If both extensions $E(v_{i-1}v_i)$ and $E(v_{i+1}v_{i+2})$ determine the top of the tower, by Lemma 4, the height of the watchtower is maximized by placing v_i and v_{i+1} at the top of their respective intervals.

Suppose that exactly one of the extensions $E(v_{i-1}v_i)$ or $E(v_{i+1}v_{i+2})$ determine the top of the watchtower, without loss of generality say $E(v_{i-1}v_i)$. By Lemma 4, the height of the watchtower is maximized when $v_{i-1}v_i$ is steepest, which occurs when $v_i = t_i$, $v_{i-1} = b_{i-1}$, and $v_{i+1} = b_{i+1}$. If neither the extension $v_{i-1}v_i$ nor $v_{i+1}v_{i+2}$ determine the top of the watchtower then the height is maximized by $v_i = b_i$ and $v_{i+1} = b_{i+1}$. □

We now present an $O(n \log n)$ time algorithm for the pessimistic 1.5D watchtower problem.

Theorem 3. *The 1.5D pessimistic watchtower problem can be computed in $O(n \log n)$ time.*

Proof. Let T be an imprecise 1.5D terrain. Consider the set L of $2n - 2$ lines formed by the top to bottom edges, and bottom to top edges, of consecutive intervals, explicitly

$$L = \{E(b_1t_2), E(t_1b_2), E(t_2b_3), E(b_2t_3), \ldots, E(t_{n-1}b_n), E(b_{n-1}t_n)\}.$$

By Lemma 4, the edge extension(s) that determine the top of a watchtower belong to L. The upper envelope of L, denoted $En(L)$, can be computed in $O(n \log n)$ time [15]. We can assume that an optimal solution includes a vertex of $En(L)$ or a vertex of T.

We consider the x-coordinates of the vertices of $En(L)$ union those of the x-coordinates terrain vertices, sorted in increasing order, we then sweep from left-to-right. For each terrain vertex, we compute the vertical distance from the bottom of the imprecise interval to $En(L)$. For each $u \in En(L)$, let v_iv_{i+1} denote the terrain edge directly below u. By Lemma 5, the base of the watchtower is in one of four configurations. We compute the vertical distance from u to t_it_{i+1}, t_ib_{i+1}, b_it_{i+1}, and b_ib_{i+1}, and take the maximum.

As we sweep, we track the maximized minimum vertical distance between $En(L)$ and the terrain. The edge(s) of $En(L)$ and the terrain edge realizing this

minimum distance determine the placement of the watchtower. Once identified, we fix the corresponding edges while placing all other imprecise points at the bottoms of their intervals. The following theorem confirms correctness.

The realization R described above is a solution to the 1.5D pessimistic watchtower problem. Let u be the point on $E(L)$ and v be the point on R that are found by the algorithm above, and let $|uv| = h$. Consider any other point on R, v'. Suppose the distance from v' to $E(L), h'$ is less than h. Then this contradicts the choice of h, because the point u' is on $E(L)$ and v' on an edge with end points at the maximum or minimum of their imprecise intervals, and this distance is considered by the algorithm.

Consider any other realization R'. By Lemma 4 and Lemma 5, the minimum height watchtower is at most equal to the minimum height watchtower for R. Thus, the realization R is a solution to the 1.5D pessimistic watchtower problem. □

Generalized k-Cell Decomposition for Visibility Planning in Polygons

1st Yeganeh Bahoo *Computer Science*
Toronto Metropolitan University
Toronto, Canada
0000-0001-5349-494

2nd Sajad Saeedi *Computer Science*
University College London
London, UK
0000-0002-6385-6127

3rd Roni Sherman *Computer Science*
Toronto Metropolitan University
Toronto, Canada
0009-0004-0542-3480

Abstract

This paper introduces a novel k-cell decomposition method for pursuit-evasion problems in polygonal environments, where a searcher is equipped with a k-modem: a device capable of seeing through up to k walls. The proposed decomposition ensures that as the searcher moves within a cell, the structure of unseen regions (shadows) remains unchanged, thereby preventing any geometric events between or on invisible regions, that is, preventing the appearance, disappearance, merge, or split of shadow regions. The method extends existing work on 0- and 2-visibility by incorporating m-visibility polygons for all even $0 \leq m \leq k$, constructing partition lines that enable robust environment division. The correctness of the decomposition is proved via three theorems. The decomposition enables reliable path planning for intruder detection in simulated environments and opens new avenues for visibility-based robotic surveillance. The difficulty in constructing the cells of the decomposition consists in computing the k-visibility polygon from each vertex and finding the intersection points of the partition lines to create the cells.

1 Introduction

Pursuit Evasion [3] is a famous problem in computational geometry, especially for robotics and game theory. Pursuit evasion involves one or more pursuers attempting to locate or catch one or more evaders in some environment. There are four main parameters which describe pursuit evasion: the type of environment, the number of pursuers and evaders, the speed of the pursuers and evaders, and the field of vision of the pursuers and evaders. In Pursuit Evasion games, there is also always a strategy involved in how the pursuer attempts to capture the evader and the evader tries to avoid capture.

The main difference between this paper and [2] is that this paper generalizes the idea for 2-visibility cell decomposition of the polygons to k-visibility with a proof of correctness that extends the two proofs in [2]. It should be noted that this generalization was not trivial. The proof of correctness for vertex-shadow and edge-shadow has been modified to work for k-visibility, with the proof of edge-shadow being condensed. The rest of the paper is organized as follows: Sec. 2 presents the background material. Sec. 3 describes the method. Sec. 4 concludes the paper.

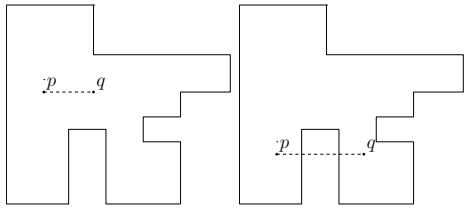
2 Background

Consider a simple polygon P in $2D$ with some evaders and a pursuer moving inside of it. The entire map is known. The pursuers and evader have infinite speed. The evader knows the pursuer's location and all of its movements. The pursuers and evaders move continuously within the polygon. As the pursuer moves, parts of the polygon might become invisible to it. Two points p and q are said to be visible when the line segment pq does not intersect the polygon. This is shown in Figure 1a. In particular, we are interested in k-visibility, where two points p and q are said to be k-visible when the segment pq intersects the polygon at most k times. An example of this is shown in Figure 1b, where two walls ($k = 2$) separate p and q.

Let p be the initial position of the pursuer, an arbitrary point within the polygon P. Each maximal connected set of points within the polygon P that is invisible to p forms what is called a shadow of p. The shadows of p are sub-polygons of P, denoted by $S_i(p)$. As the pursuer (searcher) moves continuously in P, four geometric events may occur for its shadow: merge, split, appear, or disappear [6] (See Figure 2).

The challenge is to develop a cell decomposition such that when a searcher moves within a cell, none of the four events occur [5]. To calculate the k visibility polygon, there are three algorithms available. Martins et al. [4] presented a $O(n^2)$ algorithm. Bahoo et al. [1] improved upon this with a $O(nlog(n))$ algorithm. An

even faster algorithm by Bahoo et al. [1] (for values of $k < log(n)$) runs in $O(kn)$.



(a) 0-visibility (b) 2-visibility

Figure 1: Visibility between two points p and q

3 Proposed Method

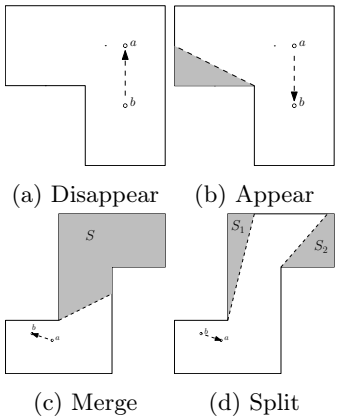
First the cell decomposition is presented, and then a proof of correctness is proposed. The proof consists of three theorems.

3.1 The K-Cell Decomposition

The k-cell decomposition is created by computing all even-valued visibility polygons, from 0-visibility to k-visibility, at each vertex of P. We then use the lines that define each polygon as the partition lines. In other words, we calculate the following visibility polygons:

- Lines of the k-visibility polygon of each vertex.
- Lines of the $k-2$-visibility polygons of each vertex.
- Lines of the $k-4$-visibility polygon of each vertex.
  ...
- Lines of the 0-visibility polygon of each vertex.

An example cell decomposition is presented in Figure 3 :



(a) Disappear (b) Appear

(c) Merge (d) Split

Figure 2: The four geometric events that may occur to the pursuer's shadow as the pursuer moves within the polygon

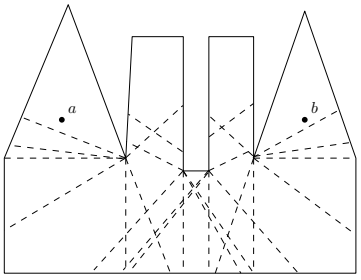


Figure 3: The cell decomposition with all decomposition lines drawn from every vertex, for $k = 2$

The intuition behind this is that a vertex always participates in an event, therefore considering all combinations of vertices at all values of k gives an upper bound solution.

3.2 Proof of Correctness of the K-Cell Decomposition

The proof consists of two theorems for two different types of shadow, and one more theorem which summarizes the two.

Definition 1 *A type 1 shadow (vertex shadow) is defined as a shadow that contains a vertex [2]. See Figure 4.*

Definition 2 *A type 2 shadow (edge shadow) is a shadow that does not include a vertex [2]. This shadow occurs only between edges. See Figure 5. Note that an edge shadow cannot occur for the case of $k = 0$.*

Definition 3 *A partition line is a line of the cell decomposition (which partitions the space into cells)*

Definition 4 *A vertex a that is critical to some point b is a vertex with both of its edges to one side of the line ab.*

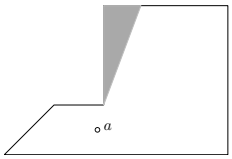


Figure 4: The shadow of type 1 - a shadow that includes a vertex

Theorem 1 *(Invariance of type 1 Shadow) When an agent moves in a cell C, the combinatorial representation of the shadow regions of type 1 remains unchanged.*

Proof. Consider a point p that is inside a cell C, a point q that is also inside the cell C, and a vertex v of

the polygon that is invisible to p, i.e., in the shadow of p (shadow of type 1) (see Figure 6a). Assume for contradiction that v is visible to q. When the pursuer moves from p to q, the pursuer crosses a line of the cell decomposition (See Figure 6b), so p and q are not in the same cell. □

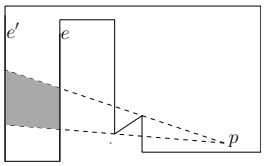
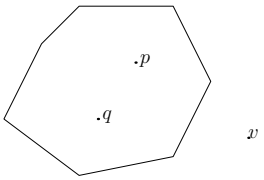
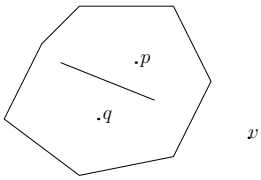


Figure 5: The shadow of type 2 (edge shadow) - a shadow that occurs between two edges



(a) A point p and a point q inside a cell C, and a vertex of a shadow of type 1



(b) A partition line between the vertex p and vertex q

Figure 6: Figures for Theorem 1

Before proceeding to the second theorem, we first introduce the necessary definitions and establish several supporting lemmas.

Consider two points, p and q, located within the same cell C of the decomposition, which denote the position of an agent at two different times as it moves within the polygon (See Figure 7). There are two edges of the polygon e and e', with e' being the next wall after e on the ray pp'. From the point p, there is a shadow of type 2 between these two edges. Two rays, pp' and pp'', define this shadow (p' and p'' being points on e'). Consider a ray emanating from q which rotates counterclockwise around q. Consider the first time this ray hits both e and e'. The intersection of this ray with e is q'. The last time this ray intersects both e and e' it intersects e' at q''.

We assign a coordinate system in which the point p is at the origin $(0, 0)$ and the point q is on the y-axis. Consider intersection points which further describe the geometric layout: Let t be the point that is the intersection between the line qq'' and pp'. Let s be the intersection between the line qq'' and pp''. This intersection points help with definitions in the proof. There exists a vertex, m, on the segment pp' that is critical to p. Call one of the edges of vertex m, the edge of m that makes the smallest angle with the x-axis (as defined by the coordinate system) e_m, as in Figure 8.

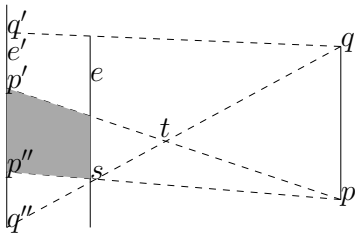


Figure 7: The shadow of p. The intersection points t and s with qq''

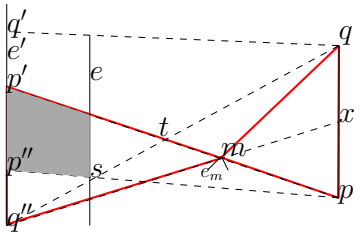


Figure 8: The existence of a vertex, m, on pp' which 'starts' the shadow of p

Lemma 1 *There exists a vertex, m, on the segment pp', with both edges lying below pp'. The vertex m is critical to p.*

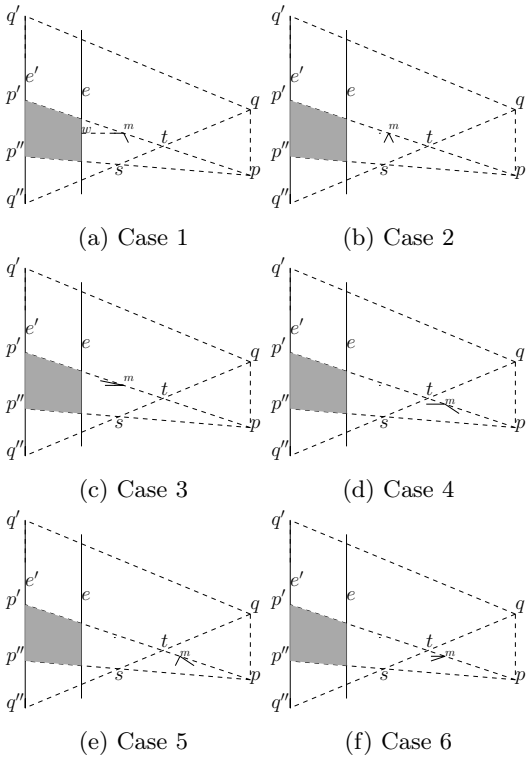
Proof. Since the ray pp' is responsible for forming a part of the shadow of p, it means that somewhere along this segment, there is a vertex which obstructs the visibility from the point p. This vertex has both of its adjacent edges below segment pp'.

□

The next Lemma will be established through proof by contradiction, as follows: we first consider that p has a shadow of Type 2 between edges e and e'. Moreover, for appear/disappear event, we suppose that the area between edges e and e' – the interior of the triangle $qq'q''$ that is between e and e' – is visible to point q.

Lemma 2 *If a k-modem that is in a cell C of the decomposition and with a shadow of type 2 moves continuously inside the cell, appear and disappear events of the shadow regions of type 2 may not happen.*

Proof. The proof is by contradiction. Assume, for contradiction, that there is no vertex in $\triangle mpq$ or in $\triangle mp'q''$ (See the two red triangles in Figure. 8) that is critical to m and forms a partition line with m. This means that as pp' is rotated around m towards q, at every vertex that is critical to m, including the other vertex of e_m (that is not m, see Figure 10), there are k or less walls on both the right and left sides of the rotated line, regardless of which of the six cases we are in (See Figure 9). Thus, there are always less than k walls below pp' (unrotated), so pp' is not the boundary



(a) Case 1 (b) Case 2

(c) Case 3 (d) Case 4

(e) Case 5 (f) Case 6

Figure 9: The Six Cases for Theorem 2

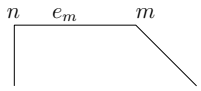


Figure 10: The other vertex of edge e_m, 'n'

of the shadow of p, a contradiction. Note that case 3 is a subcase of 1 and case 6 is a subcase of 4.

□

Lemma 3 *The k-visibility polygon drawn from each vertex ensures that no merge or split occurs when going from one cell to another.*

Proof. If suddenly one shadow splits into two while going from p to q, it means that there is a new critical vertex c that has appeared to q which was not visible at p. As such, the partition line created by the k-visibility polygon of c must intersect the segment pq. So p and q may not lie in the same cell of the decomposition, a contradiction. □

This concludes our final theorem, which is as follows:

Theorem 2 *The k-cell decomposition guaranties that for a shadow of type 2 no geometric events (i.e., appear, disappear, merge, and split) will occur while a pursuer moves continuously in a cell.*

Proof. This follows as a consequence of Lemmas 1, 2, and 3. Note that there are only two possible shadows - vertex shadow and edge shadow. □

Theorem 3 *The proposed cell decomposition is complete. In other words, while moving within a cell, no geometric events may happen to the shadows.*

Proof. This is proved by Theorem 1 and 2. □

3.3 Computational Complexity

The worst case complexity is when all vertices are critical for each other. As such, for a vertex v, there may exist $O(n)$ partition lines for each i in $0, 2, 4 \ldots, k$. So, each vertex, there are $O(nk)$ partition lines. Overall there are n vertices. As such the total number of partition lines is $O(kn^2)$. For a particular vertex, each partition line can intersect all the rest of the partition lines except the one emanating from the same vertex. Consequently, each partition line may intersect $O(kn^2 - kn)$ other partition lines. Each vertex has $O(nk)$ partition lines. So, all the partition lines emanating from a particular vertex intersect $O(nk(kn^2 - kn))$. We have overall n vertices. As such the number of intersections (vertices of the cell decomposition) is $O(k^2n^4)$.

3.4 Applications

The primary purpose of the cell decomposition is to compute a path for one or multiple pursuers that guarantees detection of an intruder, and the same approach presented by Guibas et al. [5] works on the cell decomposition presented in this paper. We extend this concept for k-modems by building this graph on the k-cell decomposition. Using this, we propose an algorithm that computes a path for detecting intruders in a given polygon using a k-modem as the pursuer, if there exists any.

4 Conclusion

This paper studied the pursuit-evasion problem under the k-visibility model, generalizing existing work on $k = 0$ and $k = 2$ visibility-based decompositions. The focus of this research was to develop a cell decomposition that allows a pursuer equipped with a k-modem to navigate within a polygonal environment to detect an intruder. To do so, the environment is split into cells, ensuring that the combinatorial representation of the shadow regions remains unchanged as an agent moves within a cell. This means that none of the key visibility events, that is, merge, split, appear, or disappear occur as the pursuer moves within a cell of the decomposition. Future work could include reducing the number of lines in the decomposition or path planning between two given points to avoid specific geometric event that might occur along the path.

This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

[1] Y. Bahoo, P. Bose, S. Durocher, and T. C. Shermer. Computing the k-visibility region of a point in a polygon. *Theory of computing systems*, 64(7):1292–1306, 2020.

[2] Y. Bahoo, A. Mohades, M. Eskandari, and M. Sorouri. 2-Modem Pursuit-Evasion Problem. In *29th European Workshop on Computational Geometry*, pages 201–204, 2013.

[3] T. Chung, G. Hollinger, and V. Isler. Search and pursuit-evasion in mobile robotics. *Auton. Robots*, 31, 11 2011.

[4] A. M. de Oliveira Martins. *Geometric optimization on visibility problems: Metaheuristic and exact solutions*. PhD thesis, Universidade de Aveiro (Portugal), 2009.

[5] L. J. Guibas, J.-C. Latombe, S. M. Lavalle, D. Lin, e. F. Motwani, Rajeev", A. Rau-Chaplin, J.-R. Sack, and R. Tamassia. Visibility-based pursuit-evasion in a polygonal environment. In *Algorithms and Data Structures*, pages 17–30, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg.

[6] J. Yu and S. M. LaValle. Shadow information spaces: Combinatorial filters for tracking targets. *IEEE Transactions on Robotics*, 28(2):440–456, 2011.

The Quota-TSP on Infinite Lines in the Plane With Obstacles

Joseph S. B. Mitchell* Linh Nguyen†

Abstract

The TSP on infinite lines in the plane is polynomially solvable [1, 3, 4]. It is open whether when obstacles are present, the problem is polynomial or NP-hard. In this abstract, we show that the Quota-TSP on infinite lines (visiting different lines yields different reward and the objective is to compute the shortest route collecting at least some reward) with obstacles is weakly NP-hard and give a fully polynomial-time approximation scheme.

1 Preliminaries

Given a set of m infinite lines $l_1, l_2, \ldots, l_m$ in the Euclidean plane $\mathbb{R}^2$, each associated with a non-negative reward $r(l_i)$, a set of h polygonal obstacles $O_1, O_2, \ldots, O_h$, and a quota Q, the objective is to compute a shortest obstacle-avoiding route that intersects a subset of the lines whose total reward is at least Q. Let n denote the total input size. We have $n = O\left(m + \sum_{i=1}^{h} |\text{vert}(O_i)|\right)$, where $|\text{vert}(O_i)|$ is the number of vertices of obstacle O_i. The Quota-TSP on infinite lines with obstacles asks for a shortest obstacle-avoiding tour γ that collects reward at least Q.

2 Localization and discretization

Let γ be an optimal solution. Denote by $\mathcal{L}(\gamma)$ the set of lines intersected by γ. Note that the minimum enclosing circle of γ also intersects all these lines (and possibly more). Within this circle, there exists a locally minimum circle that intersects a set of lines whose total reward meets the quota.

By critical placement arguments, there are only a polynomial number of such critical circles (i.e., locally minimum circles that intersect a given subset of lines). We can afford to compute each one and discard those that intersect lines whose total reward does not meet the quota. Let $\mathcal{C}$ be one such critical circle that intersects a sufficient-reward set of lines. Let r be its radius. We have $r \leq |\gamma| = O(nr)$ (proof omitted in this abstract), and γ lies within localizing square B of side length $O(nr)$ concentric with $\mathcal{C}$.

We discretize the free space within B, i.e., the region $B \setminus \bigcup_{i=1}^{h} \text{int}(O_i)$. First, we triangulate this region. Then, we overlay a regular square grid over B, with pixel side length $\delta = O\left(\frac{\varepsilon \cdot r}{n}\right)$. Each convex cell in the resulting decomposition has perimeter at most 4δ. The total number of vertices in this decomposition is $O\left(\left(\frac{n^2}{\varepsilon}\right)^2\right)$; let $\mathcal{S}_\varepsilon$ denote this set of vertices. Since γ has at most $O(n)$ vertices (proof omitted in this abstract), for purposes of computing a $(1+\varepsilon)$-approximation, it suffices to consider minimum-length tours γ' with vertices in $\mathcal{S}_\varepsilon$.

*Department of Applied Mathematics and Statistics, Stony Brook University, joseph.mitchell@stonybrook.edu
†Department of Mathematics, Florida A&M University, linh.nguyen@famu.edu

3 Inverting the objective function

Our algorithm is based on a key geometric observation: a connected set intersects a line if and only if its convex hull intersects the line. This motivates the introduction of the intermediate Reward Collecting Convex Polygon problem, for which we propose a dynamic programming algorithm to solve a discrete version. We show how it can be used to approximately solve our Quota-TSPN variant.

In the Reward Collecting Convex Polygon problem, the goal is to compute n' points $p_1, p_2, \ldots, p_{n'}$ in convex position, such that the total reward of the lines intersecting the convex polygon $\overline{\gamma} = (p_1, p_2, \ldots, p_{n'})$ is maximized, subject to a constraint that the geodesic perimeter $|\pi(p_1, p_2)| + \cdots + |\pi(p_{n'}, p_1)|$ is at most a given budget L ($\pi(p_1, p_2)$ denotes the geodesic shortest path between p_1 and p_2). Since the polygon may intersect obstacles, the perimeter is measured in the geodesic metric in the free space, not the Euclidean metric.

Let γ' be the optimal tour for the discrete Quota-TSP on infinite lines (i.e., we have a finite set of points, in this case $\mathcal{S}_\varepsilon$, which vertices of γ' come from), and let $L = |\gamma'|$. We claim that an optimal solution to the discrete Reward Collecting Convex Polygon with budget L yields an optimal solution to the discrete Quota-TSP. Let $\mathcal{L}(\gamma')$ be the set of lines intersected by γ', and consider the convex hull $\mathrm{CH}(\gamma')$. Since $\mathrm{CH}(\gamma')$ intersects all the lines that γ' does, we have $r(\mathcal{L}(\mathrm{CH}(\gamma'))) = r(\mathcal{L}(\gamma')) \geq Q$. Moreover, $\mathrm{CH}(\gamma')$ is a convex polygon whose geodesic perimeter is no greater than L. Thus, we have shown that there exists a convex polygon of (geodesic) perimeter at most L that collects reward at least Q, so the optimal solution $\overline{\gamma} = (p_1, p_2, \ldots, p_{n'})$ to the Reward Collecting Convex Polygon problem with budget L must also collect total reward at least Q. Concatenating the geodesic paths $\pi(p_1, p_2), \ldots, \pi(p_{n'}, p_1)$ into a tour $\widetilde{\gamma}$, then $r(\mathcal{L}(\widetilde{\gamma})) \geq Q$ and $|\widetilde{\gamma}| \leq L$. Since no tour strictly shorter than L can satisfy the quota (by the optimality of γ'), it follows that $|\widetilde{\gamma}| = L$ and $\widetilde{\gamma}$ is also optimal. Similarly, one can show that an optimal Quota-TSP route γ' corresponds to an optimal reward collecting convex polygon, justifying using $\mathcal{S}_\varepsilon$ as candidate turn points for solving (approximately) the Reward Collecting Convex Polygon problem.

Generally, for any $\alpha \geq 1$, if we set $L = \alpha|\gamma'|$ and compute a solution $\overline{\gamma}$ to the discrete Reward Collecting Convex Polygon problem with L as the budget, then $\overline{\gamma}$ is an α-approximation to γ', and therefore a $(1+\varepsilon)\alpha$-approximation to γ.

4 The main theorem

Theorem 4.1. *The* Quota-TSP *on infinite lines with obstacles in* $\mathbb{R}^2$ *has a fully polynomial-time approximation scheme.*

Proof. We can "guess" L to be close (within a factor of $(1+\varepsilon)$) to $|\gamma'|$. To do so, we divide the side length of the localizing square B (which is $O(n|\gamma|)$ and thus also $O(n|\gamma'|)$) into intervals of length εr, resulting in $O\left(\frac{n}{\varepsilon}\right)$ interval endpoints. The interval containing $|\gamma|$ has a right endpoint that is within a factor $(1+\varepsilon)$ to $|\gamma|$. Our algorithm performs a binary search over these endpoints: we find the first L for which a route of length at most L satisfying the quota Q exists. This guarantees that $|\gamma'| \leq L \leq (1+\varepsilon)|\gamma'|$.

For each value of L, we try all possible choices for the point with minimum y-coordinate in $\overline{\gamma}$; call this point p_1. We then solve the discrete Reward Collecting Convex Polygon problem using dynamic programming. First, we preprocess $\mathcal{S}_\varepsilon$ by discarding all points with lower y-coordinate than p_1, then sort the remaining points in clockwise angular order around p_1, breaking ties by increasing distance (i.e., farther points come later). We also add a dummy point $\overline{p} \equiv p_1$ to close the polygon. Let $\{p_1, s_1, s_2, \ldots, \overline{p}\}$ be the sorted list of candidate points. A convex chain consists of vertices appearing in increasing order in this list (assuming clockwise orientation). Without loss of generality, we consider only the clockwise case.

Each subproblem in our dynamic programming is defined by two points $s_i, s_j \in \mathcal{S}_\varepsilon$ with $i < j$, and a budget value L'. The value $DP(s_i, s_j, L')$ denotes the maximum reward collected by a convex polygonal chain (a polygonal chain with vertices in convex position) from p_1 to s_j, ending with segment $s_i s_j$, using total length at most L'. Let $\mathcal{L}(ab)$ denote the set of lines intersecting the segment ab. The recursion works as follows: for $i' < i$, where $s_{i'}$ lies to the right (i.e., same side as p) of the supporting line of $s_i s_j$ (oriented from s_i to s_j),

$$DP(s_i, s_j, L') = \max_{i'} \left\{ DP(s_{i'}, s_i, L' - |\pi(s_i, s_j)|) + r(\mathcal{L}(s_i s_j) \setminus \mathcal{L}(p_1 s_i)) \right\}.$$

See Figure 1. To maintain convexity in the chain associated with subproblem (s_i, s_j, L'), we recurse only on subproblems where $s_{i'}$ lies to the right of $s_i s_j$. Due to convexity, the lines newly intersected by segment $s_i s_j$ are always exactly $\mathcal{L}(s_i s_j) \setminus \mathcal{L}(p_1 s_i)$ and do not depend on the specific chain used in previous subproblems. Thus, the chain from p_1 to s_j of total length L' collects the most reward for its length if and only if the chain from p_1 to s_i of length $L' - |\pi(s_i, s_j)|$ does the same: this is the optimal substructure of our dynamic program.

The base cases $DP(p_1, s, |\pi(p_1, s)|)$ for all candidate turn points s can be computed by brute force. The overall optimum is given by $\max_s \{DP(s, \overline{p}, L)\}$.

However, the values $|\pi(s_i, s_j)|$ may be irrational, making exact tabulation of subproblems infeasible. To remedy this, we apply a standard "bucketing" technique: we discretize the budget into intervals, and round each segment length $|\pi(s_i, s_j)|$ up to the nearest bucket boundary. Let the bucket width be I. Since $\overline{\gamma}$ has at most n vertices, then the total extra length incurred due to rounding is at most nI. To ensure that this does not increase the total length by more than εL, we set $I = \frac{L}{\lceil \frac{n}{\varepsilon} \rceil}$. Thus, we only need to consider the following $O\left(\lceil \frac{n}{\varepsilon} \rceil\right)$ multiples of I:

$$\left\{ 0, \frac{L}{\lceil \frac{n}{\varepsilon} \rceil}, \frac{2L}{\lceil \frac{n}{\varepsilon} \rceil}, \ldots, L + \frac{nL}{\lceil \frac{n}{\varepsilon} \rceil} \right\}.$$

The new recursion reads:

$$DP(s_i, s_j, L') = \max_{i'} \left\{ DP(s_{i'}, s_i, L' - \lceil |\pi(s_i, s_j)| \rceil_I) + r(\mathcal{L}(s_i s_j) \setminus \mathcal{L}(p_1 s_i)) \right\},$$

where $\lceil x \rceil_I$ denotes rounding x up to the nearest multiple of I. The optimal solution for the rounded instance is given by $DP(\overline{p}, L + nI)$.

The running time of the algorithm is dominated by the dynamic programming. The number of subproblems is $O\left(\frac{n^4}{\varepsilon^2}\right) O\left(\frac{n^4}{\varepsilon^2}\right) O\left(\lceil \frac{n}{\varepsilon} \rceil\right) = O\left(\frac{n^9}{\varepsilon^5}\right)$. It takes $O\left(\frac{n^4}{\varepsilon^2}\right)$ time to solve each subproblem, and thus the total running time is $O\left(\frac{n^{13}}{\varepsilon^7}\right)$. We obtain a tour of length at most $(1+\varepsilon)L \leq (1+\varepsilon)^2 |\gamma'| \leq (1+\varepsilon)^3 |\gamma| = (1 + O(\varepsilon))|\gamma|$. □

We note that our approach works even if some obstacles are unbounded. If there is one infinite obstacle enclosing all other obstacles, in essence the free space is a polygonal domain with holes.

4.1 Hardness Proof

Theorem 4.2. *The* Quota-TSP on infinite lines *with obstacles is weakly NP-hard.*

Proof. We reduce from the Min-Knapsack problem: given n items with weights $\{w_1, w_2, \ldots, w_n\}$ and values $\{v_1, v_2, \ldots, v_n\}$, and a target value V, select a subset of items with total value at least V and minimum total weight.

Refer to Figure 2. Begin by drawing a circle of radius $M = 1000 \cdot \max\{w_1, w_2, \ldots, w_n\}$. Draw a second concentric circle of radius $M + \min\{w_1, w_2, \ldots, w_n\}$; without loss of generality, suppose this minimum is w_1. From the center, draw n evenly spaced (radially) narrow corridors extending to the boundary of the bigger circle. Each corridor corresponds to an item in the knapsack instance, and leads to an infinite line placed at its end. Corridor i has length exactly $M + w_i$.

If item i has weight greater than w_1, its corridor includes a zigzag section to increase its length to exactly $M + w_i$ while staying within a narrow cone (of angle much smaller than $\frac{2\pi}{1000n}$) centered along its radial direction. The zigzag section is carefully constructed not to self-intersect and is shown exaggerated in the figure for clarity.

Next, insert obstacles between every pair of adjacent corridors, leaving only a tiny clearance of width $\varepsilon < \frac{\max\{|w_i - w_j|\}}{1000n^2}$. These obstacles are narrow, zigzagging (in congruence with the corridors), and extend far outside the larger circle to ensure it is always suboptimal for the tour to go out of the outer circle.

At the end of each corridor i, place an infinite line tangent to the outer circle, corresponding to item i and providing reward v_i. The construction ensures that the only feasible way to collect reward from line i is to travel through its associated corridor.

The resulting construction, which clearly can be done in polynomial time, has n infinite lines and obstacles of total complexity $O(n)$ (the ratio $(M + w_i)/(M + w_1)$ is smaller than 2, so each corridor needs to contain only one zigzag section, thus each obstacle has constant complexity). Any optimal route for quota V will necessarily select the corridors corresponding to an optimal subset of items in the Inverse Knapsack instance. Thus, solving this instance of Quota-TSP yields a solution to Inverse Knapsack, hence we have shown (weak) NP-hardness. □

Determining the exact hardness of all the variants of TSP on infinite lines with obstacles is an intriguing open direction. One special case where we know of a solution is when the obstacles are the faces of the arrangement formed by the n lines. In that setting, both the TSP and the k-TSP (computing the shortest route visiting all or at least k lines) can be solved in polynomial time, specifically in $O(n^7)$ time [2].

References

[1] Moshe Dror, Alon Efrat, Anna Lubiw, and Joseph S. B. Mitchell. Touring a sequence of polygons. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing*, pages 473–482, 2003.

[2] Kien C. Huynh, Joseph S. B. Mitchell, Linh Nguyen, and Valentin Polishchuk. Optimizing visibility-based search in polygonal domains. In *Proceedings of the 19th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT 2024)*, pages 27:1–27:16, 2024.

[3] Xuehou Tan, Tomio Hirata, and Yasuyoshi Inagaki. An incremental algorithm for constructing shortest watchman routes. *International Journal of Computational Geometry & Applications*, 3(04):351–365, 1993.

[4] Xuehou Tan, Tomio Hirata, and Yasuyoshi Inagaki. Corrigendum to "an incremental algorithm for constructing shortest watchman routes". *International Journal of Computational Geometry & Applications*, 9(03):319–323, 1999.

A Figures

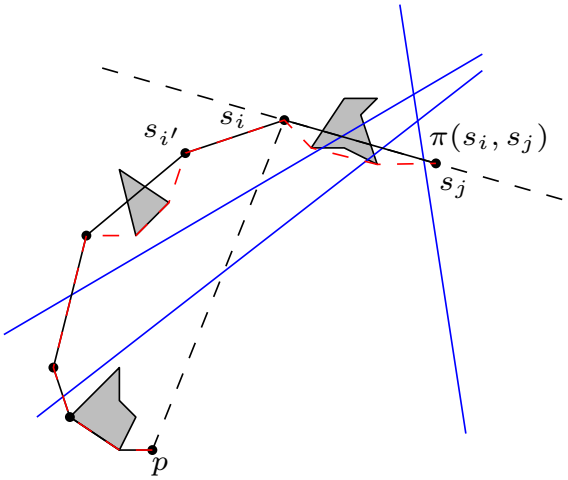


Figure 1: Dynamic programming subproblem (s_i, s_j, L').

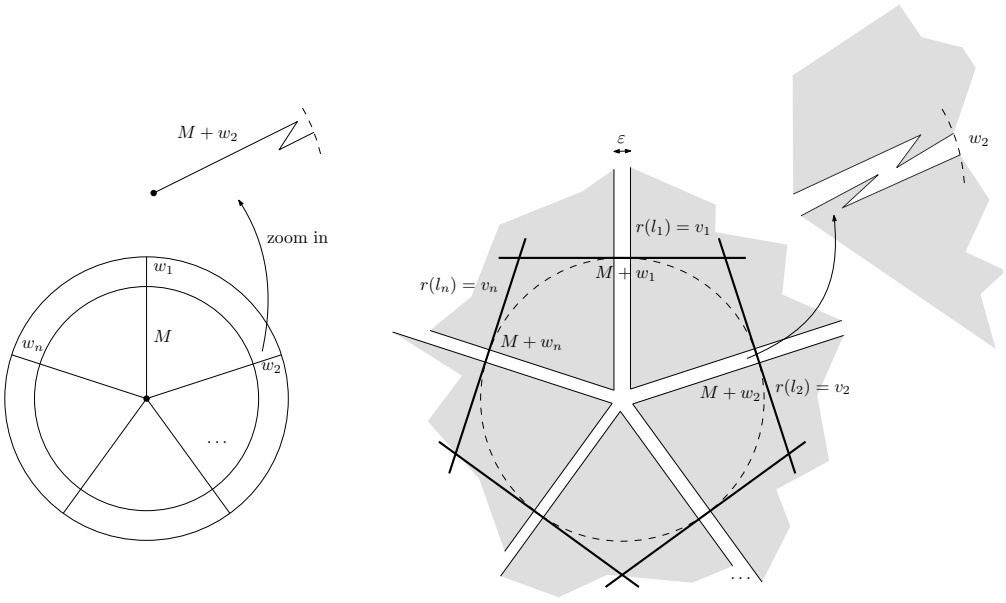


Figure 2: Reduction from Inverse Knapsack to Quota-TSP on infinite lines with obstacles.

M-Guarding Polygons In K-Visibility

Yeganeh Bahoo[1] and Ahmad Kamaludeen[1]

Department of Computer Science, Toronto Metropolitan University, Toronto, Canada
{bahoo,ahmad.kamaludeen}@torontomu.ca

1 Introduction

The Art Gallery Problem (AGP) is a classic challenge of ensuring every point in a polygon P is seen by at least one guard [12,17,15]. We study a variation with two key complexities: *M-guarding*, where each point must be seen by at least M guards, and *k-visibility*, where a line of sight can penetrate up to k polygon edges. The concept of k-visibility originates from the "Superman Problem" [14] and has since been adapted to AGP variants [1], with efficient algorithms developed for computing k-visibility regions [3,4]. The notion of M-guarding has also been explored in 0-visibility [6]. However, the combination of these two areas remains largely unexplored, with limited related work in super-guarding [9,2]. We present a theorem establishing that any polygon with holes can be 2-guarded under k-visibility where $k \geq 2$, which expands existing results in 0-visibility. We provide an algorithm that M-guards a polygon using a convex decomposition of the polygon. We show that for any even $k \geq 2$ there exists a placement of guards such that every point in the polygon is visible to $k+2$ guards. Motivated by modern applications of k-visibility in wireless communications and mapping [18,11], this paper presents a formal algorithm for M-guarding polygons using edge-restricted guards.

2 Definitions

This section introduces the key terms and concepts essential for understanding the problem of k-visibility guarding in polygons with holes. We use some basic terms that are well defined in the field of Computational Geometry, such as *polygon*,*edges reflex vertices*, *simple polygon*, *pocket* and *holes*. See [15,16] for formal definitions.

A *diagonal* is a line segment between two vertices that lies entirely within P. For this paper, an *obstacle* is a polygon edge. Two points are *k-visible* if the line segment between them crosses at most k obstacles. A *guard* is a point within a polygon that has the ability to "see" or cover some points in the interior of the polygon. A *k-visibility guard* can see through up to k obstacles. All guards in this work are restricted to the interior side of polygon edges.

3 Results

In this section, we state the contributions that this paper will provide. As a reminder in this work, our goal is to guarantee that each point in any polygon P can be seen by at least M guards when $k \geq 2$ and to establish a structured method for placing guards to ensure M-visibility. To achieve this, we present an algorithm that uses an *optimal convex decomposition*, and then assigns guards to edges while keeping the required coverage. *Optimal convex decomposition* refers to the division of a polygon P into non-overlapping polygons such that each edge of the polygon P is an edge of exactly one sub-polygon. In this paper, they are divided using diagonals, and all pieces are convex. A *real edge* refers to an edge of the original polygon. Also, a *dual graph* refers to a graph where each node represents a convex piece, and an edge exists between two nodes if their corresponding convex pieces share an edge. We begin by recognizing that for a given simple polygon P, an optimal convex decomposition can be constructed. [8]

3.1 Preliminary Results

Observation 1 *A non-convex simple quadrilateral P has at most one pocket.*

Theorem 1. *Every polygon with holes can be 2-guarded under k-visibility for $k \geq 2$.*

Proof. Let P be a polygon, and suppose we are using guards with k-visibility for some $k \geq 2$. It is known that any simple polygon (i.e., a polygon without holes) can be 2-guarded under 0-visibility [6]. Since k-visibility generalizes 0-visibility, any 2-guarding solution for a simple polygon in 0-visibility also holds for $k \geq 2$. Consider modifying P by introducing a hole, creating a new polygon P'. The original 2-guarding solution may no longer suffice under 0-visibility, as the hole may obstruct visibility. To compensate, we place guards on each edge of the hole. Let H denote this set of guards, each of which has k-visibility with $k \geq 2$.

Each guard in H can see through up to $k \geq 2$ walls, allowing them to view visibility to regions that became hidden when the hole was introduced. For convex holes, the guards on the hole's boundary can collectively see into the previously visible region multiple times, up to $|H|$ times. Non-convex holes introduce reflex vertices that can create pockets. However, by placing a guard on each edge of the hole, we ensure that every such reflex vertex is adjacent to at least one guard. Because each guard can see through at least two edges, the k-visibility condition allows these guards to recover visibility to regions that would otherwise remain hidden. Thus, the union of the original 2-guarding solution and the guards placed on the hole's edges ensures that P' remains 2-guarded under k-visibility. □

Lemma 1. *Given a monotone simple polygon P and an optimal convex decomposition of it, connecting 2 adjacent convex pieces creates 1 obstacle composed of 2 edges for any single ray of vision, which would be entirely visible in 2-visibility.*

Proof. Adding each diagonal during the decomposition can eliminate at most two reflex vertices, and reflex vertices always form a pocket [7]. Therefore we may eliminate one or two pockets. In the case where we have two pockets, they do not overlap, and there is no singular ray of vision that is blocked twice when convex pieces are merged [10]. Therefore, by doing the opposite process of merging the polygon and removing diagonals that partition the polygon, at each step, we do not invalidate the 2-visibility guards by blocking the existing guards multiple times in one move, as long as there was no guard on the removed diagonal. □

To explain the Lemma 1 in simple terms, when we connect 2 pieces in the optimal convex decomposition, there should be a pocket separating them, thus forming a concave polygon. This concave polygon would be entirely visible to any 2-visibility guard. This is the key to proving that merging pairs of convex polygon, which we will do later, maintains visibility enough that every point will be seen by 4 different guards.

Now, consider when we remove a diagonal that partitions the polygon, there may be a guard on that edge. We can move that guard to be in a different convex piece. We claim there exists a point another piece that can guard the entirety of the first convex piece.

Lemma 2. *Let A be a convex polygon. If a point g outside of A sees the entire boundary ∂A, then g sees every point in A.*

Proof. Omitted for space, see Appendix A

Algorithm 1 Sweeping algorithm for critical vertices in a pocket

Require: A reflex chain defined by vertices $v_1, v_2, \ldots, v_n$ forming a pocket, and an edge e on the boundary of region B facing the pocket.

Ensure: A segment $S \subset e$ such that every point in S sees the entire boundary of the pocket under k-visibility.

Let v_n be the vertex at the end of the reflex chain (where the sweep begins).
Let v_1 be the vertex at the start of the reflex chain.
Initialize an empty list C to store the critical vertices encountered by the sweep.
Initialize a ray L originating at v_n, rotating counterclockwise along the boundary of B.
for each reflex vertex u_i in the chain from v_n to v_1 **do**
    if the ray L encounters u_i before intersecting any other chain edge **then**
        Append u_i to the list C.
    end if
end for
Let k be the number of critical vertices found in the pocket.

Let w be the $\lceil k/2 \rceil$-th vertex in C.
Let L_w be the ray from v_n through w.
Let p be the intersection of L_w and edge e.
Let x be the reflex vertex at the endpoint of e.
Let S be the subsegment of e between p and x.
return S.

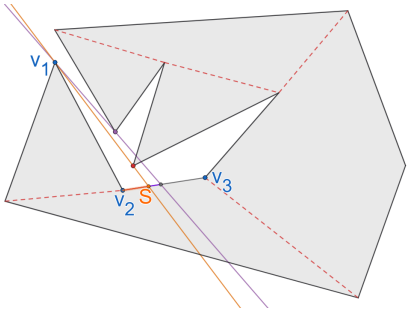


Fig. 1: The ray is cast from v_1 along the reflex chain that forms the pocket. Once we find the $(k/2)$-th critical reflex vertex (orange ray), the sweep ends

Lemma 3. *When the convex polygon B in the decomposition is separated from another convex polygon A by a diagonal d, there exists a non-empty set of points S on the boundary ∂P such that each point in S sees the entire boundary of A in 2-visibility.*

Proof (Proof by Contradiction). Let d denote the diagonal that separates A and B. Both A and B are convex. In our first case, B is not interior. We consider the strong k-visibility polygon of each edge of A. The intersection of these visibility regions and B, is a set T of points in B that, if it exists, sees the entire boundary A [5]. Since both A and B are convex and share a diagonal d, a point ϵ away from d in B can see all of A. We aim to show that this intersection is non-empty and forms our desired set S. Assume for contradiction that S is empty. That is, for all points $b \in B$, the point b cannot see the entire boundary of the convex polygon A. Now, consider a point $b \in B$ that is ϵ distance away from the diagonal d. Since A is convex, any obstruction e preventing b from seeing an edge of A must lie on the boundary of A, in B, or on a third polygonal region C:

1. If e is part of A, then visibility is blocked by the boundary of A, which is acceptable since the point b acts as a 2-visibility guard, and therefore can see through this edge.
2. e lies in B: If e is the edge that point b lies on, it can see through this point by 2-visibility. If this is some other edge, then B is no longer convex, and that is a contradiction.
3. e lies in a third polygonal region C: In this case, e must lie between $b \in B$ and some portion of ∂A. But since b is arbitrarily close to d, for e to block the view, it must intersect the visibility cone from b to ∂A near d. However, due to convexity of both A and B, and the fact that b is infinitesimally close to d, such an edge e must either intersect the interior of A, which contradicts the assumption that regions are convex and form a simple polygon, since the boundaries are intersecting. (Fig. 2)

If e lies outside of both A and B, then e cannot block any visibility from $b \in B$ to A without intersecting A, again contradicting the assumption that A is a valid convex polygon. Hence, the only possible obstructions to b seeing

all of A are edges on ∂A. But since b is arbitrarily close to d, and A is convex, there exists a view corridor from b to any edge of A. Therefore, our assumption must be false, and the set S is non-empty. Next, our second case where we consider that B can be an interior polygon. If B is an interior polygon, then instead of B, we consider the convex piece C that has a real edge and shares a vertex v with d. All intervening convex pieces between C and A will be separated by diagonals originating from v. By the same argument above, we can see that there will be no edge e blocking the visibility of the guards in C from A (other than the boundaries of C and A themselves), and so, the set S is still non-empty as expected and our proof is complete. □

Lemma 3 implies that we have all the points between ϵ and the vertex to place the guard, and this region will be determined by any reflex vertices that are within the pocket. (Fig. 3)

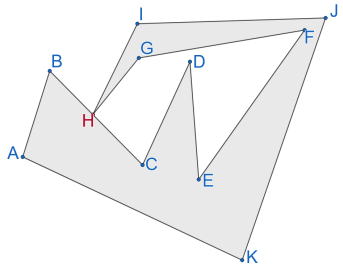


Fig. 2: An invalid polygon with a hole. The vertex H intersects the boundary of the outer polygon, violating the requirement that holes must lie entirely within the outer polygon

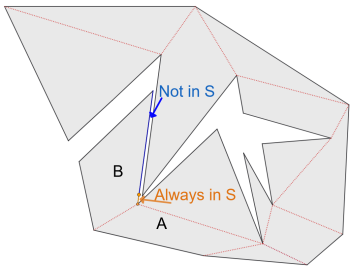


Fig. 3: A polygon that demonstrates the size of the valid area for placing a guard in the adjacent piece.

Lemma 4. *Let A and B be adjacent convex polygons in a convex decomposition of a polygon, sharing an edge, which is the diagonal d. Suppose a 0-visibility guard is initially placed on d to guard A. Then this guard may be repositioned to a point on a real edge that guards A and B in 2-visibility.*

Proof. By Lemma 3, there exists a non-empty set of points S on the boundary of the polygon P, arbitrarily close to d, such that each point in S sees the entire boundary of A under 2-visibility. By Lemma 2, any such point that sees all of ∂A also sees the entire interior of A. Thus, a 2-visibility guard originally placed on the diagonal d can be moved to a point in S without loss of coverage, since there may only be 2 new edges blocking it. Since S contains points that guard ∂A in 2-visibility, placing the guard on one of these points maintains full 2-guarding of A. □

We also claim that we do not adjust any single guard more than once. Since we are traversing the dual graph, we will be adjusting the guards forward into the next piece, or the one after that, until we reach the final ear. In this method, we will not backtrack, and thus we will not be adjusting 2 guards to the same edge.

3.2 $(k+2)$-guarding with k-visibility

Given a polygon P, we propose Algorithm 2 for $(k+2)$-guarding of P using k-visibility. Before we go into detail, we provide a short overview.

We begin by decomposing the non-convex polygon P into an optimal convex decomposition and constructing its dual graph. Adjacent convex regions are then paired along the dual graph wherever possible, leaving some regions unpaired (such as ears or leaves) when pairing is not feasible due to the structure of the dual. For each paired set of adjacent convex pieces, a guard is placed on a real edge or on the shared diagonal between them so that both regions are visible under k-visibility. For isolated convex regions, a guard is similarly placed on one of their real edges. Around each vertex, a small area containing only its two incident edges is considered. Within this area, every point can see at least one of the adjacent convex regions under k-visibility, ensuring local coverage of the boundary without obstruction by other parts of the polygon. Finally, each guard is relocated by a small ε along the polygon boundary in the direction away from the root of the dual tree. This guarantees that the guards remain on real edges after merging adjacent convex pieces and that the entire polygon remains fully covered under k-visibility. See Fig. 4 for a visual example of the algorithm. We have the following theorem:

Theorem 2. *Given a polygon P (possibly with holes) and a visibility parameter k, and let $k = 2i$ for some integer $i \geq 0$, there exists a placement of guards on the edges of P such that every point in P is $(k+2)$-guarded under k-visibility.*

Proof. This can be accomplished following the steps of Algorithm 2. The proof of the correctness of this algorithm is as follows: We prove that every point in the polygon is visible to at least $(k+2)$ guards by induction on the convex pieces.

Base Case: $n = 1$. For a single convex polygon, place one guard at the midpoint of each real edge of the ear. Since the polygon is convex, every point in the polygon is k-visible to each guard, and with at least $k+2$ edges, every point is seen by at least $k+2$ guards.

Inductive Step: Assume that for some $n \geq 1$, every point in a polygon P_n with n convex pieces is $(k+2)$-guarded. We show this holds for a polygon with $n+1$ convex pieces. Consider a polygon with $n+1$ convex pieces. Removing one convex piece yields a polygon with n convex pieces,

where every point is $(k+2)$-guarded by assumption. When we add a convex piece to a polygon, it is always an ear, and therefore has 2 real edges as shown by the well-known Two Ears Theorem [13]. For the new convex piece, if there was a guard on edge that connects the $(n+1)$-th piece to the polygon, place one guard on the real edge within the intersection of the strong segment k-visibility polygons of the n-th piece, and place the other guard anywhere on an available free edge. Otherwise, we can place 2 guards onto the available real edges. Merging the piece into P_n ensures that every point remains $(k+2)$-guarded, as long as we have placed the guards within the visibility polygons of all the segments of the adjacent piece, as proved in Lemma 1, Lemma 3, and Lemma 4. It is also possible that there is a piece inside the pocket that blocks the visibility when we combine a non-monotone polygon with a convex piece (Fig 3), So we check if the next piece is completely visible to the segment. If it is not, then we need to use Lemma 3, and find the set of points S that will guard the X by finding the critical vertices between the two pieces, X and C, which can be accomplished with Algorithm 1. A critical vertex is a reflex vertex whose adjacent edges lie in the same half-plane. When the sweep passes such a vertex, the visible region shrinks because the vertex acts as a spike that obstructs part of the pocket. Consequently, the feasible set S on the segment is determined by the endpoint of the segment and these critical vertices. See Figure 1 for a visual illustration. Therefore, every point in the polygon with $C+1$ convex pieces is visible to at least four guards, and so by the principle of mathematical induction, every point in a polygon with holes is visible to at least four guards. By induction, every point in a polygon with holes is $(k+2)$-guarded. □

As a result of Algorithm 2 and Theorem 2, it follows that there exists an upper bound on the number of guards required to k+2-guard a concave polygon under k-visibility.

Observation 2 *For any concave polygon P that is decomposed into C convex pieces, at most kC guards are sufficient to $(k+2)$-guard P.*

The decomposition used in our construction ensures that each convex piece can be $(k+2)$-guarded by at most k guards placed on its boundary. Since the decomposition yields C convex pieces, assigning at most k guards per piece requires no more than kC guards. Usually, as k increases, we can view more of the polygon, thus our number of guards should decrease. However, in this problem, as k increases, the maximum M-guarding possible increases up to the number of vertices n.

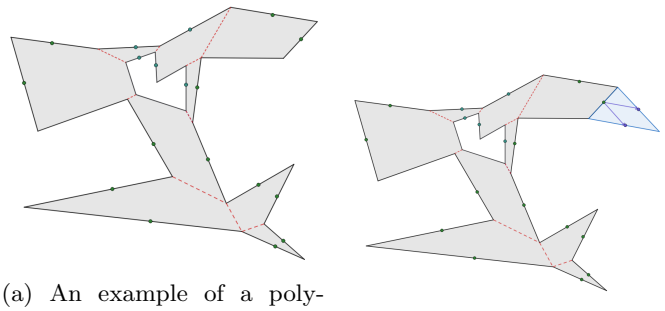
4 Conclusion

In this work, we focused on addressing M-guarding under k-visibility without attempting to minimize the number of guards. Our approach allows for new strategies that ensure multiple guards cover an area under k-visibility via an efficient algorithm, and presents a bound for the number of guards facilitating practical applications where redundancy is more critical than minimizing number of guards deployed.

Algorithm 2 $(k+2)$-Guarding a Polygon with k-visibility

Require: A non-convex polygon P and a parameter k.
Ensure: Guard set G that $(k+2)$-guards P under k-visibility.

```
Decompose P into optimal convex pieces.
Merge pieces to have ≥ k edges (except ears).
Construct the dual graph of the convex decomposition.
Let C_0 be the leftmost convex ear in the decomposition.
Place one guard at the midpoint of each real edge of C_0.
for each convex piece C_i along the dual graph path starting from C_0 do
    Perform the sweep-line algorithm (Algorithm 1) to find critical points in the pocket formed between C_i and the next piece C_{i+1}.
    if there are fewer than k/2 vertices in the pocket then
        Place guards at the midpoints of real edges of C_i.
    else
        Place the guards in the set found by Algorithm 1.
    end if
    if guards cannot be placed on real edges because not enough edges are available then
        Place the guards on the diagonals.
    end if
    Merge C_i with the current merged region.
    if a guard is placed on a diagonal that is removed during merging then
        Relocate the guard g_f to a point on a real edge of the adjacent region from which the entire boundary of the previous convex piece C_{i-1} is visible under k-visibility, by computing the intersection of C_{i-1} and the strong k-visibility polygons of every edge in C_i.
        if there are no real edges in the adjacent region then
            Let e be the real edge connected to the vertex of the diagonal d.
            Let S be the set of points on e found using Algorithm 1 with respect to the pocket and edge e.
            Relocate the guard to a point in S.
        end if
    end if
end for
return The set of k-visibility guards that (k+2)-guard the entire polygon.
```



(a) An example of a polygon after decomposition and 2-guarding

(b) An example of a polygon with an additional piece.

Fig. 4: Panels (a) and (b) show different steps of Algorithm 2.

This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

1. Aichholzer, O., Monroy, R.F., Flores-Peñaloza, D., Hackl, T., Urrutia, J., Vogtenhuber, B.: Modem illumination of monotone polygons. Comput. Geom. **68**, 101–118 (2015), https://api.semanticscholar.org/CorpusID:12402148
2. Aldous, G., Barber, S., Üngör, A.: On super-guarding convex and star-shaped polygons. In: Proceedings of the 37th Canadian Conference on Computational Geometry (CCCG). Toronto, Canada (Aug 2025), to appear
3. Bahoo, Y., Banyassady, B., Bose, P.K., Durocher, S., Mulzer, W.: A time–space trade-off for computing the k-visibility region of a point in a polygon. Theoretical Computer Science **789**, 13–21 (2019). https://doi.org/https://doi.org/10.1016/j.tcs.2018.06.017, https://www.sciencedirect.com/science/article/pii/S030439751830433X, selected Papers from the 11th International Conference and Workshops on Algorithms and Computation
4. Bahoo, Y., Bose, P., Durocher, S., Shermer, T.C.: Computing the k-visibility region of a point in a polygon. Theory of Computing Systems **64**(7), 1292–1306 (2020). https://doi.org/10.1007/s00224-020-09999-0
5. Bahoo, Y., Kundu, S., Manastyrski, K.: Segment Visibility for k-Transmitters, pp. 1–12 (12 2023). https://doi.org/10.1007/978-3-031-48882-5_1
6. Belleville, P., Bose, P., Czyzowicz, J., Urrutia, J., Zaks, J.: K-guarding polygons on the plane. In: Canadian Conference on Computational Geometry (1994), https://api.semanticscholar.org/CorpusID:18853139
7. Ghosh, S.K.: On recognizing and characterizing visibility graphs of simple polygons. In: SWAT 88: 1st Scandinavian Workshop on Algorithm Theory Halmstad, Sweden, July 5–8, 1988 Proceedings 1. pp. 96–104. Springer (1988)
8. Greene, D.H.: The decomposition of polygons into convex parts. Computational geometry **1**, 235–259 (1983)
9. Group, M.C., Akitaya, H.A., Demaine, E.D., Hesterberg, A., Lubiw, A., Lynch, J., O'Rourke, J., Stock, F.: Super guarding and dark rays in art galleries. arXiv preprint arXiv:2404.04613 (2024)
10. Keil, J.M.: Polygon decomposition. Handbook of computational geometry **2**, 491–518 (2000)
11. Kim, J., Zalat, J.A., Bahoo, Y., Saeedi, S.: Structure from wifi (sfw): Rssi-based geometric mapping of indoor environments. arXiv preprint arXiv:2403.02235 (2024)
12. Kumar Ghosh, S.: Computing the visibility polygon from a convex set and related problems. Journal of Algorithms **12**(1), 75–95 (1991). https://doi.org/https://doi.org/10.1016/0196-6774(91)90024-S, https://www.sciencedirect.com/science/article/pii/0196677491900245
13. Meisters, G.H.: Polygons have ears. The American Mathematical Monthly **82**(6), 648–651 (1975)
14. Mouawad, N., Shermer, T.: The superman problem. The Visual Computer **10**, 459–473 (1994)
15. O'Rourke, J., et al.: Art gallery theorems and algorithms, vol. 57. Oxford University Press Oxford (1987)
16. Shermer, T.C.: Recent results in art galleries (geometry). Proceedings of the IEEE **80**(9), 1384–1399 (1992)
17. Urrutia, J.: Art gallery and illumination problems. handbook on computational geometry. Elevier Science Publishers (2000)
18. Yang, Y., Xu, X., Zeng, Y., Sun, H., Hu, R.Q.: Channel knowledge map for cellular-connected uav via binary bayesian filtering. IEEE Transactions on Communications (2025)

A Proofs

Lemma 2. *Let A be a convex polygon. If a point g outside of A sees the entire boundary ∂A, then g sees every point in A.*

Proof. Let g be a guard located outside the convex polygon A, such that g sees the entire boundary ∂A. Let p be any point in the interior of A. Since A is convex, any segment between two boundary points lies entirely within A. Because g sees the entire boundary, we can select two boundary points q_1 and q_2 visible to g such that the triangle $\triangle gq_1q_2$ contains p. Since $\overline{gq_1}$ and $\overline{gq_2}$ are unobstructed and $q_1, q_2 \in \partial A$, and since A is convex, the triangle $\triangle gq_1q_2$ intersects A only in its interior. Therefore, the line segment $\overline{gp}$ lies entirely within $\triangle gq_1q_2$ and hence within the union of A and the guard's view. So p is visible to g. Hence, every point in A is visible to the guard g, and the lemma holds. □

Flex and One-Removal: Two Versions of NP-Hard Problem Stability

Samuel Cohen, Alper Üngör

Computer & Information Science & Engineering, University of Florida
[samuel.cohen,ungor]@ufl.edu

Abstract

We consider NP-Hard problems when a particular or any single element of the input set is removed. These removal options, referred to as one-removal and flex (any-removal), can change the hardness of the problem. We present new complexity results on the one-removal and flex versions of a collection of NP-Hard set/graph theory and geometric problems.

1 Introduction

Consider an NP-Hard decision problem $P(S)$, where S is a collection of objects. Define the *flex* version of P as flex-$P(S) = \bigwedge_{x \in S} P(S \setminus \{x\})$, asking if P always holds given the removal of any element in an input set. For example, flex-SUBSETSUM asks whether a set contains a subset summing to a target value when any element is removed. If P is NP-Hard then flex-P may be easy or hard; in the case of some problems including flex-PARTITION the hardness is unknown.

Suppose instead that we have an optimization problem P(S), where S is a collection of objects. Define the *one-removal* version of P as or-$P = P(S \setminus \{x\})$ where S, the solution for $P(S)$, and $x \in S$ are given.

By analyzing the hardness of the flex and one-removal versions for NP-hard problems, we can classify the problems by flexibility.

Contribution. We prove that the flex versions of SUBSETSUM, HAMILTONIANCYCLE with vertex (edge) removal, and TRIANGLEPACKING, the one-removal versions of 2DBINPACKING and 2DSTRIPPACKING are NP-Hard, and discuss open problems in flexibility analysis.

2 Background

2.1 Motivation

Robustness and fault tolerance are critical considerations in system design. For example, interdependent critical infrastructure systems (ICISs) such as electrical power networks and water networks must be built to withstand natural disasters [1, 2] . A city might like to guarantee that power will still function if any individual unit is destroyed, invoking the flex problem. Or, if the supplier must efficiently recompute network dynamics after a component is destroyed, reoptimization is invoked [3].

From a theoretical perspective, the flex and one-removal problems tell us something about the underlying solution space for various NP-Hard problems. We aim to classify some problems by flexibility and point to other open problems as areas of future research.

2.2 Flex Problem

Our flex problem asks whether a set holds a specified property when *any* of its elements are removed. This is consistent with the literature on robustness of NP-Hard problems, which asks how solutions to NP-hard problems change under small perturbations of the input. Authors typically focus on modifications involving small changes to input values, such as one changed edge weight [4]. However, resilience to element removal is not well-studied.

One exception is in graph theory, which has seen a number of results in property resilience to single edge or vertex removal. For example, researchers have shown the existence of graphs that decrease in chromatic number when any vertex is removed but keep their chromatic number when any edge is removed [5]. Dirac famously proved that complete graphs on $n \geq 3$ vertices remain Hamiltonian when up to $\lfloor n/2 \rfloor$ edges touching each vertex are removed [6]. A similar result has been proven on directed graphs [7]. On random graphs, the resilience can be bounded with respect to a generic graph property using binomial distributions and graph partitioning [8]. In this paper, we seek to continue building towards a unified theory of resilience by studying flex versions of various NP-hard problems.

2.3 One-Removal Problem

Our one-removal problem asks whether given an instance of an optimization problem *and its solution* we can easily find optimal solutions when one element of the instance set is removed. The fact that a problem is being solved with complete knowledge of a nearby solution is consistent with the literature around reoptimization, a term first coined by Schäffter [9]. Interestingly, reoptimization problems are often as hard as the original problems [10], implying that the additional information of a nearby solution might not be helpful. For example, past authors have demonstrated the NP-hardness of reoptimizing TSP and metric TSP over changes in individual edge weights [3], NP-hardness results have also been proven specifically on one-removal problems, including reoptimizing Steiner Tree over vertex removal [11], TSP over vertex removal [12], metric TSP over vertex removal [13], and max k-colorable subgraph over vertex removal [14]. Such research also frequently focuses on improving approximation ratios by using local information from the previous solution. This paper only looks at hardness of an exact solution and leave approximation ratio improvements as future work. Our goal is to expand this area of research to computational geometry by studying the hardness of one-removal on two geometric packing problems.

3 Flex Partition/SubsetSum

Here we consider three similar NP-Hard set theory problems. The flex version of the first can be solved in polynomial time, the second is an open problem, and the third will be shown to be NP-hard.

Definition 1 (EQUALCARDINALITYPARTITION (ECP))**.** *Given a set S, is it possible to partition S into two subsets of equal sum and equal cardinality?*

Definition 2 (PARTITION)**.** *Given a set S, is it possible to partition S into two subsets of equal sum?*

Definition 3 (SUBSETSUM)**.** *Given a set S and a target T, does there exist a subset of S whose sum is T?*

Theorem 1. *[15] flex-*ECP *over the integers is solvable in $O(n)$ time.*

Proof. It has been shown by King [15] in an unpublished manuscript that if S is a list of integers, then flex-ECP(S) is true if and only if all values in S are the same and $|S|$ is odd. This can be checked in $O(n)$ time. □

Whether flex-PARTITION is NP-hard is an open problem. The solution space to flex-PARTITION has some basic structure, but its exact size and shape are unknown.

Theorem 2. *flex-*SUBSETSUM *is NP-Complete.*

Proof. flex-SUBSETSUM is verifiable in polynomial time so flex-SUBSETSUM is in NP. Given an instance (S, T) of SUBSETSUM, consider flex-SUBSETSUM on $(S \cup \{T\}, T)$. Whenever an element of S which is not T is removed, there will be a subset which sums to T, namely the singleton $\{T\}$. But when T is removed, we are left with SUBSETSUM on S. Thus, SUBSETSUM(S, T) = flex-SUBSETSUM$(S \cup \{T\}, T)$, completing the reduction. □

4 Flex Hamiltonian Cycle

Definition 4 (HAMILTONIANCYCLE (HC))**.** *Given a graph G, does there exist a cycle which visits every vertex once?*

There are two options for what to remove from a graph to create a flex version of HC: vertices and edges. We define both versions below.

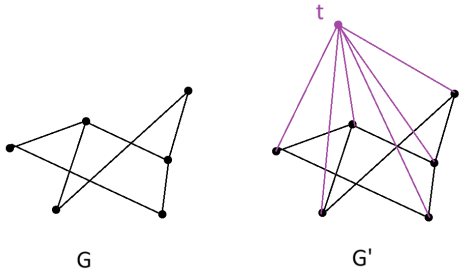
Definition 5 (flex-vertex-HAMILTONIANCYCLE (FVHC))**.** *Given a graph G, when any vertex p and its incident edges are removed, does the new graph always contain a Hamiltonian cycle?*

Definition 6 (flex-edge-HAMILTONIANCYCLE (FEHC))**.** *Given a graph G, when any edge e is removed, does the new graph always contain a Hamiltonian cycle?*

Using either version, we will end up with a problem that is as difficult as the original HAMILTONIANCYCLE problem.

Theorem 3. FVHC *is NP-Hard.*

Proof. Let G represent an instance of the HC problem. Add a new vertex t and connect it to every vertex in G. Call this new graph G', drawn below. We claim that FVHC(G') = HC(G).

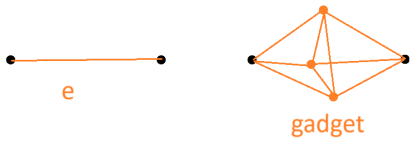


If HC(G) = true, then G contains a Hamiltonian cycle C. When t is removed, we are left with G, which has C as a Hamiltonian cycle. When any other vertex p is removed, we can replace p with t in C to get a Hamiltonian cycle, since t is connected to every vertex. Thus, FVHC(G') = true.

If HC(G) = false, then when t is removed, we are left with G, which contains no Hamiltonian cycle. Thus FVHC(G') = false. □

Theorem 4. FEHC *is NP-Hard.*

Proof. Let G be the input graph. Replace every edge e in G with the following gadget, inserting three new vertices:



If HC(G) = true, then we can still find a Hamiltonian circuit when any edge is removed since the gadget has built-in redundancy, so FEHC(G) = true.

IF HC(G) = false, then there will be no Hamiltonian circuit in the modified graph even before any edge is removed, since no new effective connections are made between any of the original vertices, so FEHC(G) = false. □

We can similarly define flex-vertex Hamiltonian-Path and flex-edge Hamiltonian-Path, which we will denote as FVHP and FEHP. Note that both FVHP and FEHP are also NP-Hard; proofs are similar to Theorems 3 and 4. We can explicitly explore the relationship between the Hamiltonian Cycle and flex versions of Hamiltonian Path.

HC(G) $\implies$ FEHP(G). If HC(G) is true, then G contains a Hamiltonian cycle. Removing an edge which is part of the cycle breaks the Hamiltonian cycle into a Hamiltonian path. Removing any other edge has no effect on the cycle, so we can take any path contained in the cycle. Thus FEHC(G) is true.

FEHP(G) $\not\implies$ HC(G). Consider two cliques K_n and K_m with $n, m \geq 4$ which share a node. When any edge is removed, the graph will still contain a Hamiltonian path; however, the entire graph does not contain a Hamiltonian cycle.

HC(G) $\implies$ FVHP(G). Removing any vertex breaks the Hamiltonian Cycle in G into a Hamiltonian Path.

We conjecture that FVHP(G) $\implies$ HC(G) for all graphs with more than two nodes, but leave it as an open problem. This would imply the equilavence of FVHP and HC.

5 Flex Geometric Packing

We review several NP-hard packing problems and show that their flex versions are all NP-Hard. Packing means placing shapes into a target region, allowing translation and rotation, without overlap.

Definition 7 (PACKRIGHTTRIANGLESINTORECTANGLE (PRTR) [16])**.** *Given n right triangles and a rectangular R, is it possible to pack the triangles into the R?*

Definition 8 (2DBINPACKING (2BP)[17])**.** *Given a bin of fixed width W and height H, a cutoff T, and a list of n rectangular items $L = (a_1, a_2, \ldots, a_n)$, where each item a_i has width w_i and height h_i such that $0 < w_i \leq W$ and $0 < h_i \leq H$, can the n items be packed into at most T bins of dimension $W \times L$?*

Theorem 5. *flex*-PRTR *and flex*-2BP *are NP-hard.*

Proof. It is already known that PRTR and 2BP are NP-Hard [16, 17]. For both problems, we reduce from the non-flex version to the flex versions as follows. Given a set of input shapes S, insert a new shape t such that t is smaller than all the shapes in S.

If Packing(S) is true, then flex-Packing($S \cup t$) is true since we can always place t in place of the missing shape. If Packing(S) is false, then flex-Packing($S \cup t$) is false since packing will fail when t is removed. Thus flex-Packing($S \cup t$) = Packing(S), completing our reduction. □

This technique of inserting a small shape to the input set can be applied to show the hardness of flex versions of other NP-Hard geometric problems, but not to all. We plan to study more problems and develop new techniques.

6 One-Removal Geometric Packing

Definition 9 (2DSTRIPPACKING (2SP) [17])**.** *Given a strip of fixed width W and infinite height, and a list of n rectangular items $L = (a_1, a_2, \ldots, a_n)$, where each item a_i has width w_i and height h_i such that $0 < w_i \leq W$, the objective is allocate all the items to the strip and minimize the total height used.*

Recall the definition of one-removal: Given an optimization problem P, a set S, the solution for $P(S)$, and an element $x \in S$, find $P(S \setminus \{x\})$. To prove that the one-removal versions of or-2BP and or-2SP are NP-Hard we will use the following lemma, adapted as a special case from Böckenhauer *et al.* [10].

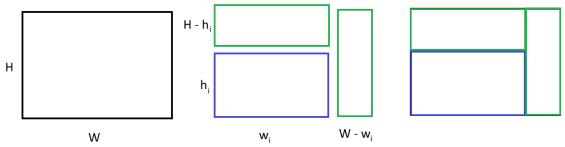
Lemma 1. *Let P be an NP-Hard optimization problem such that a deterministic algorithm can transform an efficiently solvable input instance E into any input S using a polynomial number of one-removal steps. Then or-P is NP-hard.*

Our contribution below is in constructing an efficiently solvable instance E which can be transformed into S using one-removals to satisfy the hypothesis of the lemma.

Theorem 6. *or*-2BP *is NP-Hard.*

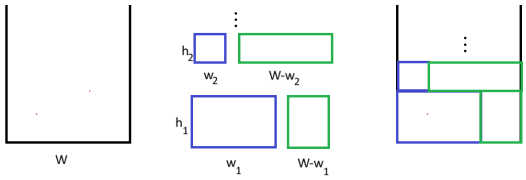
Proof. Let (L, W, H) be a problem instance where $L = (a_1, a_2, \ldots, a_n)$. Create an empty list L'. For each i, insert three rectangles into L', with dimensions $w_i \times h_i$, $(W - w_i) \times H$, and $w_i \times (H - h_i)$. This is efficiently solvable as drawn below. If we remove each $(W - w_i) \times H$ and

$w_i \times (H - h_i)$ rectangle one at a time from L', we recover the original instance L in $2n$ total steps. Thus, by Lemma 1 we have that or-2BP is NP-Hard. □



Theorem 7. *or-*2SP *is NP-Hard.*

Proof. Let (L, W) be a problem instance where $L = (a_1, a_2, \ldots, a_n)$. Create an empty list L'. For each i, insert two rectangles into L', with dimensions $w_i \times h_i$ and $(W - w_i) \times h_i$. This is efficiently solvable as drawn below. If we remove each $w_i \times (H - h_i)$ rectangle one at a time from L', we recover the original instance L in n total steps. Thus, by Lemma 1 we have that OR-2SP is NP-Hard. □



Theorems 6 and 7 can be extended to arbitrarily many dimensions using induction.

Remark 1. *While the added information of an optimal solution for* 2BP(S) *or* 2SP(S) *does not help us find a solution to the one-removal versions in polynomial time, they do give us useful approximations.* 2BP(S) *approximates* 2BP$(S \setminus x)$ *within 1 and* 2SP(S) *approximates* 2SP$(S \setminus x_i)$ *within* h_i.

We conclude in stating that this article is a work in progress towards a more unified theory of classifying NP-Hard problems by flexibility.

References

[1] Saeed Nozhati. A resilience-based framework for decision making based on simulation-optimization approach. *Structural Safety*, 89:102032, March 2021.

[2] Morteza Zare Oskouei, Hasan Mehrjerdi, Davood Babazadeh, Payam Teimourzadeh Baboli, Christian Becker, and Peter Palensky. Resilience-oriented operation of power systems: Hierarchical partitioning-based approach. *Applied Energy*, 312:118721, April 2022.

[3] Hans-Joachim Böckenhauer, Luca Forlizzi, Juraj Hromkovič, Joachim Kneis, Joachim Kupke, Guido Proietti, and Peter Widmayer. Reusing Optimal TSP Solutions for Locally Modified Input Instances. In *IFIP International Federation for Information Processing*, pages 251–270. Springer US, Boston, MA.

[4] Haris Angelidakis, Konstantin Makarychev, and Yury Makarychev. Algorithms for stable and perturbation-resilient problems. In *Proc. of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 438–451, New York, NY, June 2017.

[5] Jason I. Brown. A vertex critical graph without critical edges. *Discrete Mathematics*, 102(1):99–101, May 1992.

[6] Gabriel Andrew Dirac. Some Theorems on Abstract Graphs. *Proceedings of the London Mathematical Society*, s3-2(1):69–81, 1952.

[7] Henri Meyniel. Une condition suffisante d'existence d'un circuit hamiltonien dans un graphe oriente. *Journal of Combinatorial Theory, Series B*, 14(2):137–147, April 1973.

[8] Asaf Ferber, Rajko Nenadov, Andreas Noever, Ueli Peter, and Nemanja Škorić. Robust Hamiltonicity of random directed graphs. *Journal of Combinatorial Theory, Series B*, 126:1–23, September 2017.

[9] Markus W. Schäffter. Scheduling with forbidden sets. *Discrete Applied Mathematics*, 72(1):155–166, January 1997.

[10] Hans-Joachim Böckenhauer, Elisabet Burjons, Martin Raszyk, and Peter Rossmanith. Reoptimization of parameterized problems. *Acta Informatica*, 59(4):427–450, August 2022.

[11] Jaroslaw Byrka, Fabrizio Grandoni, Thomas Rothvoß, and Laura Sanità. An improved LP-based approximation for steiner tree. In *Proc. of the 42nd ACM symposium on Theory of computing*, pages 583–592, Cambridge, MA, June 2010.

[12] Claudia Archetti, Luca Bertazzi, and M. Grazia Speranza. Reoptimizing the traveling salesman problem. *Networks*, 42(3):154–159, 2003.

[13] Giorgio Ausiello, Bruno Escoffier, Jérôme Monnot, and Vangelis Paschos. Reoptimization of minimum and maximum traveling salesman's tours. *Journal of Discrete Algorithms*, 7(4):453–463, December 2009.

[14] Nicolas Boria, Jérôme Monnot, and Vangelis Th. Paschos. Reoptimization of maximum weight induced hereditary subgraph problems. *Theoretical Computer Science*, 514:61–74, November 2013.

[15] Jonathan King. Football And Tug-Of-War Conundrum, June 2024. Unpublished manuscript.

[16] Amy Chou. NP-Hard Triangle Packing Problems. *Research Science Institute summer program for Highschool students*, 2016.

[17] Andrea Lodi, Silvano Martello, and Michele Monaci. Two-dimensional packing problems: A survey. *European Journal of Operational Research*, 141(2):241–252, September 2002.

Enumerating Non-Crossing Hamiltonian Paths by Reachability Checks and Bidirectional Search

Randal Tuggle* Jack Snoeyink†

October 19, 2025

Abstract

Given a straight-line graph $\mathcal{G}$ embedded on a point set $P \subseteq \mathbb{R}^2$, we seek to enumerate all non-crossing Hamiltonian paths through $\mathcal{G}$. Standard backtracking algorithms extend a path from a fixed endpoint and prune infeasible extensions by checking which vertices are immediately **visible** from that endpoint, which can take $\Theta(|P|)$ per step. In place of visibility-based pruning, we introduce reachability-based pruning, which checks whether all vertices are **reachable** from a given end of the path under non-crossing constraints, at a per-step cost of $\mathcal{O}((|P| + |E(\mathcal{G})|) \cdot \alpha(|P|))$, where α is the inverse Ackermann function. Additionally, for both visibility and reachability-based pruning, we introduce a bidirectional search, which extends the path from either end to avoid early commitment to a fixed starting point. We define our reachability-based enumeration methods using a reverse search framework [7], which allows systematic generation of all paths using only polynomial space. We compare the search spaces for bidirectional and unidirectional strategies and demonstrate the practical advantages of our methods. Furthermore, we experimentally demonstrate that the stronger pruning power of reachability checks more than compensates for their additional computational overhead.

1 Introduction

Enumeration problems have been extensively studied in not only graph theory [14, 17, 21] but also computational geometry where enumeration problems often take as input a planar point set P (inducing a complete geometric graph). These often ask to list or count non-crossing structures such as Hamiltonian paths and cycles [1, 10, 23], spanning trees [8], and red-blue matchings [6], among many other non-crossing structures [2, 3, 4, 5, 7, 9, 11, 12, 13, 15, 16, 18, 19, 22]. Practical constraints often restrict allowable edges, motivating enumeration problems where the input is a straight-line embedding of a graph that is not necessarily complete [20].

In this paper, we explore enumerating non-crossing Hamiltonian paths through a straight-line graph embedding $\mathcal{G}$ on points P in the plane. That is, we seek a permutation π of the points P that defines a path with edges in the given graph that intersect only at shared endpoints. For the complete graph on P, the fastest known approach for enumerating all non-crossing Hamiltonian paths was given by Eppstein in 2023 [10]. He defines $\#\text{HAM}(P)$ to be the number of non-crossing Hamiltonian paths through the complete geometric graph with vertex set P, and $\#\text{PATH}(P)$ to be the total number of non-crossing paths (not necessarily Hamiltonian) through P. Eppstein's approach enumerates all non-crossing paths in $\mathcal{O}(|P| \cdot \#\text{PATH}(P))$ time and $\mathcal{O}(|P|^2)$ space by computing visibility polygons to identify visible but unvisited points that the current path can extend to; these extended paths become the path's children in a backtracking search tree. By proving that $\#\text{PATH}(P)$ is polynomially bounded by $\#\text{HAM}(P)$, he showed that this approach lists all non-crossing Hamiltonian paths in $|P| \cdot \#\text{HAM}(P)^{\mathcal{O}(1)}$ time. However, convex point sets show that the exponent is at least $\log_2 3 > 1.58$, since they have $\frac{|P|}{4} \cdot (3^{|P|-1} + 3)$ non-crossing paths, but only $|P| \cdot 2^{|P|-3}$ non-crossing Hamiltonian paths. Thus, if we want only the non-crossing Hamiltonian paths, then listing all non-crossing paths may incur significant unnecessary overhead.

*Department of Computer Science, UNC - Chapel Hill. E-mail: rtuggle@cs.edu
†Dept of Computer Science & SDSS, UNC - Chapel Hill. E-mail: snoeyink@cs.unc.edu

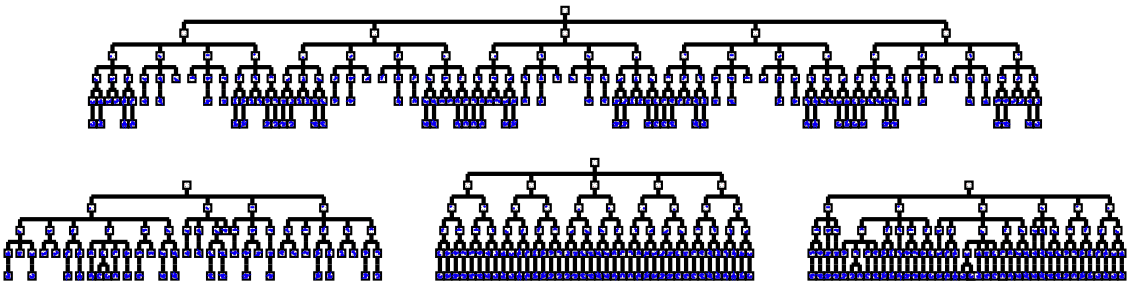


Figure 1: Search tree sizes for four algorithms on a complete graph of 5 vertices in convex position: unidirectional visibility (top with 206 nodes), bidirectional visibility (bottom left with 75 nodes, the only algorithm avoiding double-counting of a path and its reverse), unidirectional reachability (bottom middle with 116 nodes), and bidirectional reachability (bottom right with 105 nodes). For complete graphs in convex position, reachability trees have $\mathcal{O}(|P| \cdot 2^{|P|})$ nodes, while visibility trees have $\mathcal{O}(|P| \cdot 3^{|P|})$ nodes.

In place of Eppstein's visibility-based approach, we use a reachability-based approach and examine whether all vertices remain reachable under the non-crossing constraint before extending a partial path. Although reachability checks incur more work per path (checking edges rather than only points in $\mathcal{O}((|P| + |E(\mathcal{G})|) \cdot \alpha(|P|))$ time), this is offset by pruning many non-crossing paths that cannot be extended to a Hamiltonian path. In fact, for complete graphs in convex position, we reduce the exponent to 1. To support reachability checks, we store additional information ($\Theta(|P|+|E(\mathcal{G})|)$ space) for the current path, from which we can explore the tree of non-crossing paths using the reverse search technique of Avis and Fukuda [7].

For both approaches, we also introduce bidirectional search. By growing paths from both ends, our bidirectional search avoids committing to a single start vertex, which can be problematic in sparser graphs where many starting choices lead to dead ends, and it prunes the search tree more aggressively by considering only non-crossing paths that contain the initially chosen point. To foreshadow the experiments in Section 3, we provide Figure 1 to illustrate the relative size differences in the search trees for each approach. A visualization of the search trees can be found at https://rtuggle99.github.io/.

2 Search Techniques

We assume a fixed graph $\mathcal{G}$ embedded with vertex set $P \subseteq \mathbb{R}^2$ and edges represented as line segments. A path $\pi = [p_1, \ldots, p_\ell]$ consists of the directed edges $E(\pi) = \{\overrightarrow{p_1p_2}, \ldots, \overrightarrow{p_{\ell-1}p_\ell}\}$, directed from tail to head. Path operations include reverse of π, namely $\pi^{rev} = [p_\ell, \ldots, p_1]$, and adding to the tail or head: To add point p to the path $\pi = [p_1, \ldots, p_\ell]$ at the tail we use $p \circ_t \pi = [p, p_1, \ldots, p_\ell]$ or at the head we use $p \circ_h \pi = [p_1, \ldots, p_\ell, p]$. When π is empty, $p \circ_t \pi = p \circ_h \pi = [p]$. We'll use notation $\texttt{dir} \in \{h, t\}$ to denote directional operations, predicates, or variables (e.g., $\circ_{\texttt{dir}}$, etc.). Given a non-empty path π, we define $\pi_{\setminus \texttt{dir}}$ to be the subpath of π obtained by removing the point at the $\texttt{dir}$-end of π.

2.1 Visibility-based Searches

For the unidirectional visibility search, we follow the method of Eppstein [10] to define a tree such that the nodes are non-crossing paths. The root is the empty path. The parent of a non-empty path π is $\pi_{\setminus h}$. For each node π and vertex u not already in π, we have that $u \circ_h \pi$ is a child of π if $u \circ_h \pi$ is non-crossing. For the bidirectional search, we select a **pivot** point p and considers only non-crossing paths containing p. Starting from the single-point path $[p]$, the search extends the path from either end. In undirected graphs, every path corresponds to two permutations: π and its reverse π^{rev}. To avoid having to explore branches from both a path and its reverse in undirected graphs, we define a canonical form of a path as follows:

Definition 1. *The canonical form of a path π in an undirected graph is the orientation (either π or π^{rev}) where the head has a higher index than the tail.*

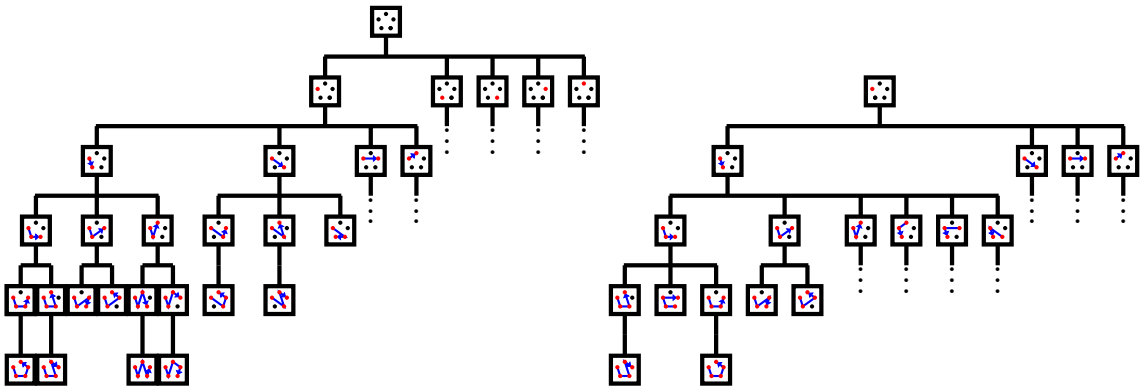


Figure 2: Unidirectional visibility search tree (left) and bidirectional visibility search tree (right) when $\mathcal{G}$ is the complete graph of 5 vertices in convex position.

If we define the children for a bidirectional search naively, a given path may have multiple parents. For example, if p is the pivot, the path $[a, p, c]$ can be reached by first adding a to the tail and then c to the head, or by adding c to the head and then a to the tail. To ensure no node has multiple parents, we define the search tree as follows: Nodes correspond to non-crossing paths (in canonical form if $\mathcal{G}$ is undirected) containing a pre-chosen pivot (by convention, we choose the vertex in $\mathcal{G}$ with the smallest degree to be the pivot). The root is the one-element path consisting of the pivot. For all other nodes π, the parent of π is (the canonical form of) $\pi_{\setminus h}$ if $\pi_{\setminus h}$ contains the pivot and (the canonical form of) $\pi_{\setminus t}$ otherwise.

2.2 Reachability-based Searches

In Figure 3, if we are restricted to extending only the head-end of π, then, since π splits the remaining vertices into several connected components (one generated by $\{q_1, q_2\}$ and another by $\{p_2\}$), we can disregard π as a valid path to extend since no non-crossing Hamiltonian path can contain π as a prefix. If, however, we are allowed to extend from both the head- and tail-end of π, the question becomes whether a non-crossing Hamiltonian path can contain π as a subpath. In this case, the head-end must extend through the component generated by the points $\{q_1, q_2\}$ immediately visible from the head-end, while the tail-end must extend through the component generated by the points $\{p_1, p_2, p_3, p_4\}$ immediately visible from the tail-end. The full version of this paper formalizes the idea of partitioning the vertices into sets according to the components they can reach and includes an $\mathcal{O}((|P| + |E(\mathcal{G})|) \cdot \alpha(|P|))$ test based on this idea to determine whether π partitions the vertices appropriately with respect to which ends of the path are available for extension. We only allow a path to be a node in the reachability-based search trees if it passes that reachability check.

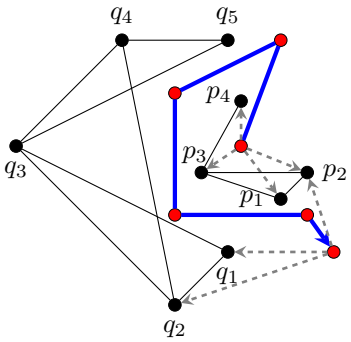


Figure 3: Suppose $\mathcal{G}$ is the complete graph on 15 vertices, embedded as shown. The blue path π partitions the remaining vertices of $\mathcal{G}$ into two components, $\{p_1, p_2, p_3, p_4\}$ and $\{q_1, q_2, q_3, q_4, q_5\}$. The head-end of π can reach either component, whereas the tail-end can reach only $\{p_1, p_2, p_3, p_4\}$.

We define trees similar to before, except, for this bidirectional search, we don't use a canonical form of a path. For the bidirectional search, same as before, we extend the tail for some period of time, and then we allow only the head to extend. To enable reverse search, we define a single, consistent rule for ordering the children of any nonempty path π: Give precedence to extensions that grow the tail-end of the current path π, and break ties using the radial ordering of the newly added point around the `dir`-end of π. (If the path π is empty, as is the case for the root of a unidirectional search, we can order its children by increasing x-coordinates, breaking ties with y).

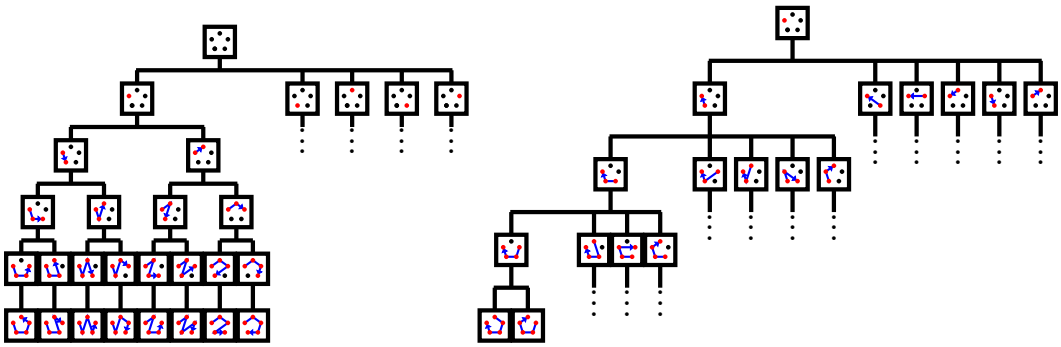


Figure 4: Unidirectional reachability search tree (left) and bidirectional reachability search tree (right) when $\mathcal{G}$ is the complete graph of 5 vertices in convex position.

3 Experiments

In this section, we explore the effectiveness of reachability pruning on both directed and undirected graphs. We select n vertices uniformly in the unit square, create the left-to-right Hamilton path, then add all other edges with probability θ. In the four approaches each search tree node is a non-crossing path; we estimate the fraction of leaves that are non-crossing Hamiltonian paths.

For each triple $((un)directed, \theta, n)$, we generate an embedding. Within each of the four search trees, we sample leaves with replacement by tracing randomly from the root 10,000 times. Trace i uses node degrees to record the probability p_i of reaching its leaf and the 0/1-indicator with $H_i = 1$ iff that leaf was Hamiltonian. We estimate the overall fraction of Hamiltonian leaves with the Horvitz–Thompson ratio: $\hat{f} = (\sum_i H_i/p_i)/(\sum_i 1/p_i)$.

Figure 5 has plots for undirected and directed graphs with several edge probabilities. Each plot shows four lines for the four search trees, with the number of vertices n on the x-axis and the fraction of Hamiltonian leaves on a log scale on the y-axis. The visibility approach has a hard time finding Hamiltonian paths; after $n = 10$, it couldn't find any Hamiltonian leaves within the 10,000 traces. In contrast, reachability pruning is quite effective at keeping the non-Hamiltonian leaves within a relatively small multiple of the Hamiltonian paths. We also note that in sparser graphs, the bidirectional approaches consistently outperform the unidirectional approaches.

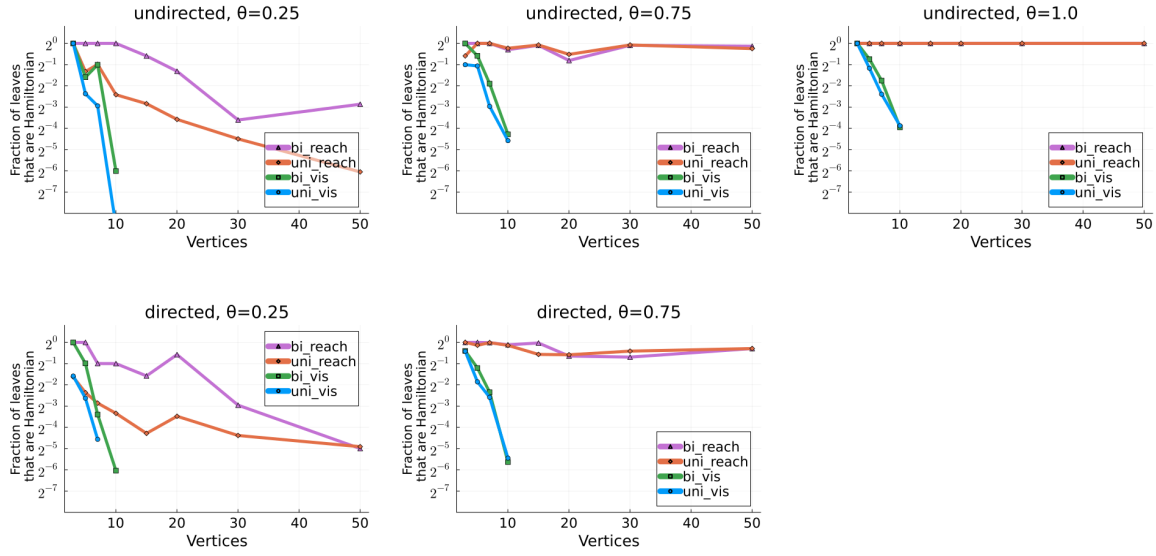


Figure 5: For a given number of vertices $n \in \{3, 5, 7, 10, 15, 20, 30, 50\}$ and edge probability $\theta \in \{0.25, 0.75, 1.0\}$ we generate a random directed or undirected graph. For each of the four search approaches nodes represent non-crossing paths; we plot, on a logarithmic y-axis, the fraction of leaves that are Hamiltonian paths.

References

[1] Oswin Aichholzer, Franz Aurenhammer, Ferran Hurtado, and Heike Krasser. Towards counting the crossing-free geometric graphs. *Theoretical Computer Science*, 233(1-2):17–33, 2001.

[2] Miklós Ajtai, Vašek Chvátal, Monroe M. Newborn, and Endre Szemerédi. Crossing-free subgraphs. In Alexander Rosa, Gert Sabidussi, and Jean Turgeon, editors, *Theory and Practice of Combinatorics: A collection of articles honoring Anton Kotzig on the occasion of his sixtieth birthday*, volume 12 of *Annals of Discrete Mathematics*, pages 9–12. North-Holland, 1982.

[3] Noga Alon, Sridhar Rajagopalan, and Subhash Suri. Long non-crossing configurations in the plane. *Fundamenta Informaticae*, 22(4):385–394, 1995.

[4] Victor Alvarez, Karl Bringmann, Radu Curticapean, and Saurabh Ray. Counting triangulations and other crossing-free structures via onion layers. *Discrete & Computational Geometry*, 53(4):675–690, 2015.

[5] Victor Alvarez and Raimund Seidel. A simple aggregative algorithm for counting triangulations of planar point sets and related problems. In *Proceedings of the Twenty-Ninth Annual Symposium on Computational Geometry*, SoCG '13, page 1–8, New York, NY, USA, 2013. Association for Computing Machinery.

[6] Andrzej Asinowski, Stefan Bereg, Seok Choi, and Chan-Su Shin. Upper and lower bounds for non-crossing bichromatic matchings. *Discrete & Computational Geometry*, 44(4):663–678, 2010.

[7] David Avis and Komei Fukuda. Reverse search for enumeration. *Discrete Applied Mathematics*, 65(1-3):21–46, 1996.

[8] Kevin Buchin, Alexander Schulz, and Barbara Vogtenhuber. On the enumeration of plane spanning trees. *Computational Geometry*, 44(4):175–189, 2011.

[9] Gi-Sang Cheon, Hong Joon Choi, Guillermo Esteban, and Minho Song. Enumeration of bipartite non-crossing geometric graphs. *Discrete Applied Mathematics*, 317:86–100, 2022.

[10] David Eppstein. Non-crossing hamiltonian paths and cycles in output-polynomial time. *arXiv preprint arXiv:2303.00147*, 2023.

[11] Philippe Flajolet and Marc Noy. Analytic combinatorics of non-crossing configurations. *Discrete Mathematics*, 204(1-3):203–229, 1999.

[12] Alfredo García, Marc Noy, and Javier Tejel. Lower bounds on the number of crossing-free subgraphs of k_n. *Computational Geometry: Theory and Applications*, 16(4):211–221, 2000.

[13] Michael Hoffmann, André Schulz, Micha Sharir, Adam Sheffer, Csaba D. Tóth, and Emo Welzl. Counting plane graphs: flippability and its applications. In János Pach, editor, *Thirty Essays on Geometric Graph Theory*, pages 303–325. Springer, 2013.

[14] Donald B Johnson. Finding all the elementary circuits of a directed graph. *SIAM Journal on Computing*, 4(1):77–84, 1975.

[15] Dániel Marx and Tillmann Miltzow. Peeling and nibbling the cactus: subexponential-time algorithms for counting triangulations and related problems. In *32nd International Symposium on Computational Geometry (SoCG)*, volume 51 of *LIPIcs*, pages 52:1–52:16. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2016.

[16] Andreas Razen and Emo Welzl. Counting plane graphs with exponential speed-up. In Cristian S. Calude, Grzegorz Rozenberg, and Arto Salomaa, editors, *Rainbow of Computer Science*, volume 6570 of *Lecture Notes in Computer Science*, pages 36–46. Springer, 2011.

[17] Ronald C Read and Robert E Tarjan. Bounds on backtrack algorithms for listing cycles, paths, and spanning trees. *Networks*, 5(3):237–252, 1975.

[18] Micha Sharir and Adam Sheffer. Counting plane graphs: cross-graph charging schemes. *Combinatorics, Probability and Computing*, 22(6):935–954, 2013.

[19] Micha Sharir, Adam Sheffer, and Emo Welzl. Counting plane graphs: perfect matchings, spanning cycles, and kasteleyn's technique. *Journal of Combinatorial Theory, Series A*, 120(4):777–794, 2013.

[20] Shin-ichi Tanigawa. Enumeration of non-crossing geometric graphs. In Ming-Yang Kao, editor, *Encyclopedia of Algorithms*, pages 638–640. Springer New York, New York, NY, 2016.

[21] Takeaki Uno. Algorithms for enumerating all perfect, maximum and maximal matchings in bipartite graphs. In *International Symposium on Algorithms and Computation*, pages 92–101. Springer, 2001.

[22] Manuel Wettstein. Counting and enumerating crossing-free geometric graphs. In *SoCG '14 : proceedings of the thirtieth annual Symposium on Computational Geometry : June 8-11, 2014, Kyoto, Japan*, pages 1 – 10, New York, NY, 2014. Association for Computing Machinery. 30th Annual Symposium on Computational Geometry (SOCG'14); Conference Location: Kyoto, Japan; Conference Date: June 8-11, 2014.

[23] Katsuhisa Yamanaka, David Avis, Takashi Horiyama, Yoshio Okamoto, Ryuhei Uehara, and Tanami Yamauchi. Algorithmic enumeration of surrounding polygons. *Discret. Appl. Math.*, 303:305–313, 2020.

Subquadratic Approximation Algorithms for Separating Two Points with Objects in the Plane

Jayson Lynch

Jack Spalding-Jamieson

Abstract

The (unweighted) **point-separation** problem asks, given a pair of points s and t in the plane, and a set of candidate geometric objects, for the minimum-size subset of objects whose union blocks all paths from s to t. Recent work has shown that the point-separation problem can be characterized as a type of shortest-path problem in a geometric intersection graph within a special lifted space. However, all known solutions to this problem essentially reduce to some form of APSP, and hence take at least quadratic time, even for special object types.

In this work, we consider the unweighted form of the problem, for which we devise subquadratic approximation algorithms for many special cases of objects, including line segments and disks. In this paradigm, we are able to devise algorithms that are fundamentally different from the APSP-based approach. In particular, we will give Monte Carlo randomized additive $+1$ approximation algorithms running in $\widetilde{\mathcal{O}}(n^{\frac{3}{2}})$ time for disks, axis-aligned line segments and rectangles, and $\widetilde{\mathcal{O}}(n^{\frac{11}{6}})$ time for line segments and constant-complexity convex polygons. We will also give deterministic multiplicative-additive approximation algorithms that, for any value $\varepsilon > 0$, guarantee a solution of size $(1+\varepsilon)\mathrm{OPT}+1$ while running in $\widetilde{\mathcal{O}}\left(\frac{n}{\varepsilon^2}\right)$ time for disks, axis-aligned line segments and rectangles, and $\widetilde{\mathcal{O}}\left(\frac{n^{4/3}}{\varepsilon^2}\right)$ time for line segments and constant-complexity convex polygons.

1 Introduction

The **point-separation** problem asks whether two points in the plane can be separated by a small number of objects. Specifically, we are given two points s and t, and a set of (weighted) candidate geometric objects $\mathcal{C}$ in the plane. Then, we wish to compute the minimum-weight subset of $\mathcal{C}$ **separating** s and t. That is, we ask for the minimum-weight subset $C \subset \mathcal{C}$ so that if we subtract every element of C from the plane, s and t then lie in different connected components. An example of this problem can be found in Figure 1.

Algorithmically, point-separation has been studied by a number of works, with a few different computational models [GKV11, CG16, CM18, KLSS21, KLS+22, SJN25a]. The most general model is what Spalding-Jamieson and Naredla [SJN25a] call the **oracle model**. In the oracle model, it is assumed that each object c is assigned a **canonical point** $x_c \in c$. Then, it is assumed that there is an $O(1)$-time oracle to answer the following form of query:

- For a pair of objects $c, c' \in \mathcal{C}$, the oracle must be able to determine if the union $c \cup c'$ contains a path from x_c to $x_{c'}$ that crosses the line segment $\overline{st}$ an even number of times, and (separately) one that crosses the line segment $\overline{st}$ an odd number of times.

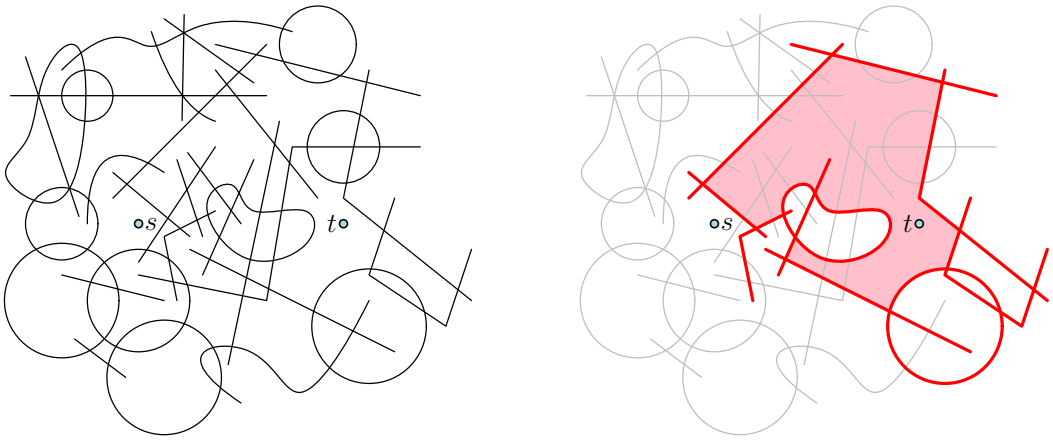


Figure 1: An instance of the (s,t) point-separation problem with objects that are simple curves, and a candidate (non-optimal) set of objects that separate s and t.[0]

Note that these operations are quite simple, and apply to a wide variety of object types, including disks, line segments, and even constant-complexity polygons and polygonal regions. A form of this model was first used by Cabello and Giannopoulos [CG16], who gave an exact polynomial-time algorithm for point-separation under this model.

More recently, Spalding-Jamieson and Naredla [SJN25a] showed that point-separation can be reduced to computing the minimum of a linear number of (unconstrained) shortest-path distances in a special **geometric intersection graph**, which is a graph whose vertices correspond to geometric objects and whose edges correspond to pairwise intersections. However, the vertices used for their graph are not the input objects of the point-separation problem, which are embedded in the plane. Rather, in their graph, the geometric objects are embedded in the "homology cover space"[1]. The geometric objects in the homology cover space they used come in pairs, each induced by a single planar object in $\mathcal{C}$. These are the *lifted* objects. The shortest-path distances they compute are then exactly the distance between the two elements of each pair. This approach is similar to a construction that has been used for maximum flow on surface graphs [CEFN23].

Spalding-Jamieson and Naredla also gave fine-grained (conditional) lower bounds on point-separation for many classes of objects, including $\Omega\left(n^{2-\varepsilon}\right)$ lower bounds for many classes of *weighted* objects, and $\Omega\left(n^{\frac{3}{2}-\varepsilon}\right)$ lower bounds for many classes of *unweighted* objects. Notably, both bounds apply for line segment objects. However, it remains open to determine if there are any non-trivial classes of objects for which point-separation can be solved in truly subquadratic time. As a step towards this goal, we present several subquadratic approximation algorithms.

1.1 Our Results

We focus on the unweighted point-separation problem. Our results are given in Table 1. The results in this paper roughly decompose into two components.

The first set of results assumes the existence of single-source shortest-path algorithms within

[0] Figure originally from Spalding-Jamieson and Naredla [SJN25a, SJN25b].

[1] For completeness: the homology cover is specifically a one-dimensional $\mathbb{Z}_2$-homology cover of the (extended) plane with small holes at s and t. The extended plane with these holes is homeomorphic to an annulus, so the homology cover space is equivalent to the non-trivial $\mathbb{Z}_2$-torsor over the annulus. That said, the steps of the reduction will be framed in elementary terms requiring no background in topology.

[2] Convex polygons and rectangles can either include or exclude their interiors (and mixing the two types is okay).

Object Type	Running Time	Randomized?
Approximation Guarantee: OPT + 1		
Disk	$\mathcal{O}(n^{3/2}(\log n)^2)$	Monte Carlo (w.h.p.)
Line segment	$\widetilde{\mathcal{O}}(n^{11/6})$	Monte Carlo (w.h.p.)
Rectilinear line segment	$\mathcal{O}(n^{3/2}(\log n)^2)$	Monte Carlo (w.h.p.)
$\mathcal{O}(1)$-complexity convex polygon[2]	$\widetilde{\mathcal{O}}(n^{11/6})$	Monte Carlo (w.h.p.)
Axis-aligned rectangle[2]	$\mathcal{O}(n^{3/2}(\log n)^2)$	Monte Carlo (w.h.p.)
Approximation Guarantee: $(1+\varepsilon)$ OPT + 1		
Disk	$\mathcal{O}\left(\frac{n(\log n)^3}{\varepsilon^2}\right)$	Deterministic
Line segment	$\widetilde{\mathcal{O}}\left(\frac{n^{4/3}}{\varepsilon^2}\right)$	Deterministic
Rectilinear line segment	$\mathcal{O}\left(\frac{n(\log n)^3}{\varepsilon^2}\right)$	Deterministic
$\mathcal{O}(1)$-complexity convex polygon[2]	$\widetilde{\mathcal{O}}\left(\frac{n^{4/3}}{\varepsilon^2}\right)$	Deterministic
Axis-aligned rectangle[2]	$\mathcal{O}\left(\frac{n(\log n)^3}{\varepsilon^2}\right)$	Deterministic
Approximation Guarantee: OPT + k		
Disk	$\mathcal{O}\left(\frac{n^2 \log n}{k}\right)$	Deterministic
Line segment	$\widetilde{\mathcal{O}}\left(\frac{n^{7/3}}{k}\right)$	Deterministic
Rectilinear line segment	$\mathcal{O}\left(\frac{n^2 \log n}{k}\right)$	Deterministic
$\mathcal{O}(1)$-complexity polyline	$\widetilde{\mathcal{O}}\left(\frac{n^{7/3}}{k}\right)$	Deterministic
$\mathcal{O}(1)$-complexity rectilinear polyline	$\mathcal{O}\left(\frac{n^2 \log n}{k}\right)$	Deterministic

Table 1: Approximation algorithms for the point-separation problem with restricted objects.

the geometric intersection graph in the homology cover. There are three such results, each of which provides a different approximation guarantee using a different method, resulting in the three sections of the table. Two of these methods further require the objects to be convex *if they intersect the line segment* $\overline{st}$.

The second set of results provide these fast single-source shortest-path algorithms within geometric intersection graphs in the homology cover. These use different geometric tools depending on the class of objects.

2 Methods

Our algorithms rely on three main components: 1) Casting the point separation problem as a shortest path problem in the geometric intersection graphs in the homology cover. 2) Approximating the minimum shortest path with a small number of exact shortest path queries. 3) Calculating exact shortest paths quickly in the homology cover using geometric properties.

2.1 Homology Cover

The geometric intersection graph in the homology cover can be viewed as a lifted form of the geometric intersection graph of $\mathcal{C}$ in the plane using canonical points and the segment $\overline{st}$. Two

vertices are created for each object. One is given a label of -1, and the other is given a label of 1. When the path between the selected canonical points passes through $\overline{st}$ an even number of times, edges between the lifted vertices of the same label are created. If the path passes through $\overline{st}$ an odd number of times, edges between the lifted vertices of opposite labels are created. See Figure 2 for an example. We call this graph $\overline{G}$, and we assign its vertices weights according to the weights of the objects.

2.2 Fast Shortest-Paths Queries

Recent work has yielded faster shortest path algorithms in a variety of geometric intersection graphs: Disks [dBC25a, dBC25b], $\mathcal{O}(1)$-complexity polylines [SJN25a], and $\mathcal{O}(1)$-complexity rectilinear polylines [GK07, GK09, Ble08]. We are able to strengthen these techniques to also apply them to $\overline{G}$. This forms a key sub-routine for our algorithms that we will apply as a black-box.

2.3 Shortest-Path Approximation

Spalding-Jamieson and Naredla [SJN25a] showed that the optimal solution to the point-separation problem corresponds to the shortest-path between a pair of vertices in $\overline{G}$ created from the same object, and that the objects in the optimal solution correspond exactly to the vertices along this path. They use this directly to give APSP-based approaches. Our algorithms instead use several different tools to obtain approximations of the global minimum:

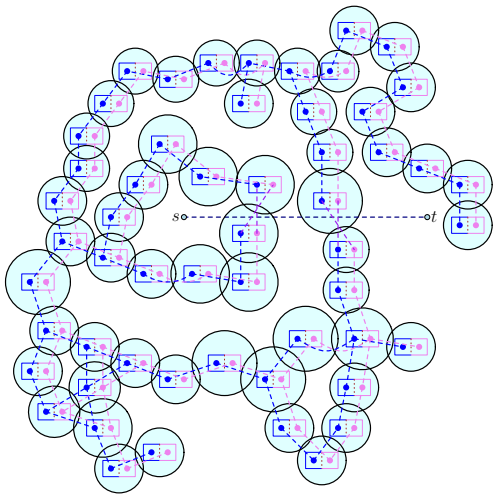


Figure 2: The intersection graph in the homology cover of a collection of disks. Vertices labelled -1 and 1 are blue and pink, respectively. Blue vertices and edges, and pink vertices and edges both appear as separate geometric intersection graphs for the disks, except for edges crossing the line segment $\overline{st}$ where they swap.

- **Random sampling**. One interesting feature of $\overline{G}$ is that all shortest-paths have complementary shortest-paths that swap all the labels along the path. As a consequence, it can be shown that the minimum shortest-path between a pair of vertices is actually a cycle, and it can be found by performing the shortest-path computation for *any* pair of complementary vertices along this cycle. Hence, if the optimal solution is sufficiently large, it can be found by random sampling.
- **Upper and lower bounds for path-lengths**. An important feature of any graph is that shortest-path lengths for a given pair of vertices also provide upper and lower bounds for shortest-paths of nearby vertices via the triangle inequality. We use lower bounds to reduce the search space of a special subroutine, and we use upper bounds to provide running time guarantees.
- **Divide and Conquer**. We use divide and conquer approaches to reduce the search space of the global point-separation problem. This kind of approach uses a special subroutine that produces a 1-hop path separator of sorts, with lower bounds for the length of every shortest-path that uses a vertex in the separator. This entire divide and conquer approach is inspired by Reif's algorithm for minimum (s,t)-cut in a planar graph [Rei83].

References

[Ble08] Guy E Blelloch. Space-efficient dynamic orthogonal point location, segment intersection, and range reporting. In *SODA*, volume 8, pages 894–903, 2008.

[CEFN23] Erin W. Chambers, Jeff Erickson, Kyle Fox, and Amir Nayyeri. Minimum cuts in surface graphs. *SIAM J. Comput.*, 52(1):156–195, 2023.

[CG16] Sergio Cabello and Panos Giannopoulos. The complexity of separating points in the plane. *Algorithmica*, 74(2):643–663, 2016.

[CM18] Sergio Cabello and Lazar Milinković. Two optimization problems for unit disks. *Comput. Geom.*, 70-71:1–12, 2018.

[dBC25a] Mark de Berg and Sergio Cabello. An $o(n \log n)$ algorithm for single-source shortest paths in disk graphs. *33rd Annual European Symposium on Algorithms (ESA 2025)*, 2025. To appear.

[dBC25b] Mark de Berg and Sergio Cabello. An $o(n \log n)$ algorithm for single-source shortest paths in disk graphs, 2025.

[GK07] Yoav Giyora and Haim Kaplan. Optimal dynamic vertical ray shooting in rectilinear planar subdivisions. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 19–28, 2007.

[GK09] Yoav Giyora and Haim Kaplan. Optimal dynamic vertical ray shooting in rectilinear planar subdivisions. *ACM Transactions on Algorithms (TALG)*, 5(3):1–51, 2009.

[GKV11] Matt Gibson, Gaurav Kanade, and Kasturi R. Varadarajan. On isolating points using disks. In Camil Demetrescu and Magnús M. Halldórsson, editors, *Algorithms - ESA 2011 - 19th Annual European Symposium, Saarbrücken, Germany, September 5-9, 2011. Proceedings*, volume 6942 of *Lecture Notes in Computer Science*, pages 61–69. Springer, 2011.

[KLS+22] Neeraj Kumar, Daniel Lokshtanov, Saket Saurabh, Subhash Suri, and Jie Xue. Point separation and obstacle removal by finding and hitting odd cycles. In Xavier Goaoc and Michael Kerber, editors, *38th International Symposium on Computational Geometry, SoCG 2022, June 7-10, 2022, Berlin, Germany*, volume 224 of *LIPIcs*, pages 52:1–52:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

[KLSS21] Neeraj Kumar, Daniel Lokshtanov, Saket Saurabh, and Subhash Suri. A constant factor approximation for navigating through connected obstacles in the plane. In Dániel Marx, editor, *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 822–839. SIAM, 2021.

[Rei83] John H Reif. Minimum s-t cut of a planar undirected network in $o(n \log^2(n))$ time. *SIAM Journal on Computing*, 12(1):71–81, 1983.

[SJN25a] Jack Spalding-Jamieson and Anurag Murty Naredla. Separating two points with obstacles in the plane: Improved upper and lower bounds. *33rd Annual European Symposium on Algorithms (ESA 2025)*, 2025. To appear.

[SJN25b] Jack Spalding-Jamieson and Anurag Murty Naredla. Separating two points with obstacles in the plane: Improved upper and lower bounds, 2025.

A draft of the full paper is appended to this document.

Subquadratic Approximation Algorithms for Separating Two Points with Objects in the Plane

Jayson Lynch

Jack Spalding-Jamieson

Abstract

The (unweighted) **point-separation** problem asks, given a pair of points s and t in the plane, and a set of candidate geometric objects, for the minimum-size subset of objects whose union blocks all paths from s to t. Recent work has shown that the point-separation problem can be characterized as a type of shortest-path problem in a geometric intersection graph within a special lifted space. However, all known solutions to this problem essentially reduce to some form of APSP, and hence take at least quadratic time, even for special object types.

In this work, we consider the unweighted form of the problem, for which we devise subquadratic approximation algorithms for many special cases of objects, including line segments and disks. In this paradigm, we are able to devise algorithms that are fundamentally different from the APSP-based approach. In particular, we will give Monte Carlo randomized additive $+1$ approximation algorithms running in $\widetilde{\mathcal{O}}(n^{\frac{3}{2}})$ time for disks, axis-aligned line segments and rectangles, and $\widetilde{\mathcal{O}}(n^{\frac{11}{6}})$ time for line segments and constant-complexity convex polygons. We will also give deterministic multiplicative-additive approximation algorithms that, for any value $\varepsilon > 0$, guarantee a solution of size $(1+\varepsilon)\mathrm{OPT}+1$ while running in $\widetilde{\mathcal{O}}\left(\frac{n}{\varepsilon^2}\right)$ time for disks, axis-aligned line segments and rectangles, and $\widetilde{\mathcal{O}}\left(\frac{n^{4/3}}{\varepsilon^2}\right)$ time for line segments and constant-complexity convex polygons.

1 Introduction

The **point-separation** problem asks whether two points in the plane can be separated by a small number of objects. Specifically, we are given two points s and t, and a set of (weighted) candidate geometric objects $\mathcal{C}$ in the plane. Then, we wish to compute the minimum-weight subset of $\mathcal{C}$ **separating** s and t. That is, we ask for the minimum-weight subset $C \subset \mathcal{C}$ so that if we subtract every element of C from the plane, s and t then lie in different connected components. An example of this problem can be found in Figure 1.

Geometric intersection graphs are graphs whose vertices represent geometric objects and whose edges are exactly intersections between those objects. Shortest-path problems in geometric intersection graphs have garnered significant interest [CS16, CS17, Skr18, CS19, WX20, BKBK+22, WZ22, WZ23, BW24, CGL24, DKP24, HHZ24, AOX25, CH25, BWB25, dBC25a, CCG+25]. The point-separation problem has recently also been shown to be a form of shortest-path problem in a geometric intersection graph [SJN25a], although not in the plane but rather in a "lifted" space.

Point-separation captures a number of real-world scenarios, particularly problems of detecting or preventing entrances/exits from a region. For instance, if you wanted to install radar sensors (each able to detect within a fixed radius) around an isolated facility to detect any unexpected visitors approaching the facility, and you had a number of candidate installation locations based

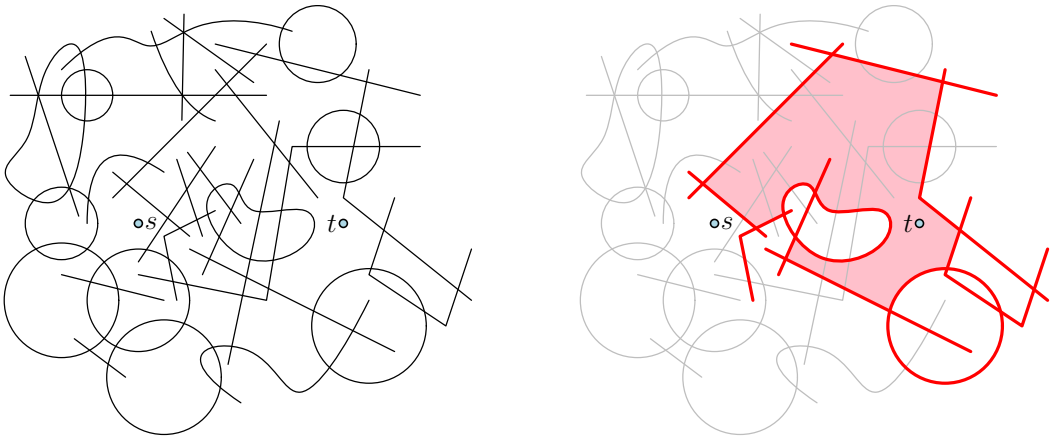


Figure 1: An instance of the (s, t) point-separation problem with objects that are simple curves, and a candidate (non-optimal) set of objects that separate s and t.[0]

on the terrain and foliage that would be protected from wildlife: Radar sensors can be quite expensive, so you'd like to minimize the number of them required to guarantee detection of anyone who approaches. The point s is your facility, the point t is at infinity (or some point sufficiently far away), and each candidate radar location becomes a disk object with the given radius.

An algorithm for point-separation also forms a sub-routine in a constant approximation-algorithm for the so-called "barrier resilience" problem [KLSS21].

Algorithmically, point-separation has been studied by a number of works, with a few different computational models [GKV11, CG16, CM18, KLSS21, KLS+22, SJN25a]. Essentially the most general model is what Spalding-Jamieson and Naredla [SJN25a] call the "oracle model", which assumes the existence of an oracle with the following properties:

- Assume that each object c has a **canonical point** $x_c \in c$ (arbitrary).
- For a pair of objects $c, c' \in \mathcal{C}$, the oracle must be able to determine if the union $c \cup c'$ contains a path from x_c to $x_{c'}$ that crosses the line segment $\overline{st}$ an even number of times, and (separately) one that crosses the line segment $\overline{st}$ an odd number of times.
- Queries to this oracle are assumed to take $\mathcal{O}(1)$ time.

Note that these operations are quite simple, and apply to a wide variety of object types, including disks, line segments, and even constant-size polygons. This model was first used by Cabello and Giannopoulos [CG16] in a slightly different form than what is presented here. In particular, they gave an exact polynomial-time algorithm for point-separation under this model. Essentially, they show that point-separation can be reduced to a linear number of "constrained" shortest-path computations in the geometric intersection graph of $\mathcal{C}$. This isn't quite analogous to the "unconstrained" shortest-path problems above, but it was an important step.

More recently, Spalding-Jamieson and Naredla [SJN25a] essentially showed that point-separation can be reduced to computing the minimum of a linear number of (unconstrained) shortest-path distances in a slightly different geometric intersection graph. Specifically, the geometric objects are embedded in a space called the "homology cover"[1], and the objects themselves come in pairs, each

[0] Figure originally from Spalding-Jamieson and Naredla [SJN25a, SJN25b].

[1] The "homology cover" is specifically a one-dimensional Z_2-homology cover of the (extended) plane with small holes at s and t. In our case, this space is homeomorphic to an annulus. That said, the steps of the reduction will be framed in elementary terms requiring no background in topology.

induced by a single planar object in $\mathcal{C}$. The distances to compute are then exactly the distance between the two elements of each pair. This approach is similar to a construction that has been used for maximum flow on surface graphs [CEFN23]. We will give a complete description of their framework for our purposes in the next section.

It can be shown that the most general form of Spalding-Jamieson and Naredla's algorithm is not fundamentally different from that of Cabello and Giannopoulos. In fact, the constrained shortest-path problem used by Cabello and Giannopoulos can be reduced to an unconstrained shortest-path problem in a graph equivalent to the one used by Spalding-Jamieson and Naredla. However, the key difference is that the reduction of Spalding-Jamieson and Naredla specifically induces a geometric representation of this graph (in the homology cover), and hence many of the methods for shortest-path problems on geometric intersection graphs can be applied with some modification. This results in faster algorithms for many classes of objects, including disks and line segments. That said, all the algorithms of Spalding-Jamieson and Naredla (and earlier work) essentially reduce to the all-pairs shortest-paths (APSP) problem, and so there is a methodological quadratic lower bound based on the output size of APSP (up to some sub-polynomial improvements from word-packing tricks).

Spalding-Jamieson and Naredla also gave fine-grained (conditional) lower bounds on point-separation for many classes of objects, including $\Omega\left(n^{2-\varepsilon}\right)$ lower bounds for many classes of *weighted* objects, and $\Omega\left(n^{\frac{3}{2}-\varepsilon}\right)$ lower bounds for many classes of *unweighted* objects. Notably, both bounds apply for line segments objects. However, it remains open to determine if there are any non-trivial classes of objects for which point-separation can be solved in truly subquadratic time. As a step towards this goal, we present several subquadratic approximation algorithms.

1.1 Related Work

There are a number of problems related to point-separation. In their recent work, Spalding-Jamieson and Naredla give a comprehensive history of these problems [SJN25b, Appendix A], including the hardness of generalizations. We will refrain from detailed discussion here, but we will briefly discuss one important piece of inspiration for our work: Algorithms for minimum (s,t)-cut in an undirected planar graph.

The minimum (s,t)-cut problem in an undirected planar graph asks, given a plane graph G and two vertices s and t, for the minimum set (or weighted set) of edges whose removal separates s and t. Note that point-separation generalizes this problem via non-crossing line segment objects. When s and t are both on the same face, Ford and Fulkerson [FJF56, Section 3] show that the minimum (s,t)-cut problem in a planar graph reduces to a single shortest-path problem in the dual graph of $G \cup \{e\}$ for a single extra edge e. Itai and Shiloach [IS79] generalized this approach to all planar graphs by solving a set of similar shortest-path problems in a generalization of this modified dual graph. Notably, they might need to compute up to a linear number of shortest-paths with their approach: Each of these computations is performed for an element of a dual shortest-path between chosen faces incident to s and t. Hence, if this dual shortest-path is long, the runtime of their algorithm is slow. Reif [Rei83] improved on this approach by employing a divide and conquer scheme. Essentially, by observing that (save for ties) none of these shortest-paths can intersect, Reif was able to recover a global optimum in near-linear time. At a high-level, Reif computes a shortest-path for a "middle" vertex that is far from both s and t, and then deletes all vertices along this path, disconnecting the graph into two independent subproblems in which the new values of s and t are twice as close together.

The time complexity of Reif's algorithm has been since been surpassed, but we have mentioned these results because they are analogous to the progress we will present on point-separation. Essen-

Object Type	Running Time	Randomized?
Approximation Guarantee: OPT + 1 (Section 4)		
Disk	$\mathcal{O}(n^{3/2}(\log n)^2)$	Monte Carlo (w.h.p.)
Line segment	$\widetilde{\mathcal{O}}(n^{11/6})$	Monte Carlo (w.h.p.)
Rectilinear line segment	$\mathcal{O}(n^{3/2}(\log n)^2)$	Monte Carlo (w.h.p.)
$\mathcal{O}(1)$-complexity convex polygon[2]	$\widetilde{\mathcal{O}}(n^{11/6})$	Monte Carlo (w.h.p.)
Axis-aligned rectangle[2]	$\mathcal{O}(n^{3/2}(\log n)^2)$	Monte Carlo (w.h.p.)
Approximation Guarantee: (1 + ε) OPT + 1 (Section 5)		
Disk	$\mathcal{O}\left(\frac{n(\log n)^3}{\varepsilon^2}\right)$	Deterministic
Line segment	$\widetilde{\mathcal{O}}\left(\frac{n^{4/3}}{\varepsilon^2}\right)$	Deterministic
Rectilinear line segment	$\mathcal{O}\left(\frac{n(\log n)^3}{\varepsilon^2}\right)$	Deterministic
$\mathcal{O}(1)$-complexity convex polygon[2]	$\widetilde{\mathcal{O}}\left(\frac{n^{4/3}}{\varepsilon^2}\right)$	Deterministic
Axis-aligned rectangle[2]	$\mathcal{O}\left(\frac{n(\log n)^3}{\varepsilon^2}\right)$	Deterministic
Approximation Guarantee: OPT + k (Section 6)		
Disk	$\mathcal{O}\left(\frac{n^2 \log n}{k}\right)$	Deterministic
Line segment	$\widetilde{\mathcal{O}}\left(\frac{n^{7/3}}{k}\right)$	Deterministic
Rectilinear line segment	$\mathcal{O}\left(\frac{n^2 \log n}{k}\right)$	Deterministic
$\mathcal{O}(1)$-complexity polyline	$\widetilde{\mathcal{O}}\left(\frac{n^{7/3}}{k}\right)$	Deterministic
$\mathcal{O}(1)$-complexity rectilinear polyline	$\mathcal{O}\left(\frac{n^2 \log n}{k}\right)$	Deterministic

Table 1: Approximation algorithms for the point-separation problem with restricted objects.

tially, the algorithms of all previous approaches, including the recent results of Spalding-Jamieson and Naredla [SJN25a], are analogous to the methods of Itai and Shiloach, in that they solve a shortest-path problem for a large number of vertices. That is, they involve computing a large number of shortest-paths within a static graph. In contrast, our two most interesting results will instead both be analogous to Reif's algorithm: We will employ a divide and conquer that finds a "path" of objects in a certain graph at each level (but *not* necessarily a shortest-path in our case), and then deletes the path *along with its* 1*-neighbourhood*. By choosing the path to pass through one particular object in the "middle" of s and t, they will become twice as close, in a sense. In order for this closeness metric to work, we will require that certain objects be convex.

1.2 Our Results

Our results are given in Table 1. The results in this paper roughly decompose into two components. Section 7 gives fast single-source shortest-path algorithms within geometric intersection graphs in the homology cover. Sections 4 to 6 give approximation schemes for the point-separation problem with convex objects which rely on many calls to a single-source shortest-path oracle. In particular, Section 4 gives a randomized scheme for additive +1 approximations, Section 5 gives a deterministic multiplicative-additive approximation-scheme (arbitrary multiplicative factor, additive +1 term),

[2]Convex polygons and rectangles can include or exclude their interiors (mixing the two types is okay).

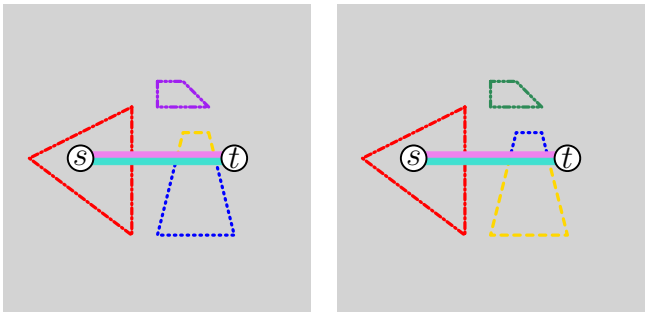


Figure 2: The "portal" construction of the homology cover. Each colour (or dot/dash pattern) is a single closed curve in the homology cover, and all of the curves are disjoint. The two solid lines are the "portals".[0]

and Section 6 gives a deterministic algorithm with parameterized additive error. The listed results come from combining these with fast single-source shortest-path algorithms of geometric intersection graphs involving convex objects, so new advances in shortest-path algorithms in geometric intersection graphs may similarly yield further improved approximation algorithms. We give the derivations for these combinations in Section 7.1.

2 Reduction to Shortest-Paths

In this section, we outline the framework of Spalding-Jamieson and Naredla [SJN25a], which reduces the point-separation problem to a type of shortest-path problem in a geometric intersection graph embedded in the "homology cover". We will focus on simplicity of presentation, using some slight simplifications. At the end, we will also add some additional tools for our purposes.

A simplifying assumption Suppose there is a single object $c \in \mathcal{C}$ that separates s and t. This also includes objects that contain s or t. It can be shown that we can test for the existence of such an object in the oracle model by checking if the canonical point x_c of an object has a path to itself contained entirely within c that crosses $\overline{st}$ an odd number of times. Hence, we can find all such objects in linear time in the oracle model, so we can assume without loss of generality that $\mathcal{C}$ contains no such objects.

The homology cover We now define the homology cover. Consider the plane with two punctures at the points s and t. Let π be the line segment $\overline{st}$. Next, create two copies of $\mathbb{R}^2 \setminus \overline{st}$, and call them P_{-1}, P_1. Each of P_{-1} and P_1 has a line-segment shaped "hole". Connect the top of the hole in P_{-1} to the bottom of the hole in P_1, and vice versa, making sure to leave the endpoints unconnected. The resulting space is what we call the **homology cover**. We will refer to P_{-1} and P_1 as the "two copies of the plane" making up the homology cover. The values -1 and 1 can also be thought of as "indicator bits". We will use similar notation for so-called "indicator bits" later too. An equivalent formulation is to create two pairs of "portals", so if a path traverses up into line segment in one copy of the plane, it comes out in the other copy (see Figure 2).

The geometric intersection graph in the homology cover Now we describe the geometric intersection graph within the homology cover that we will use. We will call this graph $\overline{G}$. More specifically, we will describe how to formulate the set of objects in the homology cover which is

used as the vertex set for this graph. Afterwards, we will also give a simplified test for when two such objects intersect.

As mentioned, we started with the set $\mathcal{C}$ of n objects within the plane, and we will construct a set of $2n$ objects called $\overline{\mathcal{C}}$. We assume that each object $c \in \mathcal{C}$ has an associated **canonical point** x_c. That is, there are exactly two points in the homology cover that project into x_c. One of them is in P_{-1}, and the other is in P_1. Call these x_c^{-1} and x_c^1, respectively. Since we assumed no single object separates s and t, the set of points in the homology cover that project into c has exactly two connected components – these will form the two objects in the homology cover induced by c. We define c^{-1} to be the connected component containing x_c^{-1}, and c^1 to be the connected component containing x_c^1.

Let $b_1, b_2 \in \{-1, 1\}$ be "indicator bits", and let c_1 and c_2 be objects in $\mathcal{C}$. A simple (computational) way to check if two objects $c_1^{b_1}$ and $c_2^{b_2}$ intersect is to check if there is a path from x_{c_1} to x_{c_2}, contained entirely within $c_1 \cup c_2$, that crosses the segment $\overline{st}$ an odd number of times if $b_1 \neq b_2$, or an even number of times if $b_1 = b_2$. This check can be easily accomplished for line segments and disks, for instance. These checks can be used to define the edges of $\overline{G}$, although in many cases, geometric methods are more desirable.

Importantly, it should be noted that $\overline{G}$ *is* a geometric intersection graph. Not only that, but it is a geometric graph of essentially the same class of objects as $\mathcal{C}$. For instance, if $\mathcal{C}$ is a collection of line segments (where a line segment is defined as the set of convex combinations of two endpoints), then so is $\overline{\mathcal{C}}$. Due to prior assumptions, this is true even though $\overline{\mathcal{C}}$ itself has some atypical properties. See Figure 3 for an example of the geometric intersection graph in the homology cover $\overline{G}$.

Useful properties for approximations We now highlight some properties of the homology cover and $\overline{G}$ that will be useful for our approximation algorithms.

First, observe that $\overline{G}$ has an inherent symmetry: Let $c_1, c_2 \in \mathcal{C}$ be objects, and let $b_1, b_2 \in \{-1, 1\}$ be indicator bits. Then $c_1^{b_1} c_2^{b_2} \in E(\overline{G})$ if and only if $c_1^{b_2} c_2^{b_1} \in E(\overline{G})$.

Finally, we arrive at the main characterization of the point-separation problem as a form of shortest-path problem:

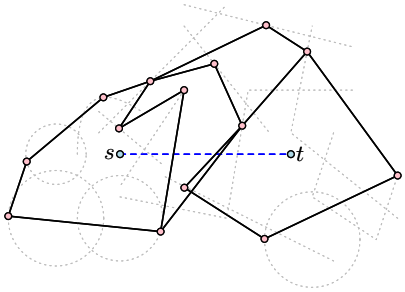
Proposition 2.1. *For a set of objects $\mathcal{C}$, and the graph $\overline{G}$, let D_c denote the elements along an arbitrary shortest-path in $\overline{G}$ from c^{-1} to c^1. Then, the objects in $\mathcal{C}$ corresponding to the vertices along this path together separate s and t – in fact, this is true of* any *path between c^{-1} and c^1 in $\overline{G}$. Let $F_c \subset \mathcal{C}$ denote the objects in this induced separating set. If D_c is the shortest such path over all $c \in \mathcal{C}$, then the set F_c is an optimal solution to the point-separation problem. Furthermore, an object $c \in C$ is a part of an optimal solution if and only if F_c itself is an optimal solution. Consequently, for $c' \in F_c$, $|F_{c'}| \leq |F_c|$ and $|D_{c'}| \leq |D_c|$.*

See Figure 3d for an example of such a path, and see Figure 3e for the corresponding set of objects.

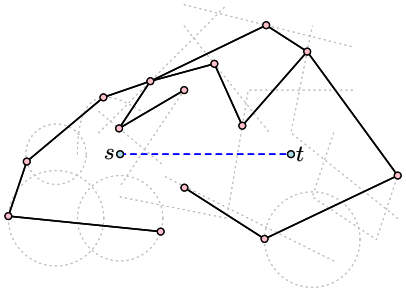
Since we are particularly interested in approximation algorithms for point-separation, we will point out an important consequence of this result:

Corollary 2.2. *Let $c, c' \in C$ be objects, and let b, b' be indicator bits. Suppose that c^b and $c'^{b'}$ are at distance d in the intersection graph $\overline{G}$ in the homology cover, or that c and c' are at distance d in the intersection graph G in the plane. Then $|D_{c'}| \leq |D_c| + 2d$, and $|F_{c'}| \leq |F_c| + d$.*

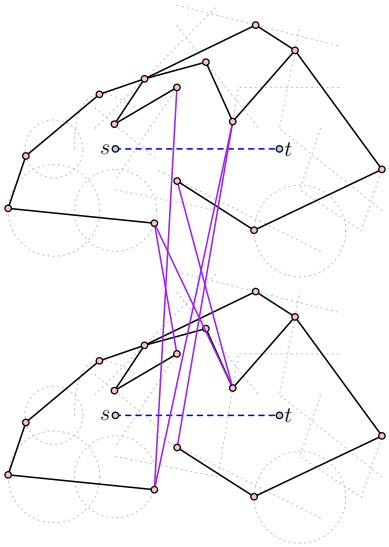
This follows from the symmetry of $\overline{G}$.



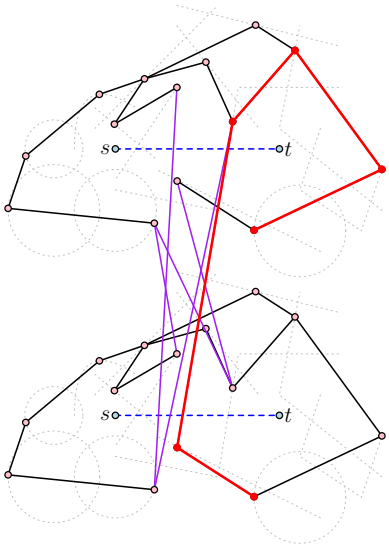
(a) The intersection graph with the path $\pi = \overline{st}$.



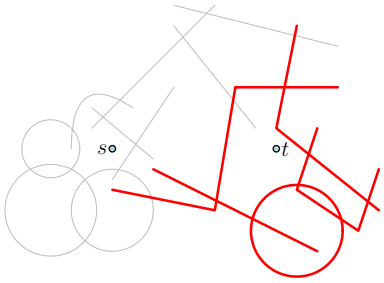
(b) The intersection graph after deleting the edges whose underlying paths cross π an odd number of times.



(c) Two copies of the intersection graph are connected with "crossing" edges (in purple).



(d) A shortest-path in the homology cover between two copies of an object (a path D_c).



(e) The corresponding set of objects that separate s and t (a set F_c).

Figure 3: How the intersection graph can be transformed into the intersection graph in the homology cover.[0]

2.1 Notation and Time Complexity

Fast single-source shortest-path algorithms are widely known for a number of geometric intersection graphs in the plane. For example, a single-source shortest-path in a disk graph with n vertices has recently been shown to be solvable in $\mathcal{O}(n \log n)$ time [BWB25, dBC25a], even though the graph itself may have $\Theta(n^2)$ edges. Some of these algorithms are also already known to extend to the homology cover. In Section 7, we will discuss such extensions, including new results.

All the algorithms we will describe in this paper will make use of such single-source shortest-path algorithms as a black-box. Every such computation will measure the distance from one copy of an object to another, so we will consistently use the notation defined in Proposition 2.1: D_c denotes the sequence of elements along the shortest-path in $\overline{G}$, and F_c denotes the corresponding separating set of objects. Furthermore, we will measure the time complexity of our algorithms *in terms* of the number of single-source shortest-path computations performed. Let $\texttt{Time}_{\text{SSSP}}(n)$ denote the time to compute D_c (and, consequently, F_c) among n objects. For instance, among disk objects, $\texttt{Time}_{\text{SSSP}}(n) \in \mathcal{O}(n \log n)$. We will also assume that $\texttt{Time}_{\text{SSSP}}(n)$ is super-additive. That

is, for problem sizes $a, b \geq 0$, $\texttt{Time}_{\text{SSSP}}(a) + \texttt{Time}_{\text{SSSP}}(b) \leq \texttt{Time}_{\text{SSSP}}(a+b)$. This will be useful in analyzing a divide and conquer approach later.

3 A Monte Carlo Algorithm

In this brief section, we will give a randomized Monte Carlo algorithm of sorts:

Theorem 3.1. *For a given pair of points s and t, and a set of objects $\mathcal{C}$, there is an iterative algorithm with the following properties:*

- *Each iteration takes $\texttt{Time}_{SSSP}(n)$ time, and produces a separating set of objects.*
- *If there exists a separating set of objects $C \subset \mathcal{C}$, then, with high probability, the algorithm will find a separating set of objects $C' \subset \mathcal{C}$ with size $|C'| \leq |C|$ in $\mathcal{O}\left(\frac{n}{|C|}\log n\right)$ iterations.*

Proof. The algorithm is quite simple: In each iteration, a uniformly random object c is sampled from $\mathcal{C}$, and then D_c and F_c (defined in Proposition 2.1) are computed. By Proposition 2.1, a set of objects with size at most $|C|$ will be found if $c \in C$, so the resulting iteration count is simply the number required to sample an element of C with high probability. □

A useful consequence of this theorem is that we can probabilistically determine if a separating set of size $\mathcal{O}(\sqrt{n})$ exists within $\mathcal{O}(\sqrt{n}\log n)$ iterations. We will use this consequence in the next section.

Another method of deriving this theorem would be to note that the so-called "VC-dimension" of the sets $\{D_c\}_{c\in\mathcal{C}}$ is bounded [dLdSV23], so we can produce an "ϵ-net" with high-probability.

4 A Fast Additive +1 Approximation

We will now present one of the most notable algorithms of this work. Specifically, we will give an additive +1 approximation of point-separation that is, in many cases, computable in subquadratic time. This algorithm will be analogous to Reif's algorithm for maximum (s,t)-flow in a planar graph. In particular, it will employ a divide and conquer approach over a path from s to t, and a form of "separators". These separators will be 1-hop path separators that are balanced relative to intersections of the line segment $\overline{st}$, but not balanced in any other sense.

Theorem 4.1. *For a given pair of points s and t, and a set of n objects $\mathcal{C}$, where each object intersecting the segment $\overline{st}$ is convex, assume the intersection of each $c \in \mathcal{C}$ with the $\overline{st}$ can be computed in constant time. Then, there is a randomized Monte Carlo algorithm that produces an additive +1 approximation for the point-separation problem in $\mathcal{O}\left((\sqrt{n}\log n)\cdot \texttt{Time}_{SSSP}(n)\right)$ time (with high probability), where $\texttt{Time}_{SSSP}(n)$ is the time required to compute a shortest-path through the geometric intersection graph in the homology cover.*

The most notable restriction of this algorithm is that it requires some of the objects to be convex. However, this is true of many of the geometric object classes we have already mentioned, such as disks and line segments.

Before giving the proof, we highlight several of the main techniques we will employ:

- The first key insight is that, using the Monte Carlo algorithm in the previous section, if the optimum solution is of size $\Omega(\sqrt{n})$, then we can find it with high probability.

- Next, we employ a divide and conquer strategy. Specifically, we will consider the intersections of the line segment $\overline{st}$ with objects. Loosely, we will use these to define an ordering of the objects intersecting the segment. The details of this ordering, and why it will be useful, are related to the assumption of convexity.

- The optimality guarantees of the divide and conquer strategy will assume that the optimum solution is of size $\mathcal{O}(\sqrt{n})$, but the running time analysis will not.

- To perform the "divide" part of the divide and conquer strategy, we shall find a path of objects in the homology cover $\overline{G}$ from c^{-1} to c^1, where c is the "middle" object in the ordering. In the simplest case, this path will be exactly D_c, although this is not true in the most challenging case, where it will not even be a shortest path. Regardless, bounds will be derived on $F_{c'}$ for every c' in the path between the copies of c, as well as its 1-hop neighbourhood. Specifically, for every such c', we will either compute $F_{c'}$, or it will be shown $|F_{c'}|$ is at least $\Omega(\sqrt{n})$ (larger than the best solution assumed to have been found by the Monte Carlo algorithm). Then, the path and its 1-hop neighbourhood will be removed from $\overline{G}$, and the remaining connected components will be recursed on separately.

- The algorithm to "combine" will be quite simple: Of every separating set F_c computed throughout all steps, output the smallest. Since we are deleting vertices from the graph in the division step, the F_c sets in lower levels of the recursion will not be "true" minimum F_c sets for those vertices, but we will argue that this is okay regardless.

Proof. We will describe each aspect of the algorithm in turn, while proving all the necessary properties for correctness and time complexity along the way.

Applying the Monte Carlo algorithm Apply Theorem 3.1 to determine if, with high probability, the optimal solution to the point-separation problem has at least $f_1(n)-1$ objects, where $f_1(n)$ is a function we will choose later. Then, if it does, we halt the algorithm, since the Monte Carlo algorithm also provides such a solution. Regardless, this step takes $\mathcal{O}\left(\left(\frac{n}{f_1(n)}\log n\right)\cdot \texttt{Time}_{\text{SSSP}}(n)\right)$ time. We call this the **Monte Carlo step**. An implementation of this step requires choosing $f_1(n)$.

Divide and conquer Henceforth, we will describe a divide and conquer approach. We will make specific approximation guarantees for every shortest-path distance. In the case that the optimal solution contains less than $f_1(n)-1$ objects, these specific approximation guarantees will also imply a global approximation on the optimal solution to the point-separation problem. Combined with the Monte Carlo step, this gives an approximation guarantee in general.

Segment intersection ordering We will create an ordering of all objects intersecting $\overline{st}$. We have assumed all objects are convex. Hence, for each object $c \in \mathcal{C}$ intersecting $\overline{st}$, that intersection must be exactly a sub-segment of $\overline{st}$. Let $[0,1]$ represent the sub-segment $\overline{st}$. For every object intersecting $\overline{st}$, we can define a corresponding sub-interval of $[0,1]$. There is one important simple property of these intervals that we will use: If two objects c and c' (both intersecting $\overline{st}$) do not mutually intersect, then their corresponding intervals do not overlap. The contrapositive of this property is that if two intervals overlap, then the corresponding objects must also overlap.

We will order all the objects intersecting $\overline{st}$ according to the left-endpoints of their respective sub-intervals (see Figure 4). For an ordering with k elements, consider a "middle" element in this ordering. That is, an element with at most $k/2$ elements both before and after it. Let c be the

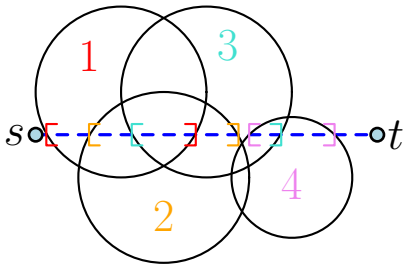


Figure 4: The convex objects intersecting $\overline{st}$ are ordered based on their "first" intersection point.

corresponding object. Suppose that we find a connected set of objects that includes c, and separates s and t (such as F_c). We will call such a set a **dividing path through** c, since the sets we will use will always correspond to paths in $\overline{G}$. Then, if we delete all the objects in the dividing path, and their 1-hop neighbourhoods, we observe the following: For two objects c', c'' both intersecting $\overline{st}$ and both not deleted in the previous process: If $c' < c < c''$ in the ordering, then c' and c'' are in different connected components of the geometric intersection graph (and hence their copies are in different connected components of $\overline{G}$) after the deletions. This is the main operation that will allow us to perform divide and conquer, since it allows us to halve the length of the sequence at each level of recursion, while effectively distributing the remaining objects among the different recursive calls.

Finding good dividing paths Let c be the "middle" object defined in the previous step. Without any constraints, finding a dividing path through c is quite simple: We could simply take D_c. However, this choice will not always suffice for our ultimate goal of tackling the point-separation problem. We will be deleting all the objects corresponding to vertices in our dividing path, as well as their 1-hop neighbourhoods. Hence, we need to be able to say something about the size of $F_{c'}$ for all such objects c', as well as all sets $F_{c''}$ even *including* some object c' that will be deleted.

The way we will accomplish this is as follows: With our final connected set of objects P[2] to be deleted along with its 1-hop neighbourhood $N_1(P)$, we will make one of the following guarantees for any object $c' \in P$ (regardless of the size of the optimum solution):

- We will have computed $F_{c'}$ directly.
- We will have computed $F_{c''}$ for some element $c'' \in P$ with $|F_{c''}| \geq f_1(n) + d$ so that c' has distance at most d to c'' in the geometric intersection graph G, $|F_{c'}| \geq |F_{c''}| - d \geq f_1(n)$.

Consequently, we also obtain the same guarantees up to an additive $+1$ term for any $p' \in N_1(P)$. We will call these guarantees the **dividing path approximation guarantees**. In general, $|P|$ may be quite large, hence why we do not wish to compute $F_{c'}$ for every object c' in P. This leads to a conflict requiring some balance: We must fulfill the dividing path approximation guarantees while minimizing the total runtime required to compute the dividing path.

The algorithm to find the dividing path P is as follows:

Algorithm 4.2. DIVIDINGPATH$(c, f_2(n,m), f_3(n,m), f_4(n,m))$

- *Compute D_c and F_c (one shortest-path computation).*

[2]Note that this is slight abuse of notation: P is a path through $\overline{G}$, and hence consists of a subset of objects from $\overline{\mathcal{C}}$ in the homology cover, although here we use it as a set of elements from $\mathcal{C}$ corresponding to the induced elements. We will continue to slightly abuse the notation P in this manner.

- *Let $m := |F_c|$. Recall that $1 \le m \le n$ by definition.*
- ***Case 1:** If $m \le f_2(n, m)$, then:*
  - *For each object $c' \in F_c$, compute $F_{c'}$.*
  - *Return $P := D_c$.*

  ($\mathcal{O}(f_2(n, m))$ shortest-path computations)
- ***Case 2:** Else if $m > f_2(n, m)$:*
  - *Denote $D_c = [c^{-1} = v_0^{b_0}, \ldots, v_m^{b_m} = c^1]$.*
  - *Select a subsequence $i_1, \ldots, i_r$ of indices of D_c so that each $j \in \{1, \ldots, m-1\}$ is in some interval $[i_l, i_{l+1}]$, and $|i_{l+1} - i_l| \le f_3(n, m)$.*
  - *For each $l \in \{1, \ldots, r\}$, compute the pair $(D_{v_{i_l}}, F_{v_{i_l}})$.*
    ($r = \mathcal{O}\left(\frac{m}{f_3(n,m)}\right)$ shortest-path computations)
  - ***Case 2a:** If every $|F_{v_{i_l}}| > f_4(n, m)$ then return the path $P := D_c$.*
    ***Case 2b:** Else, let l be the smallest index where $|F_{v_{i_l}}| \le f_4(n, m)$.*
    - *Let $P' = [u_1, \ldots, u_p]$ =DividingPath(v_{i_l}).*
    - *Return*

      $$P := \left[c^{-1} = v_0^{b_0}, v_1^{b_1}, \ldots, v_{i_l}^{b_{i_l}} = u_1, \ldots, u_p = v_{i_l}^{-b_{i_l}}, \ldots, v_1^{-b_1}, v_0^{-b_0} = c^1\right],$$

      flipping all indicator bits in P' if necessary.

This is the **dividing path algorithm**. We have described the algorithm in a very general form, since we will use it again for another proof in the next section. An implementation of the dividing path algorithm requires choosing f_2, f_3, and f_4. In this particular proof, we will choose them so that the dividing path algorithm will terminate after at most one recursive call. Moreover, we will choose f_1, f_2, f_3, f_4 as functions only in n, and not in m.

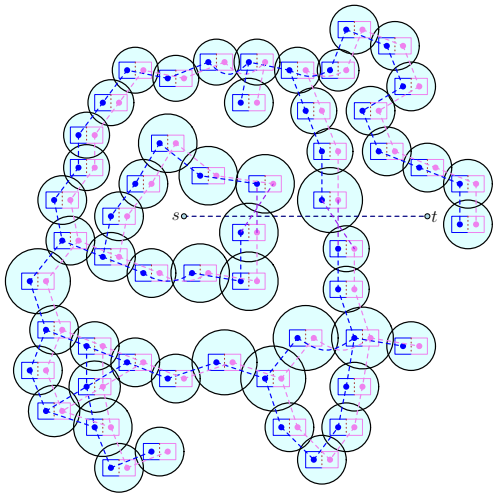
We must choose the functions f_1, f_2, f_3, f_4 to accomplish two things simultaneously: First, we must fulfill the dividing path approximation guarantees. Consider the following choices: For any fixed constant α, pick $f_1(n) = \alpha\sqrt{n}$, $f_2(n) = 2\alpha\sqrt{n}$, $f_3(n) = \alpha\sqrt{n}$, and $f_4(n) = 2\alpha\sqrt{n}$. Under these choices, case 2b occurs only if $f_4(n) = f_2(n)$. Hence, case 1 is always triggered in the second layer of recursion, and so there are at most 2 layers of recursion, greatly simplifying our analysis.

Note that the dividing path approximation guarantees are always fulfilled in case 1, so we need only consider cases 2a and 2b. In case 2a, every object c' induced by a vertex in P has $|F_{c'}| \ge f_4(n) - f_3(n) = \alpha\sqrt{n} = f_1(n)$, fulfilling the guarantee. Similarly, in case 2b, every object c' induced by a vertex in P but not P' has the same guarantee.

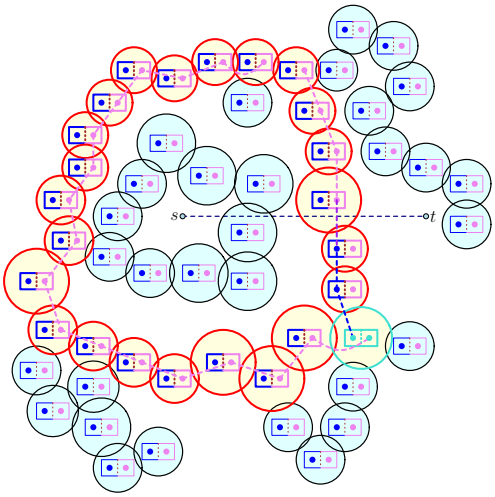
The total time complexity of the dividing path algorithm is fairly easy to analyze. In total, it takes $\mathcal{O}\left(\frac{n}{f_3(n)} + f_2(n)\right)$ shortest-path computations, which is exactly $\mathcal{O}\left(\sqrt{n} \cdot \texttt{Time}_{\text{SSSP}}(n)\right)$ time.

An example run of the dividing path algorithm is given in Figure 5.

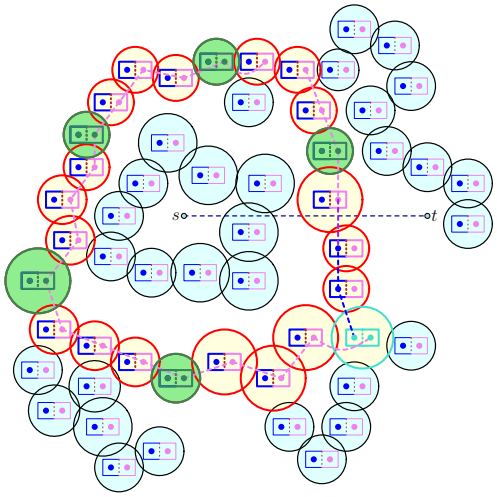
Combining Results We now complete the divide and conquer process. The algorithm is as-outlined: Take the middle object c in the ordering. Compute a dividing path P through c, and delete P and its 1-hop neighbourhood. Then, recurse on the ordering before c, and the ordering after c. Note that the residual objects form at least two connected components in G (and $\overline{G}$), and



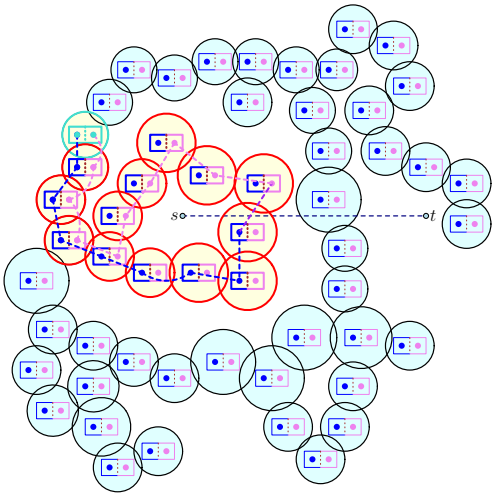
(a) The intersection graph in the homology cover.



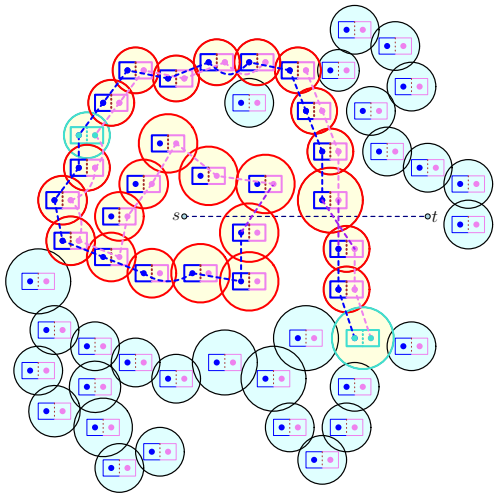
(b) A shortest-path D_c.



(c) Case 2: Well-spaced samples are chosen along D_c, and $D_{c'}$ is computed for each sample c'.



(d) Case 2b: For one of c', $D_{c'}$ is recursed on.



(e) D_c is truncated and then the concatenation of the truncated D_c, $D_{c'}$, and the complementary objects of D_c is returned.

Figure 5: The dividing path algorithm applied with one recursive call.

moreover that each connected component is on exactly one side of the union of P and its 1-hop neighbourhood, so these subproblems are disjoint.

For any c' among the residual objects, let $F_{c'}$ denote the separating set of objects induced by $D_{c'}$ among *the original set of objects*, and let $F'_{c'}$ instead take from *the residual set of objects*. Note that $|F'_{c'}| \geq |F_{c'}|$. Specifically, we have strict inequality if every shortest-path $D_{c'}$ contains some vertex in P or its 1-hop neighbourhood. We claim this is not an issue, and it suffices to approximate $|F'_{c'}|$ instead of $F_{c'}$: This follows from the final inequality stated in Proposition 2.1. Specifically, if $F_{c'}$ contains some c'' in P or its 1-hop neighbourhood, then we have a good approximation of $F_{c''}$ by the dividing path approximation guarantees, and we know that $|F_{c'}| \geq |F_{c''}|$.

Global approximation guarantees We now show that we have given an additive $+1$ approximation to the point-separation problem.

First, consider the case where the optimum solution has at least $f_1(n) - 1$ objects. In this case, the Monte Carlo step will find the exact optimum solution with high probability.

Second, consider the case where the optimum solution has less than $f_1(n) - 1$ objects. In this case, we apply the approximation guarantees we have for every $c \in \mathcal{C}$ to some c^* minimizing $|F_{c^*}|$. At some level of recursion, c^* was included in some dividing path P or its 1-hop neighbourhood. In either case, there is some c in P so that $|F_c| \leq |F_{c^*}| + 1$. Furthermore, if any element $c' \in F_c$ was not part of the residual objects at that level of recursion, it must be the case that $|F_{c'}| \leq |F_c|$. Hence, without loss of generality, we may assume that all elements of F_c were part of the residual objects at that level of recursion. We apply the dividing path approximation guarantees for $|F_c|$:

- If we have computed F_c directly, then we are done.
- If we have not computed F_c directly, then $|F_c| \geq f_1(n)$, in which case the optimum was found by the Monte Carlo step with high probability.

Time-Complexity Analysis We analyze the time complexity with no assumptions of the optimum solution's size. There are $\mathcal{O}(\log n)$ levels of recursion. At each layer, the cost of $\mathcal{O}(\sqrt{n})$ shortest-path computations on the entire graph are incurred. This is *not* the same as stating that $\mathcal{O}(\sqrt{n})$ shortest-path computations occur at each level. In fact, many more occur. However, at the ith level of the recursion there are at most 2^i different independent subproblems. Each one is individually performing $\mathcal{O}(\sqrt{n})$ shortest-path computations. However, each of the independent subproblems uses a subgraph disjoint from the subgraphs of the other subproblems. Hence, since we assumed super-additivity of shortest-path computation time (i.e., $\texttt{Time}_{\text{SSSP}}(a+b) \geq \texttt{Time}_{\text{SSSP}}(a) + \texttt{Time}_{\text{SSSP}}(b)$ where a and b are sizes of the input graphs), we can give an upper bound on the total time complexity for the divide and conquer as $\mathcal{O}((\sqrt{n}\log n) \cdot \texttt{Time}_{\text{SSSP}}(n))$. The Monte Carlo step runs in the same time complexity, so our overall time complexity is the same.

□

5 Fast Deterministic Multiplicative-Additive Approximations

In the previous section, we outlined an additive $+1$ approximation algorithm for point-separation that uses a sub-linear number of shortest-path computations. The key step used was a generic recursive routine to construct a "dividing path", which proved useful for performing divide and conquer. The dividing path algorithm template we presented was actually more general than what was required in the previous section. We will now re-use it.

In this section, we will show that by choosing a different set of parameterized functions for the dividing path algorithm, we can obtain a much faster algorithm for point-separation with a slightly weaker approximation guarantee. In particular, we will obtain a "multiplicative-additive" approximation guarantee. A key step will be to choose parameters so as to make use of the recursion, unlike Theorem 4.1 (where it only recurses at most once). In fact, the proof of the new dividing path guarantees as a whole will be quite different and significantly more complex.

Theorem 5.1. *For a given pair of points s and t, and a set of n objects $\mathcal{C}$, where each object intersecting the segment $\overline{st}$ is convex, assume the intersection of each $c \in \mathcal{C}$ with the $\overline{st}$ can be computed in constant time. Let $0 < \varepsilon \leq 2$ be some value. Then, there is a deterministic algorithm that produces a "multiplicative-additive" approximation for the point-separation problem in $\mathcal{O}\left(\left(\frac{\log n}{\varepsilon}\right)^2 \texttt{Time}_{SSSP}(n)\right)$ time, where $\texttt{Time}_{SSSP}(n)$ is the time required to compute a shortest-path through the geometric intersection graph in the homology cover. The specific approximation guarantee is that if there exists a solution to the point-separation problem of size k^*, then the algorithm produces a solution of size at most $(1+\varepsilon)k^* + 1$.*

Proof. By following the outline of the proof of Theorem 4.1, it suffices to provide parameters $f_2(n,m)$, $f_3(n,m)$, and $f_4(n,m)$ for Algorithm 4.2 that result in the dividing path algorithm using $\mathcal{O}\left(\frac{\log n}{\varepsilon^2}\right)$ shortest-path computations, in such a manner that it produces a dividing path P in G with one of the following approximation guarantees for each $c' \in P$:

- We will have computed $F_{c'}$ directly.
- We will have computed $F_{c''}$ directly for some $c'' \in P$, and we guarantee that $|F_{c'}| \geq \frac{|F_{c''}|}{1+\varepsilon}$.

Although we are making use of the same "template" for computing the dividing paths, the way we will choose our parameters and prove our result is different in a couple fundamental ways, and more complicated. Most notably: The algorithm will actually run recursively with more than constant depth, and the computed $F_{c''}$ sets referred to in the second case of the guarantees will not be the same ones that are used in Theorem 4.1.

We choose the following parameters:

$$f_3(n,m) = \max\left\{1, \frac{\varepsilon m}{3(1+\varepsilon)}\right\},$$

$$f_2(n,m) = f_4(n,m) = \frac{m}{\sqrt{1+\varepsilon}} + 3.$$

First, we establish termination and time complexity: Suppose that for some c, $|F_c| > f_2(n, |F_c|)$, and that $|F_{v_{i_l}}| \leq f_4(n, |F_c|)$. Then $|F_{v_{i_l}}| \leq f_4(n, |F_c|) < |F_c|$, so a recursive call must decrease the length of the path (the value of m). Therefore, since $m \leq n$, the depth of the recursion must be

$$\mathcal{O}\left(\frac{\log n}{\log\sqrt{1+\varepsilon}}\right) \subset \mathcal{O}\left(\frac{\log n}{\varepsilon}\right).$$

At each level where case 2 is applied, $\mathcal{O}\left(\frac{m}{f_3(n,m)}\right)$ shortest-path computations are made, which simplifies to

$$\mathcal{O}\left(\frac{m}{f_3(n,m)}\right) \subset \mathcal{O}\left(\frac{m \cdot 3(1+\varepsilon)}{\varepsilon m}\right) = \mathcal{O}\left(\frac{3(1+\varepsilon)}{\varepsilon}\right) = \mathcal{O}\left(\frac{1}{\varepsilon}\right).$$

At the final level where case 1 is applied, if we assume $1+\varepsilon \le 2$, we have

$$\begin{aligned}
m &\le \frac{m}{\sqrt{1+\varepsilon}} + 3 \\
\implies m &\le \frac{3}{1-\frac{1}{\sqrt{1+\varepsilon}}} \\
&= \frac{3\sqrt{1+\varepsilon}}{\sqrt{1+\varepsilon}-1} \\
&= \frac{3\sqrt{1+\varepsilon}(\sqrt{1+\varepsilon}+1)}{(\sqrt{1+\varepsilon}-1)(\sqrt{1+\varepsilon}+1)} \\
&= \frac{3\sqrt{1+\varepsilon}(\sqrt{1+\varepsilon}+1)}{(1+\varepsilon)-1} \\
&= \frac{3\sqrt{1+\varepsilon}(\sqrt{1+\varepsilon}+1)}{\varepsilon} \\
&\le \frac{3\sqrt{2}(\sqrt{2}+1)}{\varepsilon} =\in \mathcal{O}\left(\frac{1}{\varepsilon}\right).
\end{aligned}$$

Thus, in total, we run $\mathcal{O}\left(\frac{\log n}{\varepsilon^2}\right)$ shortest-path computations, as desired.

Next, we establish the approximation guarantees: Let c' be some element of P for which we did not compute $F_{c'}$ directly. This can only occur in case 2. Suppose c' was added to P by a level of recursion that used c as the initial input, with $m = |F_c|$. Regardless of whether or not case 2a or 2b was taken, we know that c' must have distance at most $f_3(n,m)$ to some $c'' \in P$ with $F_{c''}$ computed, and moreover that $|F_{c''}| > f_4(n,m) = f_2(n,m)$ (note that in one case, the only valid choice of c'' will be c). Hence, by Corollary 2.2, $|F_{c'}| \ge |F_{c''}| - f_3(n,m) > f_4(n,m) - f_3(n,m)$. We will show that, in this case, $f_4(n,m) - f_3(n,m) \ge \frac{m}{(1+\varepsilon)}$.

First, we will show that $f_3(n,m) = \frac{\varepsilon m}{3(1+\varepsilon)}$ in this case. Since we are in case 2, we have $m > f_2(n,m) = \frac{m}{\sqrt{1+\varepsilon}} + 3$. Hence,

$$m > \frac{3}{1-\frac{1}{\sqrt{1+\varepsilon}}} = \frac{3\sqrt{1+\varepsilon}(\sqrt{1+\varepsilon}+1)}{\varepsilon} > \frac{3(1+\varepsilon)}{\varepsilon} \implies \frac{\varepsilon m}{3(1+\varepsilon)} > 1.$$

Thus, by assuming again that $\varepsilon \le 1$, we have

$$\begin{aligned}
(1+\varepsilon)\big[f_4(n,m) - f_3(n,m)\big] &= (1+\varepsilon)\left[\frac{m}{\sqrt{1+\varepsilon}} + 3 - \frac{\varepsilon m}{3(1+\varepsilon)}\right] \\
&= m\sqrt{1+\varepsilon} + 3(1+\varepsilon) - \frac{\varepsilon m}{3} \\
&\ge m\sqrt{1+\varepsilon} - \frac{\varepsilon m}{3} \\
&= m\left(\sqrt{1+\varepsilon} - \tfrac{\varepsilon}{3}\right) \\
&\ge m,
\end{aligned}$$

where the last step follows from assuming $\varepsilon \le 1$, and hence

$$(1+\varepsilon) - \left(1+\frac{\varepsilon}{3}\right)^2 = 1+\varepsilon - \left(1+\tfrac{2\varepsilon}{3}+\tfrac{\varepsilon^2}{9}\right) = \frac{\varepsilon(3-\varepsilon)}{9} \ge 0 \implies \sqrt{1+\varepsilon} - \frac{\varepsilon}{3} \ge 1.$$

Therefore, we get $f_4(n,m) - f_3(n,m) \ge \frac{m}{1+\varepsilon}$, as desired. □

6 A Simple Deterministic Additive $+k$ Approximation

We now outline a deterministic algorithm with an approximation guarantee not attainable by the previous section, although the guarantee is weaker than that of the Monte Carlo algorithm described in Theorem 3.1. Unfortunately, the weaker guarantee is not enough for usage in Theorem 4.1.

Theorem 6.1. *For a given pair of points s and t, and a set of n objects $\mathcal{C}$, and a parameter k, there is an algorithm that produces an additive $+k$ approximation for the point-separation problem in $\mathcal{O}\left(\frac{n}{k} \cdot \mathtt{Time}_{SSSP}(n)\right)$ time.*

Proof. Assume without loss of generality that $\overline{G}$ is connected (if not, handle each connected component separately). Let T be an arbitrary spanning tree of $\overline{G}$ (for example, a single-source shortest-path tree from an arbitrary vertex). Consider an Euler tour around T, obtaining a circular sequence covering all vertices. Cut this circular sequence at some arbitrary point to obtain a sequence S covering all vertices $v_1, \ldots, v_{2n}$. The key property of this sequence is that the distance from v_i to v_j in $\overline{G}$ is at most $|i-j|$.

We pick a "net" of vertices in the sequence S. That is, let S' be the subsequence $v_1, v_{1+(2k+2)}$, $v_{1+2(2k+2)}, v_{1+3(2k+2)}, \ldots, v_{2n}$. Then, compute D_c and F_c for each c with a vertex in S'. For any $v_i \in S$, there is some $v_{1+j(2k+2)}$ in S' so that $|i-(1+j(2k+2))| \leq k$. Hence, $|F_{v_i}| \leq |F_{v_{1+j(2k+2)}}|+k$ by Corollary 2.2, so the smallest set F_c computed for c with a vertex in S' is an additive $+k$ approximation for the smallest separating set by Proposition 2.1.

Finally, since we computed a shortest-path only for each element of S' (plus one extra to compute T), and $|S'| \in \mathcal{O}\left(\frac{n}{k}\right)$, the algorithm runs in the claimed time complexity. □

This algorithm could be sped up for certain classes of objects and small values of k, since there are some techniques in geometric intersection graphs that can perform shortest-path queries faster if the shortest-path tree for a nearby vertex (ideally, a neighbouring vertex) is known [CS17]. However, such speedups would come with a poor tradeoff in the form of an additional factor of k in the time complexity.

7 Fast Shortest-Paths in the Homology Cover

In this section, we discuss methods for computing single-source shortest-paths (SSSP) in the intersection graph in the homology cover, $\overline{G}$. The object types we focus on in this paper are given in Table 2, along with two extra types. Our results are primarily based on other advances for SSSP in geometric intersection graphs, however small modifications to the algorithms are needed to apply them within the homology cover.

Object Type	$\mathtt{Time}_{\mathrm{SSSP}}(n)$	**Reference**
Disk	$\mathcal{O}(n \log n)$	Theorem 7.2
Line segment	$\widetilde{\mathcal{O}}(n^{4/3})$	[SJN25a]
$\mathcal{O}(1)$-complexity polyline	$\widetilde{\mathcal{O}}(n^{4/3})$	[SJN25a]
Rectilinear line segment	$\mathcal{O}(n \log n)$	Theorem 7.4
$\mathcal{O}(1)$-complexity rectilinear polyline	$\mathcal{O}(n \log n)$	Theorem 7.4

Table 2: Summary of SSSP running times for geometric intersection graphs in homology cover. All running times here are deterministic.

Disks For the intersection graph of disks in the plane, the fastest single-source shortest-path algorithm is quite recent. Moreover, two groups independently discovered optimal algorithms (optimal for the algebraic decision tree model):

- De Berg and Cabello [dBC25a, dBC25b] showed that disk graphs in the plane admit structure called a "clique-based contraction" with some helpful properties, and that this structure can be computed efficiently.
- Brewer and Wang [BWB25, BW25] instead made use of additively-weighted Voronoi diagrams.

Both of these groups' methods attain $\mathcal{O}(n \log n)$-time algorithms for the single-source shortest-path problem in a unit-disk graph, including the ability to compute a shortest-path tree. We will show that the method of de Berg and Cabello can be extended to the intersection graph in the homology cover in the same time complexity.

A **clique-based contraction** of a graph $G = (V, E)$ is a graph $H = (U, F)$ so that every $u \in U$ is exactly a subset of V forming a clique. Moreover, it is required that every $v \in V$ is in exactly one $u \in U$ (i.e., U is a partition of V), and that an edge $uu' \in F$ exists in H if and only if there is some edge $vv' \in E$ with $v \in u$ and $v' \in u'$.

De Berg and Cabello showed that, if G is an intersection graph of n disks $\mathcal{C}$, a clique-based contraction with $|F| = \mathcal{O}(n \log n)$ "post-contraction" edges exists, and that it can be constructed in the same time complexity. Furthermore, the clique-based contraction they construct has the property that every $u \in U$ corresponds specifically to a *stabbed clique* in G. That is, for each $u \in U$, there is some point p in the plane contained by *every* $v \in u$. The resulting lemma is:

Lemma 7.1 ([dBC25a, dBC25b, Proposition 8])**.** *For a geometric intersection graph of n disks $\mathcal{C}$ in the plane, let G be the geometric intersection graph of $\mathcal{C}$. Then, a clique-based contraction of G with $\mathcal{O}(n \log n)$ edges can be computed in $\mathcal{O}(n \log n)$ time.*

Another element of their proof involves **bichromatic intersection testing**: Given two sets B and R of disks ("blue" and "red" disks), find, for each $b \in B$, if it intersects any element of R (the element itself need-not be identified). Well-known methods exist for performing this step among disks in the plane in $\mathcal{O}((|B| + |R|) \log |R|)$ time [CS16, CS17, Klo23]. De Berg and Cabello also showed that, by combining the clique-based contraction of G and the bichromatic intersection testing algorithm, they can perform shortest-path computations in G in $\mathcal{O}(n \log n)$ time. Moreover, these are the only elements needed. That is, if we can compute a clique-based contraction of the intersection graph $\overline{G}$ in the homology cover with the same guarantees, and perform bichromatic intersection testing on the objects $\overline{\mathcal{C}}$ in the homology cover, then we can solve the single-source shortest-path problem in the same time complexity. This is exactly the method we will employ:

Theorem 7.2. *For a set of disks $\mathcal{C}$ in the plane, and points s and t, let the geometric intersection graph in the homology cover be denoted $\overline{G}$. Then, there is an algorithm that computes a single-source shortest-path tree within $\overline{G}$ in $\mathcal{O}(n \log n)$ time.*

Proof. By the method of de Berg and Cabello [dBC25a], it suffices to construct a clique-based contraction of $\overline{G}$ with $\mathcal{O}(n \log n)$ edges in the same time complexity, and give an algorithm for static intersection detection in the same time complexity. In fact, Spalding-Jamieson and Naredla already gave an algorithm for bichromatic intersection testing ("static intersection detection") in the homology cover with the required time complexity [SJN25b, Appendix C].

We now show how to compute the clique-based contraction. First, generate the clique-based contraction of G, in which all the cliques are stabbed cliques. For a stabbed clique $Q \subset \mathcal{C}$ with

stabbing point p, let p be the canonical point of all $c \in Q$ for the purposes of labelling in the homology cover. Then, to generate the cliques in $\overline{G}$, we create two copies of each stabbed clique Q, which we denote Q^{-1} and Q^1, relative to the location of the stabbing point p.

It remains to compute the edges in the clique-based contraction of $\overline{G}$. Let Q and R be cliques in the clique-based contraction of G, and let $b_1, b_2 \in \{-1, 1\}$ be indicator bits. Let the stabbing points of Q and R be denoted p_Q and p_R, respectively. Let F_Q, F_R, $F_{Q^{b_1}}$ and $F_{R^{b_2}}$ be the union of elements in each clique (de Berg and Cabello call these **flowers**). Then, Q^{b_1} and R^{b_2} intersect if and only if there is some point p in the intersection of $F_{Q^{b_1}}$ and $F_{R^{b_2}}$ so that the number of intersections of the length-2 polyline $\overline{p_R p p_Q}$ with the segment $\overline{st}$ is odd (if $b_1 \neq b_2$) or even (if $b_1 = b_2$). Moreover, we know that no disk contains s or t. Q and R intersect if one of two things occur: Either the boundaries of Q and R intersect (in which case we need only consider crossing points p), or one is contained in the other. In the latter case, we look at whether or not the segment $\overline{p_R p_Q}$ crosses the segment $\overline{st}$ – no other path can have a different parity of crossings. For the former case: de Berg and Cabello [dBC25a] actually iterate through all crossing points between the boundaries of F_Q and F_R as part of their algorithm, so we simply check each one. □

Line segments and constant complexity polylines Intersection graphs of line segments (and, consequently, constant complexity polylines) in the plane are known to have small "biclique covers": A **biclique cover** of a graph $G = (V, E)$ is a collection $\{(A_1, B_1), \ldots, (A_k, B_k)\}$ where each (A_i, B_i) is a **biclique** – $A_i, B_i \subset V$ and $A_i \times B_i \subset E$ – such that every edge $e \in E$ is in at least one biclique in S. The **size** of a biclique cover is the sum $\sum_i [|A_i| + |B_i|]$. A geometric intersection graph over n line segments (or, consequently, constant complexity polylines) in the plane have biclique covers of size $\widetilde{\mathcal{O}}(n^{\frac{4}{3}})$, or $\widetilde{\mathcal{O}}(n)$ if the segments are rectilinear [Cha23]. Moreover, these constructions are also able to compute the covers in the same time complexity (up to poly-logarithmic factors). Spalding-Jamieson and Naredla also observe that these constructions can be quite easily adapted to line segments in the homology cover [SJN25a]. This mainly uses the fact that the intersection of a line segment and a half-plane is another line segment (which does not apply for disks, hence the method above).

A biclique cover can be thought of as a kind of exact sparsification of a graph. In particular, given a biclique cover for a graph G of size K, single-source shortest-paths computations within G can be reduced to single-source shortest-path computations within a directed graph G' containing $\mathcal{O}(K)$ vertices, even in the case that the vertices are given weights [SJN25a, SJN25b] (although we are only interested in unweighted objects). The result is the following theorem:

Theorem 7.3. *For a set $\mathcal{C}$ of line segments and/or constant complexity polylines, and a pair of points s and t, let $\overline{G}$ be the corresponding intersection graph in the homology cover. Then, single-source shortest-paths in $\overline{G}$ can be computed in $\widetilde{\mathcal{O}}(n^{\frac{4}{3}})$ time, or $\widetilde{\mathcal{O}}(n)$ time if the segments/polylines are rectilinear.*

Rectilinear line segments and constant complexity rectilinear polylines For rectilinear line segments in the plane, an even faster approach to single-source shortest path is known. In particular, Chan and Skrepetos [CS17, CS19] considered the **decremental intersection detection** problem, in which a set of n objects is given, and then a sequence of $O(n)$ queries/updates are given, where each update deletes an object, and each query provides an object to test for intersection with any element of the set. If this problem can be solved in $T(n)$ time for a class of objects, then Chan and Skrepetos showed that a breadth-first search (i.e. an unweighted single-source shortest-path computation) can be performed in a geometric intersection graph for the same class of objects. As Chan and Skrepetos observe, for axis-aligned (rectilinear) segments, this problem

can be reduced to decremental ray-shooting queries, and thus can be solved in $\mathcal{O}(n \log n)$ time by known results [GK07, GK09, Ble08]. Note that this also extends to rectilinear polylines of constant complexity, since each polyline can be broken into its components. Using a straightforward method of Spalding-Jamieson and Naredla, we can extend this result to the homology cover:

Theorem 7.4. *For a set $\mathcal{C}$ of rectilinear line segments and/or constant complexity rectilinear polylines, and a pair of points s and t, let $\overline{G}$ be the corresponding intersection graph in the homology cover. Then, single-source shortest-paths in $\overline{G}$ can be computed in $\mathcal{O}(n \log n)$ time.*

Proof. It suffices to solve decremental intersection detection among line segments in the homology cover. As Spalding-Jamieson and Naredla observe [SJN25b, Appendix C], intersection tests involving n line segments in the homology cover can be reduced into pairs of intersection tests involving at most $2n$ line segments among two copies of the plane, so we can apply the method of Chan and Skrepetos [CS17, CS19] in each copy of the plane. □

7.1 Applications to Point-Separation Approximations

We briefly outline the derivations for the time complexities stated in Table 1. Each follows from combining a result with each of Theorem 7.2, Theorem 7.3, and Theorem 7.4.

The first collection follows from combining with Theorem 4.1, the second with Theorem 5.1, and the third with Theorem 6.1.

Note that, for the purposes of point-separation, any object defined by a closed curve not enclosing s or t can be assumed to contain its interior without changing the size of the optimum solution. Moreover, a convex polygon intersecting $\overline{st}$ can only enclose at most one of s or t (by Jordan curve theorem and convexity), and we have assumed (algorithmically, at the start of Section 2) that no single object separates s and t. Hence, we can assume that convex polygon objects are defined in any of the following two ways, and the point-separation solutions will be equal:

- No convex polygon contains its interior.
- All convex polygons not containing s or t contain their interiors.
- Only convex polygons intersecting the segment $\overline{st}$ contain their interiors.

Notably, these are *all* more general than the case of all convex polygons containing their interiors, since it allows for convex polygons that contain *both* s and t. This suffices for the convexity requirements of Theorem 4.1 and Theorem 5.1.

8 Conclusion and Open Questions

In this paper we provided several new subquadratic approximation algorithms for the point-separation problem with line segments, convex polygons, and disks. We gave two different approximation algorithms for the point-separation problem with convex objects, and one for more general objects. These rely on efficient single-source shortest-path calculations in homology covers.

These results leave a number of open questions. Chief among these is understanding the cases for which there are faster *exact* algorithms for the point-separation problem:

Open Question 8.1. *Is there a natural (non-trivial) class of objects for which point-separation can be solved exactly in truly subquadratic time?*

We use the term "non-trivial" since there are classes like non-crossing line segments (where the problem simplifies to planar minimum (s,t)-cut) for which this is already known.

Next, although these approximation algorithms break the subquadratic barrier of previous methods, it seems like there is room for improvement. In particular, the parameterized approximation scheme admits quasi-linear constant factor approximations, which offers hope that the polynomial overhead for the additive approximation algorithm might be reduced:

Open Question 8.2. *Are there approximation algorithms for the point-separation problem with small additive error that run in quasi-linear time?*

Additional useful possible improvements include the derandomization of both the additive approximation scheme and the SSSP:

Open Question 8.3. *Can the additive approximation scheme of Theorem 4.1 be derandomized?*

For our approximation schemes to run in subquadratic time, we currently need our shortest-path queries to run in time $\mathcal{O}(n^{1.5-\varepsilon})$ for any constant ε. Thus new results in geometric intersection graphs may also improve algorithms for point-separation.

Open Question 8.4. *For what classes of objects can the single-source shortest-path problem through the geometric intersection graph in the homology cover be solved in $\mathcal{O}(n^{1.5-\varepsilon})$ time, for a constant $\varepsilon > 0$?*

So far we have been able to extend prior algorithms from geometric intersection graphs in the plane to those in the homology cover. For the most part, these extensions are fairly natural. This brings up a natural question of whether these problems are algorithmically different for any types of geometric intersection graphs:

Open Question 8.5. *Are there any objects for which the computational complexity of finding the single-source shortest path in geometric intersection graphs in the plane is different from that of geometric intersection graphs in the homology cover?*

As briefly mentioned in the introduction, Spalding-Jamieson and Naredla [SJN25a] gave the first conditional lower bounds for point-separation with a number of classes of geometric objects [SJN25a]. In particular, they give an $\Omega\left(n^{\frac{3}{2}-\varepsilon}\right)$ lower bound for (unweighted) line segments, and one of their lower bounds could also be very slightly modified to become an $\Omega\left(n^{\frac{3}{2}-\varepsilon}\right)$ lower bound for axis-aligned rectangles. Their conditional lower bounds are for *exact* point-separation, and do not seem to naturally extend to approximations. Hence, we ask if a comparable result is possible:

Open Question 8.6. *Are there any object types for which non-trivial fine-grained lower bounds on* approximating *point-separation can be obtained?*

Lastly, although Theorem 4.1 and Theorem 5.1 compute many shortest-paths, only some of these computations need to explicitly produce the path. For the rest of them, only the *length* of the path is required. Thus, if an exact distance-oracle for $\overline{G}$ could be computed efficiently, we could speed up these algorithms:

Open Question 8.7. *Are there any object types for which an exact distance-oracle of the intersection graph in the homology cover $\overline{G}$ can be computed in $\widetilde{\mathcal{O}}\left(n^{\frac{3}{2}}\right)$ time?*

Note that an exact distance oracle that can be computed in $\mathcal{O}\left(n^{2-\varepsilon}\right)$-time would also provide an improved algorithm for *exact* point-separation. Recently, Chan, Chang, Gao, Kisfaludi-Bak, Le, and Zheng [CCG+25] have developed such an algorithm for unit disks in the plane, although it is not yet clear if their algorithm could be modified to operate in the homology cover. Approximate distance-oracles could also provide interesting results.

References

[AOX25] Shinwoo An, Eunjin Oh, and Jie Xue. Single-Source Shortest Path Problem in Weighted Disk Graphs. In Oswin Aichholzer and Haitao Wang, editors, *41st International Symposium on Computational Geometry (SoCG 2025)*, volume 332 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 7:1–7:15, Dagstuhl, Germany, 2025. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[BKBK+22] Karl Bringmann, Sándor Kisfaludi-Bak, Marvin Künnemann, André Nusser, and Zahra Parsaeian. Towards sub-quadratic diameter computation in geometric intersection graphs. In *38th International Symposium on Computational Geometry*, pages 1–16. Schloss Dagstuhl, 2022.

[Ble08] Guy E Blelloch. Space-efficient dynamic orthogonal point location, segment intersection, and range reporting. In *SODA*, volume 8, pages 894–903, 2008.

[BW24] Bruce W Brewer and Haitao Wang. An improved algorithm for shortest paths in weighted unit-disk graphs. In Rahnuma Islam Nisha, editor, *Proceedings of the 36th Canadian Conference on Computational Geometry, CCCG 2024, Brock University, St. Catharines, Ontario, Canada, July 17 - August 19, 2024*, pages 57–64, 2024.

[BW25] Bruce W. Brewer and Haitao Wang. An optimal algorithm for shortest paths in unweighted disk graphs, 2025.

[BWB25] Haitao Wang Bruce W. Brewer. An optimal algorithm for shortest paths in unweighted disk graphs. *33rd Annual European Symposium on Algorithms (ESA 2025)*, 2025. To appear.

[CCG+25] Timothy M. Chan, Hsien-Chih Chang, Jie Gao, Sándor Kisfaludi-Bak, Hung Le, and Da Wei Zheng. Truly subquadratic time algorithms for diameter and related problems in graphs of bounded vc-dimension. In *Proceedings of the 66th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2025. To appear.

[CEFN23] Erin W. Chambers, Jeff Erickson, Kyle Fox, and Amir Nayyeri. Minimum cuts in surface graphs. *SIAM J. Comput.*, 52(1):156–195, 2023.

[CG16] Sergio Cabello and Panos Giannopoulos. The complexity of separating points in the plane. *Algorithmica*, 74(2):643–663, 2016.

[CGL24] Hsien-Chih Chang, Jie Gao, and Hung Le. Computing Diameter+2 in Truly-Subquadratic Time for Unit-Disk Graphs. In Wolfgang Mulzer and Jeff M. Phillips, editors, *40th International Symposium on Computational Geometry (SoCG 2024)*, volume 293 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 38:1–38:14, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[CH25] Timothy M. Chan and Zhengcheng Huang. Faster Algorithms for Reverse Shortest Path in Unit-Disk Graphs and Related Geometric Optimization Problems: Improving the Shrink-And-Bifurcate Technique. In Oswin Aichholzer and Haitao Wang, editors, *41st International Symposium on Computational Geometry (SoCG 2025)*, volume 332 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 32:1–32:14, Dagstuhl, Germany, 2025. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[Cha23] Timothy M Chan. Finding triangles and other small subgraphs in geometric intersection graphs. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1777–1805. SIAM, 2023.

[CM18] Sergio Cabello and Lazar Milinković. Two optimization problems for unit disks. *Comput. Geom.*, 70-71:1–12, 2018.

[CS16] Timothy M Chan and Dimitrios Skrepetos. All-pairs shortest paths in unit-disk graphs in slightly subquadratic time. In *27th International Symposium on Algorithms and Computation (ISAAC 2016)*, pages 24–1. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2016.

[CS17] Timothy M. Chan and Dimitrios Skrepetos. All-pairs shortest paths in geometric intersection graphs. In Faith Ellen, Antonina Kolokolova, and Jörg-Rüdiger Sack, editors, *Algorithms and Data Structures - 15th International Symposium, WADS 2017, St. John's, NL, Canada, July 31 - August 2, 2017, Proceedings*, volume 10389 of *Lecture Notes in Computer Science*, pages 253–264. Springer, 2017.

[CS19] Timothy M. Chan and Dimitrios Skrepetos. All-pairs shortest paths in geometric intersection graphs. *J. Comput. Geom.*, 10(1):27–41, 2019.

[dBC25a] Mark de Berg and Sergio Cabello. An $o(n \log n)$ algorithm for single-source shortest paths in disk graphs. *33rd Annual European Symposium on Algorithms (ESA 2025)*, 2025. To appear.

[dBC25b] Mark de Berg and Sergio Cabello. An $o(n \log n)$ algorithm for single-source shortest paths in disk graphs, 2025.

[DKP24] Lech Duraj, Filip Konieczny, and Krzysztof Potkepa. Better Diameter Algorithms for Bounded VC-Dimension Graphs and Geometric Intersection Graphs. In Timothy Chan, Johannes Fischer, John Iacono, and Grzegorz Herman, editors, *32nd Annual European Symposium on Algorithms (ESA 2024)*, volume 308 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 51:1–51:18, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[dLdSV23] Alane de Lima, Murilo da Silva, and André Vignatti. A range space with constant vc dimension for all-pairs shortest paths in graphs. *Journal of Graph Algorithms and Applications*, 27(7):603–619, 2023.

[FJF56] Lester R Ford Jr and Delbert R Fulkerson. Maximal flow through a network. *Canadian journal of Mathematics*, 8:399–404, 1956.

[GK07] Yoav Giyora and Haim Kaplan. Optimal dynamic vertical ray shooting in rectilinear planar subdivisions. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 19–28, 2007.

[GK09] Yoav Giyora and Haim Kaplan. Optimal dynamic vertical ray shooting in rectilinear planar subdivisions. *ACM Transactions on Algorithms (TALG)*, 5(3):1–51, 2009.

[GKV11] Matt Gibson, Gaurav Kanade, and Kasturi R. Varadarajan. On isolating points using disks. In Camil Demetrescu and Magnús M. Halldórsson, editors, *Algorithms - ESA 2011 - 19th Annual European Symposium, Saarbrücken, Germany, September 5-9, 2011. Proceedings*, volume 6942 of *Lecture Notes in Computer Science*, pages 61–69. Springer, 2011.

[HHZ24] Elfarouk Harb, Zhengcheng Huang, and Da Wei Zheng. Shortest Path Separators in Unit Disk Graphs. In Timothy Chan, Johannes Fischer, John Iacono, and Grzegorz Herman, editors, *32nd Annual European Symposium on Algorithms (ESA 2024)*, volume 308 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 66:1–66:14, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[IS79] Alon Itai and Yossi Shiloach. Maximum flow in planar networks. *SIAM Journal on Computing*, 8(2):135–150, 1979.

[Klo23] Katharina Klost. An algorithmic framework for the single source shortest path problem with applications to disk graphs. *Computational Geometry*, 111:101979, 2023.

[KLS+22] Neeraj Kumar, Daniel Lokshtanov, Saket Saurabh, Subhash Suri, and Jie Xue. Point separation and obstacle removal by finding and hitting odd cycles. In Xavier Goaoc and Michael Kerber, editors, *38th International Symposium on Computational Geometry, SoCG 2022, June 7-10, 2022, Berlin, Germany*, volume 224 of *LIPIcs*, pages 52:1–52:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

[KLSS21] Neeraj Kumar, Daniel Lokshtanov, Saket Saurabh, and Subhash Suri. A constant factor approximation for navigating through connected obstacles in the plane. In Dániel Marx, editor, *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 822–839. SIAM, 2021.

[Rei83] John H Reif. Minimum s-t cut of a planar undirected network in $o(n \log^2(n))$ time. *SIAM Journal on Computing*, 12(1):71–81, 1983.

[SJN25a] Jack Spalding-Jamieson and Anurag Murty Naredla. Separating two points with obstacles in the plane: Improved upper and lower bounds. *33rd Annual European Symposium on Algorithms (ESA 2025)*, 2025. To appear.

[SJN25b] Jack Spalding-Jamieson and Anurag Murty Naredla. Separating two points with obstacles in the plane: Improved upper and lower bounds, 2025.

[Skr18] Dimitrios Skrepetos. *Shortest Paths in Geometric Intersection Graphs*. PhD thesis, University of Waterloo, 2018.

[WX20] Haitao Wang and Jie Xue. Near-optimal algorithms for shortest paths in weighted unit-disk graphs. *Discrete & Computational Geometry*, 64(4):1141–1166, 2020.

[WZ22] Haitao Wang and Yiming Zhao. Reverse shortest path problem in weighted unit-disk graphs. In *International Conference and Workshops on Algorithms and Computation*, pages 135–146. Springer, 2022.

[WZ23] Haitao Wang and Yiming Zhao. Reverse shortest path problem for unit-disk graphs. *Journal of Computational Geometry*, 14(1):14–47, 2023.

2-Face Path Unfolding the Tesseract into a Font

Soham Samanta
Greater Commonwealth Virtual School
Greenfield, MA, USA
soham2020sam@gmail.com

Hugo A. Akitaya
Miner School of Computer & Information Sciences
University of Massachusetts at Lowell
Lowell, MA, USA
hugo_akitaya@uml.edu

Erik D. Demaine
Computer Science & Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, MA, USA
edemaine@mit.edu

Martin L. Demaine
Computer Science & Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, MA, USA
mdemaine@mit.edu

***Abstract*—We study a special class of *2-face unfoldings* of the tesseract (4D hypercube) where we cut edges of its 2-skeleton and lay down its 24 square faces on the plane, forming a nonoverlapping polyomino. We explore the family of 2-face unfoldings with the maximum diameter: unfoldings whose dual graph is a path, commonly called *path unfoldings*. We implemented an algorithm to test whether a given 24-omino is a path unfolding of the 2-skeleton of the hypercube. We used this software to design a geometric font by selecting unfoldings that resemble Latin letters and digits, illustrating the intersection of computational geometry and typographic design.**

***Index Terms*—unfolding, polyominoes, tesseract, font**

I. Introduction

Unfolding polyhedra is an active area of research in discrete and computational geometry [5]. In the most famous edge-unfolding problem, the goal is to cut a subset of the edges of the given polyhedron and unfold the surface so that the faces remain connected and they lie on the plane without overlap. A typical example is the 11 edge unfoldings of the cube. We say that two faces *strongly overlap* if their interiors overlap in the unfolding. If two faces overlap only at the edges that were cut, we say that they *weakly overlap*. In this project, we allow unfoldings with weak overlaps.

We can generalize unfolding to higher-dimensional polytopes. A natural way to generalize edge unfolding is *ridge unfolding*: cut the $(d-2)$-dimensional faces (ridges) of a d-dimensional polytope and lay down the $(d-1)$-dimensional faces (facets) flat in the plane. Turney [7] showed that there are exactly 261 different (polycube) unfoldings of the tesseract (4D hypercube). Diaz and O'Rourke [6] investigated the 2D (polyomino) unfoldings of these 3D (polycube) unfoldings of the tesseract. In these unfoldings, the same square face of the tesseract appears either twice in the polyomino unfolding or not at all depending on whether or not they are in the surface of the polycube unfolding.

Recently, Akitaya and Kandarpa [2] defined an alternative definition that unfolds a d-dimensional polytope to 2D directly, called "2-face unfoldings". The *2-skeleton* is the shape that we get from the 2D faces of the polytope, gluing them together at their shared edges. A *2-face unfolding* cuts some of these shared edges in such a way that the 2-skeleton unfolds to a connected nonoverlapping 2D polyomino. For a tesseract, all 24 square faces appear exactly once, so we get a 24-omino. Akitaya and Kandarpa used this approach to list a family of such unfoldings of the tesseract via a brute-force algorithm. However, their approach could not list 2-face unfoldings whose dual graph has a large diameter.

In this paper, we focus on 2-face unfoldings that are *path unfoldings* where the dual graph forms a Hamiltonian path on the square faces of the tesseract. These dual graphs have the maximum possible diameter, placing us at the opposite extreme of Akitaya and Kandarpa [2]. The number of 2-face path unfoldings is likely very large; ignoring overlap, the number of Hamiltonian paths in the 2-face adjacency graph is over 16 trillion. In Section II, we show some structural properties of such unfoldings.

To go beyond alternating unfoldings, we also implemented an algorithm to check whether a given 24-omino is a 2-face path unfolding of the tesseract. In Section III, we describe this algorithm as well as the design of a font

of 36 2-face path unfoldings that look like the letters A–Z and digits 0–9.

II. Tesseract and Its Unfoldings

The d-*cube* is a d-dimensional hypercube. Consider the projection of the 4-cube shown in Figure 1. The tesseract has eight 3D cube faces, which consist of four opposite pairs that share no square face. We pick one such opposite pair and call them the *top* and *bottom* cubes. Call $ABCDabcd$ the *top cube* and $EFGHefgh$ the *bottom cube*. Twelve faces (2-cubes) $ADHE, CDHG, BCGF, ABFE, adhe, cdhg, bcgf, abfe, AaeE, DdhH, CcgG, BbfF$ are not part of the top or bottom cubes, so they form a cut between these cubes; we call them *side faces*.

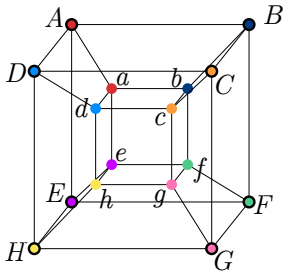


Fig. 1: Naming convention for the vertices of a tesseract (4D hypercube).

The *dual graph* of the 4-cube as is face adjacency graph, i.e., its vertices are the 2D faces of the 4-cube, and two faces are adjacent if they share a 1D edge. Given a Hamiltonian path P of the dual graph of the 4-cube, the side faces induce a set of subpaths. We call each of these subpaths a *traversal* of the side faces. If there are m such subpaths, we say that P traverses the side faces m times. We use a similar definition for the top and bottom cubes. In previous work, a subset of the authors experimentally enumerated all 35,520 2-face path unfoldings in the subclass of path unfoldings that alternate between visiting faces of the top and bottom cubes (six times each), with a visit to a single *side* face in between. We call this class the *alternating* 2-face path unfoldings of the tesseract.

III. Checking Path Unfoldings and Tesseract Font

To illustrate the breadth of path 2-face unfoldings of the tesseract, we designed a font of 36 such unfoldings that look like each letter A–Z and digit 0–9. Although we have enumerated the tens of thousands alternating path 2-face unfoldings, it is also difficult to sift through them all, and we still have not generated all path 2-face unfoldings. Thus we implemented an algorithm to check whether a given 24-omino is a path 2-face unfolding of the tesseract. This algorithm uses recursion and backtracking, trying to find Hamiltonian a path in the dual of the given polyomino while simultaneously walking on the face-adjacency graph of the tesseract. Then we used SVG Painter [4] to hand-draw possible unfoldings in the shape of a desired symbol, input the resulting SVG into our program to check whether it is a valid path unfolding, and repeated until we found a good-looking unfolding.

Figure 2 shows the resulting font. Can you figure out how to fold each unfolding into the 2-faces of the tesseract? We encourage the reader to try this puzzle before looking at the labels in Figure 3.

IV. Future Work

Our objective is to enumerate all path 2-face unfoldings of the tesseract, extending beyond the alternating case. To achieve this, we plan to use Zero-suppressed Decision Diagrams (ZDDs), a method previously employed to enumerate non-overlapping edge unfoldings of polyhedra [3]. In addition to this enumeration, we aim to construct a new FONT based on **general** edge unfoldings of the tesseract, which may yield Latin letters with more visually appealing forms.

V. Code Availability

All code from this project can be found at https://github.com/Soham2020sam/PathUnfoldingsTesseract.

Acknowledgments

Research supported in part by the NSF award CCF-2348067. We thank participants in Demaine's MIT class "Geometric Folding Algorithms: Linkages, Origami, Polyhedra" for helpful discussions on this project.

References

[1] H. A. Akitaya and S. Samanta, "Path-Unfolding the Tesseract" in Proc. 31st Fall Workshop on Computational Geometry, 2024.

[2] H. A. Akitaya and N. V. Kandarpa, "Unfolding Skeletons," in Proc. 24th Japan Conference on Discrete and Computational Geometry, Graphs, and Games, 2022.

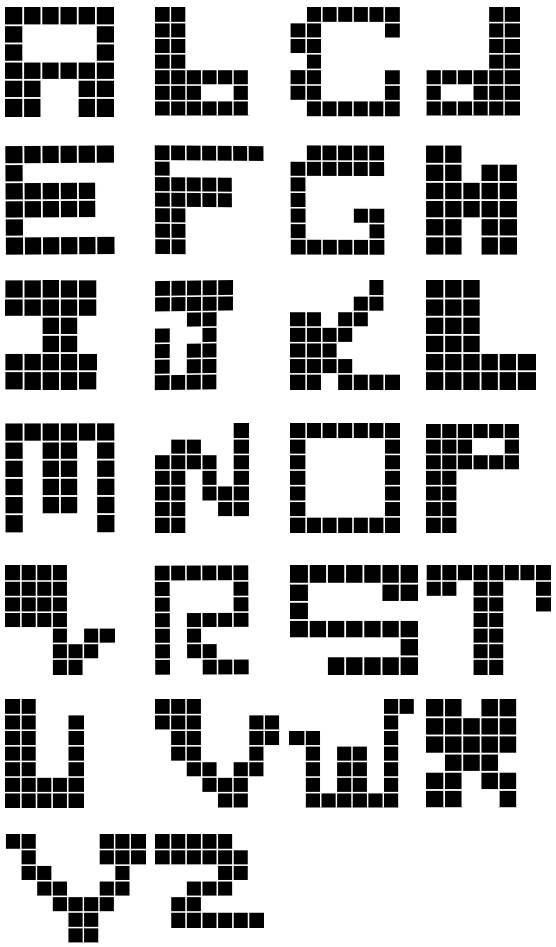
[3] T. Shiota (2024). Overlapping and Non-overlapping Unfoldings in Convex Polyhedra. Kyushu Institute of Technology.

[4] E. D. Demaine. https://erikdemaine.org/svgpainter/

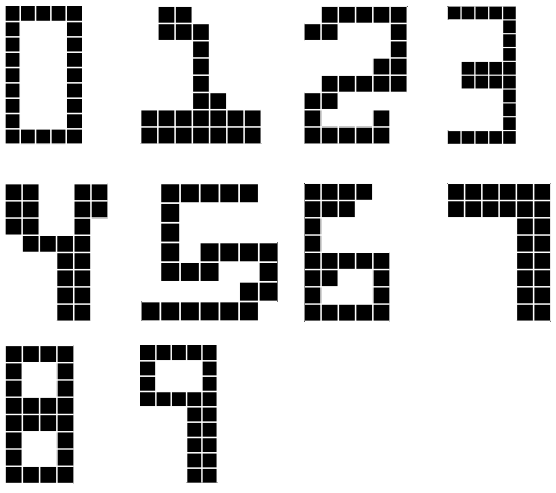
[5] E. D. Demaine and J. O'Rourke, "Geometric Folding Algorithms: Linkages, Origami, Polyhedra", Cambridge Univ. Press, 2007.

[6] G. Diaz and J. O'Rourke, "Hypercube Unfoldings that Tile $\mathbb{R}^3$ and $\mathbb{R}^2$," arXiv preprint arXiv:1512.02086, 2015. https://arxiv.org/abs/1512.02086

[7] P. D. Turney, "Unfolding the tesseract," Journal of Recreational Mathematics, vol. 17, no. 1, pp. 1–16, 1984.

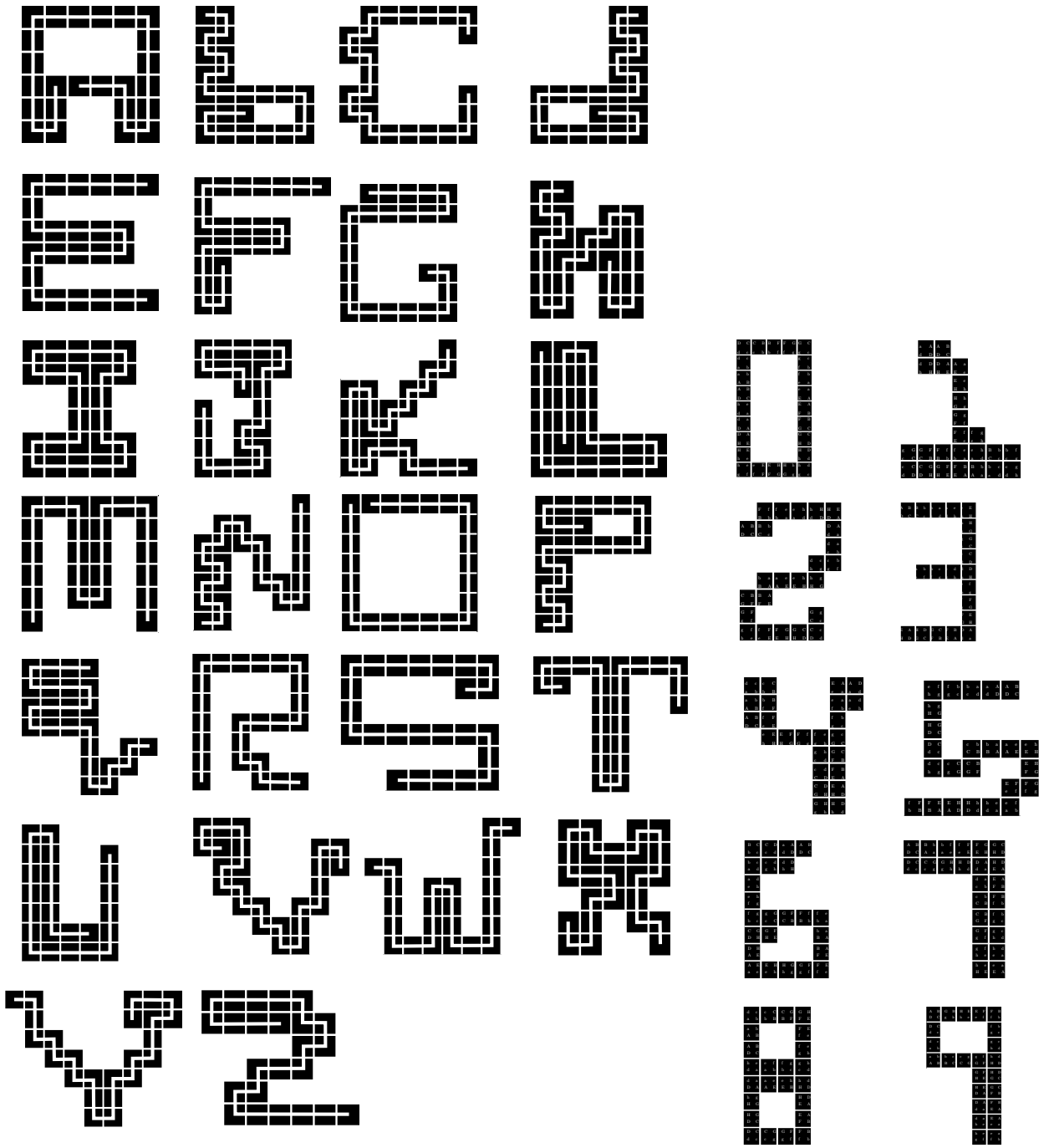


(a) A–Z



(b) 0–9

Fig. 2: Tesseract font. Each letter/digit is represented by a path 2-face unfolding of the tesseract.



(a) A–Z (b) 0–9

Fig. 3: Tesseract font with labels indicating vertices.

Statistical Analysis of Persistence Landmarks: A Hilbert Space Embedding of the Space of Persistence Diagrams

Pramita Bagchi[1], Sushovan Majhi[2], Atish Mitra[3], and Žiga Virk[4]

[1]Biostatistics and Bioinformatics, George Washington University, USA (pramita.bagchi@gwu.edu)
[2]Data Science, George Washington University, USA (s.majhi@gwu.edu)
[3]Department of Mathematical Sciences, Montana Technological University, USA (amitra@mtech.edu)
[4]Faculty of Computer and Information Science, University of Ljubljana, Slovenia (ziga.virk@fri.uni-lj.si)

Abstract

Persistence Landmarks are explicit geometrically motivated 1-Lipschitz maps from the space of persistence diagrams on n points (equipped with the Bottleneck distance) into Hilbert space, with quantified lower bounds on distortion. Such embeddings allow us to control the amount of geometric information lost due to their use as a topological summary. Since our summary lies in Hilbert space, it is easily adaptable to tools in statistics and machine learning. Viewed as a Hilbert space valued random variable, this summary obeys a strong law of large numbers and a central limit theorem, and standard statistical tests can be used for hypothesis testing using this summary.

1 Introduction

Motivation. The applications of persistent homology have been burdened with the difficulty of combining its main signature—persistence diagrams—with statistics and machine learning. While persistent homology as a construction is backed by a strong theoretical background (such as stability theorems) and practical applications in the context of data science, one of the major disadvantages to an even more widespread application is the fact that the persistence diagrams—the natural outputs of persistent homology computations—are not subsets of a Hilbert space. This is problematic since the approaches of statistics and general data analysis typically require the structure of a finite-dimensional vector space or a Hilbert space in order to apply the corresponding standard tools. As a result, there have been multiple approaches to mapping the space of persistence diagrams into a Hilbert space or a finite-dimensional Euclidean space, for example: the influential work on Fréchet means of distributions of persistence diagrams in [12], and subsequently [11], [1], [4]. For a comparative review of some of these approaches, see [3] and [2].

These approaches are typically Lipschitz maps from the space of persistence diagrams to Hilbert or Euclidean space. More specifically, if f is such a map, then $d(f(A), f(B)) \leq L \cdot d_{\mathcal{B}}(A, B)$ where d is the standard Euclidean or ℓ_2 distance, $L > 0$, and $d_{\mathcal{B}}$ is the bottleneck distance between persistence diagrams A and B. On the other hand, while many of these approaches construct an injective map f, there has been a **notable absence of quantified distortion** with respect to the Bottleneck distance, i.e., a map ψ implying $d(f(A), f(B)) \geq \psi(d_{\mathcal{B}}(A, B))$. The absence of a lower bound on distortion meant that we have no control over the amount of information lost through f. In particular, none of the mentioned approaches explains how to distinguish the images of persistence diagrams with the Bottleneck distance to at least a chosen threshold. Putting it differently, there was no proven control on the discriminative properties of the mentioned approaches.

The first such lower bound was established by the third and fourth named authors, who showed the existence of a **cover-based embedding coordinatized by landmarks** into the Hilbert space of the space of persistence diagrams $\mathcal{D}_n$ on n points (with the bottleneck distance) [7]. In a subsequent paper [8], the authors extended this existence result to all scales and constructed explicit maps and explicit distortion functions. We call this embedding a **persistence landmark**. The present work shows an actual implementation of the landmark embedding and does a statistical analysis—as described below.

Statistical Analysis. The above-mentioned cover-based embedding (persistence landmark) should satisfy the following desirable properties (compare with [4]): Considering the data to be sampled from some underlying abstract probability space, and applying the above-mentioned constructions, we can consider our topological summary (as defined by the embedding into Hilbert or Euclidean space) to be a random variable with values in some summary space $\mathcal{S}$. Let $X_1, \cdots .X_n$ be a sample of independent random variables with the same distribution as X. We would like to describe a notion of the mean μ of X and the mean $\overline{X}_n$ of the sample, and show that $\overline{X}_n$ converges to μ. We would like to have quantified $\overline{X}_n - \mu$, and be able to estimate the confidence intervals related to μ. Moreover, given two such samples for random variables X and Y with values in our summary space, we would like to be able to test the hypothesis that $\mu_X = \mu_Y$. In this project, we show that the cover-based persistence landmark embedding achieves these desired properties.

2 Preliminaries

This section provides some preliminary notation and definitions used throughout the paper.

Metric Geometry. The following definitions summarize the various concepts of embeddings that we use.

Let $f : X \to Y$ be a function between metric spaces.

1. f is said to be **Lipschitz** if there is $\Lambda > 0$ such that $d_Y(f(x_1), f(x_2)) \leq \Lambda \cdot d_X(x_1, x_2)$. We occasionally call such a map Λ-Lipschitz.

2. f is said to be a **coarse embedding** if there are non-decreasing functions $\rho_-, \rho_+ \colon [0, \infty) \to [0, \infty)$ with $\rho_-(d_X(x_1, x_2)) \leq d_Y(f(x_1), f(x_2)) \leq \rho_+(d_X(x_1, x_2))$ and with $\lim_{t\to\infty} \rho_-(t) = \infty$.

3. A coarse embedding f is said to be a **quasi-isometric** embedding if the functions $\rho_{-,+}$ are linear.

4. A quasi-isometric embedding is said to be a **bi-Lipschitz** embedding if the functions $\rho_{-,+}$ are dilations; In particular, $\rho_-(t) = a \cdot t$ and $\rho_+(t) = b \cdot t$, where $a, b > 0$.

5. f is said to be a **uniform embedding** if there are non-decreasing functions $\rho_-, \rho_+ \colon [0, \infty) \to [0, \infty)$ with $\rho_-(d_X(x_1, x_2)) \leq d_Y(f(x_1), f(x_2)) \leq \rho_+(d_X(x_1, x_2))$, with $\lim_{t\to 0^+} \rho_+(t) = 0$ and with $\rho_-(t) > 0$ whenever $t > 0$.

Persistence Diagrams. In this subsection, we recall the structure of the space of persistence diagrams. We take a slightly non-standard approach (for more details see [7], [8]). $\mathcal{D}_1$ represents the **space of persistence diagrams** on 1 point. As a set it equals $\mathcal{D}_1 = T \cup \{\Delta\}$ where $T = \{(x_1, x_2) \in \mathbb{R}^2 \mid x_2 > x_1 \geq 0\}$, while Δ is an additional point representing the diagonal $\{(x, x) \in \mathbb{R}^2 \mid x \geq 0\}$. On the space $\mathcal{D}_1$ we define the **Bottleneck distance** $d_{\mathcal{B}}$ as the minimum of two terms. The first term is the d_∞ distance between the points and corresponds to the matching between the points in the standard definition of the bottleneck metric. The second term corresponds to the matching of both points to the diagonal Δ.

Now we fix $n \in \{1, 2, \ldots\}$. To define a metric on the space of persistence diagrams on n points, we define the usual max metric $d^n_{\mathcal{B}}$ on the product space $(\mathcal{D}^n_1, d^n_{\mathcal{B}})$.

The symmetric group on n elements, S_n, acts on $(\mathcal{D}^n_1, d^n_{\mathcal{B}})$ by permutations of coordinates. The **space of persistence diagrams** on n points, $(\mathcal{D}_n, d_{\mathcal{B}})$, is the quotient $(\mathcal{D}^n_1, d^n_{\mathcal{B}})/S_n$:

1. Elements of $\mathcal{D}_n$ are orbits of the S_n action on $\mathcal{D}^n_1$. In particular, elements of $\mathcal{D}_n$ are n-tuples $[\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n]$ with identifications $[\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n] = [\bar{y}_1, \bar{y}_2, \ldots, \bar{y}_n]$ iff $\exists \psi \in S_n : \bar{x}_i = \bar{y}_{\psi(i)}, \forall i$. We will often think of persistence diagrams as multisets, i.e., collections of n points from $\mathcal{D}_1$ with potential repetitions.

2. The metric $d_{\mathcal{B}}$ is defined as the usual quotient metric.

3 Notes on the Embedding

The Main Construction. The persistence landmark-based embedding is constructed in two stages. First, the embedding is constructed at one scale, on the space $\mathcal{D}_1$, and then assembled into a map to Hilbert space. Subsequently, this construction is extended to $\mathcal{D}_n$. As the details of this stage-wise construction are substantial, we omit those details and refer the reader to [8] for a comprehensive explanation.

Example Implementation: Maps to Euclidean Space for Bounded Domains. Given a set of persistence diagrams on n points such that the points are enclosed in a bounded domain and the embedding needs a specified scale of discrimination, we can adjust the embedding constructed in the earlier sections to an explicit quasi-isometric 1-Lipschitz embedding to a Euclidean space of a specific dimension. Suppose the given collection of persistence diagrams consists of points that have coordinates in $[0, M]$, and that the resolution required is m (i.e., we do not need to distinguish between coordinates of persistence diagrams that are less than m distance apart). We choose an appropriate finite set of scales $\{R_1, \cdots, R_N\}$ and corresponding weights $\{w_1, \cdots, w_N\}$ - with the values (to be determined later). This gives an embedding $\varphi_3 : \mathcal{D}_n \to \mathbb{R}^N$ defined as $\varphi_3 = (w_k \varphi_{R_k})$, which is Lipschitz and whose distortion is explicitly determined by the choice of the set of scales and weights.

Let $L > 0$ denote the frame size, let $0 < R_1 < R_2 < \ldots < R_N \leq L$ be a chosen sequence of scales, and assume $\bar{w} = (w_1, \ldots, w_N)$ is a unit vector of weights corresponding to the chosen scales. For each i let ν_i denote the number of elements of the cover $R_i\mathcal{V}$ intersecting $\mathcal{D}_n \cap [0, L]^2$. Then the map $\varphi_3 \colon \mathcal{D}_n \cap [0, L]^2 \to \mathbb{R}^{\nu_1+\nu_2+\ldots+\nu_N}$ defined as

$$\varphi_3(x) = (w_k 2^{-n} \varphi_{R_k}(x))_{k=1}^N$$

is 1-Lipschitz and satisfies the following: if $d_{\mathcal{B}}(x, y) \geq R_i$, then $\|\varphi_3(x) - \varphi_3(y)\| \geq \frac{1}{3 \cdot 2^{n+2.5}} \sqrt{\sum_{k=1}^{i} w_k{}^2 R_k{}^2}$.

Moreover, in such a case, we have an explicit linear lower bound as follows:

$$\rho_-(t) = \left[\frac{1}{3 \cdot 2^{n+2.5}} \min \left\{ \min_{2 \leq i \leq N} \left\{ \frac{\sqrt{\sum_{k=1}^{i-1} w_k{}^2 R_k{}^2}}{R_i - R_1} \right\}, \frac{\sqrt{\sum_{k=1}^{N-1} w_k{}^2 R_k{}^2 + w_N^2 L^2}}{L - R_1} \right\} \right] (t - R_1) \tag{3.1}$$

when $t > R_1$, and $\rho_-(t) = 0$ when $0 \leq t \leq R_1$.

Implementation of the Embedding. The Persistence Landmark-based embedding has been implemented in Python.

Adjusting the landmarks and grid to improve the embedding. Adjusting the landmark locations and size and shape of the grid, we have control over the distortion of the embedding.

4 Statistical Analysis of Persistence Landmarks

Following [4], we now consider a statistical analysis of the embedding in the face of a random sample from a topological space.

Let $(\Omega, \mathcal{F}, P)$ be a probability space. Let X be a random variable on Ω, with persistence landmark Ψ, a Borel random variable with values in ℓ_2. In other words, for any $\omega \in \Omega$, $X(\omega)$ denotes a realization of point cloud data and $\Psi(X(\omega))$ is the corresponding Hilbert space embedding of the persistence diagram of $X(\omega)$.

Now let $X_1, \ldots, X_n$ be i.i.d. copies of X, and let $\Psi^1, \ldots, \Psi^n$ be the corresponding persistence landmark embedding. Their mean in ℓ_2 is denoted by $\overline{\Psi}^n$. We would like to be able to say that the mean converges to the expected persistence landmark.

Convergence of Persistence Landmarks. We now state the convergence results.

Theorem 1 (Strong Law of Large Numbers for persistence embeddings). $\overline{\Psi}^n \to E(\Psi)$ *almost surely if and only if* $E\|\Psi\| < \infty$.

landmarks

Proof. Apply the Central Limit Theorem[6] to $\Psi(X) - \mathbb{E}(\Psi(X))$. □

Next, we apply a functional to the persistence summaries to obtain a real-valued random variable that satisfies the usual central limit theorem.

Corollary 2. *Assume* $\mathbb{E}\|\Psi\| < \infty$ *and* $\mathbb{E}(\|\Psi\|^2) < \infty$. *Let*

$$Y = \|\Psi\|_2^2. \tag{4.1}$$

Let $\bar{Y}^n$ *be the mean of* $\|\Psi^1\|_2^2, \|\Psi^2\|_2^2, \ldots, \|\Psi^n\|_2^2$. *Then*

$$\sqrt{n}[\overline{Y}^n - \mathbb{E}(Y)] \xrightarrow{d} N(0, Var(Y)). \tag{4.2}$$

where $\xrightarrow{d}$ *denotes convergence in distribution and* $N(\mu, \sigma^2)$ *is the normal distribution with mean* μ *and variance* σ^2.

Confidence Intervals. Now, we obtain approximate confidence intervals for the expected values Y.

Assume that $\Psi(X)$ satisfies the conditions of Corollary 2 and that Y is a corresponding real random variable as defined in (4.1). By Corollary 2 and Slutsky's theorem, we may use the normal distribution to obtain the approximate $(1-\alpha)$ confidence interval for $\mathbb{E}(Y)$ using

$$\overline{Y}^n \pm z^* \frac{S_n}{\sqrt{n}},$$

where $S_n^2 = \frac{1}{n-1}\sum_{i=1}^n (Y_i - \overline{Y}^n)^2$, and z^* is the upper $\frac{\alpha}{2}$ critical value for the normal distribution.

Additionally, since Ψ is a Hilbert space-valued random variable, following [10] and [5], we can construct a confidence region for the mean persistence summary $\mathbb{E}(\Psi)$ using a bootstrap approach.

Hypothesis Testing. Here, we apply the above results to hypothesis testing.

Let $X_1, \ldots, X_n$ be iid copies of the random variable X and let $X_1', \ldots, X_n'$, be iid copies of the random variable X'. Assume that the corresponding persistence summaries Ψ, Ψ' lie in ℓ_2. Let Y and Y' be defined as in (4.1). Let $\mu = \mathbb{E}(Y)$ and $\mu' = \mathbb{E}(Y')$. We will test the null hypothesis that $\mu = \mu'$.

As the sample mean $\overline{Y} = \frac{1}{n}\sum_{i=1}^n Y_i$ is an unbiased estimator of μ and the sample variance $s_Y^2 = \frac{1}{n-1}\sum_{i=1}^n (Y_i - \overline{Y})^2$ is an unbiased estimator of $\mathrm{Var}(Y)$ and similarly for $\overline{Y}'$ and $s_{Y'}^2$, by Corollary 2, $\overline{Y}$ and $\overline{Y}'$ are asymptotically normal.

We use the two-sample z-test. Let

$$z = \frac{\overline{Y} - \overline{Y}'}{\sqrt{\frac{S_Y^2}{n} + \frac{S_{Y'}^2}{n'}}},$$

where the denominator is the standard error for the difference.

To compare the expectation of the persistence summaries, Hotelling's T^2 test can be used following [9]. Additionally, a permutation test can be used to compute p-values similar to [4].

5 Discussions

The nature of our work is ongoing. Using the persistence landmark cover-based embedding, we can vectorize persistence diagrams to use them for downstream machine learning tasks. Our goal is to conduct comprehensive experiments on benchmark ML datasets to compare the accuracy and computational efficiency achieved by alternative summaries, e.g., Persistence Landscapes and Persistence Images, etc. [2], [3]. Moreover, given any set of persistence diagrams arising from practical data sets, we want to create a pipeline for "learning" an optimal persistence landmark embedding by appropriate choices of landmark locations, shape, and scales of the cover elements, and weights; for example, in the bounded domain persistence landmark embedding 3.1.

References

[1] Henry Adams, Tegan Emerson, Michael Kirby, Rachel Neville, Chris Peterson, Patrick Shipman, Sofya Chepushtanova, Eric Hanson, Francis Motta, and Lori Ziegelmeier. Persistence images: A stable vector representation of persistent homology. *Journal of Machine Learning Research*, 18(8):1–35, 2017.

[2] Dashti Ali, Aras Asaad, Maria-Jose Jimenez, Vidit Nanda, Eduardo Paluzo-Hidalgo, and Manuel Soriano-Trigueros. A survey of vectorization methods in topological data analysis. *IEEE Trans. Pattern Anal. Mach. Intell.*, 45(12):14069–14080, December 2023.

[3] Danielle Barnes, Luis Polanco, and Jose A Perea. A comparative study of machine learning methods for persistence diagrams. *Front. Artif. Intell.*, 4:681174, July 2021.

[4] Peter Bubenik. Statistical topological data analysis using persistence landscapes. *J. Mach. Learn. Res.*, 16(1):77–102, January 2015.

[5] Herold Dehling, Olimjon Sh Sharipov, and Martin Wendler. Bootstrap for dependent hilbert space-valued random variables with application to von mises statistics. *Journal of Multivariate analysis*, 133:200–215, 2015.

[6] Michel Ledoux and Michel Talagrand. *Probability in Banach spaces*. Classics in Mathematics. Springer, Berlin, Germany, 2011 edition, July 2011.

[7] Atish Mitra and Žiga Virk. The space of persistence diagrams on n points coarsely embeds into hilbert space. *Proceedings of the American Mathematical Society*, 149(6):2693–2703, March 2021.

[8] Atish Mitra and Žiga Virk. Geometric embeddings of spaces of persistence diagrams with explicit distortions, 2024.

[9] Alessia Pini, Aymeric Stamm, and Simone Vantini. Hotelling's t2 in separable hilbert spaces. *Journal of Multivariate Analysis*, 167:284–305, 2018.

[10] Stefan Tappe. Linear estimators for gaussian random variables in hilbert spaces. *Theory of Probability and Mathematical Statistics*, 112:129–152, 2025.

[11] Katharine Turner. Medians of populations of persistence diagrams. *Homology Homotopy Appl.*, 22(1):255–282, 2020.

[12] Katharine Turner, Yuriy Mileyko, Sayan Mukherjee, and John Harer. Fréchet means for distributions of persistence diagrams. *Discrete Comput. Geom.*, 52(1):44–70, July 2014.

Topology-Aware Data Decomposition via the Ham Sandwich Theorem for Divide-and-Conquer Classification

Oleg Musin[1] and Pavel Snopov[1]

[1]University of Texas Rio Grande Valley, Brownsville, TX, USA

Abstract

The complexity and performance of neural networks is deeply connected to the topology of the input data. We propose a novel Divide-and-Conquer (DC) classification framework that simplifies the data's topology before learning. Our method leverages the Ham Sandwich Theorem to partition the dataset into balanced subsets. For the general case where such partitions are not unique, we introduce a selection criterion that minimizes a total persistence measure based on persistent homology. Simpler models are then trained on each topologically simplified subset. As a preliminary study, we show that even in 2D, where the cut is unique, this principled geometric partition simplifies topology and improves accuracy. Experiments on synthetic datasets show that our topology-aware decomposition allows simple networks to achieve higher classification accuracy than when trained on the full, complex dataset. This highlights the potential of using geometric data partitioning as a pre-processing step for complex learning tasks.

Introduction

The main premise of Topological Data Analysis (TDA) is that data often possesses a rich geometric and topological structure. Recent work in machine learning has shown that this structure is not just a novelty but a critical factor in model performance. The topology of data simplifies as it passes through a neural network's layers [8], and a network's ability to learn is fundamentally linked to the topological complexity of its input [4]. Highly complex data requires deeper networks, suggesting data topology is a significant bottleneck for efficient learning.

The Divide-and-Conquer (DC) paradigm, where a problem is broken into smaller sub-problems, is a technique in machine learning [9]. However, existing DC approaches typically partition the input space without considering the data's underlying topology. A random cut is unlikely to simplify the problem and may worsen it by reducing the training data for each sub-model.

In this paper, we bridge this gap by introducing a topology-aware DC framework. We propose to simplify the learning problem by first partitioning the data into topologically simpler subsets using principled geometric cuts. This is achieved by:

1. Using the **Ham Sandwich Theorem** (HST) to generate balanced partitions of the data classes.
2. When the cut is not unique (e.g., binary classification in $d > 2$), selecting the optimal partition by minimizing a **total persistence cost function**.

The classical HST guarantees a unique balanced cut when the number of classes equals the ambient dimension (d sets in $\mathbb{R}^d$). However, in our setting (binary classification in $\mathbb{R}^d$), the cut is unique only in 2D. In higher dimensions ($d > 2$), the bisecting hyperplane is not unique, and the set of all such cuts forms a manifold [6]. Our cost function allows us to search this manifold for the cut that maximally simplifies the topology.

Our experiments serve as a preliminary validation of this concept. We show that even in 2D, where the cut is unique, this single, principled geometric partition successfully simplifies the data's topology (by breaking cycles) and leads to improved classification accuracy. This provides strong evidence for the general method of using topological optimization to select cuts in higher dimensions.

Background on Mass Partitions

Mass partition problems describe the partitions we can induce on a family of measures or finite sets of points in Euclidean spaces by dividing the ambient space into pieces. One of the most famous result in this area is the Ham-Sandwich theorem alongside its discrete version:

Theorem 1 (Discrete Ham Sandwich Theorem)**.** *Let $X_1, ..., X_d$ be finite sets in $\mathbb{R}^d$. Then there exists a hyperplane that simultaneously bisects $X_1, ..., X_d$.*

There exists efficient algorithms for finding such hyperplanes in practice, e.g. when $d = 2$ there exists an algorithm that runs in $O(n)$ time, where n is the total number of points, and in general, the problem can be solved in $O(n^{d-1-\alpha(d)})$ time, where $\alpha(d) > 0$ and goes to zero as d increases [5].

Background on Persistent Homology

To measure the topological complexity, we use persistent homology (PH) [2]. PH tracks the evolution of topological features (connected components, loops, voids) over a sequence of growing simplicial complexes, known as a filtration. The result is a persistence diagram [3], a multiset of (birth, death) points summarizing the lifespan of each feature. In this work, we use the Vietoris-Rips (VR) filtration.

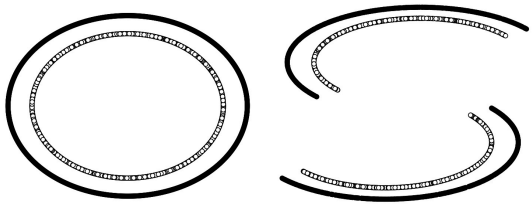
Proposed Method: Data Partitioning

We focus on the binary classification of a dataset $\mathcal{D} \subset \mathbb{R}^d$ with classes C_0 and C_1 that exhibit non-trivial topology. Our goal is to find a partition of $\mathcal{D}$ into subsets $\{\mathcal{D}_i\}$ such that their total topological complexity is lower than that of the original dataset.

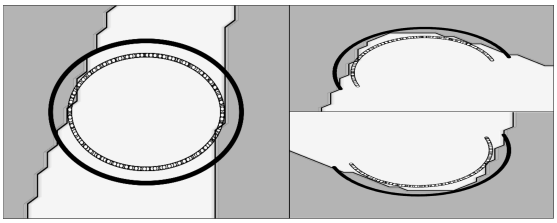
As a measure of this complexity, we use the **1D Total Persistence**, which is often used in TDA-based ML applications [8]. We define (following the notation from [8]) it as:

$$\mathrm{TC}(\mathcal{D}_i) = \sum_{\text{points in } PD_1(\mathcal{D}_i)} (\text{death} - \text{birth}),$$

where $PD_1(\mathcal{D}_i)$ is the persistence diagram of the first homology group of a subset $\mathcal{D}_i$. We expect that for a "good" partition, the total persistence $\sum_i \mathrm{TC}(\mathcal{D}_i)$ will be lower than $\mathrm{TC}(\mathcal{D})$. We focus on PD_1 as we are interested in simplifying loop-like features, which are the dominant complex structures in our test data.



(a) The partitioning of the dataset $\mathcal{D}$ into two subsets. The cut simplifies the topology by breaking the cycles.



(b) Decision boundaries. Left: Baseline FCNN fails. Right: Our 1-Split DC method succeeds.

Figure 1: Illustration of our method on the Circles dataset. (a) A Ham Sandwich cut partitions the data, simplifying its topology. (b) This allows a simple classifier to learn the decision boundary.

Finding the Optimal Hyperplane

As established in the introduction, for binary classification in $d > 2$, the Ham Sandwich cut is not unique; the set of all such cuts forms a manifold [6]. This non-uniqueness is the key to our method, as it allows us to **search for an optimal partition**.

Our proposal is to find the hyperplane h that minimizes the total persistence of the resulting subsets:

$$\min_h \left(\mathrm{TC}(\mathcal{D}_0) + \mathrm{TC}(\mathcal{D}_1)\right)$$

This optimization problem can be approached in several ways (e.g., via functional methods or different parametrizations of the cut manifold). In this preliminary work, we use a practical, iterative search method based on random projections, which is detailed in the Experiments section.

Computational Complexity

The cost of our method is a one-time pre-processing step to find the optimal cut. This is the main computational bottleneck. We must compute PD_1 for two subsets of size $\approx n/2$. The complexity is determined by the size of the VR filtration. Constructing the 2-skeleton (up to triangles) can take $O(n^3)$ time in the worst case. The subsequent matrix reduction algorithm for PH is $O(m^3)$ where m is the number of simplices. While worst-case complexity is high, modern implementations like Ripser [1] often exhibit near-linear empirical performance for low-dimensional homology.

This pre-processing cost must be weighed against the potential savings from training simpler, smaller neural networks.

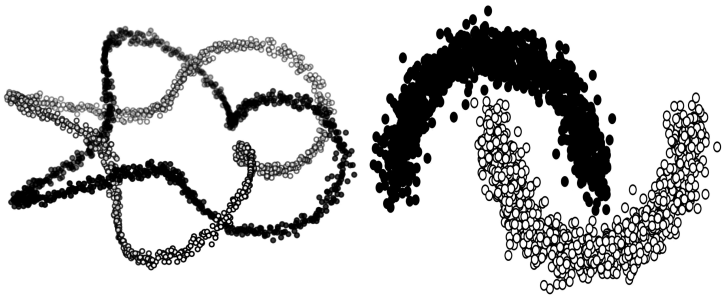


Figure 2: The synthetic datasets `CurvesOnTorus` (left) and `Moons` (right) used in the experiments.

Experiments

We evaluate our topology-aware DC approach on synthetic datasets against baseline fully-connected neural networks (FCNNs). We intentionally use simple network architectures to demonstrate that our method simplifies the problem to a degree that even these basic models can achieve high accuracy, whereas they fail on the original, complex data.

Experimental Setup

In our preliminary study, we split the dataset only once, training two smaller FCNNs on the resulting chunks of labeled data. Our method (DC (Ours)) finds its partition as follows:

1. A candidate hyperplane is generated by first projecting the high-dimensional data onto a random 2D plane.
2. The standard 2D Ham Sandwich cut is computed on the projected data.
3. This cut is lifted back to a separating hyperplane in the original space, partitioning the dataset.
4. The total persistence (TC) of this partition is calculated.

This process is repeated ($N = 100$ trials), and we select the hyperplane that yields the **minimum total persistence** as the final partition.

The Baseline FCNN is trained on the full, unpartitioned dataset. Both DC and Baseline models use the same architectures for a fair comparison, as specified by (Depth, Width) in Table 1. All models were implemented in PyTorch and trained using the Adam optimizer. We used `Ripser` for persistence computations. Models were trained for 250 epochs with a batch size of 32, and a learning rate of 1e-3.

Datasets

We used the following synthetic datasets:

- **Curves on Torus Dataset:** 3000 points sampled from two closed curves on a torus $T^2 \subset \mathbb{R}^3$. The topological invariant (number of revolutions) serves as the complexity characteristic.
- **Moons Dataset:** 2000 points sampled from two interleaving **half-circles** in $\mathbb{R}^2$.

Results and Discussion

Table 1 summarizes the test accuracy. On the `Moons` dataset (Table 1a), our method achieves near-perfect accuracy (99.73%) with a minimal (1, 2) network, demonstrating the effectiveness of the partition. The advantage is more pronounced on the `CurvesOnTorus` dataset (Table 1b), which has a more complex topology ($\beta_1 = 2$). Here, our approach reaches 96.32% accuracy with a (2, 5) configuration, significantly outperforming the baseline's 89.91%. These results show that by simplifying the data's topology first, our method enables simpler networks to solve complex classification problems more effectively.

Discussion. The results suggest that our topology-aware decomposition acts as a form of problem simplification. By partitioning the data space into topologically simpler subsets, we effectively "linearize" the learning task for the local models. This allows simple FCNNs to learn effective decision boundaries without needing the high capacity (and large amounts of data) that a single, large model would require to learn the original, complex data

Table 1: Test accuracy on all synthetic datasets (mean ± std over 10 seeds). Our topology-aware DC method consistently outperforms the baseline FCNN, especially on topologically complex data.

(a) `Moons`

Method	Config (Depth, Width)	Acc. (%)
DC (Ours)	(1, 2)	99.73 ± 0.001
DC (Ours)	(1, 3)	99.67 ± 0.002
DC (Ours)	(2, 2)	99.66 ± 0.001
Baseline	(1, 2)	88.13 ± 0.005
Baseline	(2, 2)	89.54 ± 0.038
Baseline	(1, 3)	96.77 ± 0.001

(b) `CurvesOnTorus`

Method	Config (Depth, Width)	Acc. (%)
DC (Ours)	(1, 3)	73.86 ± 0.068
DC (Ours)	(2, 3)	82.04 ± 0.081
DC (Ours)	(2, 5)	96.32 ± 0.032
Baseline	(1, 3)	71.00 ± 0.028
Baseline	(2, 3)	68.31 ± 0.075
Baseline	(2, 5)	89.91 ± 0.085

structure. Our method's strength lies in its ability to find a partition that is not just balanced, but genuinely beneficial for the learning algorithm.

However, the method has limitations. The computational cost of the search procedure makes it a heavy pre-processing step. We also acknowledge that our experiments are limited to a single cut and synthetic datasets. Future work will explore several promising directions. Firstly, we will extend this framework to multi-class problems and test its generalizability on real-world data. Secondly, instead of using only the standard Ham Sandwich theorem to generate candidate partitions, one could investigate more constrained, canonical cuts. There exist powerful extensions that guarantee additional geometric properties, for instance:

Theorem 2 ([7]). *Let $X_1, ..., X_d$ be finite sets in $\mathbb{R}^{2d}$. Then there exists a hyperplane which passes through the centers of mass $X_1, ..., X_d$ and simultaneously bisects each of these sets.*

Theorem 3 ([7]). *Let X be a finite set in $\mathbb{R}^d$. Then there exist d mutually orthogonal hyperplanes such that every pair of these hyperplanes divides X into four parts of equal parts.*

Investigating whether the additional structure imposed by such cuts (e.g., passing through centers of mass or ensuring orthogonality) provides an advantage when optimized with our topological cost function is a key research question.

References

[1] Ulrich Bauer. Ripser: efficient computation of Vietoris-Rips persistence barcodes. *J. Appl. Comput. Topol.*, 5(3):391–423, 2021. ISSN 2367-1726. doi: 10.1007/s41468-021-00071-5. URL `https://doi.org/10.1007/s41468-021-00071-5`.

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

[3] Herbert Edelsbrunner and John Harer. *Computational Topology: An Introduction*, volume 69. American Mathematical Society, 2010. doi: 10.1090/mbk/069.

[4] Ekin Ergen and Moritz Grillo. Topological Expressivity of ReLU Neural Networks. In *Proceedings of Thirty Seventh Conference on Learning Theory*, pages 1599–1642. PMLR, June 2024. URL `https://proceedings.mlr.press/v247/ergen24a.html`. ISSN: 2640-3498.

[5] Chi-Yuan Lo, J. Matoušek, and W. Steiger. Algorithms for ham-sandwich cuts. *Discrete & Computational Geometry*, 11(4):433–452, April 1994. ISSN 0179-5376, 1432-0444. doi: 10.1007/BF02574017. URL `http://link.springer.com/10.1007/BF02574017`.

[6] Oleg Musin. Borsuk-Ulam type theorems for manifolds. 140(7):2551–2560. ISSN 0002-9939, 1088-6826. doi: 10.1090/S0002-9939-2011-11094-3. URL `https://www.ams.org/proc/2012-140-07/S0002-9939-2011-11094-3/`.

[7] Oleg R. Musin. General mass partition theorems. in preparation, 2025.

[8] Gregory Naitzat, Andrey Zhitnikov, and Lek-Heng Lim. Topology of deep neural networks. *Journal of Machine Learning Research*, 21(184):1–40, 2020. URL `http://jmlr.org/papers/v21/20-345.html`.

[9] Alex Nowak, David Folqué, and Joan Bruna. Divide and conquer networks. In *International Conference on Learning Representations (ICLR)*, 2018. URL `https://openreview.net/forum?id=B1jscMbAW`.

Vietoris–Rips Shadow for Euclidean Graph Reconstruction

Rafal Komendarczyk[1], Sushovan Majhi[2], and Atish Mitra[3]

[1]Mathematics Department, Tulane University, USA; rako@tulane.edu
[2]Data Science Program, George Washington University, USA; s.majhi@gwu.edu
[3]Mathematics Department, Montana Technological University, USA; amitra@mtech.edu

Abstract

The shadow of an abstract simplicial complex $\mathcal{K}$ with vertices in $\mathbb{R}^N$ is a subset of $\mathbb{R}^N$ defined as the union of the convex hulls of simplices of $\mathcal{K}$. The Vietoris–Rips complex of a metric space $(\mathcal{S}, d)$ at scale β is an abstract simplicial complex whose each k-simplex corresponds to $(k+1)$ points of $\mathcal{S}$ within diameter β. In case $\mathcal{S} \subset \mathbb{R}^2$ and $d(a,b) = \|a-b\|$ the standard Euclidean metric, the natural shadow projection of the Vietoris–Rips complex is already proved by Chambers et al. to induce isomorphisms on π_0 and π_1. We extend the result beyond the standard Euclidean distance on $\mathcal{S} \subset \mathbb{R}^N$ to a family of path-based metrics, $d^{\varepsilon}_{\mathcal{S}}$. From the pairwise Euclidean distances of points in $\mathcal{S}$, we introduce a family (parametrized by ε) of path-based Vietoris–Rips complexes $\mathcal{R}^{\varepsilon}_{\beta}(\mathcal{S})$ for a scale $\beta > 0$. If $\mathcal{S} \subset \mathbb{R}^2$ is Hausdorff-close to a planar Euclidean graph $\mathcal{G}$, we provide quantitative bounds on scales β, ε for the shadow projection map of the Vietoris–Rips complex of $(\mathcal{S}, d^{\varepsilon}_{\mathcal{S}})$ at scale β to induce π_1-isomorphism. This paper first studies the homotopy-type recovery of $\mathcal{G} \subset \mathbb{R}^N$ using the abstract Vietoris–Rips complex of a Hausdorff-close sample $\mathcal{S}$ under the $d^{\varepsilon}_{\mathcal{S}}$ metric. Then, our result on the π_1-isomorphism induced by the shadow projection lends itself to providing also a geometrically close embedding for the reconstruction. Based on the length of the shortest loop and large-scale distortion of the embedding of $\mathcal{G}$, we quantify the choice of a suitable sample density ε and a scale β at which the shadow of $\mathcal{R}^{\varepsilon}_{\beta}(\mathcal{S})$ is homotopy-equivalent and Hausdorff-close to $\mathcal{G}$.

1 Introduction

Given a metric space $(\mathcal{S}, d_{\mathcal{S}})$ and scale $\beta > 0$, the *Vietoris–Rips complex*, denoted $\mathcal{R}_{\beta}(\mathcal{S})$, is defined as an abstract simplicial complex having a k-simplex for every subset of $\mathcal{S}$ with cardinality $(k+1)$ and diameter less than β. In the last decade, Vietoris–Rips complexes have gained considerable attention in the topological data analysis community due to their relatively straightforward computational schemes regardless of the dimension of the data, compared to some of the alternatives such as Čech and α-complexes. The theoretical understanding of the topology of Vietoris–Rips complexes at different scales is generally extremely elusive. Nonetheless, the far and wide use of Vietoris–Rips complexes—particularly in the field of shape reconstruction—can be attributed to their ability to *reconstruct* the topology of a hidden shape $\mathcal{X}$ even when constructed on a noisy sample $\mathcal{S}$ "around" $\mathcal{X}$; [9, 3, 10, 14, 15, 12].

There are many real-world applications where the unknown shape $\mathcal{X}$ and the sample $\mathcal{S}$ are hosted in a Euclidean space $\mathbb{R}^N$ within a small Hausdorff proximity. In case $\mathcal{X}$ belongs to a nice enough class of shapes, the Vietoris–Rips complex of $\mathcal{S}$ (possibly under a non-Euclidean metric) has been shown to successfully reconstruct $\mathcal{X}$ up to homotopy-type; pivotal developments include [3, 10] for compact subsets of positive reach, [14] for Euclidean submanifolds, [14] for Euclidean graphs, [12] for more general geodesic subspaces of curvature bounded above.

This paper is devoted to Euclidean graph reconstruction—both *topologically* and *geometrically*. Graph structures are ubiquitous in real-world applications. In practice, Euclidean data or *sample* $\mathcal{S} \subset \mathbb{R}^N$ sometimes approximate an (unknown) graph $\mathcal{G}$ that is realized as a subset of the same Euclidean space $\mathbb{R}^N$ with a controlled Hausdorff distance $d_H(\mathcal{S}, \mathcal{G})$. Despite the prevalence of graph structures to be recovered from noisy samples, the theoretical challenges ensuing from their vanishing reach (and μ-reach [5]) make most of the existing results for nice enough spaces untenable for geometric graphs.

Topological Reconstruction Our study of the topological reconstruction of Euclidean graphs via Vietoris–Rips complexes of the sample is inspired by the recent developments in the reconstruction of graphs [14] and general geodesic subspaces [12], using a non-Euclidean, path-based metric for the output Vietoris–Rips complexes. The sample $\mathcal{S}$ comes equipped with the Euclidean distance between pairs of points. Even when such a sample exhibits a sufficiently small Hausdorff–closeness to $\mathcal{G}$, the *Euclidean* Vietoris–Rips complex generally fails to be homotopy

equivalent to the underlying graph. Near the vertices of $\mathcal{G}$, the presence of small redundant cycles in the Euclidean Vietoris–Rips complex of $\mathcal{S}$ is often unavoidable; see Figure 2.

For this reason, the Euclidean metric on $\mathcal{S}$ is not deemed an appropriate metric for building the Vietoris–Rips complexes on $\mathcal{S}$ to obtain a topologically faithful reconstruction of the unknown graph. Instead, the authors of [8, 14, 12] considered the Vietoris–Rips complexes of the sample under a family of path-based metrics $(\mathcal{S}, d_{\mathcal{S}}^{\varepsilon})$ ([11, Definition 2.6]) in their reconstruction schemes. Under this metric, for sufficiently small scale β, we show that the Vietoris–Rips complex $\mathcal{R}_{\beta}^{\varepsilon}(\mathcal{S})$ is homotopy equivalent to $\mathcal{G}$.

Our topological reconstruction extends and improves the above-mentioned works in the following directions. Although the authors of [8, 14] considered embedded graph reconstruction in a similar setting, a noteworthy limitation was the use of the global distortion of $\mathcal{G}$, which is known to become infinite in the presence of the cusp-like structures in $\mathcal{G}$. We successfully mitigate the caveat by putting forward the large-scale distortion ([11, Definition 2.8]) as a more robust, alternative sampling parameter in our reconstruction scheme. In addition, our proof techniques are much simpler than [14], with reconstruction guarantees under much weaker sampling conditions. We also mention that the large-scale distortion was introduced in [12] for the reconstruction of spaces more general than Euclidean graphs. However, we present a more direct proof, which avoids using their two main ingredients—Hausmann's theorem [9] and Jung's theorem [13]—resulting in a much weaker sampling condition in the special case of graphs.

Based on the length $\ell(\mathcal{G})$ of the shortest loop and large-scale distortion $\delta_{\beta}^{\varepsilon}(\mathcal{G})$ of the embedding of $\mathcal{G}$, we show how to choose a suitable density parameter ε and a scale β such that $\mathcal{R}_{\beta}^{\varepsilon}(\mathcal{S})$ is homotopy-equivalent to $\mathcal{G}$. For a proof, see [11, Theorem 3.1].

Theorem 1.1 (Topological Reconstruction). *Let $\mathcal{G} \subset \mathbb{R}^N$ be a compact, connected metric graph. Fix any $\xi \in \left(0, \frac{1}{4}\right)$. For any positive $\beta < \frac{\ell(\mathcal{G})}{4}$, choose a positive $\varepsilon \leq \frac{\beta}{3}$ such that $\delta_{\beta}^{\varepsilon}(\mathcal{G}) \leq \frac{1+2\xi}{1+\xi}$. If $\mathcal{S} \subset \mathbb{R}^N$ is such that $d_H(\mathcal{G}, \mathcal{S}) < \frac{1}{2}\xi\varepsilon$, then we have a homotopy equivalence $\mathcal{R}_{\beta}^{\varepsilon}(\mathcal{S}) \simeq \mathcal{G}$.*

Geometric Reconstruction Topologically faithful reconstructions are only useful to estimate the homological features—such as the Betti numbers, Euler characteristic, etc—of the hidden shape $\mathcal{X}$. A more challenging yet more useful paradigm is *geometric reconstruction*: to output a subset $\widetilde{\mathcal{X}}$ in the same host Euclidean space $\mathbb{R}^N$ computed from $\mathcal{S}$ such that $\widetilde{\mathcal{X}}$ is not only homotopy equivalent but also Hausdorff-close to $\mathcal{X}$.

Despite aiding in homotopy equivalent reconstruction, as an abstract simplicial complex, Vietoris–Rips complexes fail to provide an embedding in the same host Euclidean space. For a geometric reconstruction of Euclidean shapes, it's most natural to consider the shadow of the Vietoris–Rips complexes. The *shadow* of an abstract simplicial complex $\mathcal{K}$ with vertices in $\mathbb{R}^N$ is a subset of $\mathbb{R}^N$ defined as the union of the convex hulls of simplices of $\mathcal{K}$; see Definition 2.1 for more details.

The shadow of a general simplicial complex with Euclidean vertices is notorious for being topologically unfaithful. However, when considering the Vietoris–Rips complex of a finite points in $\mathbb{R}^2$ under the Euclidean metric, the shadow project map has been shown in [4] to induce isomorphisms on both π_0 and π_1. Furthermore, the authors show that the projection map fails to induce surjection on π_1 for any $N \geq 4$ and fails to induce an injective homomorphism on π_k for any $N \geq 2$ and $k \geq 2$. The curious case of $N = 3$ was later partially resolved in [1] by proving that the shadow projection induces a surjection on π_1.

In this paper, we consider the Vietoris–Rips complexes of a sample $\mathcal{S} \subset \mathbb{R}^2$, constructed under a (possibly non-Euclidean) family (parametrized by ε) of path-based metrics $d_{\mathcal{S}}^{\varepsilon}$ on $\mathcal{S}$. The phenomenal utility of such path-based metrics has recently been demonstrated by the authors of [8, 14, 12] in the context of shape reconstruction beyond smooth submanifolds. If $\mathcal{S}$ is Hausdorff-close to a Euclidean graph $\mathcal{G}$, we provide quantitative bounds on scales β, ε for the shadow projection map of the Vietoris–Rips of $(\mathcal{S}, d_{\mathcal{S}}^{\varepsilon})$ at scale β to induce π_1-isomorphism. This leads to the following pragmatic geometric reconstruction scheme using the quantity Θ ([11, Eq. 9]) and the shadow radius $\Delta(\mathcal{G})$ of $\mathcal{G}$ as introduced in [11, Definition 5.1].

Theorem 1.2 (Geometric Reconstruction). *Let $\mathcal{G} \subset \mathbb{R}^2$ a graph having properties [A1–A4] from 3.1. Fix any $\xi \in \left(0, \frac{1-\Theta}{6}\right)$. For any positive $\beta < \min\left\{\Delta(\mathcal{G}), \frac{\ell(\mathcal{G})}{18}\right\}$, choose a positive $\varepsilon \leq \frac{(1-\Theta)(1-\Theta-6\xi)}{12}\beta$ such that $\delta_{\beta}^{\varepsilon}(\mathcal{G}) \leq \frac{1+2\xi}{1+\xi}$. If $\mathcal{S} \subset \mathbb{R}^2$ is such that $d_H(\mathcal{G}, \mathcal{S}) < \frac{1}{2}\xi\varepsilon$, then the shadow $\mathcal{S}(\mathcal{R}_{\beta}^{\varepsilon}(\mathcal{S}))$ is homotopy equivalent to $\mathcal{G}$. Moreover, $d_H(\mathcal{S}(\mathcal{R}_{\beta}^{\varepsilon}(\mathcal{S})), \mathcal{G}) < \left(\beta + \frac{1}{2}\xi\varepsilon\right)$.*

2 Shadows of Simplicial Complexes

Let $\mathcal{K}$ be an abstract simplicial complex with vertices in $\mathbb{R}^N$, i.e., $\mathcal{K}^{(0)} \subset \mathbb{R}^N$. In this section, we define the shadow (or *geometric projection*) of $\mathcal{K}$ as a subset of $\mathbb{R}^N$ and study the natural shadow projection map.

The *shadow projection* map $p\colon |\mathcal{K}| \to \mathbb{R}^N$ sends a vertex $v \in \mathcal{K}^{(0)}$ to the corresponding point in $\mathbb{R}^N$ then extends linearly to all points of the geometric realization $|\mathcal{K}|$. Note that p is continuous.

Definition 2.1 (Shadow)**.** We define the *shadow* of $\mathcal{K}$ as its image under the projection map p, i.e.,

$$\mathcal{S}(\mathcal{K}) \coloneqq \bigcup_{\sigma=[v_0,v_1,\ldots,v_k]\in\mathcal{K}} \mathrm{Conv}(\sigma),$$

where $\mathrm{Conv}(\cdot)$ denotes the convex hull of a subset in $\mathbb{R}^N$.

Since the shadow is a polyhedral subset of $\mathbb{R}^N$, it can be realized by the N-dimensional skeleton of $\mathcal{K}$ by Carathéodory's theorem [7]. We now describe a special simplicial complex decomposition of the *shadow* in $\mathbb{R}^2$; see Figure 1. We call it the *shadow complex* and denote it by $\mathcal{SC}(\mathcal{K})$.

(A) A *shadow vertex* $v \in \mathcal{SC}(\mathcal{K})$ is either $v \in \mathcal{K}^{(0)}$ or a transverse intersection of $p(e_1)$ and $p(e_2)$ for edges $e_1, e_2 \in \mathcal{K}^{(1)}$. The transverse intersections are shown in red in Figure 1.

(B) Triangulate the planar shadow using the shadow vertices such that a shadow edge or face does not contain any other vertices of the shadow. Consequently, the realization of any shadow simplex $\sigma \in \mathcal{SC}(\mathcal{K})$ is contained in $p(\tau)$ for some $\tau \in \mathcal{K}$.

Remark 2.2. Note that the triangulation described by (B) above may not be unique. We abuse notation here to denote any such triangulation by $\mathcal{SC}(\mathcal{K})$.

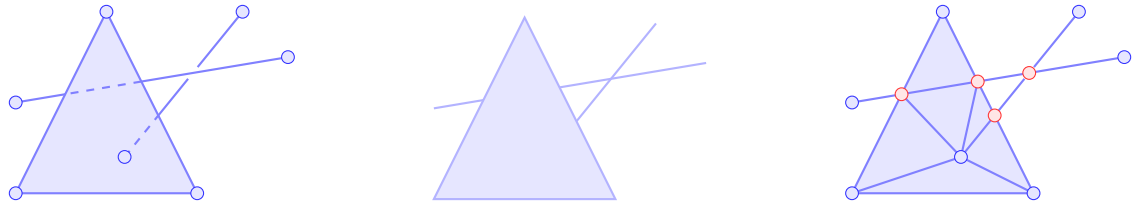


Figure 1: [Left] An abstract simplicial complex $\mathcal{K}$ with planar vertices has been depicted. [Middle] The shadow $\mathcal{S}(\mathcal{K})$ has been shown as a subset of the plane. [Right] The shadow complex $\mathcal{SC}(\mathcal{K})$ has been drawn. The new shadow vertices due to transverse intersection are shown in red.

We use the following notation throughout the rest of the paper. We denote the simplices of an abstract simplicial complex $\mathcal{K}$ by square braces, e.g., $[ABC]$ for an abstract simplex on vertices $A, B, C \in \mathbb{R}^N$. We denote the convex-hull of Euclidean points without any adornment, e.g., ABC denotes the Euclidean (filled in) triangle formed by the vertices A, B, C. Lastly, the Euclidean length of a line segment AB is specified by $\overline{AB}$.

2.1 π_1-epimorphism in $\mathbb{R}^2$

We provide a sufficient condition for the abstract simplicial complex $\mathcal{K}$ so that p induces an epimorphism on the fundamental group.

As already described in the introduction, we mention here the works of [4], where the authors considered the homotopy equivalence of the shadow projection in the particular case when $\mathcal{K}$ is the (Euclidean) Vietoris–Rips complex of a finite, planar point set. The projection was shown to induce π_1-isomorphism. We generalize such results to any simplicial complex $\mathcal{K}$ by proposing a general *lifting condition* to ensure π_1-epimorphism of the projection map p. For geometric graph reconstruction, we apply the condition to the Vietoris–Rips complex of planar samples but under a possibly non-Euclidean metric: the ε-path metric. In particular, we infer below that when $\mathcal{K}$ is the Vietoris–Rips complex under the Euclidean metric (as considered in [4]), $\mathcal{K}$ automatically satisfies the lifting condition.

Let $\mathcal{K}$ be a simplicial complex with vertices in $\mathbb{R}^2$. Any path γ in the geometric realization of its 1-skeleton $|\mathcal{K}^{(1)}|$ can be described by a sequence of oriented edges in $\mathcal{K}$, and its projection $p(\gamma)$ must also form a path in the 1-skeleton of the shadow $\mathcal{S}(\mathcal{K})$, such paths are further referred to as *shadow paths*. However, the converse is not generally true, i.e., every shadow path is not necessarily the projection of a path in the geometric realization of $\mathcal{K}$. Nonetheless, every shadow path can be *lifted up to homotopy* to $\mathcal{K}$ under the following lifting condition.

Definition 2.3 (Shadow Path Lifting)**.** We say that an abstract simplicial complex $|\mathcal{K}|$ with $\mathcal{K}^{(0)} \subset \mathbb{R}^2$ satisfies the *lifting condition* up to homotopy if whenever the images AB and CD of two edges $[AB]$ and $[CD]$ of $\mathcal{K}$ intersect, then there exist vertices $E, F \in \mathcal{K}^{(0)}$ such that either

(A) $[ABE]$ and $[CDE]$ are simplices of $\mathcal{K}$, or

(B) $[AEF]$ and $[CDEF]$ are simplices of $\mathcal{K}$ with EF intersecting AB.

We remark that in the setting where $\mathcal{K}$ is the Euclidean Vietoris–Rips complex (as considered in [4]), one can choose E to be one of the four initial vertices that is closest to the intersection of AB and CD to meet condition (A).

The following theorem guarantees that p induces an epimorphism on the fundamental group of $\mathcal{K}$. See [11, Theorem 4.4] for a proof.

Theorem 2.4 (π_1–epimorphism)**.** *Let $\mathcal{K}$ satisfy the above lifting condition. Then, the projection map p induces an epimorphism on the fundamental groups.*

3 Geometric Reconstruction using Vietoris–Rips Shadow

This section considers the geometric reconstruction of a Euclidean graph $\mathcal{G}$ from a noisy Euclidean sample $\mathcal{S}$ using the shadow of Vietoris–Rips complexes. We assume that both the graph and sample are hosted in the plane, i.e., $N = 2$. For topological reconstruction, $\mathcal{G}$ is assumed to be only compact and connected. However, for geometric reconstruction, we impose a few more *geometric regularity* assumptions on $\mathcal{G}$.

3.1 Assumptions for Geometric Reconstruction

(A1) $\mathcal{G}$ is compact and connected with $\mathcal{E}(\mathcal{G}) < \infty$;

(A2) any two edges of $\mathcal{G}$ are incident to at most one vertex;

(A3) each (open) edge of $\mathcal{G}$ is at least C^1;

(A4) the tangents of each pair of incident edges $e_1, e_2 \in \mathcal{E}(\mathcal{G})$ of $\mathcal{G}$ make an angle in $(0, \pi]$, and is denoted by $\angle e_1 e_2$. We set $\angle e_1 e_2 = \infty$ if e_1, e_2 are not incident.

3.2 Homotopy Equivalence

Our π_1-isomorphism argument for the shadow projection is presented in Theorem 1.2. The proof is based on a series of technical lemmas to ensure that the shadow lifting condition (2.3) is satisfied for a sample near a Hausdorff-close graph $\mathcal{G}$. See [11] for the proof.

In many settings, it is desirable to recover a 1–dimensional proxy for the embedded graph $\mathcal{G} \subset \mathbb{R}^2$. Since the shadow complex $\mathcal{S} = \mathcal{S}(\mathcal{R}^{\varepsilon}_{\beta}(\mathcal{S}))$ is a planar polygon, a natural choice for this proxy is the medial axis of the shadow; see [2] for a definition. In the planar case, this medial axis is itself a geometric graph, as shown in [6].

4 Discussions

The current work successfully provides guarantees for a homotopy-type recovery of an embedded metric graph in $\mathbb{R}^N$ from the Vietoris-Rips complexes of a Hausdorff-close Euclidean sample. Moreover, we prove the π_1-isomorphism of the natural shadow projection of the path-based Vietoris-Rips complexes of a sample lying in a Hausdorff proximity of a planar graph. Since the topological reconstruction works in any host dimension $N \geq 2$, the immediate next step is to consider three-dimensional graphs for geometric reconstruction. The difficulty lies in the elusive nature of the shadow beyond the plane; the shadow projection of (even) Euclidean Vietoris–Rips is not fully known to induce π_1-isomorphism in $\mathbb{R}^3$ [1]. Nonetheless, the exploration remains very relevant to practical applications of Euclidean graph reconstruction. The study sparks several interesting future research directions. Euclidean graphs are the simplest, albeit interesting, class of geodesic spaces one can consider. It is reasonable to believe that the geometric recovery of more general geodesic spaces—such as bounded curvature spaces considered by [12]—can similarly be approached using the shadow of homotopy equivalent Vietoris-Rips complexes.

References

[1] Michał Adamaszek, Florian Frick, and Adrien Vakili. On Homotopy Types of Euclidean Rips Complexes. *Discrete & Computational Geometry*, 58(3):526–542, October 2017. Publisher: Springer New York LLC.

[2] Dominique Attali, Jean-Daniel Boissonnat, and Herbert Edelsbrunner. Stability and computation of medial axes: a state-of-the-art report. In *Mathematical foundations of scientific visualization, computer graphics, and massive data exploration*, Math. Vis., pages 109–125. Springer, Berlin, 2009.

[3] Dominique Attali, André Lieutier, and David Salinas. Vietoris-rips complexes also provide topologically correct reconstructions of sampled shapes. In *Proceedings of the twenty-seventh annual symposium on Computational geometry*, pages 491–500, 2011.

[4] Erin W. Chambers, Vin de Silva, Jeff Erickson, and Robert Ghrist. Vietoris–Rips complexes of planar point sets. *Discrete & Computational Geometry*, 44(1):75–90, Jul 2010.

[5] Frédéric Chazal, David Cohen-Steiner, and André Lieutier. A sampling theory for compact sets in euclidean space. In *Proceedings of the twenty-second annual symposium on Computational geometry*, pages 319–326, 2006.

[6] Hyeong In Choi, Sung Woo Choi, and Hwan Pyo Moon. Mathematical theory of medial axis transform. *Pacific J. Math.*, 181(1):57–88, 1997.

[7] Jürgen Eckhoff. Helly, Radon, and Carathéodory type theorems. In *Handbook of Convex Geometry*, pages 389–448. Elsevier, 1993.

[8] Brittany Terese Fasy, Rafal Komendarczyk, Sushovan Majhi, and Carola Wenk. On the reconstruction of geodesic subspaces of $\mathbb{R}^n$. *International Journal of Computational Geometry & Applications*, 32(01n02):91–117, 2022.

[9] Jean-Claude Hausmann et al. On the vietoris-rips complexes and a cohomology theory for metric spaces. *Annals of Mathematics Studies*, 138:175–188, 1995.

[10] Jisu Kim, Jaehyeok Shin, Frédéric Chazal, Alessandro Rinaldo, and Larry Wasserman. Homotopy reconstruction via the Čech complex and the Vietoris-Rips complex. In *36th International Symposium on Computational Geometry (SoCG 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.

[11] Rafal Komendarczyk, Sushovan Majhi, and Atish Mitra. Vietoris–rips shadow for euclidean graph reconstruction, 2025.

[12] Rafal Komendarczyk, Sushovan Majhi, and Will Tran. Topological stability and Latschev-type reconstruction theorems for spaces of curvature bounded above. *arXiv:2406.04259 [math.AT]*, 2024.

[13] Urs Lang and Viktor Schroeder. Jung's theorem for alexandrov spaces of curvature bounded above. *Annals of Global Analysis and Geometry*, 15:263–275, 1997.

[14] Sushovan Majhi. Vietoris–Rips complexes of metric spaces near a metric graph. *Journal of Applied and Computational Topology*, pages 1–30, 2023.

[15] Sushovan Majhi. Demystifying latschev's theorem: Manifold reconstruction from noisy data. *Discrete & Computational Geometry*, May 2024.

A Extra Figures

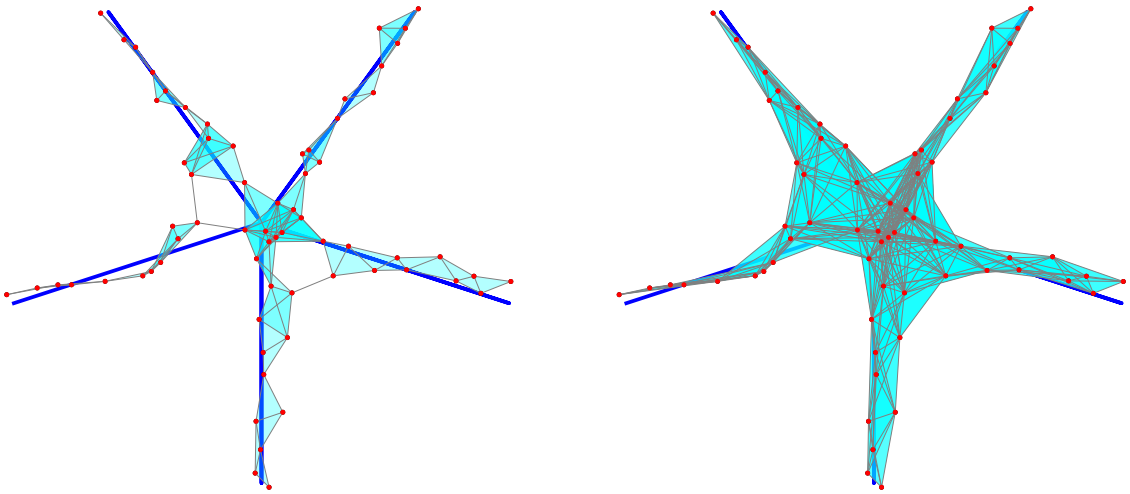


Figure 2: Reconstruction of a 5–pronged graph from a Hausdorff close sample via the shadow $\mathcal{S}(\mathcal{R}^{\varepsilon}_{\beta}(\mathcal{S}))$. The [TOP LEFT] figure shows a shadow of the Euclidean Rips complex, which is topologically inaccurate (projected edges in grey and faces in light blue). The [TOP RIGHT] figure shows $\mathcal{S}(\mathcal{R}^{\varepsilon}_{\beta}(\mathcal{S}))$, which reflects the correct homotopy type (Theorem 1.2). Further homotopy equivalent simplifications of $\mathcal{S}(\mathcal{R}^{\varepsilon}_{\beta}(\mathcal{S}))$ are shown in the [BOTTOM LEFT] and [BOTTOM RIGHT] figures. The [BOTTOM LEFT] shows the planar triangulation of the shadow $\mathcal{S}(\mathcal{R}^{\varepsilon}_{\beta}(\mathcal{S}))$ with marked green boundary. The purple geometric graph shows an approximation of the medial axis.

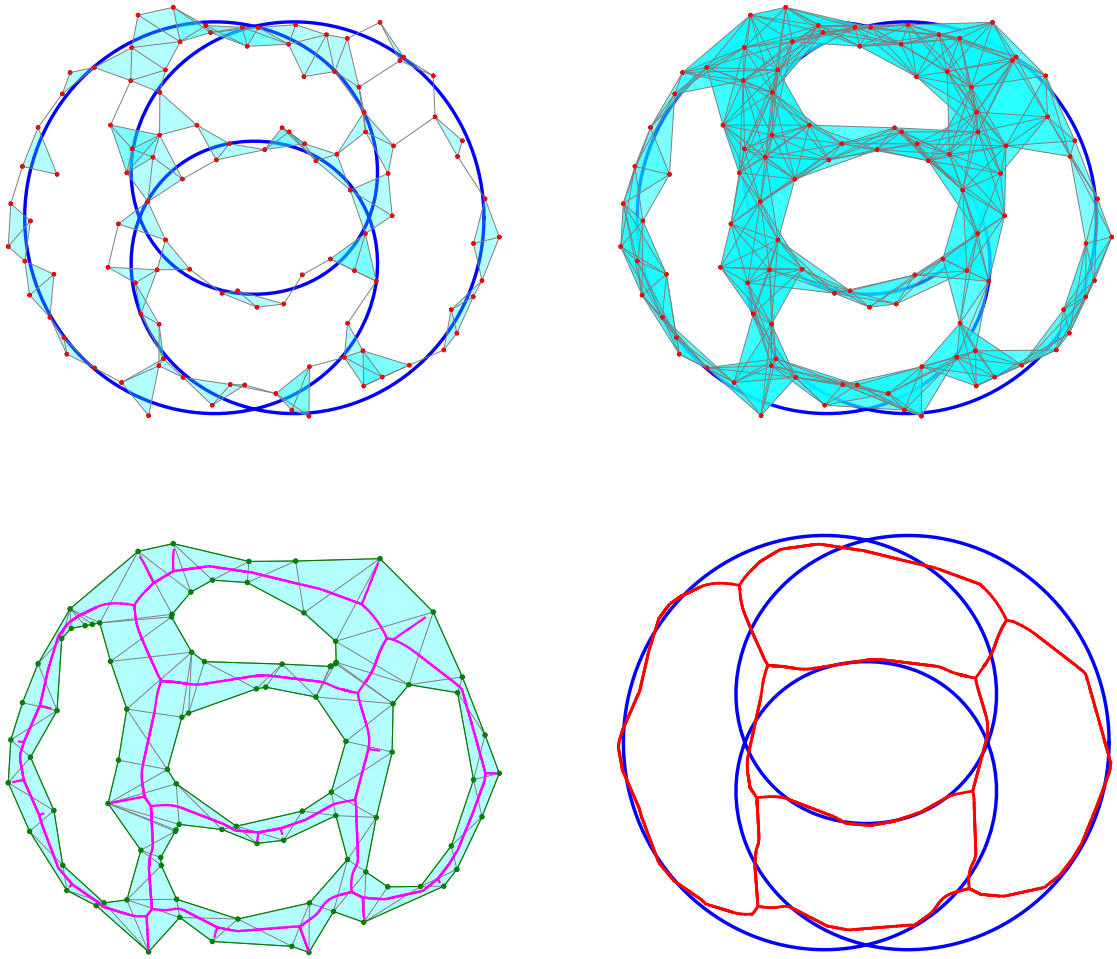


Figure 3: Similar to Figure 2, here the geometric graph under reconstruction is a closed curve with no leaves. The [BOTTOM LEFT] panel shows the entire medial axis of the shadow complex $\mathbb{S}$ (purple). The [BOTTOM RIGHT] panel displays a pruned version of the medial axis (red); see [2] for pruning strategies for medial axes.

PRODUCT RANGE SEARCH PROBLEM

OLIVER CHUBET, ANVI KUDARAYA, NIYATHI KUKKAPALLI, AND DONALD R. SHEEHY

Abstract. Given a metric space, a standard metric range search would find all points within some given distance of a given point. If we have two different metrics d_1 and d_2 on a set, define a product range query as a point p and two radii r_1 and r_2. The result will be all points within distance r_1 with respect to d_1 and within r_2 with respect to d_2. In other words, it is the intersection of two searches. We present two data structures for approximate product range search in doubling metrics. Both are adapted from greedy trees, a data structure that can efficiently answer approximate range searches in doubling metrics. The first data structure is a generalization of the Range Tree from computational geometry using greedy trees rather than binary trees. The second data structure is a single greedy tree constructed on the product of the two metrics.

1. Introduction

The metric range search problem is as follows: given a set of points P in a metric space, preprocess P into a data structure where queries are of the form (c, r) where c is a point in P and $r \geq 0$ is a real number. Such a query returns the points in $P \cap \mathsf{ball}(c, r)$. In one dimension, the metric balls are intervals and products of intervals are axis-aligned boxes. The corresponding range search problem is known as *orthogonal range search*. Thus, orthogonal range search is a product of one-dimensional range searches. In this paper, we generalize the metric range search problem to product metric ranges searches in doubling metrics. For two metrics, d_1 and d_2, the queries will be of the form (p, r_1, r_2) and the result will be:

$$\{x \in X \mid \mathsf{d}_1(x, p) \leq r_1 \text{ and } \mathsf{d}_2(x, p) \leq r_2\}.$$

In other words, it is the intersection of two range searches (p, r_1) with respect to d_1 and (p, r_2) with respect to d_2. We call this the **Product Range Search Problem**. In this paper, we present and analyze two data structures for product range search in doubling metrics. The first product range tree is a multi-level data structure. The construction uses the classical paradigm of cascading multiple decomposition schemes. The second product range tree is a greedy tree on the product metric itself. However, the query algorithm must be modified to accommodate products of balls, which may not correspond to a ball in the product metric.

In settings where distance computations can be computed efficiently, there are several efficient data structures available, and some authors even consider range search a "solved" problem [2]. However, there are still cases when range queries over the full space may be impractical due to expensive metric computations. Recently, product range search approaches have been considered for cases where metric computations are expensive to compute, but the metric admits some decomposability into a product, such as is the case for the Ulam metric, discrete Frechet distance, and dynamic time warping [1, 8]. Other applications that can be formulated as product problems are multi-key range queries in databases [4]. Product-metric range search has recently been studied for metrics that decompose into l_p products of l_∞ or l_1 spaces [3]. To the best of our knowledge, product range search has not been studied for products of non-normed spaces.

This research was supported by the NSF (CCF-23491790).

1.1. **Related Work.** There are a plethora of data structures for range search in product metrics, but most are for Euclidean space. For coordinate-wise comparisions, the *k-d tree* partitions the plane into rectangular regions and has query time of $O(n^{1-1/d}+k)$ for a d-dimensional query [7]. There are also range trees for orthogonal range queries, which have a query time of $O(\log^{d-1} n+k)$ using fractional cascading [7].

Ball trees are the simplest hierarchal structure for range search in metric spaces. They are defined by recursively partitioning the data set, represented as a binary tree [11]. Each node of the tree represents a metric ball, storing a center and radius. Range searches are performed using branch and bound, where nodes can be pruned when the entire ball is disjoint from the query ball. Greedy trees are a ball trees constructed using farthest point sampling to achieve packing and covering guarantees. The algorithms we propose in this work are built with greedy trees because they are simple, implementable, and admit strong theoretical guarantees, but other similar trees could work.

2. Background

2.1. **Product Metrics.** Let X be a set of points, and $\mathsf{d}_1, \mathsf{d}_2, \ldots, \mathsf{d}_m$ metrics on X. The *l_∞-product metric* d is

$$\mathsf{d}(x,y) = \max_{1\le i\le m} \mathsf{d}_i(x,y).$$

For clarity, when referring to a ball in metric d_i, we use the notation $\mathsf{ball}_{\mathsf{d}_i}$, however when referring to a ball in the product metric itself we may omit the subscript.

2.2. **Product Range Search.** Given a set of points P and m metrics $\mathsf{d}_1, \ldots, \mathsf{d}_m$, a product range search query is a point q, radii $r_1, \ldots, r_m$, and an approximation parameter ε. The output contains all points in the intersection of the metric balls centered at a query point q, and only points approximately contained in the intersection. That is,

$$\bigcap_{i=1}^{m} \mathsf{ball}_{\mathsf{d}_i}(q, r_i) \subseteq \mathsf{output} \subset \bigcap_{i=1}^{m} \mathsf{ball}_{\mathsf{d}_i}\left(q, (1+\varepsilon)r_i\right).$$

2.3. **Doubling Dimension and Packing.** Doubling dimension offers a proxy for volume in finite metric spaces. The *doubling constant* ρ is the minimum integer ρ such that any ball of radius r can be covered with ρ balls of radius $\frac{r}{2}$. We define the *doubling dimension* on a metric (X, d) as $\dim(X, \mathsf{d}) := \log_2(\rho)$.

Lemma 2.1 (Subadditivity of Dimension)**.** *Let X be a finite set, with d, the product of metrics d_1 and d_2. Then* $\dim(X, \mathsf{d}) \le \dim(X, \mathsf{d}_1) + \dim(X, \mathsf{d}_2)$.

A set S is *r-packed* if for any distinct $a, b \in S$ it follows $\mathsf{d}(a,b) \le r$ for any $a, b \in S$.

Lemma 2.2 (Standard Packing Lemma [10])**.** *Let (X, d) be a metric space and $\delta = \dim X$. If the set Z is r-packed and can be covered with a ball of radius R then*

$$|Z| \le \left(\frac{4R}{r}\right)^{\delta}.$$

2.4. **Greedy Permutation and Greedy Trees.** Let $P = (p_0, p_1, \cdots p_{n-1})$ be a set of points in a metric space with metric d where the ith prefix is denoted at $P_i = \{p_0, p_1, \cdots p_{i-1}\}$, or the first i points. The *greedy permutation* is an ordering of the points P such that

$$\mathsf{d}(p_i, P_i) = \max_{j\ge i} \min_{q\in P_i} \mathsf{d}(p_j, q).$$

The point p_0 is chosen arbitrarily. A greedy permutation can be computed in $O(n \log \Delta)$ time for low-dimensional data, where Δ is the ratio of farthest to smallest pairwise distances in P [9]. There are also $O(n \log n)$ time approximation algorithms [6, 9].

2.5. **Range Search in Greedy Trees.** We analyze our range search algorithms using the proximity search framework for ball trees established by Chubet et al. [5]. A range search maintains a heap H containing *viable nodes* that can intersect the query range. A range search maintains the invariant that any point in the query range is either covered by some node in H or has been added to the output. Additionally, each point appears in at most one node in H or the output at any given time. The width w of a search is the maximum number of nodes in H during any given iteration of the algorithm. The height h of a search is the maximum number of times a node containing a(n arbitrary) point p splits.

A useful property of greedy trees is that they can be constructed recursively. In particular, merging two greedy trees is more efficient than constructing a greedy tree on the same point set.

Lemma 2.3. *(Chubet et al. [6])*

- *A greedy tree can be constructed in* $2^{O(\delta)} n \log n$ *time.*
- *Two greedy trees on* n *points can be merged in* $2^{O(\delta)} n$ *time.*
- *A greedy tree is stored in* $O(n)$ *space.*

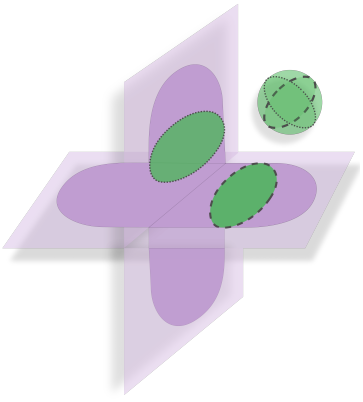
3. Greedy Range Tree on Product Metric

A greedy range tree (see Figure 1a) on a set P and metrics $\mathsf{d}_1, \ldots, \mathsf{d}_m$ is defined recursively, as follows:

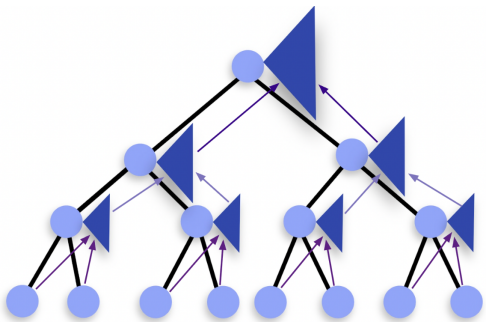
- The *primary tree* is a greedy tree on all of P using metric d_1.
- Each node v of the primary tree points to an *auxiliary* greedy range tree on the subset $\mathsf{points}(v)$ with metrics $\mathsf{d}_2, \ldots, \mathsf{d}_m$.

The construction of a greedy range tree is also recursive (see Figure 1b). First, we construct a greedy tree for d_1. We observe that for a node u with children u_L and u_R, we have $\mathsf{points}(u) = \mathsf{points}(u_L) \cup \mathsf{points}(u_R)$. So, rather than constructing the greedy tree on $\mathsf{points}(u)$ for each node u, we merge the trees of its children. For the m metrics, a bottom-up merging process is repeated recursively across $m-1$ levels. Note that each point is contained in at most one node per depth from the root, so each point participates in $O(\log \Delta)$ merges. Thus, each recursive level adds a multiplicative factor of $\log \Delta$ to the build time. Note that for simplicity our analysis uses $\Delta = \max_{1 \le i \le m} \Delta_i$, where Δ_i is the spread of P under metric d_i.

Theorem 3.1. *Let* $\mathsf{d}_1, \ldots, \mathsf{d}_m$ *be* m *metrics on an* n*-point set* P*. A product range tree can be constructed in* $2^{O(\delta_*)} n \log^{m-1} \Delta$ *time and has space complexity* $O(n \log^{m-1} \Delta)$*, where* $\delta_* = \max_{1 \le i \le m} \delta_i$*.*



(A) Each node of a greedy range tree corresponds to an intersection of balls from each metric.



(B) The auxiliary greedy range trees are built bottom-up. Each node merges the auxiliary trees of its children.

3.1. **Query Time.** Now we discuss how to perform greedy range tree queries, starting with the case of only 2 metrics. We first query the greedy tree built on the first metric d_1, and return a collection of nodes that are approximately contained in $\mathsf{ball}_{\mathsf{d}_1}(q, r_1)$. This takes $(2+\frac{1}{\varepsilon})^{O(\delta_1)} \log \Delta$ time and returns a collection of $(2+\frac{1}{\varepsilon})^{O(\delta_1)}$ nodes. Then, we query each node returned with (q, r_2) in d_2. This takes $(2+\frac{1}{\varepsilon})^{O(\delta_2)} \log \Delta$ time per node. We continue similarly for each metric. Note that the resulting collection size may increase by a factor of $(2+\frac{1}{\varepsilon})^{O(\delta_i)}$ for each layer i of the greedy range tree. The bound on the output size of each layer follows from the bounded width of a greedy tree range search [5].

Theorem 3.2. *An m-metric query in a greedy range tree takes $(2+\frac{1}{\varepsilon})^{O(\delta)} \log \Delta + O(k)$, where $\delta = \sum_{1 \le i \le m} \delta_i$, and k is the size of the output.*

4. Greedy Tree on Product Metric

A greedy tree on the product metric is a greedy tree that uses the product metric to compute distances. However, the query algorithm is modified to accommodate product range search queries, which are more complex than typical range search queries. By Lemma 2.3, the build time is $2^{O(\delta)} n \log n$ and the space is $O(n)$.

4.1. **Query Algorithm.** The following algorithm is similar to range search in a ball tree [5], since we are querying on one greedy tree. However, the pruning rules are modified to account for all query radii.

Input: Greedy tree T, radii $r_1, \ldots, r_m$, $\varepsilon > 0$

(1) Initialize max-heap H with the root of T.
(2) While H is not empty:
  (a) Remove node v with maximum radius r from H.
  (b) If $\mathsf{d}_i(q, v) \le (1+\varepsilon) r_i - r$ for all i, then add v to output.
  (c) Else if v is not a leaf:
    (i) Split v into children v_L and v_R with radii r_L and r_R respectively.
    (ii) If $\mathsf{d}_i(q, v_R) \le r_i + r_R$ for all i add v_R to H.
    (iii) If $\mathsf{d}_i(q, v_L) \le r_i + r_L$ for all i add v_L to H.

4.2. **Query Time.** We analyze the query time for the case of two metrics.

Theorem 4.1. *Let d_1 and d_2 be metrics with doubling dimensions δ_1 and δ_2 respectively. Then a product range query $(q, r_1, r_2, \varepsilon)$ takes*

$$A^{\delta_2} \left(2 + \frac{1}{\varepsilon}\right)^{O(\delta_1+\delta_2)} \log \Delta + O(k)$$

time, where k is the size of the output, and $A = \frac{\max\{r_1, r_2\}}{\min\{r_1, r_2\}}$.

The bound follows from a packing argument used to bound the width of the search. The ratio of largest query radius to smallest appears in the packing bound because we keep nodes in the heap that intersect the largest metric ball in the product, however, we may also need to split the nodes to the scale of the smallest query radius.

References

[1] Peyman Afshani and Anne Driemel. On the complexity of range searching among curves. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 898–917. SIAM, 2018.

[2] Pankaj K Agarwal, Jeff Erickson, et al. Geometric range searching and its relatives. *Contemporary Mathematics*, 223(1):56, 1999.

[3] Alexandr Andoni, Piotr Indyk, and Robert Krauthgamer. Overcoming the l 1 non-embeddability barrier: Algorithms for product metrics. In *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 865–874. SIAM, 2009.

[4] Jon Louis Bentley and Jerome H Friedman. Data structures for range searching. *ACM Computing Surveys (CSUR)*, 11(4):397–409, 1979.

[5] Oliver Chubet, Parth Parikh, Donald R Sheehy, and Siddharth Sheth. Proximity search in the greedy tree. In *Symposium on Simplicity in Algorithms (SOSA)*, pages 332–342. SIAM, 2023.

[6] Oliver Chubet, Don Sheehy, and Siddharth Sheth. Simple construction of greedy trees and greedy permutations. *arXiv preprint arXiv:2412.02554*, 2024.

[7] Mark De Berg, Marc Van Kreveld, Mark Overmars, and Otfried Schwarzkopf. Computational geometry: introduction. In *Computational geometry: algorithms and applications*, pages 1–17. Springer, 2008.

[8] Ioannis Z Emiris and Ioannis Psarros. Products of euclidean metrics, applied to proximity problems among curves: Unified treatment of discrete fréchet and dynamic time warping distances. *ACM Transactions on Spatial Algorithms and Systems (TSAS)*, 6(4):1–20, 2020.

[9] Sariel Har-Peled and Manor Mendel. Fast construction of nets in low dimensional metrics, and their applications. In *Proceedings of the twenty-first annual symposium on Computational geometry*, pages 150–158, 2005.

[10] Robert Krauthgamer and James R Lee. Navigating nets: simple algorithms for proximity search. In *SODA*, volume 4, pages 798–807, 2004.

[11] Stephen M Omohundro. Five balltree construction algorithms. 1989.

Fully Packed and Ready to Go: High-Density, Rearrangement-Free, Grid-Based Storage and Retrieval

Tzvika Geft*, Kostas Bekris*, and Jingjin Yu*

***Abstract*— We consider an ordered storage and retrieval problem: a set of uniform-sized, labeled loads (e.g., containers, pallets, or totes) must be placed in a 2D grid storage area as they arrive sequentially, and then be retrieved in some (possibly different) order. Each load occupies a grid cell and may be moved, e.g., by a robot, along the cardinal directions. Such storage systems arise in logistics, industrial, and transportation domains, where space utilization and retrieval time are critical metrics. To maximize space utilization, loads must be densely packed with some loads blocking access to others, which raises a key question: How should one store the loads to minimize costly rearrangements, i.e., the number of relocated loads, during retrieval? We identify conditions, alongside efficient algorithms, for achieving either zero or near-optimal rearrangements under different knowledge assumptions. While the online case (i.e., no prior knowledge of the storage and retrieval sequences) induces a trade-off between density and rearrangement, we show that even partial prior knowledge essentially eliminates the trade-off. When the sequences are fully known, we further provide an intriguing characterization: rearrangement can always be eliminated if the grid's open side (used to access the loads) is at least 3 cells wide, even for full capacity storage.**

Full version: `https://arxiv.org/pdf/2505.22497`

I. Introduction

The past two decades have witnessed the dramatic rise of robotics and automation technologies for transporting uniform-sized *loads* or *items* (e.g., containers, pallets, totes, etc.) in logistics applications. Notable examples range from thousands of mobile robots roaming in a warehouse helping order fulfillment [1] to automated cranes transporting containers at shipping yards [2]. Oftentimes, operations at logistics hubs have two distinct phases: first, the storage of incoming loads, and later, their retrieval for further transport. A central challenge these systems must contend with is the trade off between maximizing space utilization and storage/retrieval efficiency, as denser storage necessarily makes arbitrary load access more difficult. Relocations of loads during retrieval is a key metric to minimize, as each relocation incurs costs due to time-consuming pick-up and drop-off operations.

This work investigates the coupling between storage density and rearrangement efforts by asking: How far can we maximize storage space utilization while minimizing load rearrangement? We focus on 2D grid-based storage systems, akin to Puzzle-Based Storage (PBS) [5], where each grid cell can store a load and loads can be moved along the grid by a mobile robot, as long as the motion is collision-free. We now define the setting formally.

Problem definition. Consider a rectangular $r \times c$ grid storage space W with r rows and c columns. Fix the orientation of W so that its bottom row, also called *front* row, is the open side of W through which loads are stored/retrieved. See Figure 2. The loads have distinct labels $1, \ldots, n$, with $n \leq rc$. The *density* of the W is $n/(rc)$. Each load occupies exactly one grid cell. Each load can be moved by a robot via a path of empty cells along the four cardinal directions (up, down, left, or right). Specifically, the following types of *actions* are valid:

- Storage: A robot can *store* a load in an empty cell v in W via a path from any cell in the front row to v.
- Retrieval: A robot can *retrieve* a load from W via a path from its current cell to any cell in the front row.
- Relocation: A robot can *relocate* a load within W to an empty cell v via a path from its current cell to v.

Loads must be all stored and then retrieved according to prescribed sequences. The departure sequence, i.e., the order in which loads are to be retrieved, is fixed to be $D = (1, \ldots, n)$ without loss of generality. We denote the arrival sequence, i.e., the order in which loads must be stored, by $A = (a_1, \ldots, a_n)$. We study the following problem:

Storage and Retrieval with Minimum Relocations (`StoRMR`): Given a storage area W, and arrival and departure sequences A and D, respectively, find a minimum-length sequence of actions that stores all loads per the sequence A and then retrieves the loads per the sequence D.

We further consider various degrees of prior knowledge: The problem is **offline** when A and D are known in advance. Otherwise, the problem is online, where two versions are examined: (i) **Online with lookahead**: D is fully known, but A is revealed online, one load at a time, which we call an *arrival lookahead* of 1. That is, the departure position of each arriving load is available.[1] (ii) **Fully online**: Neither A

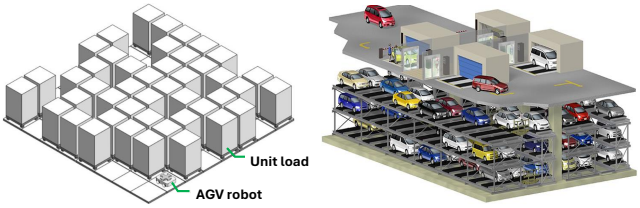


Fig. 1: Application examples. Left: Grid-based storage using robotic vehicles (AGVs) for transferring loads [3]. The AGV can go beneath a load and can move in all four grid directions. Right: Illustration of an automated parking garage where vehicles are the load to be autonomously placed and retrieved [4]. Similar to the first case, AGVs can go under vehicles to transport them.

*Computer Science Dept., Rutgers University, New Brunswick NJ, USA

[1] In the full version, we also consider a larger lookahead, i.e., additional foresight of A.

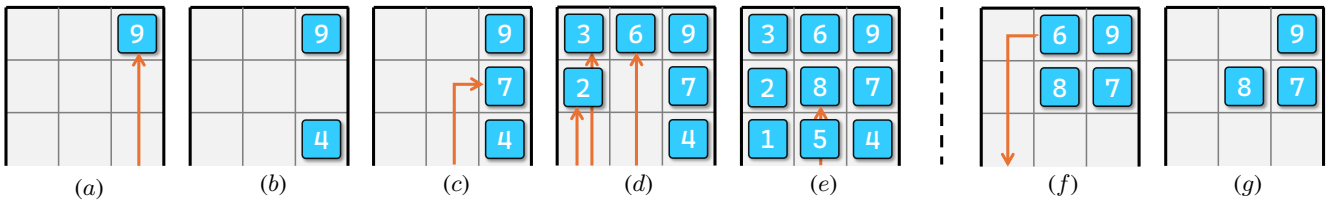


Fig. 2: A rearrangement-free solution for an arrival sequence $A = (9, 4, 7, 3, 6, 2, 1, 8, 5)$ and departure sequence $D = (1, 2, 3, 4, 5, 6, 7, 8, 9)$ for a (3×3) grid accessible only from the bottom. Snapshots are illustrated from left to right. (a) The first arriving (load) 9 can be directly stored at the top using a (straight) upward path. (b) Next, 4 arrives and can be stored in front of 9, leaving the space in between. (c) 7 is stored using an upward path with a single turn, i.e., a column-adjacent path. (d) 3, 6, 2, can be stored as shown using upward paths. (e) 1, 8, 5 can be stored similarly. At this stage, all loads have arrived. (f) 1, 2, 3, 4, and 5 can be retrieved sequentially directly using downward paths. Then, 6 can be retrieved using a downward path with a turn. (g) 7, 8, and 9 can be retrieved using downward paths.

nor D are known in advance, but the number of loads n is known. In the best case, $2n$ actions (a storage action and a retrieval action for each load) are necessary. We call such solutions *rearrangement-free*.

Contributions. Our key insight is that the density–rearrangement tradeoff can be (nearly) eliminated given prior information on the arrival and departure sequences. The precise contributions are as follows:

- **Offline setting:** Relocation may be required when the number of columns is $c \leq 2$. For any $c \geq 3$, however, it is *always* possible to avoid relocations, for any number of loads, even at full capacity.
- For the **online with lookahead** setting strong guarantees remain possible:
  - If $n \leq r(c-1)+1$, i.e., when it is possible to keep a single column nearly empty, there exists a rearrangement-free solution.
  - When $c \leq r$, a solution with at most $r-1$ relocations exists for any density, yielding a $9/8$-approximation for the total number of actions.
- **Fully online setting:** Without prior information, density must be sacrificed to limit rearrangements. We characterize the tradeoff by providing the maximum achievable density for a given bound on the number of actions allowed per load. In particular, guaranteeing no relocations requires a natural aisle-based layout, with maximum density $2/3$. (see Figure 7, left)

All the positive results are accompanied by (near) linear-time algorithms in the number of loads, which determine the loads' storage and retrieval paths. Besides the guarantees on minimizing relocations, our solutions result in desirable paths for accessing loads. More specifically, in our rearrangement-free solutions, each path used for storage and retrieval lies in one column with a possible additional lateral segment to a cell adjacent to that column, i.e., each path is *distance-optimal* up to an additive factor of 1.

II. Related Work

A sizable body of work considers storage systems with uniform-sized loads. In this brief review, we distinguish prior work along two main axes. The first is whether storage/retrieval is *ordered*, with sequence(s) specified in advance, or *online*, where requests arrive with little prior information. The second is the motion model: *Grid-based* systems, which include this work, allow loads to move along both axes of the grid. In *stack-based* systems, often motivated by shipyards, loads are placed in vertical stacks with Last-In-First-Out (LIFO) access, so that a load can only be retrieved after removing those above it. Under this dichotomy, ordered storage and retrieval have been studied almost exclusively in stack-based systems, whereas for grid-based systems, they have received little attention.

Grid-based storage. Gue [6] studies rectangular grid warehouses under a *depth* bound limiting the number of blocking objects. Puzzle-Based Storage (PBS) [5] systems consist of dense 2D grids with as few as a single empty "escort" cell: loads are moved into the escort, treating other loads as movable obstacles. PBS works typically focus on online retrieval of one or a few loads without a strict order, and generally omit a storage phase [7]. Ordered retrieval has been examined in decentralized conveyor-based systems [8], where loads arrive randomly and must be output in a given sequence. These systems rely on frequent relocations and dedicate grid rows or columns to motion rather than storage, in contrast to our maximum storage density, rearrangement-free focus. **Stack-based storage.** Ordered retrieval has focused primarily on stack-based systems, in which the LIFO constraint substantially alters the problem structure: even deciding whether relocations can be avoided under given arrival and departure sequences is NP-hard [9]. The classic Block Relocation or Pre-marshalling problems [10, 11] focus on minimizing relocations during/just before retrieval after storage is complete; these too remain NP-hard, precluding rearrangement-free guarantees.

III. The Offline Setting

This section presents a characterization of the existence of rearrangement-free solutions in the offline setting. First, we observe that relocations may be necessary to support 100% density for narrow grid openings, as shown in Figure 3:

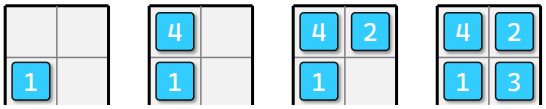


Fig. 3: Consider an instance with $A = (1, 4, 2, 3)$ and $D = (1, 2, 3, 4)$. Given that load 1 must depart first, it has to be stored in the front to avoid relocations. This forces the above-shown storage sequence (or its vertical mirror, where 1 is placed on the bottom right). This leaves load 2 buried behind loads 3 and 4. This means it cannot be retrieved without a rearrangement.

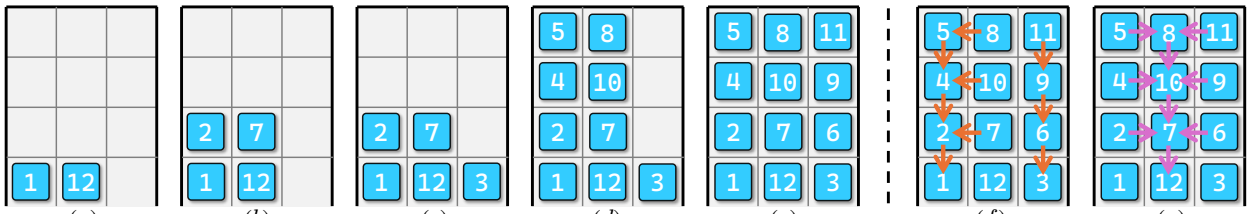


Fig. 4: Running Algorithm 1 on the input $D' = (\underset{1}{12}, \underset{2}{7}, \underset{3}{3}, \underset{4}{1}, \underset{5}{10}, \underset{6}{8}, \underset{7}{9}, \underset{8}{11}, \underset{9}{6}, \underset{10}{4}, \underset{11}{2}, \underset{12}{5})$. (a)(b) Partial solutions after the first two iterations, where non-equal matching loads of D and D' are put in the left two columns C_1 and C_2. (c) In the third interaction, D and D' have an equal matching load 3, which is put in C_3. (d) The next two iterations fill up columns C_1 and C_2. This leads to the solution following case 1 from here on. (e) The leftover loads in C_3 are filled up to complete the arrangement. The solution with arrows showing the guaranteed local adjacencies for $[12]$ and D' is shown in (f) and (g), respectively.

Observation 1. *For a 2×2 storage space, relocations may be necessary.*

We now proceed with key definitions and notation: An *arrangement* $\mathcal{A}$ of the loads is an injective mapping that specifies a cell in W, i.e., a (row, column) pair, for each load. Two loads are *adjacent* in a given arrangement if they are located in horizontally or vertically adjacent grid cells. An arrangement $\mathcal{A}$ is defined to *satisfy* an arrival (resp. departure) sequence A (resp. D), if all the loads can be stored (resp. retrieved) according to sequence A (resp. D) with one action per load, where $\mathcal{A}$ is the final (resp. initial) arrangement. Fix $D = [n]$, where $[n] \coloneqq (1, \ldots, n)$, leaving A as the sole sequence that can change per problem instance.

Observation 2. *An arrangement $\mathcal{A}$ satisfies an arrival sequence A if and only if $\mathcal{A}$ satisfies the departure sequence in which A is reversed.*

Following Observation 2, the problem of finding a rearrangement-free solution is equivalent to finding an arrangement $\mathcal{A}$ that satisfies two departure sequences, namely the true departure order $[n]$ as well as a permutation D' of $[n]$, where D' is the reverse of A in the original input. For example, given $A = (1, 4, 2, 3)$ and $D = (1, 2, 3, 4)$ as in Fig. 3, the two departure orders to be satisfied are the original one $D = (1, 2, 3, 4)$ and $D' = (3, 2, 4, 1)$. We assume a density of 1 (otherwise, the situation is simpler).

It suffices to check the following *local adjacency conditions* to determine whether a given arrangement $\mathcal{A}$ satisfies a departure sequence D':

Observation 3. *An arrangement $\mathcal{A}$ satisfies a departure sequence $D' = (d_1, \ldots, d_n)$ if and only if every load d_i is either in the bottom row or is adjacent in $\mathcal{A}$ to an load d_j that departs earlier, i.e., $j < i$.*

Let D' be an input permutation of $[n]$. The following algorithm constructs a valid arrangement $\mathcal{A}$ by iteratively assigning placements bottom-up, ensuring the adjacency conditions of Observation 3 are met for $[n]$ and D'. Recall that C_1, C_2, C_3 are the columns in left-to-right order.

Algorithm 1. In the first stage, the algorithm iterates jointly over $[n]$ and D' in order and repeats the following: Let x and y be the first two items in $[n]$ and D', respectively, which have not been assigned a cell. If $x \neq y$, assign x and y to the lowest unassigned cells of C_1 and C_2, respectively. Otherwise, if $x = y$, assign x to the lowest unassigned cell in C_3. Repeat this process until one (or more) of the columns becomes fully assigned, at which point we proceed to the next stage, which has two cases:

Case 1: If C_1 and C_2 are fully assigned (notice that since we assign items to them together, they become fully assigned together), we continue filling up C_3 (bottom-up) by adding the remaining unassigned items of $[n]$ (in order).

Case 2: Otherwise, C_3 becomes fully assigned first. In this case, we fill in C_2 (bottom-up) with the remaining items of D' until C_2 is filled up. Then, we fill C_1 similarly with the remaining items of $[n]$.

Theorem 1. *For an $r \times 3$ storage area with $r \geq 1$, there is an offline rearrangement-free solution. The solution can be found in $O(n)$ time, where n is the number of loads.*

Proof sketch. One must verify that the adjacency conditions of Observation 3 hold for $[n]$ and D' in the arrangement output by Algorithm 1. Figure 4 illustrates one case. □

We now use Theorem 1 to obtain rearrangement-free solutions for the more general case of three or more columns.

Theorem 2. *For an $r \times c$ storage area with $r > 1$, an offline rearrangement-free solution always exists if and only if $c \geq 3$. The solution can be found in $O(n \log r)$ time.*

Proof. Assume a density of 1; otherwise, the situation is simpler. The argument proceeds in two stages. First, iterate over the $c - 3$ leftmost columns in left-to-right order, filling up a column at a time as follows (see Figure 5 for an example): Let C denote the current column, and let $s = (a_{(1)}, a_{(2)}, \ldots, a_{(r)})$ denote the next r loads in A (the original non-reversed sequence) sorted per departure order. We proceed to place each load $a_{(i)}$ in the cell of column C located on the (i)-th row away from the front row. The placement also ensures that each load can be retrieved directly using only C, as the load below it departs earlier.

After the first stage, we apply the algorithm from Theorem 1 to the remaining three columns and inherit the theorem's properties. Finally, the running time follows from sorting r loads at a time for at most n/r batches. □

IV. Online setting with lookahead

This section analyzes the variant where the position of each arriving load in the departure sequence D is known but

the arrival sequence A is not given. The proof of Theorem 2 makes it clear that when D is known, it is possible to fill up columns as loads arrive without fully knowing A, for most of the storage space, while ensuring a rearrangement-free solution. Proposition 1 formalizes this observation.

Proposition 1. *For an $r \times c$ storage area and $r(c-1)+1$ (or fewer) loads, a rearrangement-free solution always exists using an arrival lookahead of 1.*

Proof. Let us assume there are $n = r(c-1)+1$ loads, as it is straightforward to handle fewer loads. The goal is to fill the leftmost $c-1$ columns top-down such that at the departure stage, the loads in each column C_i depart after the loads at column C_{i+1} have departed. We store each load in a column at the current topmost available cell based on its departure order: Load 1 is stored at C_c, at its front cell. Loads $2, \ldots, 2+r-1$ are stored at C_{c-1}, loads $2+r, \ldots, 2+2r-1$ are stored at C_{c-2}, and so on. This approach results in a straight upward path for each storage action. Each retrieval is possible via the adjacent column on the right or a straight downward path. □

The observation leading to Proposition 1 allows to show that if W is square-shaped or wider (at its open side), full-capacity storage with limited relocations is possible.

Theorem 3. *For an $r \times c$ storage area with $r \leq c$, there is always a solution with at most $r-1$ relocations using an arrival lookahead of 1.*

Proof sketch. We generalize the column-filling approach from Proposition 1 to a path-filling approach, i.e., we define a sequence of paths, each of which will contain loads departing only after the loads on the next path. The first $c-r$ paths are the leftmost $c-r$ columns. The remaining portion of W is an $r \times r$ square. Take the next path to be the square's leftmost column and top row, i.e., an L-shaped path. Define the next path similarly for the remaining $(r-1) \times (r-1)$ square, and so on; see Figure 6. Denote the resulting sequence of paths by $\pi_1, \ldots, \pi_c$. Each path contains a cell on the front row, so we can fill each path with arriving loads back to front (similarly to a stack). Loads are assigned to the paths based on their departure order: Load 1 is assigned to π_c (which is a single cell), loads $2, 3, 4$ are assigned to π_{c-1}, and so on, with the last $|\pi_1|$ departing loads assigned to π_1. Upon retrieval, we may require a single relocation for each load stored on a corner of an L-shaped path. □

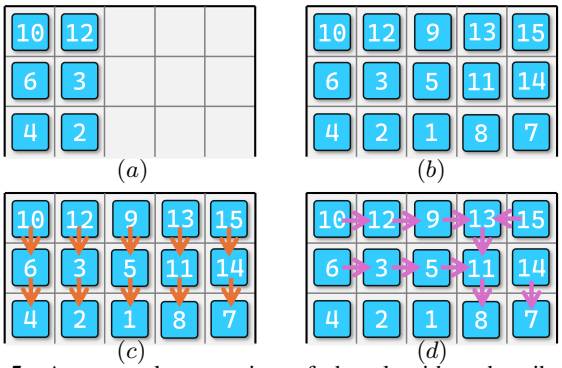


Fig. 5: An example execution of the algorithm described in Theorem 2. $A = (4, 10, 6, 12, 2, 3, 9, 15, 1, 14, 13, 7, 5, 11, 8)$ and $D = [15]$. (a) Knowing the first three loads to arrive are $4, 10, 6$, we store them in the order they depart, which is $4, 6, 10$ from bottom to top. These go to C_1. Similarly, the next three arrivals are stored in C_2 as $2, 3, 12$, from bottom to top. (b) For the next 9 arrivals, we run the algorithm from Theorem 1. For this, we turn the remaining loads of A backward to get $D' = (8, 11, 5, 7, 13, 14, 1, 15, 9)$ and the corresponding portion of D is $(1, 5, 7, 8, 9, 11, 13, 14, 15)$. (c)(d) Arrows showing how departures and arrivals can be handled without rearrangements, respectively. Note that the arrows for arrivals are drawn backwards to be consistent with Fig. 4.

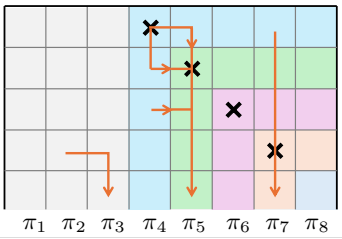


Fig. 6: The storage strategy of Theorem 3 for a (5×8) grid. Paths π_1–π_3 are the left three vertical columns. Paths π_4–π_7 are L-shaped and distinguished using different colors. π_8 is only the bottom right cell. Corner cells are marked with crosses, and the possible retrieval paths are shown using arrows.

One can verify that the $r-1$ bound on relocations translates to a $9/8$-approximation algorithm for minimizing total actions.

V. The Fully Online Setting

In the fully online setting, density must be limited to minimize rearrangements, and the arrangement of the loads in W needs to be carefully chosen:

Theorem 4. *To guarantee at most a actions for each fully online storage and retrieval, the density must be at most $2a/(2a+1)$. The bound is asymptotically tight: a $2a/(2a+1) - \varepsilon$ density is achievable for a small $\varepsilon > 0$.*

We obtain the density upper bound using a result of Gue [6], and realize it using aisle-based arrangements; see Figure 7.

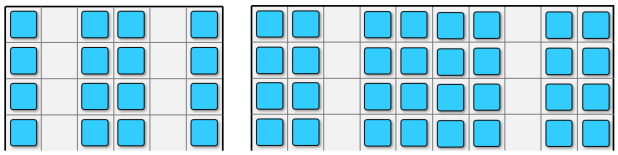


Fig. 7: Left: A 4×6 storage area with 1-deep aisles. Right: A 4×10, 2-deep aisle arrangement.

VI. Conclusion

We establish a range of conditions for (nearly) rearrangement-free storage and retrieval of uniform loads in dense 2D grids. The results highlight the practical utility of the setup and are promising for real-world adoption. When considering the NP-hardness of related stack-based problems [9, 12], our findings, especially Theorem 1, are surprising, as at first glance some variants may also appear NP-hard. Future directions include sharpening our understanding of the online cases, interleaving storage and retrieval, and extensions to irregular grids.

REFERENCES

[1] P. R. Wurman, R. D'Andrea, and M. Mountz, "Coordinating hundreds of cooperative, autonomous vehicles in warehouses," *AI magazine*, vol. 29, no. 1, pp. 9–9, 2008.

[2] V. Bela, "China stakes global dominance in race to build intelligent ports," https://www.scmp.com/news/china/science/article/3250341/china-stakes-global-dominance-race-build-intelligent-ports, 2024, accessed: 2025-01-24.

[3] A. Yalcin, "Multi-agent route planning in grid-based storage systems," Ph.D. dissertation, Europa-Universität Viadrina Frankfurt, 2017.

[4] "Hidden like secret bases – automated multistory parking facilities," https://web-japan.org/trends/11_tech-life/tec170223.html, accessed: 2025-01-24.

[5] K. R. Gue and B. S. Kim, "Puzzle-based storage systems," *Naval Research Logistics (NRL)*, vol. 54, no. 5, pp. 556–567, 2007.

[6] K. R. Gue, "Very high density storage systems," *IIE transactions*, vol. 38, no. 1, pp. 79–90, 2006.

[7] Y. Bukchin and T. Raviv, "A comprehensive toolbox for load retrieval in puzzle-based storage systems with simultaneous movements," *Transportation Research Part B: Methodological*, vol. 166, pp. 348–373, 2022.

[8] K. R. Gue, O. Uludag, and K. Furmans, "A high-density system for carton sequencing," in *Proceedings of the international material handling research colloquium*, 2012.

[9] S. Boge and S. Knust, "The parallel stack loading problem minimizing the number of reshuffles in the retrieval stage," *Eur. J. Oper. Res.*, vol. 280, no. 3, pp. 940–952, 2020.

[10] C. Lersteau and W. Shen, "A survey of optimization methods for block relocation and premarshalling problems," *Computers & Industrial Engineering*, vol. 172, p. 108529, 2022.

[11] M. Caserta, S. Schwarze, and S. Voß, "Container rehandling at maritime container terminals: A literature update," *Handbook of terminal planning*, pp. 343–382, 2020.

[12] I. K. Hanou, M. M. de Weerdt, and J. Mulderij, "Moving trains like pebbles: A feasibility study on tree yards," in *ICAPS*. AAAI Press, 2023, pp. 482–490.

Undecidability of Tiling with a Tromino

MIT–ULB CompGeom Group* Zachary Abel† Hugo Akitaya‡ Lily Chung§
Erik D. Demaine§ Jenny Diomidova§ Della Hendrickson§ Stefan Langerman¶
Jayson Lynch§

Abstract

Given a periodic placement of copies of a tromino (either  or ), we prove co-RE-completeness (and hence undecidability) of deciding whether it can be completed to a plane tiling. By contrast, the problem becomes decidable if the initial placement is finite, or if the tile is a domino instead of a tromino (in any dimension). As a consequence, tiling a given periodic subset of the plane with a given tromino ( or ) is co-RE-complete.

We also prove co-RE-completeness of tiling the entire plane with two polyominoes (one of which is disconnected and the other of which has constant size), and of tiling 3D space with two connected polycubes (one of which has constant size). If we restrict to tiling by translation only (no rotation), then we obtain co-RE-completeness with one more tile: two trominoes for a periodic subset of 2D, three polyominoes for the 2D plane, and three connected polycubes for 3D space.

Along the way, we prove several new complexity and algorithmic results about periodic (infinite) graphs. Notably, we prove that Periodic Planar (1-in-3)SAT-3, 3DM, and Graph Orientation are co-RE-complete in 2D and PSPACE-complete in 1D; we extend basic results in graph drawing to 2D periodic graphs; and we give a polynomial-time algorithm for perfect matching in bipartite periodic graphs.

1 Introduction

Given one or more ***prototiles*** (shapes) and a target ***space*** (e.g., the plane), a ***tiling*** [GS87] is a covering of the space with nonoverlapping copies of the prototiles, called ***tiles***, without gaps or overlaps. By default, we allow the copies to translate, rotate, and reflect, though reflections do not affect our (or most) results, and we will also consider translation-only tiling. In this paper, we study three fundamental computational problems about tilings:

Problem 1 (***d***D Tiling)**.** *Given one or more prototiles, can they tile d-dimensional Euclidean space?*

*Artificial first author to highlight that the other authors (in alphabetical order) worked as an equal group. Please include all authors (including this one) in your bibliography, and refer to the authors as "MIT–ULB CompGeom Group" (without "et al.").

†Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA, USA, zabel@mit.edu

‡Miner School of Computer and Information Sciences, University of Massachusetts, Lowell, MA, USA, hugoakitaya@gmail.com. Supported by NSF grant CCF-2348067.

§Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA, USA, {lkdc,edemaine,diomidova,della,jaysonl}@mit.edu

¶Computer Science Department, Université libre de Bruxelles, Belgium, stefan.langerman@ulb.ac.be. S. Langerman is Directeur de Recherche du F.R.S.-FNRS.

Problem 2 (***d***D Tiling Completion)**.** *Given one or more prototiles, and given some already placed tiles, can this partial placement be extended to a tiling of d-dimensional Euclidean space?*

Problem 3 (***d***D Subspace Tiling)**.** *Given one or more prototiles, and given a subset of d-dimensional Euclidean space, can the prototiles tile that space?*

Problem 1 is a special case of Problem 2 (with no preplaced tiles), and Problem 2 is a special case of Problem 3 (where the preplaced tiles form the excluded subspace). In Problems 2 and 3, there are multiple ways to specify the preplaced tiles or subspace respectively:

1. **Finite**: There are finitely many preplaced tiles, or finitely many excluded regions from d-dimensional space, and we encode each explicitly.
2. **Periodic**: The preplaced tiles or excluded regions are periodic in $d' \leq d$ dimensions, and we encode the fundamental domain and the d' translation vectors along which to repeat the fundamental domain. (Our results use $d' = d$.)
3. **Eventually periodic**: The preplaced tiles or excluded regions are periodic outside a finite region, so we use a hybrid: a periodic encoding, plus an explicit finite list of exceptions (excluded/included preplaced tiles or excluded regions). (Our results do not use this form of the problems, but we mention it for completeness.)

All three problems have been shown ***undecidable*** (solved by no finite algorithm) in a variety of settings. Such undecidability proofs generally simulate a Turing machine, where finding an (infinite) tiling corresponds to the machine running forever, which shows ***co-RE-hardness***. Recently, Demaine and Langerman [DL25] proved that these problems are in co-RE in very general settings, and thus co-RE-hardness in fact establishes ***co-RE-completeness***.

Table 1 summarizes the history of many such results, focusing on Problem 1, but also capturing Problem 3 in the form of a periodic "piece". In general, we aim for undecidability under the following objectives: (1) Minimize the target dimension d. In addition to integer d, we define $d = i + \frac{1}{2}$ to consist of i infinite real dimensions plus one bounded dimension given by a real interval. For example, 2.5D means $\mathbb{R}^2 \times [a, b]$ for some $a, b \in \mathbb{R}$; (2) Minimize the number of distinct prototiles required; (3) Simplify the prototile shapes; (4) Minimize the preplaced tiles or excluded subspace.

In this paper, we improve the state-of-the-art for all three problems. In this abstract, we focus on the tile completion problem. See the full version [GAA+25] for details on the other results.

We obtain particularly tight results for tiling completion with a single prototile:

1. Given a periodic preplacement of copies of a single tromino ( or ) in 2D, tiling completion is co-RE-complete, and hence undecidable. As a consequence, there are periodic preplacements that can be completed but only aperiodically. This undecidability result is tight by the following contrasting results:
2. Given a periodic preplacement of copies of a single tromino ( or ) in *1.5D*, tiling completion is PSPACE-complete, and hence decidable.
3. Given a *finite* preplacement of copies of a single tromino ( or ) in 2D, tiling completion is NP-complete, and hence decidable.

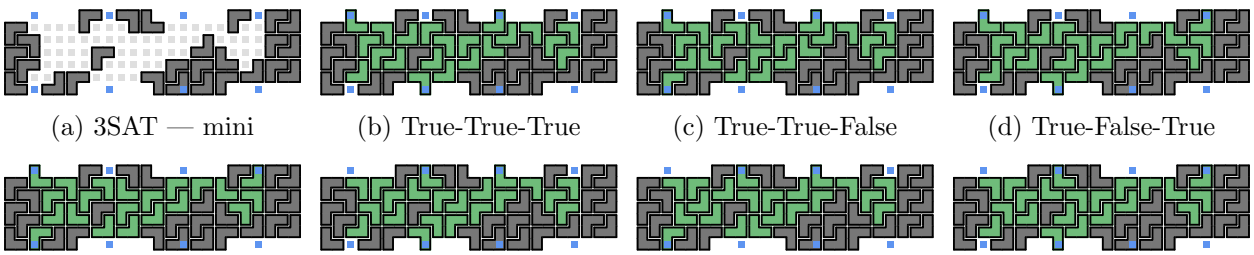
Dim.	Number/types of pieces		Result	Date
	Tiling by translation	by rotation + translation		
dD	**1 disconnected polycube + periodic**	n/a	undecidable [GT25]	2023-09
4D	4 connected polycube	n/a	undecidable [YZ24d]	2024-09
4D	3 connected polycube	n/a	undecidable [YZ24a]	2024-12
3D	6 connected polycube	n/a	undecidable [YZ25b]	2024-08
3D	3 connected polycube	n/a	undecidable [YZ25a]	2025-07*
3D	**2 connected polycube**	n/a	undecidable [Kim25b]	2025-08*
2.5D	**3 connected polycube**	**2 connected polycube**	undecidable	NEW
2D	n connected polyomino	n connected polyomino	undecidable [Gol70]	1970
2D	11 connected polyomino	5 connected polyomino	undecidable [Oll09]	2009-04
2D	10 connected polyomino	n/a	undecidable [Yan23]	2023-02
2D	9 connected polyomino	n/a	undecidable [Yan25]	2024
2D	8 connected polyomino	n/a	undecidable [YZ24b]	2024-03
2D	7 polyomino	n/a	undecidable [YZ24c]	2024-12
2D	7 orthoconvex polyomino	n/a	undecidable [YZ25d]	2025-06*
2D	4 disconnected polyomino	n/a	undecidable [YZ25c]	2025-06*
2D	n/a	3 polygons, or 2 polygons + periodic	undecidable, co-RE complete [DL25]	2024-09
2D	4 polyhex	**2 polyhex** or **3 connected polyomino**	undecidable [Sta25]	2025-06*
2D	**5 connected polyomino**	**3 connected polyomino**	undecidable [Kim25a]	2025-08*
2D	**3 polyomino: 2 connected + 1 disconnected**	**2 polyomino: 1 connected + 1 disconnected**	undecidable	NEW
2D	**2 tromino + periodic**	**1 tromino + periodic**	undecidable	NEW
2D	2 polyomino	1 polyomino	OPEN	—
dD	domino + periodic	domino + periodic	polynomial	NEW
2D	1 connected polyomino	n/a	decidable, $O(n)$ [Win15]	2015
2D	1 disconnected polyomino	n/a	decidable, periodic [Bha20]	2016-02
1.5D	**2 tromino + periodic**	**1 tromino + periodic**	PSPACE-complete	NEW
1.5D	**3 polyomino: 2 connected + 1 disconnected**	**2 polyomino: 1 connected + 1 disconnected**	PSPACE-complete	NEW
1D	n polyomino	n polyomino	decidable, $O(n)$ [GT23]	2023

Table 1: Past and new (un)decidability results for tiling Euclidean space, in various dimensions, and tiling by translation (left) or by rotation/translation (right). Bold indicates current hardness record holders. Dates are year-month, and "*" indicates recent independent work. Dashed lines separate undecidable from decidable. Our results are highlighted in blue and red. We give some intuition for the red result below.

4. Given a periodic preplacement of *dominoes* in dD for any $d \geq 1$, tiling completion can be solved in polynomial time. This is a consequence of the mathematical property that, if a periodic preplacement of dominoes in dD can be completed to a tiling, then it can be completed periodically with the same period.

Our results are the first to prove undecidability of tiling completion with a *single prototile*. The earliest undecidability result for tiling completion was by Robinson [Rob71], who used 36 Wang tiles (which can be implemented by 36 polyominoes [Gol70]) and required only finite preplacement. Yang [Yan13] showed how rule 110 can be simulated using 6 Wang tiles, and thus 6 polyominoes. (Rule 101 is a one dimensional cellular automaton where the next state of a cell depends on the its current state and the current state of the two neighboring cells, similar to Conway's Game of Life.) Together with Cook's undecidability of rule 110 for an eventually periodic initial configuration [Coo04], this implies undecidability of tiling completion with 6 polyominoes and eventually periodic preplacement [DL25], using only translations. More recently, this bound was lowered for fixed general polygons to 3 using finite preplacement, or 2 using eventually periodic preplacement [DL25]. In addition to reducing to 1 shape and simplifying the shape to a polyomino, we characterize the exact size of polyomino (3) and dimension (2) required for undecidability, while (necessarily) requiring periodic preplacement.

We show, in the full version, that the problem of finding a satisfying assignment to an infinite instance of Planar 3SAT given by a planar periodic graph is undecidable. We reduce from this problem to the problem of tile completion. Figure 1 shows the clause gadget for trominoes. There are four blue dots at the top and four blue dots at the bottom of the gadget. The first, second and third literal are encoded by the top left, top right, and bottom left pairs of blue dots, respectively. The gadget ensures that exactly one, among the pair of blue dots of a literal, is covered by an trominoes contained in the gadget. The truth assignment is then encoded by which of the blue dots is covered. The gadget was found by computer search and we used computer search to verify that a False-False-False assignment cannot be extended to a tiling.



(a) 3SAT — mini (b) True-True-True (c) True-True-False (d) True-False-True

(e) True-False-False (f) False-True-True (g) False-True-False (h) False-False-True

Figure 1: 3SAT for trominoes.

Acknowledgments

This work grew out of two research groups: the MIT Hardness Group and the MIT CompGeom Group. We thank the other members of these groups — in particular, Josh Brunner, Craig Kaplan, Hayashi Layers, Anna Lubiw, Joseph O'Rourke, Mikhail Rudoy, and Frederick Stock — for helpful discussions.

References

[Bha20] Siddhartha Bhattacharya. Periodicity and decidability of tilings of $\mathbb{Z}^2$. *American Journal of Mathematics*, 142(1):255–266, 2020.

[Coo04] Matthew Cook. Universality in elementary cellular automata. *Complex Systems*, 15(1):1–40, 2004.

[DL25] Erik D. Demaine and Stefan Langerman. Tiling with three polygons is undecidable. In *Proceedings of the 41st International Symposium on Computational Geometry (SoCG 2025)*, pages 39:1–39:16, Kanazawa, Japan, June 2025.

[GAA+25] ULB CompGeom Group, Zachary Abel, Hugo Akitaya, Lily Chung, Erik D Demaine, Jenny Diomidova, Della Hendrickson, Stefan Langerman, and Jayson Lynch. Undecidability of tiling with a tromino. *arXiv preprint arXiv:2509.07906*, 2025.

[Gol70] Solomon W. Golomb. Tiling with sets of polyominoes. *Journal of Combinatorial Theory*, 9(1):60–71, 1970.

[GS87] Branko Grünbaum and G. C. Shephard. *Tilings and Patterns*. W. H. Freeman, 1987.

[GT23] Rachel Greenfeld and Terence Tao. Undecidable translational tilings with only two tiles, or one nonabelian tile. *Discrete & Computational Geometry*, 70(4):1652–1706, 2023.

[GT25] Rachel Greenfeld and Terence Tao. Undecidability of translational monotilings. *Journal of the European Mathematical Society*, 2025. Originally arXiv:2309.09504.

[Kim25a] Yoonhu Kim. Undecidability of tiling the plane with a set of 5 polyominoes. arXiv:2508.10067, 2025. https://arxiv.org/abs/2508.10067.

[Kim25b] Yoonhu Kim. Undecidability of translational tiling with 2 polycubes. arXiv:2508.11725, 2025. https://arxiv.org/abs/2508.11725.

[Oll09] Nicolas Ollinger. Tiling the plane with a fixed number of polyominoes. In *Proceedings of the 3rd International Conference on Language and Automata Theory and Applications (LATA 2009)*, volume 5457 of *Lecture Notes in Computer Science*, pages 638–647, Tarragona, Spain, April 2009. Springer.

[Rob71] Raphael M. Robinson. Undecidability and nonperiodicity for tilings of the plane. *Inventiones Mathematicae*, 12(3):177–209, September 1971.

[Sta25] Jack Stade. Two tiling is undecidable. arXiv:2506.11628, 2025. https://arxiv.org/abs/2506.11628.

[Win15] Andrew Winslow. An optimal algorithm for tiling the plane with a translated polyomino. In *Algorithms and Computation (ISAAC 2015)*, volume 9472 of *Lecture Notes in Computer Science*, pages 3–13. Springer, 2015.

[Yan13] Jed Chang-Chun Yang. *Computational Complexity and Decidability of Tileability*. PhD thesis, University of California, Los Angeles, 2013.

[Yan23] Chao Yang. Tiling the plane with a set of ten polyominoes. *International Journal of Computational Geometry & Applications*, 33(03n04):55–64, 2023.

[Yan25] Chao Yang. On the undecidability of tiling the plane with a set of nine polyominoes (in chinese). *SCIENCE CHINA Mathematics (Chinese Edition) [SCIENTIA SINICA Mathematica]*, 55(6):1113–1122, 2025.

[YZ24a] Chan Yang and Zhujun Zhang. Undecidability of translational tiling with three tiles. arXiv:2412.10646, 2024. https://arxiv.org/abs/2412.10646.

[YZ24b] Chao Yang and Zhujun Zhang. A proof of Ollinger's conjecture: undecidability of tiling the plane with a set of 8 polyominoes. arXiv:2403.13472, 2024. https://arxiv.org/abs/2403.13472.

[YZ24c] Chao Yang and Zhujun Zhang. Translational aperiodic sets of 7 polyominoes. arXiv:2412.17382, 2024. https://arxiv.org/abs/2412.17382.

[YZ24d] Chao Yang and Zhujun Zhang. Undecidability of translational tiling of the 4-dimensional space with a set of 4 polyhypercubes. arXiv:2409.00846, 2024. https://arxiv.org/abs/2409.00846.

[YZ25a] Chao Yang and Zhujun Zhang. On the undecidability of tiling the 3-dimensional space with a set of 3 polycubes. arXiv:2508.00192, 2025. https://arxiv.org/abs/2508.00192. In Chinese.

[YZ25b] Chao Yang and Zhujun Zhang. Undecidability of translational tiling of the 3-dimensional space with a set of 6 polycubes. *Proceedings of the American Mathematical Society*, 153(8):3541–3554, May 2025. Originally arXiv:2408.02196.

[YZ25c] Chao Yang and Zhujun Zhang. Undecidability of translational tiling of the plane with four tiles. arXiv:2506.19295, 2025. https://arxiv.org/abs/2506.19295.

[YZ25d] Chao Yang and Zhujun Zhang. Undecidability of translational tiling of the plane with orthogonally convex polyominoes. arXiv:2506.12726, 2025. https://arxiv.org/abs/2506.12726.

Planar Emulators for Geometric Intersection Graphs

(Extended Abstract)*

Hsien-Chih Chang† Jonathan Conroy‡ Zihan Tan§ Da Wei Zheng¶

1 Introduction

Given a set of geometric objects $\mathcal{S}$, the *intersection graph* of $\mathcal{S}$, denoted $G_{\mathcal{S}}$, is a graph with vertex set $\mathcal{S}$, such that an edge exists between two objects if and only if they intersect. The most general form of geometric intersection graph in the plane is the class of *string graphs*, where vertices are arbitrary connected curves. One may also study intersection graphs of more restricted class of objects, such as disks, fat objects, or axis-aligned rectangles.

Our goal is to study *distances* between objects on a geometric intersection graph G. The *distance* between two vertices u and v in G, denoted $\delta_G(u,v)$, is the number of edges in the shortest path between u and v. (Note that we assume edges are unweighted; if we allowed arbitrary edge weights then all metrics are realizable as string graph metrics, as the complete graph is a string graph.[1]) Distance problems on geometric intersection graphs have attracted a great deal of attention, including diameter computation [CS19b, CS19a, BKBK+22, CGL24, CCG+25], clustering [BFI23, FRSS25], spanners [YXD12, CCV10, Bin20, CT23, CH23], low-diameter decomposition and its applications to LP rounding [Lee17, KLSS21, KLS+22, LPS+24], and much more. Many results have been obtained for restricted classes of intersection graphs rather than the most general string graphs.

In partial explanation, several results on unit-disk graphs (UDGs) can be traced back to the existence of a $O(1)$-distortion *planar spanner*: every unit-disk graph G has a subgraph H which is planar, such that for every pair of vertices (u,v), we have $\delta_G(u,v) \le \delta_H(u,v) \le O(1)\cdot\delta_G(u,v)$. Planar metrics are well studied, and tools from planar graphs (in particular, shortest-path separators) can be combined with the planar spanner to design algorithms for UDGs. The planar spanner for UDGs is helpful for distance problems even beyond the O(1)-approximate regime; for example, it is a crucial ingredient for a near-linear time $(1+\varepsilon)$-approximate diameter algorithm and a compact $(1+\varepsilon)$-approximate distance oracle [CS19b], and in the design of coresets for clustering [BFI23]. (Note that one can also obtain shortest-path separators on UDG directly, not using a planar spanner [YXD12, HHZ24]; this is another way in which the metric structure of UDGs is similar to that of planar metrics.)

The use of planar techniques for geometric intersection graphs so far has seemed somewhat limited to UDGs and closely related classes: for example, the only classes of intersection graphs with planar spanners (that we are aware of) are Euclidean weighted UDGs [LCW02], unweighted UDGs [Bin20], and Euclidean weighted unit-square graphs [BFI23]. General string graphs could have much more

*This is an extended abstract. All details are deferred to the full paper.

†Department of Computer Science, Dartmouth College. Email: hsien-chih.chang@dartmouth.edu.

‡Department of Computer Science, Dartmouth College. Email: jonathan.conroy.gr@dartmouth.edu

§Department of Computer Science and Engineering, University of Minnesota. Email: ztan@umn.edu

¶Institute of Science and Technology Austria. Email: dzheng@ista.ac.at

[1]For some classes of geometric intersection graphs, one can instead impose a natural weight on the edges—for example, a unit-disk graph may have edge weights set to be the Euclidean distance between the disk centers. We are primarily concerned with unweighted string graphs.

complex interactions than UDGs. Unlike disks and squares, the strings may be *piercing* and thus permit non-local interactions like segments and rectangles; moreover, an n-vertex string graph may even require a representation (i.e. a drawing on the plane) with description complexity exponential in n [KM91].

As a result, it might come as a surprise that a planar distance sketching structure still exists for *general* string graphs, even in the $O(1)$-distortion regime! Our main result vastly generalizes the previous spanner constructions on UDGs: every string graph admits an $O(1)$-distortion *planar emulator*. Philosophically speaking, our result suggests that metrics on string graphs are not so different from planar metrics.

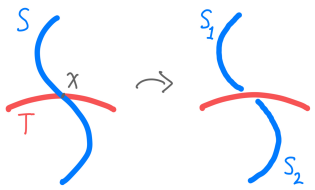
Theorem 1.1. *For an absolute constant c, the following is true: For any unweighted string graph G, there is a planar graph H with $V(H) = V(G)$, such that for every pair of vertices (u, v), we have $\delta_G(u, v) \le \delta_H(u, v,) \le c \cdot \delta_G(u, v)$. Moreover, if one is given a representation of G, the graph H can be computed in time polynomial in the description complexity of the representation.*

As one application, we can obtain $(1+\varepsilon)$-approximate distance oracles for string graphs, by combining (in a non-black-box manner) a result on distance oracles in planar graphs [CCL+23] with our Theorem 1.1.

Corollary 1.2. *For any n-vertex string graph G and any $\varepsilon > 0$, there is a data structure using $O_\varepsilon(n)$ words of space*[2] *such that, given any two query vertices $u, v \in V(G)$, in $O_\varepsilon(1)$ time we can return a distance estimate $\tilde{\delta}(u, v)$ such that $\delta_G(u, v) \le \tilde{\delta}(u, v) \le (1 + \varepsilon) \cdot \delta_G(u, v) + O(1)$.*

Parallel work. A similar result to our Theorem 1.1 has recently been announced[3] by James Davies: every (unweighted) string graph is *quasi-isometric*[4] to an (unweighted) planar graph. This work was done independently and concurrently.

2 Technical Overview



We begin by describing a natural, but *flawed* approach at creating a planar emulator for a string graph $G_{\mathcal{S}}$ on a set of strings $\mathcal{S}$. Let S and T be two strings that cross at some point $x \in \mathbb{R}^2$. The operation of *shattering* of S at x can be thought of as cutting apart S at x to produce two new strings S_1 and S_2, both of which *touch* T but neither of which *crosses* T at x. Given a set of strings $\mathcal{S}$, we could transform their intersection graph $G_{\mathcal{S}}$ into a planar graph by the following procedure: for every pair of strings that cross, shatter one of them to remove the crossing.[5] Each shattering procedure removes a crossing at the cost of changing distances by an O(1) amount — the single string S is turned into two strings S_1 and S_2 which are within distance 2 of each other in the new intersection graph (as witnessed by the path $[S_1, T, S_2]$). After all crossings are removed by the shattering procedure, it is not difficult to see that the resulting intersection graph on the shattered strings $\mathcal{S}'$ is planar. However, distances on the shattered graph $G_{\mathcal{S}'}$ could look very different than on the original $G_{\mathcal{S}}$. When the intersection graph G is a clique (for example), some string S might cross every other string, so S could be shattered into $\Theta(n)$ pieces, and these pieces could be at distance $\Theta(n)$ from each other. In this case, shattering doesn't seem helpful in producing a planar emulator.

[2] we work in the Word RAM model

[3] see the 31th minute mark of https://youtu.be/U1puthN5tq8?t=1861

[4] This is essentially equivalent to the existence of a $O(1)$-distortion emulator, up to some quirks in the definition. A quasi-isometry implies the existence of a $O(1)$-distortion emulator, and our proof of our emulator implies a quasi-isometry.

[5] When a string S is shattered into S_1 and S_2, one of the strings S_1 or S_2 is chosen arbitrarily to represent S, and the other string is treated as a Steiner vertex.

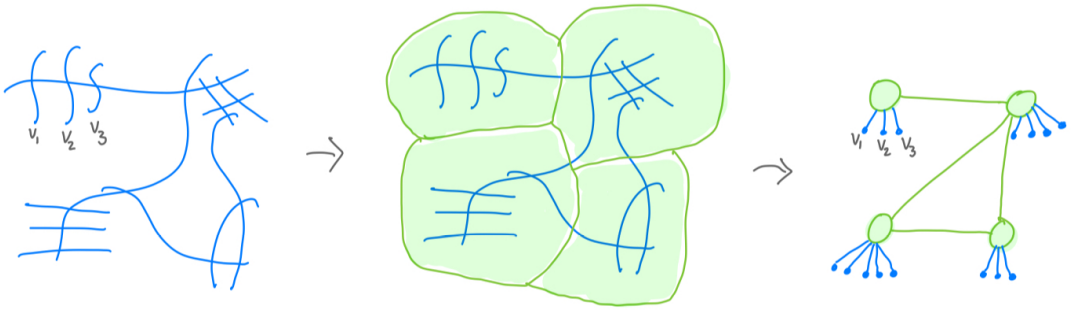


Figure 1. (Left) A string graph G; (Middle) A partition of $\mathbb{R}^2$ into clusters provided by Lemma 2.1; (Right) A planar emulator for G obtained by contracting all clusters and adding stars.

On the other hand, it is easy to find a planar emulator H when G is a clique: just take H to be a star graph. This gives us the following hope: we could try to remove some crossings using the shattering procedure (only shattering each string $O(1)$ times), and hope that the remaining intersections happen in local "clusters" that can be replaced with stars.

> **Goal.** Given strings $\mathcal{S}$, find a partition of $\mathbb{R}^2$ into connected clusters such that (1) each string intersects only $O(1)$ clusters, and (2) for each cluster C, the strings touched by C are all within $O(1)$ distance of each other in $G_{\mathcal{S}}$.

Given such a partition, it is fairly straightforward to find a planar emulator, by shattering along the boundary of each cluster and then replacing each cluster with a star; see the proof of Theorem 1.1 below for the details, and Figure 1 for an example. Now, a key insight is that our stated goal is reminiscent of the notion of *scattering partition* introduced by Filtser for general graphs [Fil24].

> **Scattering partition.** A graph G has a $O(1)$-scattering partition if, for any distance $\Delta > 0$, there is a partition of $V(G)$ into connected clusters such that (1) any path of length Δ intersects only $O(1)$ clusters, and (2) each cluster has diameter $O(\Delta)$.

Our intuition is that the scattering partition with $\Delta = O(1)$ is similar to the partition in our goal. Unfortunately, scattering partition is only known for some very restricted graph classes, including trees, cactus graphs [Fil24], and series-parallel graphs [HL22]. Recently, however, Chang *et al.* [CCL+23] constructed a very similar (but slightly weaker) object, the *shortcut partition*, for every *planar graph* (and later for every minor-free graph [CCL+24]). Our main technical contribution, summarized in Lemma 2.1, is to delicately adapt the planar construction of [CCL+23] to (essentially) achieve our goal.

Lemma 2.1. *There is a constant $\alpha = O(1)$ such that, given a set of strings $\mathcal{S}$, there is a partition of $\mathbb{R}^2$ into connected regions $\mathcal{C}$ called clusters, such that:*

- [Low-hop.] *Define the cluster graph $\hat{H}$ to be the (planar, unweighted) string graph obtained by shattering along the boundary of every cluster in $\mathcal{C}$ and then contracting the strings in each cluster.*[6] *For any string $S \in \mathcal{S}$, if S intersects clusters C_1 and C_2, then $\delta_{\hat{H}}(C_1, C_2) \leq \alpha$.*
- [Diameter.] *For any cluster $C \in \mathcal{C}$, if strings $S_1, S_2 \in \mathcal{S}$ both intersect C, then $\delta_{G_{\mathcal{S}}}(C_1, C_2) \leq \alpha$.*

From here we can prove Theorem 1.1.

[6] More formally, $\hat{H}$ is the graph with vertex set $\mathcal{C}$ where an edge exists between clusters $C_1, C_2 \in \mathcal{C}$ iff C_1 and C_2 touch (that is, $\partial C_1 \cap \partial C_2 \neq \emptyset$) and there is some string $S \in \mathcal{S}$ that intersects both C_1 and C_2. It is not hard to see that H is planar.

Proof (of Theorem 1.1): Let $\mathcal{C}$ be the set of clusters, α be the constant, and $\hat{H}$ be the (planar) cluster graph from Lemma 2.1. Initialize $H \leftarrow \hat{H}$. For every string $S \in \mathcal{S}$, create a new vertex in H representing S, and add an edge between this vertex and an arbitrary cluster $C \in V(\hat{H})$ that intersects S; see Figure 1. Set all edge lengths of H to 2α. We claim H is a $O(1)$-distortion planar emulator for the string graph $G_{\mathcal{S}}$.

First we prove an upper bound on distances in H: for any $u, v \in \mathcal{S}$, we claim $\delta_H(u,v) \leq O(1) \cdot \delta_{G_{\mathcal{S}}}(u,v)$. It suffices to show that for every edge (u,v) in $G_{\mathcal{S}}$, we have $\delta_H(u,v) \leq O(\alpha^2) = O(1)$. Indeed, the fact that $u \cap v \neq \emptyset$ implies there is some cluster $C \in \mathcal{C}$ that intersects both u and v. The [low-hop] property implies that there is a path in H between C and u (resp. C and v) with $O(\alpha)$ hops; as each edge has length 2α, the path has length $O(\alpha^2) = O(1)$ so we are done.

Now we prove a lower bound on distances in H: for any $u, v \in \mathcal{S}$, we claim $\delta_{G_{\mathcal{S}}}(u,v) \leq \delta_H(u,v)$. For any two string $u, v \in \mathcal{S}$, let $P_H = [u, C_1, C_2, \ldots, C_\ell, v]$ be a shortest path in H between strings u and v; note that every vertex other than the endpoints represents some cluster $C_i \in \mathcal{C}$. As every edge in H has length α, there are $\frac{\|P_H\|}{2\alpha}$ edges in P_H.[7] Now, for each cluster C_i along the path P_H, let $c_i \in \mathcal{S}$ be an arbitrary string that intersects C_i, and let P be the shortest path in the string graph $G_{\mathcal{S}}$ that walks from u to c_1, then to c_2, and so on until reaching v. We claim that $\|P\| \leq \|P_H\|$. It suffices to show that, for any c_i and c_{i+1}, we have $\delta_{G_{\mathcal{S}}}(c_i, c_{i+1}) \leq 2\alpha$. Indeed, the edge $(C_i, C_{i+1}) \in E(H)$ implies that there is some string c' that intersects both clusters C_i and C_{i+1}, and the [diameter] property implies that $\delta_{G_{\mathcal{S}}}(c_i, c_{i+1}) \leq \delta_{G_{\mathcal{S}}}(c_i, c') + \delta_{G_{\mathcal{S}}}(c', c_{i+1}) \leq 2\alpha$. □

2.1 Some intuition for the proof of Lemma 2.1

Dealing with a path. First observe that Lemma 2.1 is easier to prove in the special case where the intersection graph $G_{\mathcal{S}}$ consists of a shortest path π (called the "spine") together with some strings within $O(1)$ distance of π. We start by selecting a set $\mathcal{P}$ of well-separated strings along π: add the first string on π to $\mathcal{P}$, then walk $O(1)$ steps along π and add the next string to $\mathcal{P}$, and repeat. For every string $P \in \mathcal{P}$, we create a cluster C_P, where we promise that C_P will only intersect strings within $O(1)$ distance from P. (The precise definition of the clusters is somewhat involved, but one should roughly think of a Voronoi partition according to distances in $G_{\mathcal{S}}$.) Each cluster C_P satisfies the [diameter] property. Moreover, it follows from the fact that π is a shortest path that any string S intersects only $O(1)$ clusters.

Partition into paths. To prove Lemma 2.1 on general string graphs, we aim to first partition $\mathbb{R}^2$ into *spined clusters* $\mathcal{C}$ where (1) every string intersects few spined clusters, and (2) for each spined cluster C, the strings intersecting C consist of a shortest path plus strings within $O(1)$ distance of the path.[8] To do this, we follow the algorithm of [CCL+23] (which itself builds on [BLT07]) for planar graphs. To begin, they select an arbitrary vertex v on the outer face and initialize π to be a $O(1)$ neighborhood around v. They then repeatedly perform the following process: add π to the set of spined clusters $\mathcal{C}$; delete all vertices in π; then look at the two external edges adjacent to π, connecting π to vertices v_1' and v_2', and set π to be the shortest path from v_1' to v_2' plus all vertices within $O(1)$ distance of this path. In this way, π is a "thick path" that acts as a *separator* as it moves across all vertices in the outer face. We can perform a similar idea in string graphs, albeit with technical complications.[9] The "separator" property of π guarantees that each string S can intersect only 2 spined clusters (otherwise S would be deleted). There may be still be parts of the graph which are not covered by any spined cluster, so we recurse in a manner similar to [CCL+23].

[7] We use the notation $\|P\|$ to mean the total (weighted) length of path P.

[8] By the argument above, these spined clusters are easy to deal with, and can be partitioned independently. For technical reasons, we can't actually find these spined clusters, but this is the intuition.

[9] One key observation is that, even though the path in a string graph does not act as a separator, the path *together with its neighbors* acts a separator. Another idea is that, in order to ensure all clusters are disjoint, we can shatter along the boundary of π and compute shortest paths in this shattered graph.

References

[BFI23] Sayan Bandyapadhyay, Fedor V Fomin, and Tanmay Inamdar. Coresets for clustering in geometric intersection graphs. In *39th International Symposium on Computational Geometry (SoCG 2023)*, pages 10–1. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2023.

[Bin20] Ahmad Biniaz. Plane hop spanners for unit disk graphs: Simpler and better. *Computational Geometry*, 89:101622, 2020.

[BKBK+22] Karl Bringmann, Sándor Kisfaludi-Bak, Marvin Künnemann, André Nusser, and Zahra Parsaeian. Towards sub-quadratic diameter computation in geometric intersection graphs. In *38th International Symposium on Computational Geometry*, pages 1–16. Schloss Dagstuhl, 2022.

[BLT07] Costas Busch, Ryan LaFortune, and Srikanta Tirthapura. Improved sparse covers for graphs excluding a fixed minor. In *Proceedings of the twenty-sixth annual ACM symposium on Principles of distributed computing*, pages 61–70, 2007.

[CCG+25] Timothy M. Chan, Hsien-Chih Chang, Jie Gao, Sándor Kisfaludi-Bak, Hung Le, and Da Wei Zheng. Truly subquadratic time algorithms for diameter and related problems in graphs of bounded vc-dimension, 2025. To appear in FOCS 2025.

[CCL+23] Hsien-Chih Chang, Jonathan Conroy, Hung Le, Lazar Milenkovic, Shay Solomon, and Cuong Than. Covering planar metrics (and beyond): O(1) trees suffice. In *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 2231–2261. IEEE, 2023.

[CCL+24] Hsien-Chih Chang, Jonathan Conroy, Hung Le, Lazar Milenković, Shay Solomon, and Cuong Than. Shortcut partitions in minor-free graphs: Steiner point removal, distance oracles, tree covers, and more. In *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 5300–5331. SIAM, 2024.

[CCV10] Nicolas Catusse, Victor Chepoi, and Yann Vaxès. Planar hop spanners for unit disk graphs. In *International Symposium on Algorithms and Experiments for Sensor Systems, Wireless Networks and Distributed Robotics*, pages 16–30. Springer, 2010.

[CGL24] Hsien-Chih Chang, Jie Gao, and Hung Le. Computing diameter+ 2 in truly-subquadratic time for unit-disk graphs. In *40th International Symposium on Computational Geometry (SoCG 2024)*, pages 38–1. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2024.

[CH23] Timothy M Chan and Zhengcheng Huang. Constant-hop spanners for more geometric intersection graphs, with even smaller size. In *Proc. 39th International Symposium on Computational Geometry (SoCG 2023)*. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2023.

[CS19a] Timothy M Chan and Dimitrios Skrepetos. All-pairs shortest paths in geometric intersection graphs. *Journal of Computational Geometry*, 10(1):27–41, 2019.

[CS19b] Timothy M Chan and Dimitrios Skrepetos. Approximate shortest paths and distance oracles in weighted unit-disk graphs. *Journal of Computational Geometry*, 10(2):3–20, 2019.

[CT23] Jonathan Conroy and Csaba Tóth. Hop-spanners for geometric intersection graphs. *Journal of Computational Geometry*, 14(2):26–64, Dec. 2023. URL: https://jocg.org/index.php/jocg/article/view/4475, doi:10.20382/jocg.v14i2a3.

[Fil24] Arnold Filtser. Scattering and sparse partitions, and their applications. *ACM Transactions on Algorithms*, 20(4):1–42, 2024.

[FRSS25] Zachary Friggstad, Mohsen Rezapour, Mohammad R. Salavatipour, and Hao Sun. A QPTAS for Facility Location on Unit Disk Graphs. In Pat Morin and Eunjin Oh, editors, *19th International Symposium on Algorithms and Data Structures (WADS 2025)*, volume 349 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 27:1–27:18, Dagstuhl, Germany, 2025. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. URL: https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.WADS.2025.27, doi:10.4230/LIPIcs.WADS.2025.27.

[HHZ24] Elfarouk Harb, Zhengcheng Huang, and Da Wei Zheng. Shortest Path Separators in Unit Disk Graphs. In Timothy Chan, Johannes Fischer, John Iacono, and Grzegorz Herman, editors, *32nd Annual European Symposium on Algorithms (ESA 2024)*, volume 308 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 66:1–66:14, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. URL: https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.ESA.2024.66, doi:10.4230/LIPIcs.ESA.2024.66.

[HL22] D Ellis Hershkowitz and Jason Li. O (1) steiner point removal in series-parallel graphs. In *30th Annual European Symposium on Algorithms (ESA 2022)*, pages 66–1. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2022.

[KLS+22] Neeraj Kumar, Daniel Lokshtanov, Saket Saurabh, Subhash Suri, and Jie Xue. Point separation and obstacle removal by finding and hitting odd cycles. *arXiv preprint arXiv:2203.08193*, 2022.

[KLSS21] Neeraj Kumar, Daniel Lokshtanov, Saket Saurabh, and Subhash Suri. A constant factor approximation for navigating through connected obstacles in the plane. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 822–839. SIAM, 2021.

[KM91] Jan Kratochvíl and Jiří Matoušek. String graphs requiring exponential representations. *Journal of Combinatorial Theory, Series B*, 53(1):1–4, 1991. URL: https://www.sciencedirect.com/science/article/pii/009589569190050T, doi:10.1016/0095-8956(91)90050-T.

[LCW02] Xiang-Yang Li, Gruia Calinescu, and Peng-Jun Wan. Distributed construction of a planar spanner and routing for ad hoc wireless networks. In *Proceedings. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 3, pages 1268–1277. IEEE, 2002.

[Lee17] James R Lee. Separators in region intersection graphs. In *8th Innovations in Theoretical Computer Science Conference (ITCS 2017)*, pages 1–1. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2017.

[LPS+24] Daniel Lokshtanov, Fahad Panolan, Saket Saurabh, Jie Xue, and Meirav Zehavi. A 1.9999-approximation algorithm for vertex cover on string graphs. In *40th International Symposium on Computational Geometry (SoCG 2024)*, volume 293, page 72. Dagstuhl Publishing, 2024.

[YXD12] Chenyu Yan, Yang Xiang, and Feodor F Dragan. Compact and low delay routing labeling scheme for unit disk graphs. *Computational Geometry*, 45(7):305–325, 2012.

Forbidden-Minor Preserving Support Graphs for Distance Balls in Directed Graphs

(Extended Abstract)

Reilly Browne* Hsien-Chih Chang†

October 19, 2025

1 Introduction

A fundamental concept in the study of optimization problems is the notion of a *set system* (also known as a *hypergraph*). In the most general sense, a *set system* $(U, \mathcal{R})$ is a tuple containing a universe U of elements and a family $\mathcal{R}$ of subsets of U. A *support graph* of a set system $(U, \mathcal{R})$ is an undirected graph H on the elements of U such that for every set R in $\mathcal{R}$, the subgraph H_R of H induced by the elements of R is connected. One perspective is to view support graphs as the connectivity analog of other well-studied notions of *sketches*: sparsifiers for cuts, spanners for distances, and coresets for clustering.

Naturally, we would hope the sketch to retain properties of the original structure, perhaps more. Van Cleemput [18] and Voloshina and Feinberg [19] used the existence of a *planar* support graph to give meaningful notion of planarity in the hypergraph setting. As such, support graphs have seen interest in (hyper)graph drawing community [11, 12, 4]. In the realm of optimization, Pyrga and Ray [15] used support graphs to construct linear-sized ε-nets for several set systems with bounded VC dimension, including halfspaces in $\mathbb{R}^3$ and pseudodisks (more generally, r-admissable regions) in $\mathbb{R}^2$. Here the support graphs were required to be *sparse* (i.e. with linearly many edges) in order to bootstrap existing ε-net sizes down to $O(\frac{1}{\varepsilon})$. The existence of linear-sized ε-nets has many applications; for example, $O(1)$-approximation for set cover and hitting set problems can be obtained using multiplicative weight updates [3] (see also [6, 13]). Later on support graphs that are *planar* were used by Mustafa and Ray [14] to show a PTAS for various optimization problem in the geometric setting. Instead of ε-net, they exploited the existence of *sublinear-size* separators in planar graphs to show that an $O_\varepsilon(1)$-swap local search algorithm gives a $(1+\varepsilon)$-approximation for minimum geometric hitting set. The same paradigm of analyzing local search with separators on planar support was extended to other problems, such as maximum independent set and generalized set cover (including dominating set and hitting set) in pseudodisks and non-piercing regions [5, 7, 9, 16]. More recently, Roy [1] used planar supports to design a PTAS for covering the boundary of a simple orthogonal polygon with rectangles. (A survey on the many use of support graphs can be found in Raman and Singh [17].)

Distance ball systems. The existence of support graphs that are planar are so far almost entirely restricted to set systems that are pseudodisks [15] and their generalizations [16] (with the exception of Roy [1], which works with a special system within orthogonal polygons). For piercing systems like axis-parallel rectangles, there are configurations[1] where no planar support can exist. Consequently, support graphs were viewed as objects whose existence is only guaranteed when the underlying system is *planar-like* in some way. Our goal of this article is to shift this narrative: instead of focusing on planarity, a support graph of set system $(U, \mathcal{R})$ is really a *connectivity sketch* that exists for every graph G on U — as long as the sets in $\mathcal{R}$ are defined to be *distance balls* on G. Both planarity and sparseness come as a byproduct of the fact that support graphs can be constructed as a *minor* of G. (See Theorem 1.1 for a precise statement.) This change in perspective is justified; while distance

*Department of Computer Science, Dartmouth College. Email: reilly.browne.gr@dartmouth.edu.
†Department of Computer Science, Dartmouth College. Email: hsien-chih.chang@dartmouth.edu.

[1]Take k disjoint vertical rectangles and k disjoint horizontal rectangles and cross them in a grid-like pattern.

balls are not pseudodisks or even non-piercing, distance balls capture many good properties of pseudodisks and can be viewed as a discrete analog. This connection can further be made concrete by the equivalence of planar metrics and polygonal domains: it is known that any system of geodesic disks in a polygonal domain in $\mathbb{R}^2$ can be represented as a system of distance balls in a planar graph, and vice versa [2]. Therefore we can interpret our result as constructing support graphs for geodesic disks in a polygonal domain.

Terminologies. Let's first introduce the terminology needed for concrete discussion. We consider the distance balls in an arbitrary *directed* graph. We refer to the digraph that our distance ball system lives on as the *underlying graph* G. Note that all of our results apply to distance balls in undirected graphs. A *distance ball* D can be defined using a pair (c_D, r_D) where c_D is a vertex of G called the *center*, and a *radius* r_D which is a real number. Distance ball D is then defined to be the set of all vertices v whose shortest directed path from c_D has length less than r_D.

Given a system $(U, \mathcal{R})$, the *dual system* $(\mathcal{R}, U)$ has the elements of U treated as subsets of $\mathcal{R}$: if element e in U is a member of set R in $\mathcal{R}$ in the primal system, then R is a member of *set* e in the dual system. We use the term *dual support* of $(U, \mathcal{R})$ to refer to a support graph of the dual system $(\mathcal{R}, U)$. (We give an example of a dual support in the left subfigure of Figure 1.) The dual of a system of distance balls can also be thought of as a system of distance balls on the same underlying graph with the edge orientations flipped. Thus, we do not differentiate the two types of supports, but instead describe the support for the dual system since it is conceptually easier.

For other more complicated applications, sometimes we need to work with set systems that have two different types of sets. We define an *intersection system* $(U, \mathcal{R}, \mathcal{B})$ as a set system where the elements are the sets of $\mathcal{R}$ and the family of sets is $\mathcal{B}$, and we consider a set $R \in \mathcal{R}$ to be a "member" of set $B \in \mathcal{B}$ if there is at least one element of U is in the intersection $R \cap B$. We define an *intersection support* to be a support graph of an intersection system $(U, \mathcal{R}, \mathcal{B})$. In other words, an *intersection support* is an undirected graph $H(\mathcal{R}, \mathcal{B})$ where $\mathcal{R}$ and $\mathcal{B}$ are two families of subsets of some universe U and $H(\mathcal{R}, \mathcal{B})$ is a graph on $\mathcal{R}$ such that for every set $B \in \mathcal{B}$, all sets in $\mathcal{R}$ that contain a common element with B form a connected subgraph of $H(\mathcal{R}, \mathcal{B})$. (We give an example of an intersection support in the right subfigure of Figure 1.) Our notion of *intersection support* is equivalent to Raman and Ray's notion of planar support for intersection hypergraphs [16], except we do not require that our support graphs be planar. In our case, we are looking at the situation where $\mathcal{R}$ and $\mathcal{B}$ are both sets of distance balls that lie in some underlying graph G. It is important to note that this is a generalization of both dual and primal support for the case of distance balls, since we can always consider the case where $\mathcal{R}$ is a set of radius-0 balls on every vertex to get a primal support and the case where $\mathcal{B}$ is the set of radius-0 balls on every vertex to get a dual support.

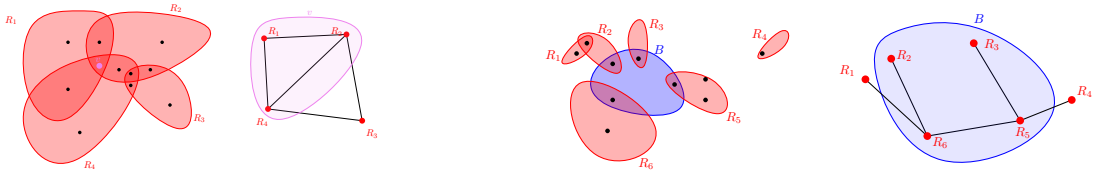


Figure 1. Left: A set system with a dual support. The red regions containing the violet vertex v form a connected subgraph in the support. Right: An intersection system with an intersection support. The red regions intersecting blue set B form a connected subgraph in the support.

Main results. We start by giving a short proof that we can construct a dual support for a system of distance balls, by contracting the underlying graph into a minor.

Theorem 1.1 (Dual Support). *Any system of distance balls $\mathcal{R}$ on an underlying digraph G has a dual support $H(\mathcal{R})$ such that if G does not contain a minor K with minimum vertex degree two[2], then neither does $H(\mathcal{R})$.*

After showing how to construct a forbidden-minor preserving dual support, we then show how to extend the technique to obtain an intersection support which also preserves forbidden minors.

Theorem 1.2 (Intersection Support). *Any system of distance balls $(\mathcal{R}, \mathcal{B})$ on an underlying graph G has an intersection support $H(\mathcal{R}, \mathcal{B})$ such that if G does not contain a (contraction-)minor K with minimum vertex degree two, then neither does $H(\mathcal{R}, \mathcal{B})$.*

[2]Unfortunately the min-degree requirement in the statement cannot be dropped. For example, if G is P_2-minor-free, we have a set of disjoint vertices. But there can be two balls centered on the same vertex, and thus connected in the dual support, which implies the existence of an edge and thus the support is not P_2-minor-free. Still the restriction is not severe as most of the interesting minor-free graph classes satisfy the min-degree condition, including K_h-minors for $h \geq 3$, grid minors, and all forbidden minors for bounded-genus graphs.

Applications of our results. An important property of our support construction is that we only ever contract edges that are within radius Δ of a distance ball center c_D, where Δ is the maximum radius of all distance balls. This means that if G has *polynomial expansion* and we are considering constant radius balls, the support graph will also have polynomial expansion, allowing us to use the sublinear separators from Har-Peled and Quanrud [10] to get the same PTAS results for Δ-balls as Raman and Ray [16] achieved for non-piercing regions. If Δ is not bounded by a constant, we can achieve the same when G is minor-free [20].

Another natural use of these techniques is for problems relating to geodesic disks in a polygonal domain on some fixed genus surface. Again by the equivalence between planar metrics and polygonal domains mentioned [2], our techniques naturally give planar supports for geodesic disks in polygonal domains that lie in $\mathbb{R}^2$, as well as bounded-genus supports for geodesic disks in polygonal domains on bounded-genus surfaces.

2 Existence of Dual Support

To begin constructing the dual support for a system of balls $\mathcal{R}$ in a directed graph G, we will first modify G slightly and reduce to the case where *every ball in $\mathcal{R}$ has the same radius*. We will refer to the new graph as G'. We augment G by adding a new node corresponding to each ball in $\mathcal{R}$. (For clarity, we refer to original vertices of G as *vertices* and new vertices corresponding to $\mathcal{R}$ as *nodes*.) For each ball R in $\mathcal{R}$, we create a single node, x_R, which has only one edge: an outgoing edge into c_R, the center of the ball R. We give the edge a weight of $r_{\max} - r_R$, where $r_{\max}$ is defined to be $\max_{R' \in \mathcal{R}} r_{R'}$. This allows us to instead consider each R in $\mathcal{R}$ to be centered at x_R, all with the same radius $r_{\max}$, and the balls will still define the same set system $\mathcal{R}$ over the vertices in G. Finally, remove all vertices which are not contained in any ball of $\mathcal{R}$ and slightly perturb the edge weights (without changing any ball containment relations) so that every vertex v has a unique closest ball center in $\{x_R : R \in \mathcal{R}\}$.

To build the dual support, we construct a *Voronoi partition* of G' with respect to the set of nodes. A *Voronoi partition* $\mathcal{F}$ of a graph G' with respect to a set of nodes S is a partition of $V_{G'}$ into subsets such that for every node $R \in S$, there is a corresponding subset $F_R \in \mathcal{F}$ which contains all vertices u in G' for which $d(x_R, u) < \min_{x_R \in S \setminus \{x_R\}} d(x_{R'}, u)$. For clarity, we will refer to each set F_R in $\mathcal{F}$ as the *Voronoi cell* of R. In our case, we construct the Voronoi partition $\mathcal{F}_{\mathcal{R}}$ of G' with respect to $\{x_R : R \in \mathcal{R}\}$, which we identify as $\mathcal{R}$. Each cell F_R in the Voronoi partition is a connected subset of G' because for every vertex v in F_R reachable from x_R, the vertices on the shortest path between x_R and v must belong to F_R as well. Thus we can simply contract every F to a (contraction)-minor on only the nodes in $\mathcal{R}$. This final graph H, we claim, is a dual support for $(G, \mathcal{R})$.

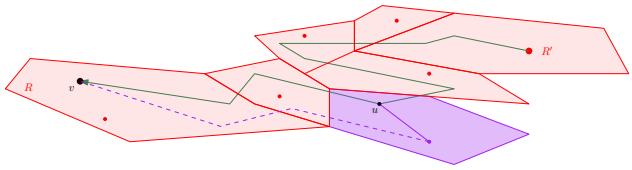


Figure 2. An illustration of the proof to Lemma 2.1 that π can only pass through Voronoi cells of $\mathcal{R}_v$. The purple region is the Voronoi cell F_Q.

Lemma 2.1. *For any vertex v in G', the set of balls $\mathcal{R}_v \subseteq \mathcal{R}$ that contain v must induce a connected subgraph in H.*

Proof: Consider any vertex v which is contained in at least two balls in $\mathcal{R}$. Vertex v must be in the Voronoi cell F_R of some ball R by construction of G'. We claim that for any other ball R' of $\mathcal{R}$ that also contains v, there is a path from $x_{R'}$ to v in G', denoted as π, that only passes through Voronoi cells corresponding to balls that contain v. Since we have removed all vertices which have no incoming path from any ball, every vertex in G' must be in some cell, so it suffices to show that every vertex along π is in a cell corresponding to a ball that contains v.

Assume that some vertex u along π is contained in some Voronoi cell F_Q corresponding to a ball Q that does not contain v. By definition of Voronoi cell, we have that $d(x_Q, u) < d(x_{R'}, u)$. By the definition of shortest paths, we know that $d(x_{R'}, v) = d(x_{R'}, u) + d(u, v)$. By triangle inequality, we then have the following:

$$d(x_Q, v) \leq d(x_Q, u) + d(u, v) < d(x_{R'}, u) + d(u, v) = d(x_{R'}, v).$$

However, $d(x_{R'}, v) \leq r_{\max}$ and $d(x_Q, v) > r_{\max}$ hold since v is in R' but not in Q. This is a contradiction. □

Proof (of Theorem 1.1): Construct H as described above. By Lemma 2.1, we know that H satisfies the definition of a dual support for system $(V_G, \mathcal{R})$. It remains to show that if G has no minor K with minimum degree two, then neither can H. This is done by arguing that G' has no such minors, and then the fact that H is a minor of G' concludes the proof.

When constructing G', we only perform two types of operations that would affect the existence of minors: deleting vertices, and adding in nodes $\{x_R\}$ of degree one. Let K be a minor model (as a collection of disjoint vertex subsets called *supernodes*) of G' that has minimum degree two. If K only contains vertices but not nodes in G', then K must appear in G as well (even though G' was obtained by removing some vertices from G). So for K to not be a minor of G, at least one supernode η of K must contain some degree-1 x_R that was added in G'.

For an edge (η, η') to exist in K, there must be a vertex/node $u \in \eta$ and a vertex/node $u' \in \eta'$ such that (u, u') is an edge of G'. For η containing x_R, it must be the case that x_R is not the only vertex/node in η, since η must have degree at least 2. Because η is connected, the neighbor of x_R is also in η and the (sole) edge incident on x_R must be contracted. This, however, means that there can be no edge (x_R, u') in G' that corresponds to an edge (η, η') in K, so there is a minor K' isomorphic to K where x_R was simply deleted and thus x_R is not in η. Since this holds for all x_R, there must be some minor isomorphic to K which is a minor of G, which is a contradiction. □

3 Existence of Intersection Support

We now shift focus to the notion of intersection support. We reuse the same construction of G', but in this case adding nodes for both $\mathcal{R}$ and $\mathcal{B}$. It is easy to show that if G (and our augmentation G') is undirected, then the above trick of contracting the Voronoi partition with respect to $\mathcal{R}$ works: Consider each $\mathcal{R}$ node to be the center of a distance ball with radius $2r_{\max}$. If a ball R from $\mathcal{R}$ and B from $\mathcal{B}$ share a vertex in G, then the distance ball centered at x_R in G' of radius $2r_{\max}$ must contain x_B. By Lemma 2.1, we know that all $R' \in \mathcal{R}$ containing x_B must induce a connected subgraph in H, satisfying the definition of intersection support.

3.1 Constructing an intersection support

To construct an intersection support for a directed graph G, we first construct G', using $\mathcal{R} \cup \mathcal{B}$ as our distance balls for the node set. This means that we create a node for each $D \in \mathcal{R} \cup \mathcal{B}$ and delete all vertices that are not contained in any such D. We will additionally label each vertex (and node) with whether it is reachable from a ball in $\mathcal{R}$ (***red***) or if it is *only* reachable from balls in $\mathcal{B}$ (***blue***). Considering only the red vertices at first, we apply the same procedure of constructing the Voronoi partition with respect to $\mathcal{R}$ and contracting each cell. We refer to this graph as H. H should now only contain blue vertices, blue nodes and red nodes. To define edge weights of H, we will say that edges between two red nodes have infinite weight, whereas all other edges inherit the minimum weight whenever two edges merge due to an edge contraction. From here, we flip the orientations of every edge in H, including those incident on nodes. We then construct a Voronoi partition with respect to the red nodes $\mathcal{R}$, and contract every cell, in this case yielding a graph H' only on the red nodes.

We first show that H' is a support graph, as was done with Lemma 2.1 for the dual system. In the interest of space, the proof of Lemma 3.1 is in Section A of the appendix.

Lemma 3.1. *For every blue ball $B \in \mathcal{B}$ in a system of distance balls $(\mathcal{R}, \mathcal{B})$ on an underlying graph G, the set of red balls $\mathcal{R}_B \subseteq \mathcal{R}$ that intersects B nontrivially must induce a connected subgraph in H'.*

Proof (of Theorem 1.2): Construct H' described above as our intersection support. By Lemma 3.1, we know that H' satisfies the definition of our intersection support, so it remains to show that H' contains no (contraction)-minor with minimum degree two that is not also a minor of G. The construction of H' begins with the construction of a graph G' almost equivalent to the one from Theorem 1.1, but for set system $(G, \mathcal{R} \cup \mathcal{B})$. From the proof of Theorem 1.1, we know that this G' contains no minor K of minimum degree two. Since we only perform contractions and vertex deletions after constructing G', we know that H' is a minor of G', and thus also contains no minor K. □

To conclude, any system of distance balls admit an intersection support which contains the same forbidden minors as the underlying graph. Both the dual and intersection supports can be computed in polynomial time. The primary bottleneck is the construction of the Voronoi partitions, which Erwig [8] showed to be computable in $O(m + n \log n)$ time for graphs with m edges and n vertices.

References

[1] Aniket Basu Roy. Covering Simple Orthogonal Polygons with Rectangles. In Alina Ene and Eshan Chattopadhyay, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2025)*, volume 353 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 2:1–2:23, Dagstuhl, Germany, 2025. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. URL: https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.APPROX/RANDOM.2025.2, doi:10.4230/LIPIcs.APPROX/RANDOM.2025.2.

[2] Sujoy Bhore, Balázs Keszegh, Andrey Kupavskii, Hung Le, Alexandre Louvet, Dömötör Pálvölgyi, and Csaba D. Tóth. *Spanners in Planar Domains via Steiner Spanners and non-Steiner Tree Covers*, pages 4292–4326. URL: https://epubs.siam.org/doi/abs/10.1137/1.9781611978322.145, arXiv:https://epubs.siam.org/doi/pdf/10.1137/1.9781611978322.145, doi:10.1137/1.9781611978322.145.

[3] Hervé Brönnimann and Michael T. Goodrich. Almost optimal set covers in finite VC-dimension. *Discrete & Computational Geometry*, 14(4):463–479, 1995. doi:10.1007/BF02570718.

[4] Kevin Buchin, Marc Van Kreveld, Henk Meijer, Bettina Speckmann, and Kevin Verbeek. On planar supports for hypergraphs. In *International Symposium on Graph Drawing*, pages 345–356. Springer, 2009.

[5] Timothy M. Chan and Sariel Har-Peled. Approximation algorithms for maximum independent set of pseudo-disks. *Discrete & Computational Geometry*, 48(2):373–392, Sep 2012. doi:10.1007/s00454-012-9417-5.

[6] Kenneth L. Clarkson and Kasturi Varadarajan. Improved approximation algorithms for geometric set cover. *Discrete & Computational Geometry*, 37(1):43–58, 2007. doi:10.1007/s00454-006-1273-8.

[7] Stephane Durocher and Robert Fraser. Duality for geometric set cover and geometric hitting set problems on pseudodisks. In *CCCG*, 2015.

[8] Martin Erwig. The graph voronoi diagram with applications. *Networks: An International Journal*, 36(3):156–163, 2000.

[9] Sathish Govindarajan, Rajiv Raman, Saurabh Ray, and Aniket Basu Roy. Packing and Covering with Non-Piercing Regions. In Piotr Sankowski and Christos Zaroliagis, editors, *24th Annual European Symposium on Algorithms (ESA 2016)*, volume 57 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 47:1–47:17, Dagstuhl, Germany, 2016. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. URL: https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.ESA.2016.47, doi:10.4230/LIPIcs.ESA.2016.47.

[10] Sariel Har-Peled and Kent Quanrud. Approximation algorithms for polynomial-expansion and low-density graphs. *SIAM Journal on Computing*, 46(6):1712–1744, 2017. arXiv:https://doi.org/10.1137/16M1079336, doi:10.1137/16M1079336.

[11] D. S. Johnson and H. O. Pollak. Hypergraph planarity and the complexity of drawing venn diagrams. *Journal of Graph Theory*, 11(3):309–325, 1987. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/jgt.3190110306, arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/jgt.3190110306, doi:10.1002/jgt.3190110306.

[12] Michael Kaufmann, Marc van Kreveld, and Bettina Speckmann. Subdivision drawings of hypergraphs. In Ioannis G. Tollis and Maurizio Patrignani, editors, *Graph Drawing*, pages 396–407, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[13] Sören Laue. Geometric set cover and hitting sets for polytopes in $\mathbb{R}^3$. In Susanne Albers and Pascal Weil, editors, *25th International Symposium on Theoretical Aspects of Computer Science (STACS 2008)*, volume 1 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 543–554, Dagstuhl, Germany, 2008. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. ISSN: 1868-8969. URL: https://drops.dagstuhl.de/opus/volltexte/2008/1328, doi:10.4230/LIPIcs.STACS.2008.1328.

[14] Nabil H. Mustafa and Saurabh Ray. Improved results on geometric hitting set problems. *Discrete & Computational Geometry*, 44(4):883–895, Dec 2010. doi:10.1007/s00454-010-9285-9.

[15] Evangelia Pyrga and Saurabh Ray. New existence proofs epsilon-nets. SCG '08, page 199–207, New York, NY, USA, 2008. Association for Computing Machinery. doi:10.1145/1377676.1377708.

[16] Rajiv Raman and Saurabh Ray. Constructing planar support for non-piercing regions. *Discrete & Computational Geometry*, 64(3):1098–1122, 2020. doi:10.1007/s00454-020-00216-w.

[17] Rajiv Raman and Karamjeet Singh. On hypergraph supports. In *European Conference on Combinatorics, Graph Theory and Applications*, pages 774–783, 2023.

[18] W. M. van Cleemput. Hypergraph models for the circuit layout problem. *Applied Mathematical Modelling*, 1(3):160–161, 1976.

[19] A. A. Voloshina and V. Z. Feinberg. Planarity of hypergraphs. In *Doklady Akademii Nauk Belarusi*, volume 28, pages 309–311, Minsk, Belarus, 1984. Academii Nauk Belarusi. In Russian.

[20] Christian Wulff-Nilsen. Separator theorems for minor-free and shallow minor-free graphs with applications. In *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, pages 37–46, 2011. doi:10.1109/FOCS.2011.15.

A Proof of Lemma 3.1

Proof (of Lemma 3.1): Assume that $|\mathcal{R}_B| \geq 2$, as otherwise this holds trivially. Consider the closest red vertex u to B (smallest distance $d(x_B, u)$). By the construction of H, u was contracted into some red node $R \in \mathcal{R}$. In fact, $R \in \mathcal{R}_B$, $|\mathcal{R}_B| = 0$ by the definition of a red vertex. Since we always choose the minimum weight edge when merging, we know that x_B must be in the Voronoi cell of R in H. It suffices to show that the remaining $R' \in \mathcal{R}_B \setminus \{R\}$ must have a path to R within $H'[\mathcal{R}_B]$.

Consider a vertex v in $R' \cap B$, and the shortest paths $\pi(x_{R'}, v)$ and $\pi(x_B, v)$. We know that every vertex v' in $\pi(x_{R'}, v)$ must be in the G' Voronoi cell of R'. We claim that every vertex v' in $\pi(x_B, v)$ is either a red vertex in the G' Voronoi cell of a ball in $\mathcal{R}_B$ or a blue vertex in the H Voronoi cell of a ball in $\mathcal{R}_B$. This lets use the union of $\pi(x_{R'}, v)$ and $\pi(x_B, v)$ to trace a path in H' from R' to R.

If v' is in $\pi(x'_B, v)$, then, by definition of shortest path, $d(x_B, v') \leq d(x_B, v)$ and thus v' is in B. With this in mind, we can split into cases based on whether v' is a red vertex or a blue vertex.

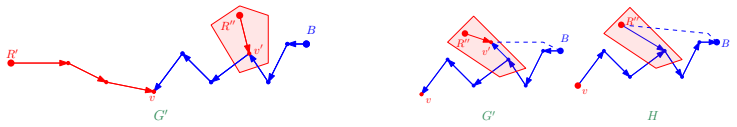


Figure 3. Left: a red vertex v' of $\pi(c_B, v)$ must lie in the Voronoi cell of a red ball R' intersecting with B. Right: A blue vertex v' of $\pi(c_B, v)$ must lie in the H Voronoi cell of R' intersecting with B.

- If v' is red then v' must be in the Voronoi cell (in G') of some red ball R'' that contains v'. Since v' is also in B, this means that $R'' \cap B \neq 0$, which implies that $R'' \in \mathcal{R}_B$.
- If v' is blue, there must be some red vertex u' which is of the smallest distance, $d(v', u')$, among all red vertices. Since v is a red vertex, we know that $d(v', u') \leq d(v', v)$, hence:

$$d(x_B, u') \leq d(x_B, v') + d(v', u') \leq d(x_B, v') + d(v', v) = d(x_B, v) \leq r_{\max}$$

  This implies that u' must be in B.

  Since u' is red, u' must be in some Voronoi cell of a red ball R'' (in G'). After the orientation flip and the construction of H, u' is contracted into $x_{R''}$, and thus $x_{R''}$ must be the closest red node to v'. This means v' is in the R'' Voronoi cell in H. Since $u' \in R'' \cap B$, $R'' \in \mathcal{R}_B$, thus every vertex in $\pi(x_B, v)$ is contracted into a red node which is in $\mathcal{R}_B$.

This means that we can simply perform a walk from R to R' in H' by going to the red ball corresponding to each vertex along $\pi(x_{R'}, v)$ and then backwards along $\pi(x_B, v)$. Thus, $\mathcal{R}_B$ must form a connected subgraph in H'. □

Johnson-Lindenstrauss Lemma Beyond Euclidean Geometry

Chengyuan Deng* Jie Gao* Kevin Lu* Feng Luo* Cheng Xin*

Abstract

The Johnson-Lindenstrauss (JL) lemma is a cornerstone of dimensionality reduction in Euclidean space, but its applicability to non-Euclidean data has remained limited. This paper extends the JL lemma beyond Euclidean geometry to handle general dissimilarity matrices that are prevalent in real-world applications. We present two complementary approaches: First, we show the JL transform can be applied to vectors in pseudo-Euclidean space with signature (p, q), providing theoretical guarantees that depend on the ratio of the (p, q) norm and Euclidean norm of two vectors, measuring the deviation from Euclidean geometry. Second, we prove that any symmetric hollow dissimilarity matrix can be represented as a matrix of generalized power distances, with an additional parameter representing the uncertainty level within the data. In this representation, applying the JL transform yields multiplicative approximation with a controlled additive error proportional to the deviation from Euclidean geometry. Our theoretical results provide fine-grained performance analysis based on the degree to which the input data deviates from Euclidean geometry, making practical and meaningful dimension reduction accessible to a wider class of data. We validate our approaches on synthetic and real-world datasets, demonstrating the effectiveness of extending JL transform to non-Euclidean settings. Full version in Neurips 2025.[1]

1 Introduction

The Johnson-Lindenstrauss (JL) lemma [1] stands as a cornerstone result in dimensionality reduction. It states that random linear projection can reduce the dimensionality of datasets in Euclidean space, while approximately preserving pairwise distances. Formally,

Proposition 1.1 (Johnson-Lindenstrauss Lemma)**.** *For any set of n points $x_1, x_2, \dots x_n$ in $\mathbb{R}^d$ and $\varepsilon \in (0, 1)$, there exists a map $f : \mathbb{R}^d \to \mathbb{R}^m$, where $m = O(\log n/\varepsilon^2)$ such that for any $i, j \in [n]$,*

$$(1 - \varepsilon)\|x_i - x_j\|_2 \leq \|f(x_i) - f(x_j)\|_2 \leq (1 + \varepsilon)\|x_i - x_j\|_2 \tag{1}$$

In modern algorithm design, the JL lemma is widely used as a key component or pre-processing step for high-dimensional data analysis. In addition to effectiveness, one crucial reason for its significant impact is that the JL lemma can be achieved by random linear maps, which are data-independent and easy to implement. Johnson-Lindenstrauss Lemma has found numerous applications in machine learning, from the most immediate applications in approximating all pairs distances (when the input data is high-dimensional), to approximate nearest neighbor search [2], approximate linear regression [3], clustering [4, 5, 6, 7], functional analysis [8] and compressed sensing [9].

Note that two conditions must be met to apply the JL lemma: (1) the data points lie in high-dimensional Euclidean space, and (2) the coordinates of all points are available. However, many real-world applications do not satisfy these two conditions. First, the dissimilarity measures used in modern data analysis are often non-Euclidean and sometimes even non-metric. Common examples include Minkowski distance, cosine similarity, Hamming distance, Jaccard index, Mahalanobis distance, Chebyshev distance, and Kullback–Leibler (KL) divergence, etc. Psychological studies have long observed that human similarity judgments do not conform to metric properties [10]. Second, high-dimensional coordinates may be unavailable or costly to obtain, whereas pairwise dissimilarities are easier to access. In recommendation systems, computing user or item embeddings can be expensive, while estimating pairwise dissimilarities (e.g. from co-click or co-purchase data) is relatively efficient. To date, our understanding of the JL lemma deviating from these two classical conditions mainly revolves around ℓ_p metrics and lower bound results.

In this paper, we study how to apply the Johnson-Lindenstrauss transform (a.k.a. random linear projection) in non-Euclidean, non-metric settings. We consider the input as a dissimilarity matrix D of size $n \times n$ where D_{ij} is the dissimilarity between item i and j. We make only two assumptions for the dissimilarity measure: symmetric ($D_{ij} = D_{ji}$) and reflexive ($D_{ii} = 0$). Note that we do not require the triangle inequality to be satisfied, therefore the

*Rutgers Uniersity, {cd751, jg1555, kll160, fluo, cx122}@rutgers.edu

[1] https://neurips.cc/virtual/2025/poster/118378

dissimilarity can be non-metric. In short, the input is a symmetric hollow dissimilarity matrix, and the expected output is a low-dimensional embedding of the original dataset that approximately preserves pairwise distances.

The major challenge we encounter in this setting is two-fold. The first is to obtain *good* coordinates from a generic input dissimilarity matrix to represent the data, before we can even apply the JL transform. Second, we need a geometric characterization of the non-Euclidean non-metric setting, to show how different our setting stands from Euclidean geometry, which the JL Lemma is based on.

Before we explain our results, we first mention existing lower bounds for dimension reduction in non-Euclidean settings. JL Lemma considers Euclidean spaces only. Naturally, one asks whether such a result is possible for other spaces. There are several non-trivial dimension reduction results for ℓ_1 norm [11] and ℓ_p norm [12]. However, the target embedding dimension for ℓ_1, ℓ_p and nuclear norm is necessarily polynomial in n for constant distortion [13, 14, 15]. These lower bounds remind us that a worst-case guarantee on low distortion and logarithmic target dimension is not possible. Rather, our results provide error analysis which depends on parameters characterizing how the input data deviates from Euclidean geometry. In other words, we have fine-grained performance analysis.

Our Contributions. We present two approaches to recover coordinates that fit into the non-Euclidean non-metric setting. For both approaches, we generalize the Euclidean norm ℓ_2 to a new form to capture the input data geometry, together with certain geometric parameters indicating how far it deviates from Euclidean geometry. We give error analysis on the dissimilarity distortion, which may involve an additive term whose magnitude is proportional to the geometric parameters.

At the heart of our first approach is the observation that any symmetric hollow matrix can be written as the distance matrix of vectors in *pseudo-Euclidean space* [16, 17, 18]. Here the distance between two vectors $x = (x_1, x_2, ...x_n)$ and $y = (y_1, y_2, ...y_n)$ is captured by a bilinear form of signature (p, q), which is defined as $\langle x, y\rangle_{p,q} = \sum_{i=1}^{p} x_i y_i - \sum_{i=p+1}^{p+q} x_i y_i$. The squared (p, q)-distance between x and y is $\|x - y\|_{p,q}^2 = \langle x - y, x - y\rangle$. When $p = n, q = 0$, it is the squared Euclidean norm. We will give a more detailed introduction later. At this point, it suffices to interpret the parameter p resembling and q negating the Euclidean space. Our first result shows the generalization of the JL lemma to the pseudo-Euclidean geometry.

Theorem 1.2. *For any set of n points $x_1, x_2, \dots x_n$ in $\mathbb{R}^{p,q}$ and $\varepsilon \in (0, 1)$, there exists a map $f : \mathbb{R}^{p,q} \to \mathbb{R}^{p',q'}$, where $p', q' = O(\log n/\varepsilon^2)$ such that for any $i, j \in [n]$,*

$$(1 - \varepsilon \cdot C_{ij})\|x_i - x_j\|_{p,q}^2 \le \|f(x_i) - f(x_j)\|_{p',q'}^2 \le (1 + \varepsilon \cdot C_{ij})\|x_i - x_j\|_{p,q}^2, \tag{2}$$

where $C_{ij} = \left|\frac{\|x_i - x_j\|_E^2}{\|x_i - x_j\|_{p,q}^2}\right|$.

Our second result takes a different route. We prove that a symmetric hollow matrix is also a matrix of *generalized power distances*. Given two points x and y with respective radius r_x and r_y, the generalized power distance is defined as $\|x - y\|_E^2 - (r_x + r_y)^2$, where $\|x - y\|_E^2$ denotes the Euclidean norm. This is the squared length of the tangent line segments of two balls centered at x, y with radii r_x, r_y. Again, at this point, it suffices to interpret r_x, r_y as a measure of deviation from Euclidean space, with both a geometric meaning and a statistical interpretation of distance between points with uncertainties.

We show that any input symmetric hollow matrix D of size n can be written as the generalized power distance matrix of n points $\{p_i\}$ with the same radius $r = \sqrt{|e_n|}/2$, where e_n is the smallest eigenvalue of the Gram matrix of D. This linear algebra result may be of independent interest. Only when D is a Euclidean distance matrix, all eigenvalues are non-negative and $r = 0$. Our second result follows from applying the JL transform on the ball centers (i.e. $\{p_i\}$). We obtain a $(1 \pm \varepsilon)$ multiplicative approximation of the generalized power distance with an additive error of $4\varepsilon r^2$.

Theorem 1.3. *Given any $n \times n$ symmetric hollow dissimilarity matrix, D, with power distance representation by $\{(p_i, r)\}$ where $r = \sqrt{|e_n|}/2$ and e_n is the smallest eigenvalue of $\mathrm{Gram}(D) = -\frac{1}{2}CDC$ with $C = 1 - J/n$ and J is an all 1 matrix, there exists a map $f : \mathbb{R}^n \to \mathbb{R}^m$, where $m = O(\log n/\varepsilon^2)$ such that for any $i \ne j \in [n]$,*

$$\begin{aligned}(1 - \varepsilon)\,\mathrm{Pow}((p_i, r), (p_j, r)) - \varepsilon 4r^2 &\le \mathrm{Pow}((f(p_i), r), (f(p_j), r)) \\ &\le (1 + \varepsilon)\,\mathrm{Pow}((p_i, r), (p_j, r)) + \varepsilon 4r^2.\end{aligned}$$

To complement Theorem 1.3, we are able to extend the techniques in [19] and give a lower bound of $\Omega(\log n/\varepsilon^2)$ on the target dimension to achieve the multiplicative and additive factors as mentioned. Note that $m = \Omega(\log n/\varepsilon^2)$ matches the JL lemma lower bound.

Experiments. We implement both methods of JL transform with respect to the above results and evaluate their performances on 10 datasets. We observe that the results corroborate with our theoretical results, and outperform classical JL transform consistently on non-Euclidean datasts. Our codes are on the Anonymous Github[2].

2 Algorithms for Non-Euclidean Johnson-Lindenstrauss Transforms

Theorem 1.2 and Theorem 1.3 provide two ways of applying the random linear projection in non-Euclidean settings. We present the algorithms for both cases below. Recall the input dissimilarity matrix D is symmetric and hollow. As for **computational complexity**, both methods start with D and we necessarily need to recover the 'coordinates' first. This can be done by running the singular value decomposition (SVD) on the Gram matrix in $O(n^3)$ time. If it happens that the coordinates are to be learned from a representation learning module, we can skip this part and directly apply dimension reduction. When applying standard JL transform we use random projection of Gaussian vectors. One can use any JL transform for example [20, 21, 22, 23, 24] for this purpose.

2.1 Pseudo-Euclidean JL Transform

We show how to obtain vectors X in Pseudo-Euclidean space such that the intervals square of x_i, x_j is precisely D_{ij}. We first perform centralization to obtain the Gram matrix $B = \text{Gram}(D) = -CDC/2$, where $C = I - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^T$ is the centering matrix and $\mathbf{1}_n$ is a vector of ones. Since $-CDC/2$ is a symmetric matrix, its eigenvalues are real, denoted as $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$. We take p as the number of non-negative eigenvalues and q to be the number of negative eigenvalues. $p + q = n$. Further, suppose $U \in \mathbb{R}^{n\times n}$ is the orthogonal vectors, we have

$$B = -CDC/2 = U\,\text{Diag}(\lambda_1, \cdots, \lambda_n)U^T$$

Now we can recover the coordinates of the n elements as n-dimensional vectors. Specifically X is a matrix of dimension $n \times n$ with the columns representing the coordinate vectors of the n points.

$$X = (x_1, \cdots, x_n) = \text{Diag}(\sqrt{|\lambda_1|}, \cdots, \sqrt{|\lambda_n|}) \cdot U^T$$

This way, we have $B = -CDC/2 = X^T\Lambda X$, with $\Lambda = \text{Diag}(1, \cdots, 1, -1, \cdots, -1)$ as a $n \times n$ diagonal matrix with the first p diagonal elements as 1 and the remaining diagonal elements as -1. Equivalently, we have

$$D_{ij} = (x_i - x_j)^T\Lambda(x_i - x_j) = \langle x_i - x_j, x_i - x_j\rangle.$$

Algorithm. Find the vectors X of n points in pseudo-Euclidean space following above steps. For each point x in the embedding, split it into $x^{(p)}$ and $x^{(q)}$. Then, we use standard JL transform to project into $\mathbb{R}^{p'}$ and $\mathbb{R}^{q'}$ respectively where p' and q' are the specified target dimension. Return the projected points $(x^{(p')}, x^{(q')})$ along with their (p', q') signature.

2.2 Power distance JL Transform

Given two balls centered at $p, q \in \mathbb{R}^d$, with radii r_p, r_q, we define the *generalized power distance* as:

$$\text{Pow}((p, r_p), (q, r_q)) = \|p - q\|_E^2 - (r_p + r_q)^2. \tag{3}$$

Equation 3 measures the distance of the internal tangents between two disjoint circles (the tangent line that keeps two circles on different sides. See the middle picture in Figure 1). Note that there is another second generalized power distance between two circles centered at p_i of radius r_i given by $||p_1 - p_2||_E^2 - (r_1 - r_2)^2$, which measures the distance of the external tangents between the circles (the tangent line that keeps two circles on the same side).

The key observation that allows for the JL transform is any symmetric hollow dissimilarity matrix can be written as the matrix of power distances between n weighted points, formally as below.

Lemma 2.1. *Given any $n \times n$ symmetric hollow dissimilarity matrix, D, we can rewrite $D = E + 4r^2(I - J)$ where $r \in \mathbb{R}^+$, E is a Euclidean distance matrix, I is an $n \times n$ identity matrix, $J = \mathbf{1}_n\mathbf{1}_n^T$. More specifically E is a Euclidean distance matrix if and only if $2r^2 \geq |e_n|$ where e_n is the least eigenvalue of* $\text{Gram}(D)$.

Algorithm. Find e_n, the smallest eigenvalue of the Gram matrix of D. Set $r = \sqrt{|e_n|}/2$. Add $4r^2(J - I)$ to D to find the new Euclidean distance matrix, E. Recover the Euclidean coordinates X' such that $E_{ij} = \|x_i' - x_j'\|^2$. Perform standard JL transform on X' to dimension m. Return the resulting points along with their radii r.

[2] `https://anonymous.4open.science/r/Non-Euclidean-Johnson-Lindenstrauss-1673`

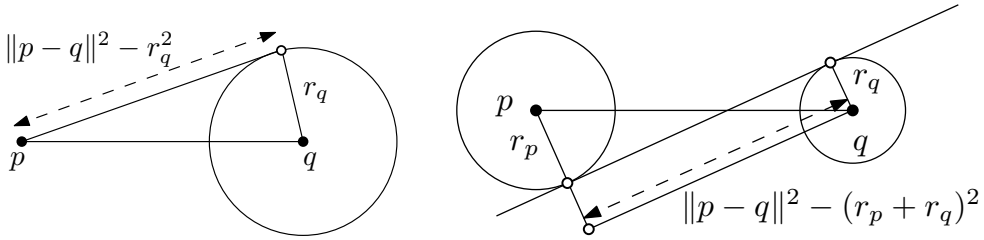


Figure 1: Left: power distance from a point p to a ball at q of radius r_q; Right: power distance between two balls at points p, q with radius r_p and r_q respectively

3 Experiments

In this section, we present experimental results of the proposed JL transforms in non-Euclidean settings with only a dissimilarity matrix as input. First, we validate our theoretical results by showing the approximation error on datasets that are highly non-Euclidean. Next, we evaluate the algorithms on real-world datasets, with the classical JL transform as a baseline.

Datasets. We use two **synthetic datasets** that are made non-Euclidean: Random-simplex and Euclidean-ball. At a high level, for the Random-simplex, given a dataset of size n, each point is constructed such that its first $n-1$ coordinates form a simplex, while the final coordinate dominates the pairwise distances. This design induces a large negative eigenvalue in the Gram matrix. The Euclidean-ball dataset, inspired by Delft's balls [25], consists of n balls with varying radii. The distance between two balls is defined as the minimal distance between any two points on their surfaces, resulting in dissimilarities that violate the triangle inequality. For **real-world data**, We consider three categories: genomics, image and graph data. The genomics data includes three cancer-related datasets from the Curated Microarray Database (CuMiDa) [26]. Following the practice in prior work [18], we obtain dissimilarities with entropic affinity. We also test two celebrated image datasets: MNIST and CIFAR-10, each with 1000 images randomly sampled. We use the measures mentioned in [27] to calculate the dissimilarities. The graph datasets are selected from the SNAP project. Details are shown in Table 1, with more to follow in Appendix.

Dataset	Simplex	Ball	Brain	Breast	Renal	MNIST	CIFAR10	Email	Facebook	Mooc
Size	1000	1000	130	151	143	1000	1000	986	4039	7047
$\#\ \{\lambda < 0\}$	900	887	53	59	57	454	399	465	1566	268
Metric	✗	✗	✗	✗	✗	✓	✓	✓	✓	✓

Table 1: Non-Euclidean/Non-metric Datasets used in experiments

Performance on Relative Error. We compare two proposed JL transforms with the classical JL transform on all datasets and report the relative error, which is defined as the maximum among $\frac{|D_{ij}-\hat{D}_{ij}|}{D_{ij}}$ for all (i,j) pairs. $\hat{D}_{ij}$ is the dissimilarity matrix obtained from any JL transform. It is a suitable metric because all three algorithms have different theoretical guarantees, but the end goal is always preserving pairwise dissimilarities. In Table 2 we show both the worst-case (defined above) and average relative error. A smaller value indicates better performance.

We observe JL-PE performs the best for genomics data and JL-power performs the best for the rest, on both metrics. The significant improvement implies when the JL transform matches with geometry, we can expect really good results. The image datasets report *inf* for JL because different images are projected to the same point. In a few cases Jl-PE has slightly worse relative error than JL, but note that this might be the outcome of really large factor C, which the performance of JL-PE is based on.

Method	Simplex	Ball	Brain	Breast	Renal	MNIST	CIFAR10	Email	Facebook	MOOC
JL Max	6.47e5	5.97e5	1.02e12	7.09e10	3.42e12	inf	inf	3.85e4	1.18e6	2.94e5
JL-PE Max	1.11e6	5.31e5	**8,21e4**	**262.20**	**2.37e4**	8.47e5	2.24e6	3.24e6	2.51e7	1.35e7
JL-Power Max	**109.18**	**12.102**	3.11e7	1.92e6	1.17e8	**85.76**	**55.74**	**781.45**	**82.74**	**24.22**
JL Ave	442.65	43.78	1.11e9	9.86e7	1.50e9	inf	inf	15.64	12.62	39.83
JL-PE Ave	47.59	23.71	**8.16**	**1.15**	**6.60**	15.93	18.24	13.79	52.52	63.44
JL-Power Ave	**13.52**	**1.002**	3.73e4	3.12e3	5.57e4	**1.47**	**1.61**	**1.60**	**1.32**	**2.04**

Table 2: Max and Average Relative Error of All Datasets on Three JL Transforms.

References

[1] William B Johnson and Joram Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. *Contemporary mathematics*, 26(189-206):1, 1984.

[2] Piotr Indyk and Assaf Naor. Nearest-neighbor-preserving embeddings. *ACM Trans. Algorithms*, 3(3):31–es, August 2007.

[3] Mert Pilanci and Martin J Wainwright. Randomized sketches of convex programs with sharp guarantees. *IEEE Trans. Inf. Theory*, 61(9):5096–5115, September 2015.

[4] Konstantin Makarychev, Yury Makarychev, and Ilya P. Razenshteyn. Performance of Johnson-Lindenstrauss transform for k-means and k-medians clustering. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC*, pages 1027–1038, 2019.

[5] Zachary Izzo, Sandeep Silwal, and Samson Zhou. Dimensionality reduction for Wasserstein barycenter. *Advances in neural information processing systems*, 34:15582–15594, 2021.

[6] Shyam Narayanan, Sandeep Silwal, Piotr Indyk, and Or Zamir. Randomized dimensionality reduction for facility location and single-linkage clustering. In *Proceedings of the 38th International Conference on Machine Learning, ICML*, volume 139 of *Proceedings of Machine Learning Research*, pages 7948–7957. PMLR, 2021.

[7] Luca Becchetti, Marc Bury, Vincent Cohen-Addad, Fabrizio Grandoni, and Chris Schwiegelshohn. Oblivious dimension reduction for k-means: beyond subspaces and the Johnson-Lindenstrauss lemma. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2019, page 1039–1050, New York, NY, USA, 2019. Association for Computing Machinery.

[8] William B Johnson and Assaf Naor. The Johnson-Lindenstrauss lemma almost characterizes Hilbert space, but not quite. *Discrete & Computational Geometry*, 43(3):542–553, 2010.

[9] Richard Baraniuk, Mark Davenport, Ronald DeVore, and Michael Wakin. A simple proof of the restricted isometry property for random matrices. *Constructive approximation*, 28:253–263, 2008.

[10] Amos Tversky and Itamar Gati. Similarity, separability, and the triangle inequality. *Psychol. Rev.*, 89(2):123–154, March 1982.

[11] Jiří Matoušek. On variants of the Johnson–Lindenstrauss lemma. *Random Struct. Algorithms*, 33(2):142–156, September 2008.

[12] James R Lee, Manor Mendel, and Assaf Naor. Metric structures in ℓ_1: dimension, snowflakes, and average distortion. *Eur. J. Comb.*, 26(8):1180–1190, November 2005.

[13] Gideon Schechtman. Dimension reduction in l_p, $0 < p < 2$. *arXiv [math.MG]*, October 2011.

[14] Assaf Naor, Gilles Pisier, and Gideon Schechtman. Impossibility of dimension reduction in the nuclear norm. *Discrete Comput. Geom.*, 63(2):319–345, March 2020.

[15] Bo Brinkman and Moses Charikar. On the impossibility of dimension reduction in ℓ_1. *J. ACM*, 52(5):766–788, September 2005.

[16] Lev Goldfarb. A unified approach to pattern recognition. *Pattern Recognition*, 17(5):575–582, 1984.

[17] Elzbieta Pekalska and Robert P. W. Duin. *The Dissimilarity Representation for Pattern Recognition: Foundations And Applications (Machine Perception and Artificial Intelligence)*. World Scientific Publishing Co., Inc., USA, 2005.

[18] Chengyuan Deng, Jie Gao, Kevin Lu, Feng Luo, Hongbin Sun, and Cheng Xin. Neuc-MDS: Non-Euclidean multidimensional scaling through bilinear forms. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems (NeurIPS 2024)*, volume 37, pages 121539–121569, December 2024.

[19] Kasper Green Larsen and Jelani Nelson. Optimality of the Johnson-Lindenstrauss lemma. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 633–638, 2017.

[20] Dimitris Achlioptas. Database-friendly random projections: Johnson-Lindenstrauss with binary coins. *J. Comput. Syst. Sci.*, 66(4):671–687, June 2003.

[21] Nir Ailon and Bernard Chazelle. The fast Johnson–Lindenstrauss transform and approximate nearest neighbors. *SIAM J. Comput.*, 39(1):302–322, January 2009.

[22] Anirban Dasgupta, Ravi Kumar, and Tamás Sarlos. A sparse Johnson-Lindenstrauss transform. STOC '10, page 341–350, New York, NY, USA, 2010. Association for Computing Machinery.

[23] Rong Yin, Yong Liu, Weiping Wang, and Dan Meng. Extremely sparse Johnson-Lindenstrauss transform: From theory to algorithm. In *2020 IEEE International Conference on Data Mining (ICDM)*, pages 1376–1381. IEEE, 2020.

[24] Daniel M Kane and Jelani Nelson. Sparser Johnson-Lindenstrauss transforms. *J. ACM*, 61(1):1–23, January 2014.

[25] Robert P W Duin and Elżbieta Pękalska. Non-Euclidean dissimilarities: Causes and informativeness. In *Structural, Syntactic, and Statistical Pattern Recognition*, pages 324–333. Springer Berlin Heidelberg, 2010.

[26] Bruno Cesar Feltes, Eduardo Bassani Chandelier, Bruno Iochins Grisci, and Márcio Dorn. CuMiDa: an extensively curated microarray database for benchmarking and testing of machine learning approaches in cancer research. *Journal of Computational Biology*, 26(4):376–386, 2019.

[27] Rishi Sonthalia, Greg Van Buskirk, Benjamin Raichel, and Anna Gilbert. How can classical multidimensional scaling go wrong? In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 12304–12315, 2021.

Computing Dominating Sets in Disk Graphs with Centers in Convex Position*

Anastasiia Tkachenko† Haitao Wang‡

Abstract

Given a set P of n points in the plane and a collection of disks centered at these points, the disk graph $G(P)$ has vertex set P, with an edge between two vertices if their corresponding disks intersect. We study the dominating set problem in $G(P)$ under the special case where the points of P are in convex position. The problem is NP-hard in general disk graphs. Under the convex position assumption, however, we present the first polynomial-time algorithm for the problem. Specifically, we design an $O(k^2 n \log^2 n)$-time algorithm, where k denotes the size of a minimum dominating set. For the weighted version, in which each disk has an associated weight and the goal is to compute a dominating set of minimum total weight, we obtain an $O(n^5 \log^2 n)$-time algorithm.

1 Introduction

Let $P = \{p_1, \ldots, p_n\}$ be a set of n points in the plane, where each point $p_i \in P$ is assigned a radius $r_{p_i} \geq 0$. Let D_{p_i} denote the disk centered at p_i with radius r_{p_i}. The *disk graph* $G(P)$ has P as its vertex set, with an edge between p_i and p_j if the two disks D_{p_i} and D_{p_j} intersect, i.e., $|p_i p_j| \leq r_{p_i} + r_{p_j}$, where $|p_i p_j|$ is the Euclidean distance of p_i and p_j. We are interested in the dominating set problem in disk graphs. A *dominating set* of $G(P)$ is a subset $S \subseteq P$ such that every vertex in $G(P)$ is either in S or adjacent to a vertex in S. The *dominating set problem* is to find a dominating set of minimum cardinality. In the *weighted case*, each point of P has a weight, and the goal is to find a dominating set minimizing the total weight. The problem is NP-hard even in unit-disk graphs [8]. Approximation algorithms have been proposed, e.g., [3, 9, 12, 16].

Convex position setting and previous work. In light of the above hardness results, exploring structured settings that may allow efficient algorithms is of particular interest. In this paper, we consider the dominating set problem in disk graphs in a *convex position* setting where every point of P appears as a vertex of the convex hull of P. This setting models deployments along perimeters (e.g., fences, shorelines, or rings of sensors). We emphasize that while the points of P are in convex position, the disks themselves may not be. In convex position, Tkachenko and Wang [20] gave $O(n^3 \log^2 n)$ and $O(kn \log n)$ algorithms for weighted and unweighted dominating set in *unit-disk* graphs (with k the optimum size). Other results in the convex position setting for problems that are NP-hard in general include: maximum independent set problem solvable in $O(n^{7/2})$ time [20]); the k-center problem solved by Choi, Lee, and Ahn [7] in $O(n^3 \log n)$ time, and its discrete variant in $O(n^2 \log^2 n)$ [20]; Singireddy, Basappa, and Mitchell gave an $O(n^4 k^2)$ time solution for dispersion problem, which later was improved to $O(n^{7/2} \log n)$ [18, 20]. Even classically easy problems have tailored treatments here, e.g., the linear-time Voronoi diagram construction of Aggarwal, Guibas, Saxe, and Shor [1]. See also [4–6, 15, 17, 19].

Our result. We study the dominating set problem in disk graphs under the convex position setting. While polynomial-time algorithms are known for the unit-disk case [20], allowing disks of varying radii introduces significant challenges (e.g., maximum clique for unit-disks and disk graphs [2, 8, 10, 11, 13, 14]). Nevertheless, we attempt to extend the techniques of [20] to the general disk graphs. This proves far from trivial, as many properties of the unit-disk case no longer hold. Even so, we uncover new structural observations and develop novel algorithmic techniques. Consequently, we establish that the dominating set problem in disk graphs under the convex position setting can be solved in polynomial time. Specifically, we present an $O(n^5 \log^2 n)$-time algorithm for the weighted case. Furthermore, given a size bound k, we can compute a minimum-weight dominating set of size at most k (if one exists) in $O(k^2 n^3 \log^2 n)$ time. For the unweighted case, we design a more efficient algorithm, achieving a running time of $O(k^2 n \log^2 n)$, where k denotes the size of a minimum dominating set. In particular, when $k = O(1)$, our algorithm runs in $O(n \log^2 n)$ time.

*A full version of this paper has been submitted for review.

†Kahlert School of Computing, University of Utah, Salt Lake City, UT 84112, USA. anastasiia.tkachenko@utah.edu

‡Kahlert School of Computing, University of Utah, Salt Lake City, UT 84112, USA. haitao.wang@utah.edu

2 Our algorithms

In this section, we present structural properties of dominating sets in $G(P)$ and sketch our algorithms. In Section 2.1, we present the algorithm for computing a minimum-weight dominating set. Our approach is a dynamic programming algorithm that leverages a key structural property established in Lemma 1. This property enables us to decompose the original problem into smaller subproblems amenable to dynamic programming. The minimum dominating set problem, Section 2.2, is solved similarly with improved efficiency using a greedy strategy.

Notation. Let $\mathcal{H}(P)$ denote the convex hull of P. We treat P as a cyclic sequence of the points ordered counterclockwise along $\mathcal{H}(P)$, that is $P = \langle p_1, p_2, \ldots, p_n \rangle$. We use a *sublist* to refer to a contiguous subsequence of P. Multiple sublists are said to be *consecutive* if their concatenation is also a sublist. For any two points p_i and p_j in P, we define $P[i, j]$ as the sublist of P from p_i counterclockwise to p_j, inclusive. Note that if $i = j$, then $P[i, j] = \langle p_i \rangle$. We also denote by $P(i, j]$ the sublist $P[i, j]$ excluding p_i, and similarly for other variations, e.g., $P[i, j)$ and $P(i, j)$. For any two points $p_i, p_j \in P$, we define $|D_{p_i} D_{p_j}| = |p_i p_j| - (r_{p_i} + r_{p_j})$.

Structural properties. For a sublist α of P, we say that a point $p_i \in P$ *dominates* α if the disk D_{p_i} intersects D_p for all points $p \in \alpha$. For two points $p_i, p_j \in P$, if D_{p_i} intersects D_{p_j}, then we also say that p_i *dominates* p_j (and similarly, p_j dominates p_i). Suppose $S \subseteq P$ is a dominating set of $G(P)$. Let $\mathcal{A}$ be a partition of P into (nonempty) disjoint sublists such that for every sublist $\alpha \in \mathcal{A}$, there exists a point in S that dominates α. An *assignment* is a mapping $\phi : \mathcal{A} \to S$ that assigns every sublist $\alpha \in \mathcal{A}$ to exactly one point $p_i \in S$ such that p_i dominates α. For each $p_i \in S$, we define the *group* of p_i, denoted by $\mathcal{A}_{p_i}$, as the collection of all sublists $\alpha \in \mathcal{A}$ that are assigned to p_i under ϕ. Depending on the context, $\mathcal{A}_{p_i}$ may also refer to the set of points of P in all its sublists. An assignment $\phi : \mathcal{A} \to S$ is *line-separable* if for every two points $p_i, p_j \in S$, there exists a line ℓ such that the points of $\mathcal{A}_{p_i}$ lie entirely on one side of ℓ, while the points of $\mathcal{A}_{p_j}$ lie on the other side. The following lemma proves a *line-separable property*.

Lemma 1. *Suppose S is an optimal dominating set of $G(P)$. Then there exists a partition $\mathcal{A}$ of P and a line-separable assignment $\phi : \mathcal{A} \to S$ such that (1) for every point $p_i \in S$, $p_i \in \mathcal{A}_{p_i}$, i.e., the group $\mathcal{A}_{p_i}$ contains p_i itself, and (2) any two adjacent sublists of $\mathcal{A}$ are assigned to different points of S.*

For each center $p_i \in S$, we call the sublist of $\mathcal{A}_{p_i}$ containing p_i the *main sublist* of p_i. We further have the following lemma, which will be instrumental in our algorithm design.

Lemma 2. *Let S be an optimal dominating set of $G(P)$, and let $\phi : \mathcal{A} \to S$ be the assignment given by Lemma 1. Then, S has a point p_i whose group $\mathcal{A}_{p_i}$ only has a single sublist (which is the main sublist of p_i).*

2.1 The weighted dominating set problem

In this section, we present our algorithm for computing a minimum-weight dominating set in the disk graph $G(P)$. For each point $p_i \in P$, let $w_i > 0$ denote its weight. For any subset $P' \subseteq P$, define $w(P') = \sum_{p_i \in P'} w_i$.

We consider the following *bounded-size problem*: Given an integer k, find a dominating set $S \subseteq P$ of minimum total weight in $G(P)$ subject to $|S| \leq k$. Solving this problem with $k = n$ yields a minimum-weight dominating set for $G(P)$. Let W^* denote the total weight of a minimum-weight dominating set of size at most k.

Algorithm overview. Our algorithm is a dynamic program with k iterations. In each t-th iteration, $1 \leq t \leq k$, we compute for each point $p_i \in P$ a set $\mathcal{L}_t(i)$ of $O(n^2)$ sublists of P, with each sublist $L \in \mathcal{L}_t(i)$ associated with a value $w'(L)$ and a subset $S_L \subseteq P$, such that the following *algorithm invariants* are maintained: (1) $w(S_L) \leq w'(L)$; (2) S_L dominates L; (3) $p_i \in S_L$; (4) $|S_L| \leq t$. Define $\mathcal{L}_t = \cup_{p_i \in P} \mathcal{L}_t(i)$.

The following notation will be used in the rest of this paper.

Definition 1. *For two points $p_i, p_j \in P$ ($p_i = p_j$ is possible), define a_i^j as the index of the first point p of P counterclockwise from p_j such that $|D_{p_i} D_p| > 0$, and b_i^j the index of the first point p of P clockwise from p_j such that $|D_{p_i} D_p| > 0$ (if $|D_{p_i} D_{p_j}| > 0$, then $a_i^j = b_i^j = j$). If $|D_{p_i} D_p| \leq 0$ for all points $p \in P$, then let $a_i^j = b_i^j = 0$.*

Algorithm description. Initially, $t = 1$, and our algorithm computes two indices a_i^i and b_i^i as defined in Definition 1 for each $p_i \in P$. We will show in Lemma 4 that this can be done in $O(n \log^2 n)$ time. Then, for each $p_i \in P$, let $L = P(b_i^i, a_i^i)$, $S_L = \{p_i\}$, $w'(L) = w_i$, and $\mathcal{L}_1(i) = \{L\}$. Obviously, all algorithm invariants hold for L. This completes the first iteration of the algorithm.

Suppose that for each t', $1 \leq t' \leq t - 1$, we have computed a collection $\mathcal{L}_{t'}(i)$ of $O(n^2)$ sublists for each $p_i \in P$, with each sublist $L \in \mathcal{L}_{t'}(i)$ associated with a value $w'(L)$ and a point set $S_L \subseteq P$, such that the algorithm invariants hold for L, i.e., $w(S_L) \leq w'(L)$, S_L dominates L, $p_i \in S_L$, and $|S_L| \leq t'$. The t-th iteration of the algorithm works as follows. For each point $p_i \in P$, we perform the following procedures.

Counterclockwise/clockwise processing procedures. For each other point $p_j \in P$, we proceed as follows. For each t' with $1 \leq t' \leq t-1$, and each point $p_z \in P[i,j]$, we do the following. Refer to Figure 1. We first perform a *minimum-value enclosing sublist query* on $\mathcal{L}_{t'}(i)$ to find the sublist $L_1 \in \mathcal{L}_{t'}(i)$ of minimum $w'(L_1)$ such that $P[i,z] \subseteq L_1$. Let p_{z_1} be the counterclockwise endpoint of L_1. Then we perform another minimum-value enclosing sublist query on $\mathcal{L}_{t-t'}$ to find the sublist $L_2 \in \mathcal{L}_{t-t'}$ of minimum $w'(L_2)$ with $P[z_1+1,j] \subseteq L_2$. Let p_{z_2} be the counterclockwise endpoint of L_2. Next, we compute the index $a_i^{z_2+1}$. By construction, the union of the following four sublists are consecutive and thus form a sublist of P: $P(b_i^i, a_i^i)$, L_1, L_2 and $P(z_2, a_i^{z_2+1})$. Denote this combined sublist by $L(t',m)$, or L for simplicity. We set $S_L = S_{L_1} \cup S_{L_2}$ and $w'(L) = w'(L_1) + w'(L_2)$. Among all such sublists $L(t',m)$ with $1 \leq t' \leq t-1$ and $p_z \in P[i,j]$, we keep the one with minimum $w'(L(t',m))$ and add it to $\mathcal{L}_t(i)$.

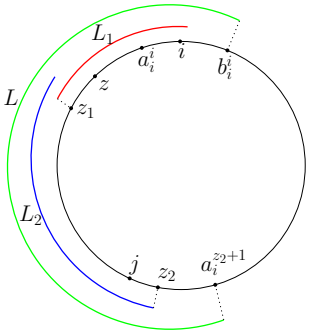


Figure 1: Illustrating the relative positions of points of P (only their indices are shown): the circle represent $\mathcal{H}(P)$.

Symmetrically, we perform a *clockwise processing procedure* for p_j, which also adds at most one sublist to $\mathcal{L}_t(i)$.

Bidirectional processing procedures. We also perform *bidirectional processing procedures* for p_i. For each pair of points (p_x, p_y) such that p_y, p_i, p_x are in counterclockwise order along $\mathcal{H}(P)$, we do the following for each t' with $2 \leq t' \leq t-1$. Refer to Figure 2. We perform a minimum-value enclosing sublist query on $\mathcal{L}_{t'}(i)$ to find the sublist $L_x \in \mathcal{L}_{t'}(i)$ of minimum $w'(L_x)$ such that $P[i,x] \subseteq L_x$. Similarly, we perform a minimum-value enclosing sublist query on $\mathcal{L}_{t+1-t'}(i)$ to find the sublist $L_y \in \mathcal{L}_{t+1-t'}(i)$ of minimum $w'(L_y)$ such that $P[y,i] \subseteq L_y$. By definition, the union of $P(b_i^i, a_i^i)$, L_x, and L_y are consecutive and thus form a sublist of P, denoted by $L(x,y,t')$ or simply L. We set $S_L = S_{L_x} \cup S_{L_y}$ and $w'(L) = w'(L_x) + w'(L_y) - w_i$. For a fixed pair (p_x, p_y), among all sublists $L(x,y,t')$, $2 \leq t' \leq t-1$, we keep the one with minimum $w'(L(x,y,t'))$ and add it to $\mathcal{L}_t(i)$. In this way, the bidirectional processing procedure for p_i adds $O(n^2)$ sublists to $\mathcal{L}_t(i)$.

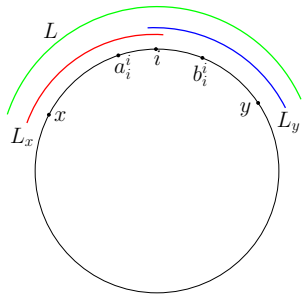


Figure 2: Illustrating L_x, L_y, and L.

After the k-th iteration, among all sublists of $\mathcal{L}_k$ that are P, we find the one L^* of minimum $w'(L^*)$ and return S_{L^*} as our optimal dominating set of size at most k. If no sublist of $\mathcal{L}_k$ is P, then we report that a dominating set of size at most k does not exist.

Algorithm correctness, time analysis, and implementation. The following lemma establishes the correctness of the algorithm. The proof (by induction on t) is based on Lemmas 1,2 (see the appendix for details).

Lemma 3. *S_{L^*} is an optimal dominating set and $W^* = w'(L^*)$.*

As for the time analysis, in the t-th iteration, $1 \leq t \leq k$, the algorithm performs $O(tn^3)$ minimum-value enclosing sublist queries: Given a sublist L of P, compute the minimum value sublist containing L in a set $\mathcal{L}$ of sublists. It was shown in [20] that each of these operations can be performed in $O(\log^2 m)$ time after an $O(m \log m)$ time preprocessing with $m = |\mathcal{L}|$. In addition, in the t-th iteration, the algorithm performs $O(tn^2)$ operations to compute the indices a_i^j and b_i^j. To this end, the following lemma provides a data structure.

Lemma 4. *We can construct a data structure for P in $O(n \log^2 n)$ time such that the indices a_i^j and b_i^j can be computed in $O(\log^2 n)$ time for any two points $p_i, p_j \in P$.*

The total size of the involved sets of $\mathcal{L}_t$ is $O(n^3)$, therefore the t-th iteration of the algorithm takes $O(tn^3 \log^2 n)$ time. As there are k iterations, we conclude with the following theorem, which in turn leads to the corollary.

Theorem 1. *Given a number k and a set of n weighted disks whose centers are in convex position in the plane, we can find in $O(k^2 n^3 \log^2 n)$ time a minimum-weight dominating set of size at most k in the disk graph, or report that no such dominating set exists.*

Corollary 1. *Given a set of n weighted disks whose centers are in convex position in the plane, we can compute a minimum-weight dominating set in the disk graph in $O(n^5 \log^2 n)$ time.*

2.2 The unweighted case

For the unweighted case, the goal is to compute a smallest dominating set in the disk graph $G(P)$. By setting the weights of all points of P to 1 and applying Corollary 1, one can solve the unweighted problem in $O(n^5 \log^2 n)$ time. Below, we provide an algorithm of $O(k^2 n \log n)$ time, where k is the size of the smallest dominating set.

Algorithm description. We still proceed in iterations $t = 1, 2 \ldots$. In each t-th iteration, we compute for each $p_i \in P$ a set $\mathcal{L}_t(i)$ of $O(t)$ sublists, with each sublist L associated with a subset $S_L \subseteq P$, such that the following algorithm invariants are maintained: (1) S_L dominates L; (2) $p_i \in S_L$; (3) $p_i \in L$; (4) $|S_L| \leq t$. Define $\mathcal{L}_t = \cup_{p_i \in P} \mathcal{L}_t(i)$. Hence, $|\mathcal{L}_t| = O(nt)$.

Let k be the smallest dominating set size. We show that after k iterations, $\mathcal{L}_k$ is guaranteed to contain a sublist that is P. As such, if a sublist that is P is computed for the first time in the algorithm, then we can stop the algorithm.

Initially, $t = 1$, and our algorithm does the following for each $p_i \in P$. We compute $L = P(b_i^i, a_i^i)$, and set $S_L = \{p_i\}$. We let $\mathcal{L}_1(i) = \{L\}$. Obviously, all algorithm invariants hold for L.

In general, suppose that for each $t' \in [1, t-1]$, we have computed a collection $\mathcal{L}_{t'}(i)$ of $O(t')$ sublists for each $p_i \in P$, with each sublist $L' \in \mathcal{L}_{t'}(i)$ associated with a subset $S_{L'} \subseteq P$, such that the algorithm invariants hold, i.e., L' is dominated by $S_{L'}$, $p_i \in S_{L'}$, $p_i \in L'$, and $|S_{L'}| \leq t'$. The t-th iteration works as follows. For each point $p_i \in P$, we perform the following procedures that now incorporate greedy strategies.

Counterclockwise/clockwise processing procedures. For each point $p_i \in P$, for each $t' \in [1, t-1]$, we do the following. One can still refer to Figure 1 (but the notation z can be ignored). By our algorithm invariants for t', every sublist of $\mathcal{L}_{t'}(i)$ contains p_i. We first find the sublist of $\mathcal{L}_{t'}(i)$ whose counterclockwise endpoint is farthest from p_i (in the counterclockwise direction). Let L_1 denote the list and let p_{z_1} be the counterclockwise endpoint of L_1. We then perform a *counterclockwise farthest enclosing sublist query* on $\mathcal{L}_{t-t'}$ to compute a sublist containing p_{z_1+1} such that its counterclockwise endpoint is farthest from p_{z_1+1} (in the counterclockwise direction). Let L_2 be the sublist and z_2 the counterclockwise endpoint of L_2. Next, we compute the index $a_i^{z_2+1}$. By construction, the union of the following four sublists is a (contiguous) sublist of P: $P(b_i^i, a_i^i)$, L_1, L_2, and $P(z_2, a_i^{z_2+1})$. Denote this combined sublist by $L(t')$, or simply L. We set $S_L = S_{L_1} \cup S_{L_2}$. Among the $O(t)$ sublists $L(t')$ computed above for all $t' \in [1, t-1]$, we keep the one L whose counterclockwise endpoint is farthest from p_i and add it to $\mathcal{L}_t(i)$.

Symmetrically, we perform a clockwise processing procedure for p_i.

Bidirectional processing procedures. For each point $p_i \in P$, for each $t' \in [2, t-1]$, we perform the following bidirectional processing procedure. One can still refer to Figure 2 (but the notation x and y in the figure can be ignored). We find the sublist L_x in $\mathcal{L}_{t'}(i)$ whose counterclockwise endpoint is farthest from p_i (along the counterclockwise direction), and the sublist L_y in $\mathcal{L}_{t+1-t'}(i)$ whose clockwise endpoint is farthest from p_i (along the clockwise direction). The following three sublists form a (contiguous) sublist of P: $P(b_i^i, a_i^i)$, L_x, and L_y. Let L be the union of the above three sublists. We set $S_L = S_{L_x} \cup S_{L_y}$, and add L to $\mathcal{L}_t(i)$.

If any sublist $L \in \mathcal{L}_t$ is P, then we stop the algorithm and return S_L as a smallest dominating set. Otherwise, we continue on the next iteration.

Algorithm correctness, time analysis, and implementation. The following lemma demonstrates the correctness of the algorithm, whose proof can be found in the appendix.

Lemma 5. *If the algorithm first time computes a sublist L that is P, then S_L is a smallest dominating set of $G(P)$.*

In each t-th iteration with $1 \leq t \leq k$, we perform $O(tn)$ operations for computing indices a_i^j and b_i^j and $O(tn)$ counterclockwise/clockwise farthest enclosing sublist queries: Given a point $p \in P$, find from a set $\mathcal{L}$ of sublists a sublist containing p with the farthest counterclockwise/clockwise endpoint from p. We can construct a data structure for $\mathcal{L}$ in $O(m \log m)$ time such that each such query can be answered in $O(\log m)$ time with $m = |\mathcal{L}|$ [20]. Additionally, computing indices a_i^j and b_i^j takes $O(\log^2 n)$ time by Lemma 4. The total size of the involved sets of $\mathcal{L}$ is $O(tn)$. Hence, the total time of each t-th iteration is $O(tn \log^2 n)$. As there are k iterations, we conclude this section with the following theorem.

Theorem 2. *Given a set of n disks whose centers are in convex position in the plane, a smallest dominating set of the disk graph can be computed in $O(k^2 n \log^2 n)$ time, where k is the smallest dominating set size.*

References

[1] Alok Aggarwal, Leonidas J. Guibas, James Saxe, and Peter W. Shor. A linear-time algorithm for computing the Voronoi diagram of a convex polygon. *Discrete and Computational Geometry*, 4:591–604, 1989. doi:10.1007/BF02187749. 1

[2] Alok Aggarwal, Hiroshi Imai, Naoki Katoh, and Subhash Suri. Finding k points with minimum diameter and related problems. *Journal of Algorithms*, 12(1):38–56, 1991. doi:10.1016/0196-6774(91)90022-Q. 1

[3] Frank Ángel Hernández Mira, Ernesto Parra Inza, José Marıia Sigarreta Almira, and Nodari Vakhania. A polynomial-time approximation to a minimum dominating set in a graph. *Theoretical Computer Science*, 930:142–156, 2022. doi:10.1016/j.tcs.2022.07.020. 1

[4] Ahmad Biniaz, Mahdi Amani, Anil Maheshwari, Michiel Smid, Prosenjit Bose, and Jean-Lou De Carufel. A plane 1.88-spanner for points in convex position. *Journal of Computational Geometry*, 7(1):520–539, 2016. doi:10.20382/jocg.v7i1a21. 1

[5] Onur Çağırici and Bodhayan Roy. Maximum clique of disks in convex position. In *Abstracts of Computational Geometry: Young Researchers Forum (CG:YRF)*, pages 21:1–21:3, 2018. URL: https://www.computational-geometry.org/old/YRF/cgyrf2018.pdf. 1

[6] Bernard Chazelle, Herbert Edelsbrunner, Michelangelo Grigni, Leonidas Guibas, Micha Sharir, and Emo Welzl. Improved bounds on weak ϵ-nets for convex sets. In *Proceedings of the 25th Annual ACM Symposium on Theory of Computing (STOC)*, pages 495–504, 1993. doi:10.1145/167088.167222. 1

[7] Jongmin Choi, Jaegun Lee, and Hee-Kap Ahn. Efficient k-center algorithms for planar points in convex position. In *Proceedings of the 18th Algorithms and Data Structures Symposium (WADS)*, pages 262–274, 2023. doi:10.1007/978-3-031-38906-1_18. 1

[8] Brent N. Clark, Charles J. Colbourn, and David S. Johnson. Unit disk graphs. *Discrete Mathematics*, 86:165–177, 1990. doi:10.1016/0012-365X(90)90358-O. 1

[9] Minati De and Abhiruk Lahiri. Geometric dominating-set and set-cover via local-search. *Computational Geometry*, 113(C):102007, 2023. doi:10.1016/j.comgeo.2023.102007. 1

[10] D. Eppstein and Jeff Erickson. Iterated nearest neighbors and finding minimal polytopes. *Discrete and Computational Geometry*, 11(1):321–350, 2007. 1

[11] Jared Espenant, J. Mark Keil, and Debajyoti Mondal. Finding a maximum clique in a disk graph. In *Proceedings of the 39th International Symposium on Computational Geometry (SoCG)*, pages 30:1–30:17, 2023. doi:10.4230/LIPIcs.SoCG.2023.30. 1

[12] Matt Gibson and Imran A. Pirwani. Algorithms for dominating set in disk graphs: Breaking the $\log n$ barrier. In *Proceedings of the 18th Annual European Symposium on Algorithms (ESA)*, pages 243–254, 2010. doi:10.1007/978-3-642-15775-2_21. 1

[13] J.E. Hopcroft and R.M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM J. on Comput.*, 2(4):225–231, 1973. 1

[14] J. Mark Keil and Debajyoti Mondal. The maximum clique problem in a disk graph made easy. In *Proceedings of the 41st International Symposium on Computational Geometry (SoCG)*, pages 63:1–63:16, 2025. doi:10.4230/LIPIcs.SoCG.2025.63. 1

[15] Andrzej Lingas. On approximation behavior and implementation of the greedy triangulation for convex planar point sets. In *Proceedings of the 2nd Annual Symposium on Computational Geometry (SoCG)*, pages 72–79, 1986. doi:10.1145/10515.10523. 1

[16] Nabil H. Mustafa and Saurabh Ray. Improved results on geometric hitting set problems. *Discrete and Computational Geometry*, 44:883–895, 2010. doi:10.1007/s00454-010-9285-9. 1

[17] Dana S. Richards and Jeffrey S. Salowe. A rectilinear steiner minimal tree algorithm for convex point sets. In *Proceedings of the 2nd Scandinavian Workshop on Algorithm Theory (SWAT)*, volume 447, pages 201–212, 1990. doi:10.1007/3-540-52846-6_90. 1

[18] Vishwanath R. Singireddy, Manjanna Basappa, and Joseph S.B. Mitchell. Algorithms for k-dispersion for points in convex position in the plane. In *Proceedings of the 9th International Conference on Algorithms and Discrete Applied Mathematics (CALDAM)*, pages 59–70, 2023. doi:10.1007/978-3-031-25211-2_5. 1

[19] Anastasiia Tkachenko and Haitao Wang. Computing maximum cliques in unit disk graphs. In *Proceedings of the 37th Canadian Conference on Computational Geometry (CCCG)*, pages 283–291, 2025. URL: https://cccg-wads-2025.eecs.yorku.ca/cccg-papers/6B2.pdf. 1

[20] Anastasiia Tkachenko and Haitao Wang. Dominating set, independent set, discrete k-center, dispersion, and related problems for planar points in convex position. In *Proceedings of the 42nd International Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 73:1–73:20, 2025. doi:10.4230/LIPIcs.STACS.2025.73. 1, 3, 4