Separating Two Points with Obstacles in the Plane: Improved Upper and Lower Bounds

Jack Spalding-Jamieson
Independent, Vancouver, Canada

Anurag Murty Naredla
University of Bonn, Germany

Abstract

Given two points in the plane, and a set of "obstacles" given as curves through the plane with assigned weights, we consider the **point-separation** problem, which asks for a minimum-weight subset of the obstacles separating the two points. A few computational models for this problem have been previously studied. We give a unified approach to this problem in all models via a reduction to a particular shortest-path problem, and obtain improved running times in essentially all cases. In addition, we also give fine-grained lower bounds for many cases.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases obstacle separation, point separation, geometric intersection graph, Z_2-homology, fine-grained lower bounds

Digital Object Identifier 10.4230/LIPIcs.ESA.2025.90

Related Version *Full Version*: `https://arxiv.org/abs/2504.17289` [17]

Acknowledgements This work originated from a problem discussion group in the Fall of 2021, whose members collectively contributed to the discussion and development of some of this work's preliminary results in the "arrangement model", as well as a better understanding of the problem as a whole. In addition to the authors, this group included (in alphabetical order) Sam Barr, Therese Biedl, Reza Bigdeli, Anna Lubiw, Jayson Lynch, Karthik Murali, and Graeme Stroud.
In addition, we want to thank Jeff Erickson, Elizabeth Xiao, and Da Wei Zheng for helpful discussions about homology.

1 Introduction

Given points s and t in the plane, and a weighted set of **obstacles** $\mathcal{C}$ defined by simple closed curves (possibly also including their interiors), the (s,t) **point-separation problem** asks for a minimum-weight subset C of $\mathcal{C}$ such that any path from s to t intersects some obstacle in C. Equivalently, s and t are in different connected components of $\mathbb{R}^2 \setminus (\cup_{\gamma \in C} \gamma)$. We say such a subset **separates** s and t. An example of this problem can be found in Figure 1.

This is a natural problem that arises in various scenarios. As a toy application of this problem: Suppose every night your dog runs from his doghouse to your backpack to eat your homework, taking an unpredictable route (making use of windows and doors). You have noticed that your dog will forget about your homework if it smells a treat. You have a number of candidate locations to place treats, and you'd like to ensure your dog is distracted from your homework every day using the fewest treats possible. See Figure 2 for this example. Similar applications also arise when considering security (e.g., replacing the batteries in the fewest number of your dead security cameras to cover all possible paths from the entrance to your bank vault). Additionally, (s,t) point-separation has found an application as a tool for constant-factor approximation of the well-studied APX-hard problem "barrier-resilience" [13].

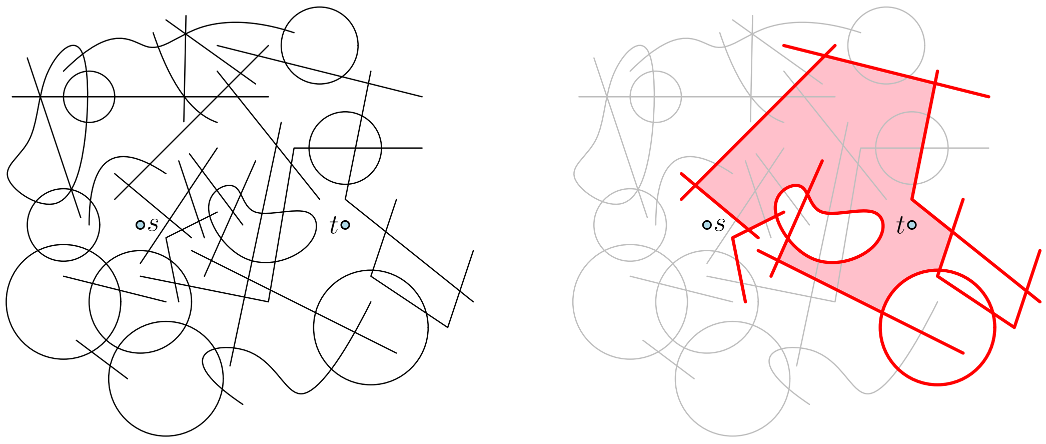


Figure 1 An instance of the (s, t) point-separation problem.

Item	Eating locations (◆)	Walls	Couch	Windows
Cost	1	0	0	∞

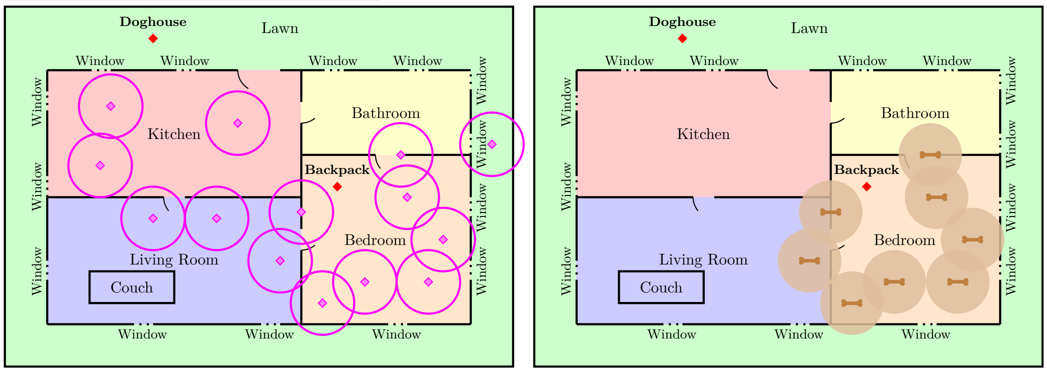


Figure 2 An example of the point-separation problem applied to placing distracting doggy treats around a house, with the minimum-weight solution on the right.

For brevity, we will henceforth say "curve" to mean a simple closed curve in $\mathbb{R}^2 \setminus \{s, t\}$. The algorithmic complexity of the (s, t) point-separation problem depends on the chosen computational model of the obstacles. Previous works use a few different models. We categorize and name the different classes of models used as follows:

- **Specific Obstacle-Type Models**: Assume the set of curves $\mathcal{C}$ takes on a special form with a standard representation, such as a set of disks, circles, or line segments.
- **The Oracle-based Intersection Graph Model**: Assume the existence of several $O(1)$-time oracles that would allow the computation of the **intersection graph** G of $\mathcal{C}$ (the graph whose vertices are exactly the curves $\mathcal{C}$ and whose edges correspond to pairwise intersections of these curves), as well as some additional information for each edge related to s and t (detailed in Section 2).
- **The Arrangement Model**: Assume the arrangement of $\mathcal{C}$ is provided as input, in the form of a plane (multi-)graph, with the faces corresponding to s and t labelled. This is a more graph-theoretic formulation. See Figure 3 for an example of such an arrangement.

Prior works discussing this problem each focused on only one of these paradigms, and the names for each paradigm are our own. We will not assume general position, so no paradigm is strictly more general than the others, since there are arrangements of n obstacles with $\Theta(n^2)$ pairwise intersections, but only $O(n)$ unique intersection points.

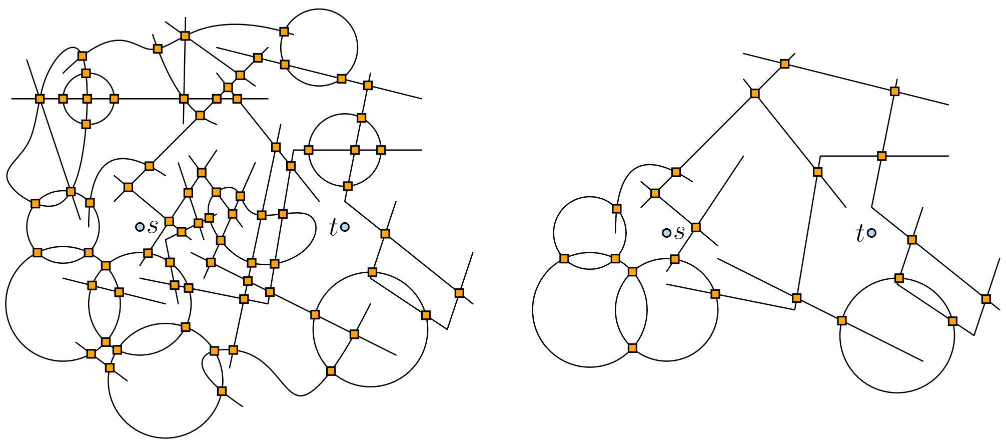


Figure 3 An arrangement representation of the instance in Figure 1, as well as an arrangement of a smaller instance.

In our work, our key method is to frame all models of this problem in terms of a "homology cover". Homology is a very broad topic in algebraic topology, which we will not attempt to summarize in this paper. We aim our paper at a typical computational geometry audience, and we will not assume prior knowledge of homology. We will present the necessary aspects in Section 2.

1.1 Prior Work (Brief)

In this subsection, we very briefly outline some key aspects of prior work. A significantly extended form of this section is in the appendix of the full version.

Most importantly for our methods, Kumar, Lokshtanov, Saurabh and Suri [13, Section 6] describe an algorithm that runs in polynomial time in the *arrangement model*. Their algorithm implicitly makes use of the homology cover (perhaps unintentionally), in the same sense that we will use it. In fact, their algorithm is in some ways analogous to an algorithm of Chambers, Erickson, Fox, and Nayyeri [5] for minimum-cuts (and maximum-flow) in surface graphs. These two algorithms are the main inspiration for our approach to the (s, t) point-separation at a high-level, in *all* models.

1.2 Results and Organization

We obtain improved algorithms for the (s, t)-point separation problem in several cases, which we outline in Table 1. As mentioned earlier, all of our positive results make use of a reduction to a shortest-path problem in the so-called "homology cover". We discuss this formulation in Section 2, and then again more rigorously in the appendix of the full version. Using this reduction, the upper bounds are then given in Section 3.

We also obtain several fine-grained lower bounds in Section 4. Our lower bounds also all have a shared foundation, which will be stated in Theorem 13. The resulting lower bounds are based on a few different hypotheses, and we give their details in the appendix of the full version.

Table 1 The time complexities of our algorithms for various obstacle models. In all cases, $n := |\mathcal{C}|$. For the arrangement model, k denotes the vertex count of the arrangement, and m is the number of obstacle-vertex incidences (so $m \geq k$). The notation $\Omega^*(\cdot)$ hides sub-polynomial factors. The notation † denotes that this is only true for one particular mutual dependence of k, m, and n (and is a slight abuse of notation). ω is the matrix-multiplication exponent ($\omega < 2.371339$ [2]). The lower bound results are given in the full version.

Model	Weights	Old	New	Cond. L.B.
Oracle	Yes	$O(n^3)$ [3]	$\widetilde{O}(n^{(3+\omega)/2})$ (Thm. 4)	$\Omega^*\left(n^2\right)$
Oracle	No	$O(n^3)$ [3]	$O(n^\omega \log n)$ (Thm. 5)	$\Omega^*(n^{3/2})$
Arrangement	Yes	$O(km^2 \lg k)$ [13]	$O(km + k^2 \lg k)$ (Thm. 6)	$\Omega^*\ (km)^\dagger$
Segments	No	$O(n^3)$ [3]	$O(n^{7/3} \log^{1/3} n)$ (Cor. 9)	$\Omega^*(n^{3/2})$
Axis-aligned segments	No	$O(n^3)$ [3]	$O(n^2 \log\log n)$ (Cor. 9)	None
Unit Disks	No	$O(n^2 \log^3 n)$ [4]	$O(n^2 \log n)$ (Cor. 9)	None
Disks	No	$O(n^3)$ [3]	$O(n^2 \log n)$ (Cor. 9)	None
$O(1)$-length polylines	No	$O(n^3)$ [3]	$O(n^{7/3} \log^{1/3} n)$ (Cor. 9)	$\Omega^*\left(n^{3/2}\right)$
$O(1)$-length rectilinear polylines	No	$O(n^3)$ [3]	$O(n^2 \log\log n)$ (Cor. 9)	$\Omega^*(n^{3/2})$
Segments	Yes	$O(n^3)$ [3]	$\widetilde{O}(n^{7/3})$ (Cor. 12)	$\Omega^*\left(n^2\right)$
Axis-aligned segments	Yes	$O(n^3)$ [3]	$\widetilde{O}(n^2)$ (Cor. 12)	None
$O(1)$-length polylines	Yes	$O(n^3)$ [3]	$\widetilde{O}(n^{7/3})$ (Cor. 12)	$\Omega^*\left(n^2\right)$
$O(1)$-length rectilinear polylines	Yes	$O(n^3)$ [3]	$\widetilde{O}(n^2)$ (Cor. 12)	$\Omega^*(n^2)$

2 Homology and Obstacles [Informal]

In this section, we will explain the main topological tools we will use to approach the (s, t) point-separation problem. However, this will be an information section, not requiring prior knowledge of any aspect of topology. Rather, we will give an equivalent formulation of the important pieces using simpler tools from computational geometry. We give a more formal treatment in the appendix of the full version. Throughout this section, we will make (implicit) assumptions of general position and such, but the deferred formal treatment does not need these.

At a high-level, there are two main steps to the constructions of our approach. First, we will discuss what it means for a *simple curve* to separate s and t, and what tools exist to classify such curves. Second, we will discuss what it means for a *set of obstacles* to separate s and t. That is, when does the union of the obstacles *contain* a simple curve separating s and t?

The simplest possible case of asking whether a simple curve separates s and t is characterized by the **point-in-polygon** problem, which asks: Given a point p and a (simple) polygon P, is p inside P? In fact, this is equivalent to asking if P separates p and the point

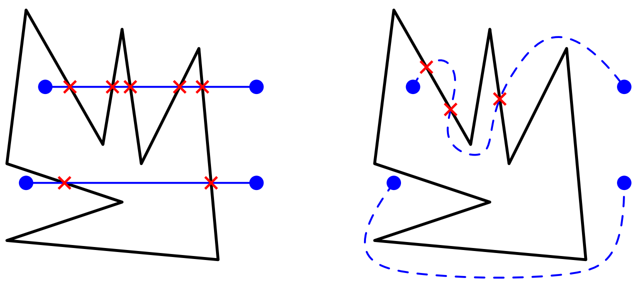


Figure 4 A demonstration of the algorithm for the point-in-polygon problem (left), as well as the problem of testing whether or not a polygon separates two points (middle). On the right, alternative paths for the separation problem are given.

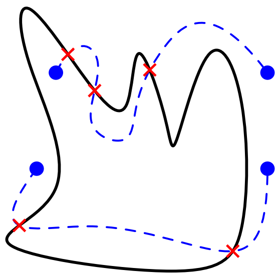
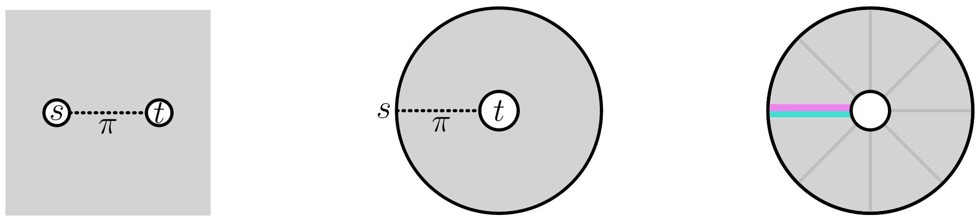


Figure 5 Examples of point-pairs separated or not separated by a given closed path between the pair, and the corresponding intersections of those curves.

at infinity. There is a folklore algorithm for this problem that takes any ray r starting at p, and counts the number of intersections between r and P. Then, P contains p if and only if the number of intersections is odd. In fact, essentially the same algorithm can be used to test whether or not P separates two points p_1 and p_2. Rather than using a ray, take the segment $\overline{p_1p_2}$. Count the number of intersections between $\overline{p_1p_2}$ and P. The count is odd if and only if P separates p_1 and p_2. Moreover, there is in fact nothing special about rays or line segments in this problem. *Any* (closed) path between p_1 and p_2 would cross P the same number of times, modulo 2. See Figure 4 for examples for these algorithms.

All three of these algorithms are equivalent in a sense. In fact, there is a further generalization: Given two points s and t on the sphere, a simple and closed curve C (not covering s or t), and any $s - t$ path π, C separates s and t if and only if π crosses C an odd number of times (regardless of the choice of π). Since the extended plane is homeomorphic to the sphere, a ray in the plane corresponds to a (simple) path in the sphere. See Figure 5 for an example.

The problems we have discussed so far are all completely static, and the methods do not provide much structure for solving more difficult problems. One of the most common ways to extend the point-in-polygon problem is to fix the polygon (or curve) P, and aim to support fast queries of points. This problem is known as "point-location", and it is well-studied in computational geometry [12]. However, we want a different sort of structure: We have a fixed pair of points s and t, and we wish to classify the curves that separate them. Since we have fixed s and t, we can also fix the path π between them – in most cases, we will use the line segment $\overline{st}$. Then, the problem of classifying curves that separate s and t becomes the problem of classifying curves that cross $\overline{st}$ an odd number of times.

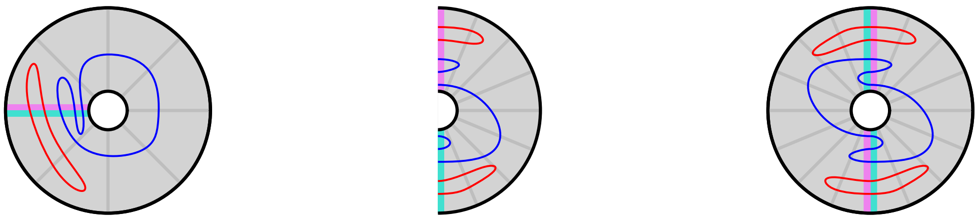


(a) s and t are punctures, π is a path between them.

(b) s and t are enlarged to become holes, and the space is an annulus.

(c) The two sides of π can be coloured to aid later notation.

Figure 6 A demonstration of how the (extended) plane with two punctures is homeomorphic to the annulus.



(a) An annulus with a specified path π.

(b) The annulus is sliced open along the specified path π.

(c) Two copies of the sliced annulus are glued together.

Figure 7 The "cut and glue" construction of the homology cover, as well as how it maps a closed curve that separates the two boundaries (blue) and one that does not (red).

It will be helpful for demonstration to make some transformations to the space we work in. That is, we will perform a sequence of homeomorphisms. We start with the extended plane with the two marked points s and t, and the line segment $\overline{st}$ (see Figure 6a). No curves we will be classifying cross s or t, so we may assume there are punctures at s and t. We can enlarge these punctures into holes with a homeomorphism. Next, since we have the extended plane, we can make one of the punctures the outer face, obtaining an annulus (see Figure 6b and Figure 6c). In performing these steps, the line segment $\overline{st}$ becomes a path between the inner and outer boundaries of the annulus.

We now present a method for constructing an important space called the **homology cover**[1]. Take the specified path π between the two boundaries (see Figure 7a) and slice it open (see Figure 7b). Then, create a second copy of the sliced annulus, and glue them together in a way that matches up the orientations of the sliced ends (see Figure 7c).

Alternatively, an equivalent construction is to create two copies of the original space (whether that be the extended plane or the annulus), and use each side of the path from s to t as a (separate) "portal" between the two copies. A more general form of this "portal" idea has been studied in the form of "portalgons" [14, 15], of which the homology cover is essentially a special case. See Figure 8 for an example of this construction.

One important aspect of this construction is that it involves the connection of two identical copies of the annulus (i.e., the extended plane with punctures s and t). In Figure 7, we also show how this construction transforms two curves – one separating the two boundaries, and one not separating them. The structure we will make use of to study curves separating s and t in the plane ultimately stems from an important set of facts:

[1] We use the term "homology cover" to refer to the one-dimensional $\mathbb{Z}_2$-homology cover of the annulus.

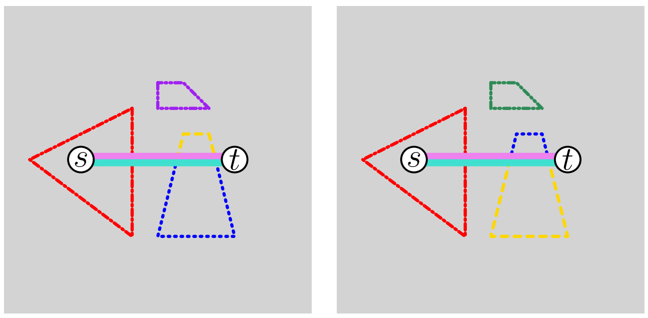


Figure 8 The “portal” construction of the homology cover. Each colour (or dot/dash pattern) is a single closed curve in the homology cover. The two solid lines are the portals.

▶ **Fact 1.** *For a simple curve C in the annulus (or the plane) that gets mapped to the set C' in the homology cover, and a point p along C that gets mapped to corresponding points p_1 and p_2 in the homology cover, the following are all equivalent:*

- *C separates s and t.*
- *C' separates the two boundaries in the homology cover.*
- *C' has one connected component (i.e., it is one closed curve instead of two).*
- *The points p_1 and p_2 are connected by a path through C'.*

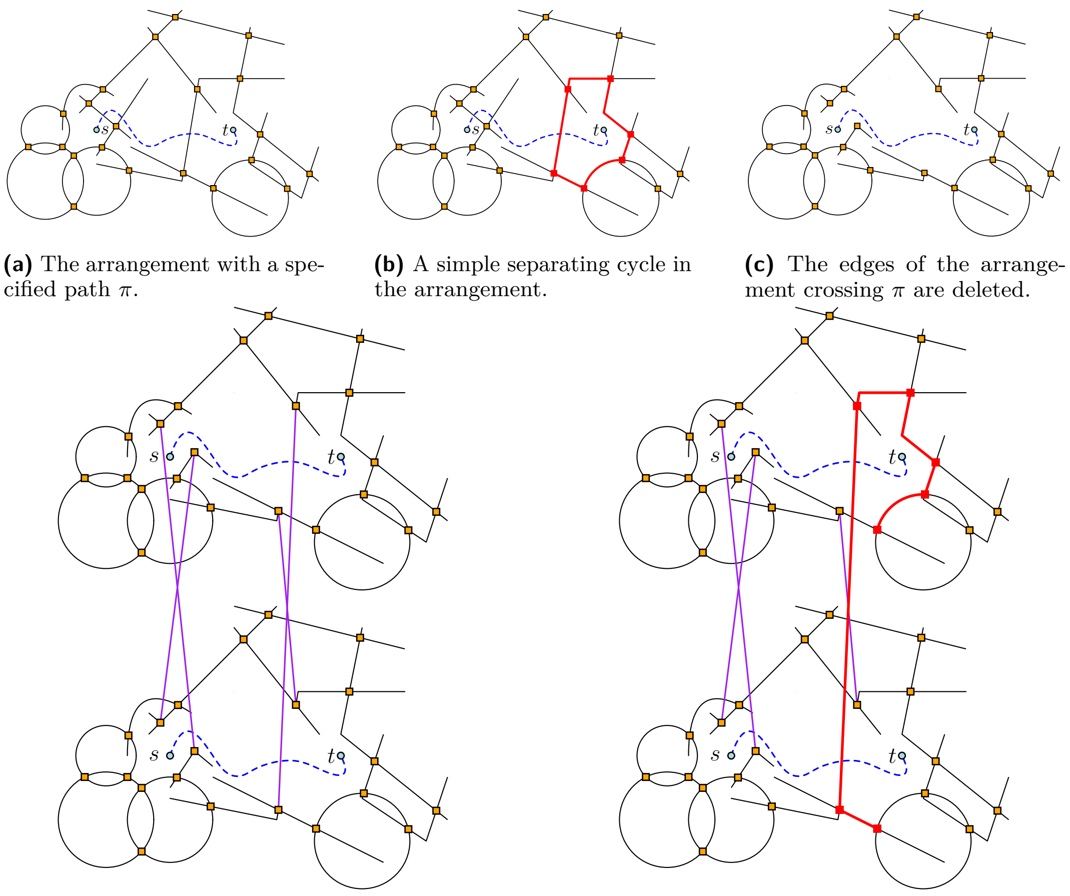
It is this last characterization that will be the most critical for obstacles. In particular, a consequence of this characterization is that if two corresponding points p_1 and p_2 in the homology cover (corresponding in the sense that they are identical points in different copies of the annulus/plane) have a simple path π between them, then that path can be mapped to a closed curve separating s and t.

2.1 Obstacles and the Homology Cover

We now have tools for characterizing *closed curves* that separate s and t. We'd now like to characterize *sets of obstacles* that separate s and t. In other words, we'd like to characterize unions of closed curves (obstacles) that contain a closed curve separating s and t. We'll give two different tools for this, first for the arrangement model, then for the oracle model. For simplicity, we will work exclusively with obstacles that do not *individually* separate s and t (that is, obstacles that get mapped to *two* separate closed curves in the homology cover). We will reduce to this case algorithmically at the start of Section 3.

In the arrangement model of the (s, t) point-separation problem, we are given a plane graph D (the arrangement) with specified faces s and t (can be obtained from points via point-location), and a set of obstacles σ, each given as a connected subgraph. The homology cover has a simple graph-theoretic interpretation in this framework: Take any (simple) dual-path π of faces from s to t – this is will serve as the “portal”. Next, create two copies D_1, D_2 of D, each with the faces s and t removed. We will create a combined graph $\overline{D}$ starting from the union of D_1 and D_2. For each edge $e = uv$ used in the dual-path π, delete e from both D_1 and D_2 (inside $\overline{D}$). Denote the corresponding vertices of u and v in each of D_1 and D_2 as u_1, v_1 and u_2, v_2, respectively. Then create new edges u_1v_2 and u_2v_1 in $\overline{D}$. This form of the homology cover is visualized in Figure 9.

With this particular arrangement structure Kumar, Lokshtanov, Saurabh and Suri [13, Section 6] built a graph using one vertex per obstacle-vertex incidence (in each copy of the plane graph). We will build a smaller graph of similar form to theirs. First, for each obstacle



(a) The arrangement with a specified path π.

(b) A simple separating cycle in the arrangement.

(c) The edges of the arrangement crossing π are deleted.

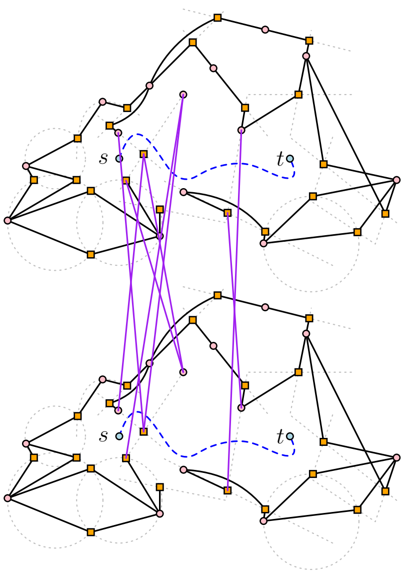
(d) Two copies of are created, and the deleted edges are added as “crossing” edges (in purple).

(e) A path between two copies of the same original vertex that would project onto the simple separating cycle in the original space from Figure 9b.

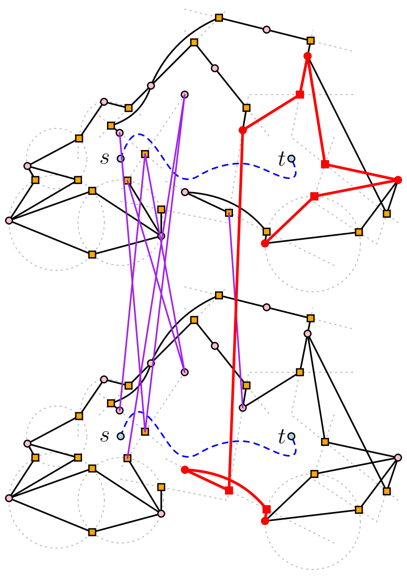
Figure 9 The construction of the homology cover for the second arrangement given in Figure 3.

$C \in \sigma$ (which induces a subgraph of D), pick some arbitrary **canonical point** $x \in C$. Since this choice is arbitrary, it will sometimes be useful to assume it is a specific point – this will primarily be useful for specific obstacle types. Assume that x is also a vertex in the subgraph of D induced by C (and if it is not, modify D so that it is, with either a subdivision or a new degree-1 vertex). Note that since C is connected in D, every other vertex $x' \in C$ is connected to x by only edges in C. Given a plane graph D with faces s and t, a dual-path π, obstacles σ, and canonical points for each obstacle, the **auxiliary graph** H is a bipartite graph constructed as follows:

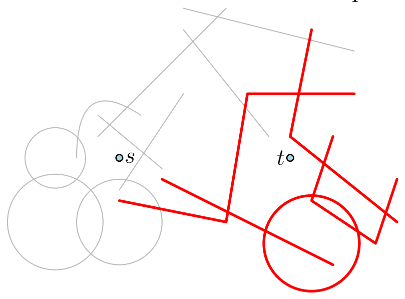
- The first set of vertices are the copies of arrangement vertices in the homology cover. For a vertex $v \in V(D)$, we denote these two copies v^+ and v^-.
- The second set of vertices are the copies of the obstacles/canonical points in the homology cover. For an obstacle $\gamma \in \sigma$, we denote these two copies as $(\gamma, -)$ and $(\gamma, +)$.
- The values $-$ and $+$ in each of the above are **indicator bits**, and can be stored as 0 and 1, respectively.
- A vertex copy v^{b_1} has an edge connecting it to a canonical point copy (γ, b_2) when:
  - $v \in \gamma$ (that is, the point v belongs to the obstacle γ in the arrangement), and...
  - The canonical point x of γ has a path through the edges of γ to v whose intersection



(a) A canonical point is added along each obstacle, and edges are created from the canonical point to each arrangement vertex along the obstacle, subject to the parity conditions.



(b) A shortest-path between the two copies of an obstacle's canonical point. The obstacles whose canonical points lie in this path form a set of obstacles separating s and t.



(c) The set of obstacles induced by the shortest path in Figure 10b.

Figure 10 The construction of the auxiliary graph for the homology cover given in Figure 9.

with the edges of π has size $b_1 + b_2$ modulo 2 (i.e., if $b_1 = b_2$, the size must be even, and if $b_1 \neq b_2$, the size must be odd). Equivalently, the projected connected component of γ containing the canonical point copy (γ, b_2) also contains v^{b_1}.

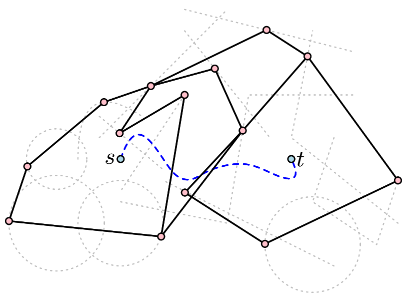
We visualize an example of the auxiliary graph in Figure 10.

The auxiliary graph is useful because the shortest path between any two corresponding vertex copies or (separately) any two corresponding canonical point copies both correspond exactly to the solution to the (s,t) point-separation problem. The high-level construction to prove this fact is as follows:

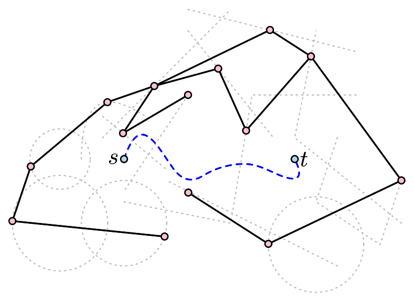
- Suppose there is a set of obstacles C. We'd like to determine when C separates s and t.
- Denote the set of copies of the obstacles in C in the homology cover as C', so that every element of C' is a closed curve representing one of the two connected components of an element of C projected into the homology cover.
- By Fact 1, we deduce that C separates s and t if and only if there is a path from v^+ to v^- along the union of obstacle copies in C', for some arrangement vertex v among pairs of elements in C only.

- Moreover, since each obstacle is connected, we may further assume that any such path visits each obstacle copy at most once, and moreover that it only visits each *obstacle* at most once.
- We may further assume that any such path visits the canonical point copy of each such obstacle copy, by inserting a path to and then from the canonical point copy while the obstacle copy is visited. Note that the resulting path may not be simple in the plane.
- Such paths also directly correspond to paths in the auxiliary graph, using only "obstacle vertices" corresponding to elements of C and the "arrangement vertices" incident to pairs of elements in C.
- With appropriate weights, the problem of finding the minimum-weight set C with this property then reduces to the problem of finding the pair v^+, v^- with the shortest distance in the auxiliary graph, so this is a shortest-path problem!
- A similar argument can show that an alternative algorithmic formulation is to find the pair $(\gamma, -), (\gamma, +)$ with the shortest-distance in the auxiliary graph.

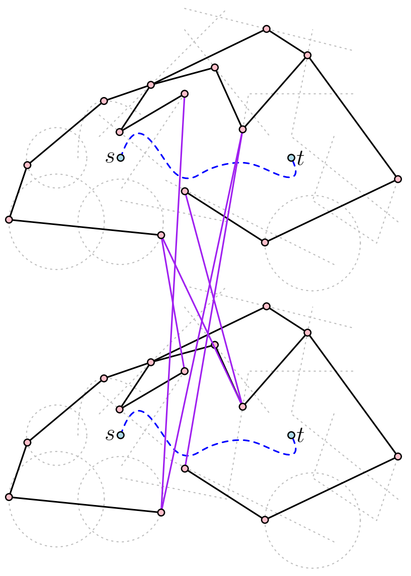
This essentially completes the set of tools necessary for the arrangement model. These arguments are given in more detail in the appendix of the full version.



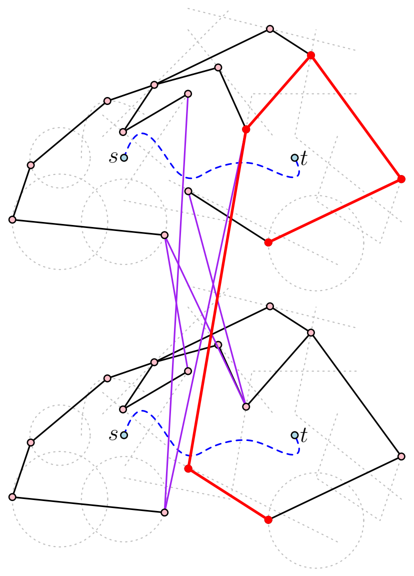
(a) The intersection graph with the path π.



(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 obstacle.

Figure 11 How the intersection graph can be transformed into the intersection graph in the homology cover.

For the oracle model, the purpose of the oracles will be to construct the geometric intersection graph of the obstacle *copies* in the homology cover (which we will denote as $\overline{G}$). We will call $\overline{G}$ the **intersection graph in the homology cover**. See Figure 11 for an example. We will present two different constructions of this graph. They are equivalent, and each of them is quite simple, but they will serve two different purposes: The first will clarify which types of oracle queries are necessary, and provide an algorithmic construction. The second will instead build on the tools we have for the arrangement model, proving the correctness of another shortest-paths approach.

Let $\mathcal{C}$ denote the set of obstacles in the plane. We assume the canonical points for each obstacle are given (or implied) – they will be used by the oracle. We also assume some simple $s-t$ path π is fixed – this will also be used by the oracle. In most cases, π will be $\overline{st}$. The first construction of the graph is as follows:

- For each obstacle $\gamma \in \mathcal{C}$, create two vertices $(\gamma, -)$ and $(\gamma, +)$.
- For each pair of obstacles γ_1, γ_2 with canonical points x_1, x_2, and values $b_1, b_2 \in \{-, +\}$, we connect vertices (γ_1, b_1) and (γ_2, b_2) by an edge if there is a path from x_1 to x_2 in $\gamma_1 \cup \gamma_2$ crossing π exactly k times, for some k where $k \equiv b_1 + b_2 \pmod 2$.

Testing for this condition is actually the *only* type of oracle query needed (although we will also require support for the case when $\gamma_1 = \gamma_2$ later). Cabello and Giannopoulos [3] used a larger set of query types, but together they can be used to perform this type by carefully choosing the canonical points and fixing π to $\overline{st}$, so our variant of the oracle model is slightly more general (although practically equivalent).

We'd like a similar algorithmic property for the intersection graph in the homology cover $\overline{G}$ that we have for the auxiliary graph H. The second construction will be what gives us that property, by constructing $\overline{G}$ *from* H:

- For each arrangement vertex v^b in H, let its adjacent vertices be denoted $x_1, \ldots, x_k$. Delete v^b and create a clique over $\{x_1, \ldots, x_k\}$.
- After performing this for every arrangement vertex v^b, the result is *exactly* the intersection graph in the homology cover $\overline{G}$.
- Therefore, the set of vertices visited by the shortest path from some $(\gamma, -)$ to $(\gamma, +)$ (over all possible $\gamma \in \mathcal{C}$, breaking ties arbitrarily) corresponds to a minimum-weight set of obstacles separating s and t.

For specific obstacle types, we will usually work with $\overline{G}$, although we will do it implicitly. Hence, with the tools from this section, we can now discuss algorithms in all three model types as solutions to a certain form of shortest-path problem.

3 Algorithmic Results

In this section, we will devise algorithms for the (s, t) point-separation problem based on our homology cover structures. We will devise algorithms for all three model types: Some that work with the arrangements of curves, some that work with the "oracle model" (the intersection graph in the homology cover), and some that work with specific types of obstacles. Most of our algorithms are fairly simple reductions to various known shortest-path algorithms, greatly simplifying some of the previous approaches to the (s, t) point-separation problem. However, some of them are more involved.

In the previous section, we assumed no obstacle individually separated s and t. Before moving on, we quickly show this is enough:

▶ **Observation 2.** *Let $\mathcal{C}$ be a set of obstacles, and let s and t be points. Suppose an oracle exists that, for an obstacle $\gamma \in \mathcal{C}$, determines whether γ itself separates s and t, all in $O(1)$ time. Let $\mathcal{C}_0$ be the set of all obstacles that do. Then an instance of the weighted* *(unweighted) (s,t) point-separation problem over $\mathcal{C}$ can be reduced to an instance of the weighted (unweighted) (s,t) point-separation problem over $\mathcal{C} \setminus \mathcal{C}_0$ in $O(|\mathcal{C}|)$ time. In particular, such a reduction returns the best solution out of the solved subproblem (if any), and each of the individual obstacles forming solutions (if any).*

3.1 General Weighted Obstacles in Oracle Model

With no significant further work, we already obtain some naïve algorithms by using a result in the appendix of the full version (or the constructions in Section 2):

▶ **Theorem 3.** *For points s and t, and a weighted set of obstacles $\mathcal{C}$ that have k pairwise intersections, the (s,t) point-separation problem can be solved in $O(|\mathcal{C}| \cdot k + |\mathcal{C}|^2 \log |\mathcal{C}|)$ time.*

Proof. Compute the intersection graph in the homology cover $\overline{G}$ of $\mathcal{C}$. For each $\gamma \in \mathcal{C}$, compute the shortest path from $(\gamma, -)$ to $(\gamma, +)$ in $\overline{G}$. The smallest such path induces the solution by a result in the appendix of the full version. Each shortest path can be computed in $O(k + |\mathcal{C}| \log |\mathcal{C}|)$ time by Dijkstra's algorithm with a Fibonacci heap [11], and there are $O(|\mathcal{C}|)$ total such paths. ◀

Note that Theorem 3 matches the result of [3], although the algorithm is much simpler. When $k \in \Theta(|\mathcal{C}|^2)$ (i.e., dense intersection graphs), this algorithm runs in $O(|\mathcal{C}|^3)$ time, which could also be obtained by running Floyd-Warshall for APSP instead of Dijkstra. It is not known if the general form of weighted APSP can be solved in "truly" subcubic time (see the appendix of the full version). However, since the weights are applied to vertices, not edges, there is a known faster algorithm (also discussed in the appendix of the full version):

▶ **Theorem 4.** *For points s and t, and a weighted set of obstacles $\mathcal{C}$, the (s,t) point-separation problem can be solved in $\widetilde{O}(|\mathcal{C}|^{(3+\omega)/2})$ time in the oracle model, where $2 \le \omega < 2.371339$ is the matrix multiplication exponent.*

Proof. Apply the same reduction to APSP as before, but use the algorithm of Abboud, Fischer, Jin, Williams, and Xi [1] for APSP with real vertex weights. ◀

3.2 General Unweighted Obstacles in Oracle Model

We are also interested in the (s,t) point-separation problem for unit weights, which corresponds to shortest paths for unit weights. In this case, we can obtain faster algorithms:

▶ **Theorem 5.** *For points s and t, and an* unweighted *set of obstacles $\mathcal{C}$, the (s,t) point-separation problem can be solved in $O(|\mathcal{C}|^\omega \log |\mathcal{C}|)$ time, where $2 \le \omega < 2.371339$ is the matrix multiplication exponent.*

Proof. Compute the intersection graph in some homology cover $\overline{G}$ of $\mathcal{C}$, and then run unweighted undirected APSP [16] on $\overline{G}$, for $O(|\mathcal{C}|^\omega \log |\mathcal{C}|)$ total time. ◀

3.3 General Weighted Obstacles in Arrangement Model

We can also obtain results using the arrangement instead of the intersection graph. The simplest of these is a slight modification of Theorem 3:

▶ **Theorem 6.** *For points s and t, a weighted set of obstacles $\mathcal{C}$, and an arrangement D, σ of $\mathcal{C}$, where σ is given as lists of vertices, let $m = \sum_{c \in \sigma} |c|$ be the total number of obstacle-vertex incidences. Then, the (s,t) point-separation problem can be solved in $O(\min(|\mathcal{C}|, |V(D)|) \cdot m + \min(|\mathcal{C}|, |V(D)|)^2 \log \min(|\mathcal{C}|, |V(D)|))$ time.*

Proof. Let $\overline{H}$ be the auxiliary graph in the homology cover, which has $O(m)$ vertices and edges. By a result in the appendix of the full version, it suffices to do one of the following:

- Compute the single-source shortest-path (SSSP) tree from each obstacle in $\overline{H}$.
- Compute the SSSP tree from each arrangement vertex in $\overline{H}$.

Let $k = \min(|\mathcal{C}|, |V(D)|)$. Each SSSP computation can be performed in $O(m + k\log k)$ time, and k such computations suffice to solve the problem. ◀

This theorem in particular is of note because as we will see later, it is essentially optimal assuming the APSP conjecture, at least in the case when $|\mathcal{C}| = \Theta(|V(D)|^2)$ and $m = \Theta(|\mathcal{C}|)$, as we will see in Section 4.

3.4 Restricted Obstacle Classes without Weights

As we will discuss in the appendix of the full version, Chan and Skrepetos [6, 7] studied APSP for several forms of unweighted/undirected geometric intersection graphs. Their intersection graphs are in the plane, but with some extra work it is possible to study intersection graphs in the homology cover, and consequently the unweighted (s, t) point-separation problem.

▶ **Theorem 7.** *For points s and t, and an* unweighted *set of obstacles $\mathcal{C}$. Let $\overline{\mathcal{C}}$ be the set of $2|\mathcal{C}|$ "mapped" obstacles in the homology cover. Let $SI(n, m)$ ("static intersection") be the time complexity for checking if each of n different objects in $\overline{\mathcal{C}}$ intersects any object in some subset $C \subset \overline{\mathcal{C}}$ of size $m = |C|$. Assume $SI(n, m)$ is super-additive, so that $SI(n_1, m_1) + SI(n_2, m_2) \leq SI(n_1 + n_2, m_1 + m_2)$. Then (s, t) point-separation over $\mathcal{C}$ can be solved in $O(nSI(n, n))$ time.*

Proof. Run APSP for the intersection graph in the homology cover via the algorithm of Chan [10, Theorem 2] (see further discussion in the appendix of the full version). ◀

To use this result, we need efficient algorithms for static intersection in the homology cover:

▶ **Lemma 8.** *For points s and t, the following values of $SI(n, n)$ hold for restricted obstacle types in the homology cover:*

Obstacle Class	$SI(n, n)$
General Disks	$O(n \log n)$
Axis-Aligned Line Segments	$O(n \log\log n)$
Arbitrary Line Segments	$O(n^{4/3} \log^{1/3} n)$

The proof of the lemma is left to the appendix of the full version. At a high-level, all cases are first reduced to the planar static intersection problem. In the case of line segments, this becomes the standard planar static intersection problem for line segments. For disks, this is not the case, and the algorithm is more involved.

The combination of these two results give an important corollary:

▶ **Corollary 9.** *For points s and t, the unweighted (s, t) point-separation problem can be solved in $O(n^2 \log\log n)$ time for axis-aligned line segments (or $O(1)$-length rectilinear polylines), $O(n^{7/3} \log^{1/3} n)$ time for line segments (or $O(1)$-length polylines), and $O(n^2 \log n)$ time for general disks or circles that do not contain both s and t.*

3.5 Restricted Obstacle Classes with Weights

We will now present a method for solving the weighted (s, t) point-separation problem using a tool called "biclique covers", which are essentially a tool for a type of graph sparsification.

For a graph $G = (V, E)$, a **biclique** in G is a complete bipartite subgraph ($A \times B \subset E$, where $A, B \subset V$ are disjoint). The **size** of a biclique is the number of vertices it contains ($|A| + |B|$). A **biclique cover** is a collection of bicliques in G covering the edges E, and its **size** is the the sum of all sizes in its bicliques. Biclique covers of geometric intersection graphs in two-dimensions are well-studied. We summarize known results in Table 2.

Table 2 Known results for biclique covers of 2D geometric intersection graphs. All of these results are stated and proven by Chan [8], although essentially all of the steps have appeared in a number of prior works. The notation $\widetilde{O}$ hides logarithmic factors. The notation $\widetilde{O}_k$ further assumes that k is constant. Note that axis-aligned line segment intersection graphs are K_3-free.

Graph Type	Cover Size	Construction Time
line segment intersection	$\widetilde{O}(n^{4/3})$	$\widetilde{O}(n^{4/3})$
Axis-aligned line seg. intersection	$\widetilde{O}(n)$	$\widetilde{O}(n)$
k-clique-free line seg. intersection	$\widetilde{O}_k(n)$	$\widetilde{O}_k(n)$

Biclique covers are useful in our case because they can be used for faster *vertex-weighted* shortest paths. In particular:

▶ **Lemma 10.** *Let $G = (V, E)$ be an n-vertex (undirected) graph with vertex-weights admitting a biclique cover of size $S(n)$ that can be constructed in $T(n)$ time. Then APSP over G can be solved in $O(T(n) + n \cdot S(n) \log(n))$ time.*

We leave the proof to the appendix of the full version.

These results aren't quite enough for the (s, t) point-separation problem, since what we need is actually a biclique cover *in the homology cover*. Fortunately, we can construct this:

▶ **Lemma 11.** *Let s and t be designated points in the plane, and let $\mathcal{C}$ be a set of line segments with $n = |\mathcal{C}|$. Let $\overline{G}$ be the intersection graph in the homology cover, and let G be the intersection graph in the plane. Then $\overline{G}$ has a homology cover of size $\widetilde{O}(n^{4/3})$ that can be found in $\widetilde{O}(n^{4/3})$ time. Moreover, if G contains no k-clique (including if $\mathcal{C}$ is a set of rectilinear segments, in which case G contains no 3-clique), then $\overline{G}$ has a homology cover of size $\widetilde{O}_k(n)$ that can be found in $\widetilde{O}_k(n)$ time.*

We leave the proof of this to the appendix of the full version as well. The proof is similar to that of Theorem 7.

The combination of these results gives the following theorem:

▶ **Theorem 12.** *For a weighted set of obstacles $\mathcal{C}$, and points s, t, the (s, t) point-separation can be solved in $\widetilde{O}(n^{7/3})$ time if $\mathcal{C}$ is a set of line segments, $\widetilde{O}(n^2)$ time if $\mathcal{C}$ is a set of axis-aligned line segments, and $\widetilde{O}_k(n^2)$ time if $\mathcal{C}$ is a set of line segments whose intersection graph has no k-clique. The same bounds hold if each obstacle is an $O(1)$-length polyline among lines of the same properties.*

Note that a biclique cover for $O(1)$-length polylines can be recovered from biclique covers over the individual line segments (adjusted slightly so that the line segments in the same polyline do not overlap).

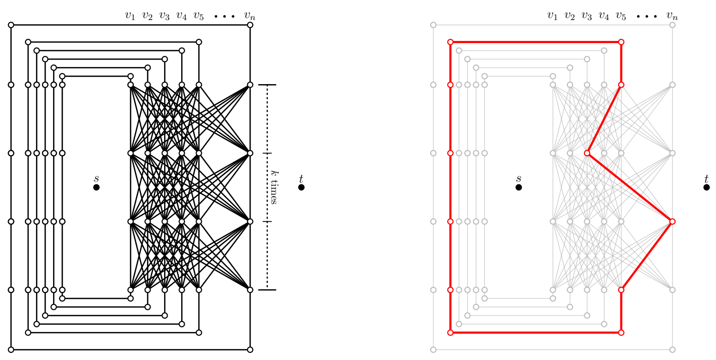

v_1 v_2 v_3 v_4 v_5 $\cdots$ v_n
s
t
k times
v_1 v_2 v_3 v_4 v_5 $\cdots$ v_n
s
t

Figure 12 A demonstration of the sub-cubic reduction from min-weight k-walk to the (s, t) point-separation problem with line segments. Empty circles are used to annotate the ends segments.

4 Lower Bounds

In this section, we present several related fine-grained lower bounds for specific cases of the (s, t) point-separation problem. The main intermediate tool is the problem of finding a minimum-weight walk of length k in a directed graph, for a given k (or detecting if any walk of length k exists). All of our lower bounds are based on the following unified theorem:

▶ **Theorem 13.** *For a positively edge-weighted directed graph $G = (V, E)$ with n vertices and m edges with maximum edge-weight W, and an integer k, there exists three sets $\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3$ of $2km + 6m$ obstacles, each with a weight equal to that of some edge in G, so that each of the following properties holds for one set:*

- *All obstacles are line segments.*
- *All obstacles are length-2 polylines, and the total number of unique intersection points of the obstacles is $k + 2(k + 1)m + 4m = \Theta(km)$.*
- *All obstacles are length-3 rectilinear polylines.*

Moreover, there are points s and t in the plane so that G has a walk of length k with weight at most w if and only if there is some subset of $\mathcal{C}_i$ (for any $i \in \{1, 2, 3\}$) that separates s and t and has weight at most $w + (k + 6)W$, so long as $w \leq kW$. Furthermore, each of $\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3$ can be constructed in time proportional to their sizes.

Proof. For each property, we will construct a reduction from min-weight k-walk (for fixed k) in a directed edge-weighted graph to point separation, with the stated guarantees. The constructions for each property are essentially the same. We will focus on the line segment case, and explain the modifications afterwards.

The construction is visualized in Figure 12. For each vertex v_i ($i \in \{1, \ldots, n\}$), assign the column $x = i$. For each value $r \in [k + 1]$, assign the row $y = r$. For each directed edge $(v_i, v_j) \in E$, with weight w_{v_i,v_j} create k line segments, each also with weight w_{v_i,v_j}. Each should go from the ith column to the jth column, and the rth row to the $(r + 1)$th row (for each $r \in [k + 1]$). We call these the **edge segments**. Pick s to be some point to the left of the columns, and t to be some point to the right. For the ith column, create a set of $k + 6$ rectilinear line segments connecting the top point of the ith column to the bottom, so that the path formed by these line segments lies to the left of s at its x coordinate, ensuring that these line segments do not intersect the corresponding line segments generated for other columns, nor do they cross the minimal orthogonal rectangle containing the previous set of line segments. We call these the **vertex segments**. These two segment types form the entirety of the obstacles, so the stated time complexity follows. We need only prove correctness of the reduction.

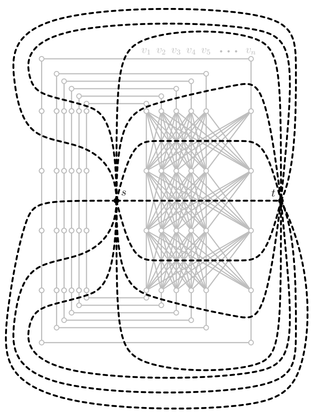


Figure 13 The finite set of (s,t) paths that need to be considered for the reduction of min-weight k-walk to the (s,t) point-separation problem with line segments.

A k-walk through G starting and ending at a vertex $v_i \in V$ corresponds exactly to a connected set of edge segments from the point $(i,0)$ to (i,k), and both also have the same weight. The forward direction of the reduction follows: Given such a set of edge segments, a set of vertex segments of weight exactly $(k+6)W$ exists (those for the ith column) so that the union of the two separates s and t, and has the specified weight. For the backwards direction of the reduction: For a set $C \subset \mathcal{C}$ with weight at most $w + (k+6)W$, if $w \leq kW$, then it is only possible to have one maximal connected set of vertex segments in C. Each of the sets of (potential) obstacles crossed by paths in Figure 13 must have at least one obstacle chosen from them for the whole set to separate s and t, so at least one such maximal connected set of vertex segments must be in C. Hence, the total weight of the remaining obstacles in C is w, and some subset of these obstacles must themselves be a connected set of edge segments (one per row) corresponding exactly to a k-walk in G.

The two modified constructions with polylines are obtained by replacing the edge segments with polylines that have the desired properties. These cases are visualized in Figure 14. ◀

There are a number of fine-grained lower bounds for min-weight k-walk and k-walk detection. In particular, all such bounds hold for *fixed* k and graphs where k-walks and k-cycles coincide (which is itself always true for $k = 3$ in graphs with no self-loops, and is otherwise implied by a property called "k-circle-layered"). In the appendix of the full version, we discuss the following results that follow from known fine-grained bounds and Theorem 13:

If (s,t) point-separation can be solved in time…	for n obstacles that are…	then it would imply a new state-of-the-art algorithm for…
$O(n^{3/2-o(1)})$	weighted line segments	edge-weighted APSP
$O(n^{2-o(1)})$	weighted line segments	minimum-weight k-clique
$O(n^{2-o(1)})$	weighted length-3 rectilinear polylines	minimum-weight k-clique
$O(n^{3/2-o(1)})$	unweighted line segments	max-3-SAT
$O(n^{3/2-o(1)})$	unweighted length-3 rectilinear polylines	max-3-SAT
$O(n^{3/2-o(1)})$	weighted length-2 polylines with $O(\sqrt{n})$ unique intersection points and 3 intersection points per obstacle	edge-weighted APSP

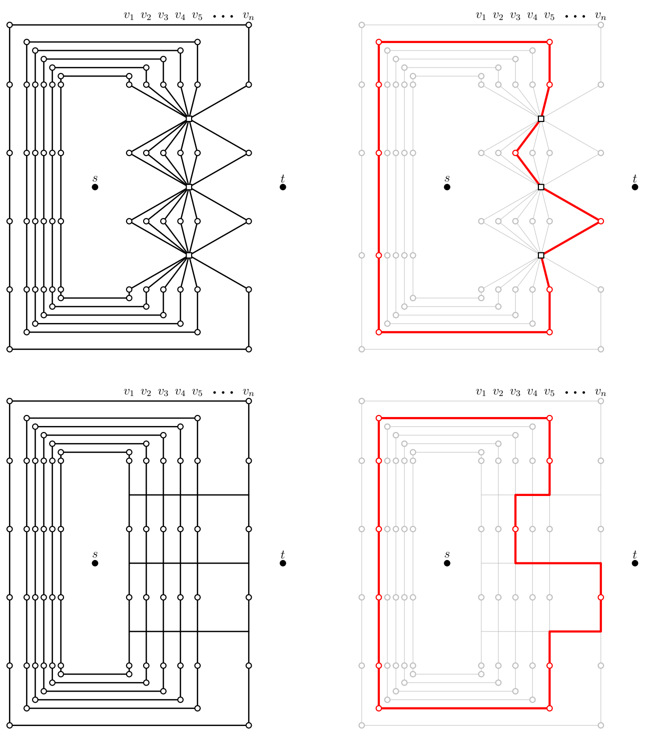


Figure 14 A demonstration of the reduction from min-weight k-walk to the (s, t) point-separation problem for length-2 polylines with few total intersection points, or length-3-rectilinear polylines.

5 Conclusion

We have discussed several upper and lower bounds for the (s, t) point-separation problem in various cases, and we conclude by briefly listing some of the most interesting open problems:

- Is there a near-quadratic algorithm for general weighted obstacles? Can a conditional lower bound (based on any popular hypotheses) excluding this possibility be formulated?
- Can a (truly) sub-quadratic algorithm be devised for disks, unit disks, or axis-aligned segments? Is there a non-trivial fine-grained lower bound in any of these cases?
- Are there faster algorithms or stronger lower bounds for the *unweighted* versions of the problem?

We note that it seems very likely that unit disks would admit a *slightly* sub-quadratic algorithm, since APSP is known to be solvable for unit disk graphs in (essentially) slightly sub-quadratic time [9], and the methods seem as though this method could plausibly be extended to the homology cover.

References

1 Amir Abboud, Nick Fischer, Ce Jin, Virginia Vassilevska Williams, and Zoe Xi. All-pairs shortest paths with few weights per node. In *Proceedings of the 57th Annual ACM Symposium on Theory of Computing*, pages 1956–1964, 2025. doi:10.1145/3717823.3718240.

2 Josh Alman, Ran Duan, Virginia Vassilevska Williams, Yinzhan Xu, Zixuan Xu, and Renfei Zhou. More asymmetry yields faster matrix multiplication. In Yossi Azar and Debmalya Panigrahi, editors, *Proceedings of the 2025 Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2025, New Orleans, LA, USA, January 12-15, 2025*, pages 2005–2039. SIAM, 2025. doi:10.1137/1.9781611978322.63.

3 Sergio Cabello and Panos Giannopoulos. The complexity of separating points in the plane. *Algorithmica*, 74(2):643–663, 2016. doi:10.1007/s00453-014-9965-6.

4 Sergio Cabello and Lazar Milinković. Two optimization problems for unit disks. *Comput. Geom.*, 70-71:1–12, 2018. doi:10.1016/j.comgeo.2017.12.001.

5 Erin W. Chambers, Jeff Erickson, Kyle Fox, and Amir Nayyeri. Minimum cuts in surface graphs. *SIAM J. Comput.*, 52(1):156–195, 2023. doi:10.1137/19M1291820.

6 Timothy M. Chan. More algorithms for all-pairs shortest paths in weighted graphs. In David S. Johnson and Uriel Feige, editors, *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007*, pages 590–598. ACM, 2007. doi:10.1145/1250790.1250877.

7 Timothy M. Chan. More algorithms for all-pairs shortest paths in weighted graphs. *SIAM J. Comput.*, 39(5):2075–2089, 2010. doi:10.1137/08071990X.

8 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. doi:10.1137/1.9781611977554.CH68.

9 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. doi:10.4230/LIPIcs.ISAAC.2016.24.

10 Timothy M. Chan and Dimitrios Skrepetos. All-pairs shortest paths in geometric intersection graphs. *J. Comput. Geom.*, 10(1):27–41, 2019. doi:10.20382/JOCG.V10I1A2.

11 Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2022.

12 Mark de Berg, Otfried Cheong, Marc J. van Kreveld, and Mark H. Overmars. *Computational geometry: algorithms and applications, 3rd Edition*. Springer, 2008. doi:10.1007/978-3-540-77974-2.

13 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. doi:10.1137/1.9781611976465.52.

14 Maarten Löffler, Tim Ophelders, Rodrigo I. Silveira, and Frank Staals. Shortest Paths in Portalgons. In Erin W. Chambers and Joachim Gudmundsson, editors, *39th International Symposium on Computational Geometry (SoCG 2023)*, volume 258 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 48:1–48:16, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SoCG.2023.48.

15 Tim Ophelders, Maarten Löffler, Rodrigo I Silveira, and Frank Staals. Shortest paths in portalgons. *Journal of Computational Geometry*, 15(2):174–221, 2024.

16 Raimund Seidel. On the all-pairs-shortest-path problem in unweighted undirected graphs. *J. Comput. Syst. Sci.*, 51(3):400–403, 1995. doi:10.1006/JCSS.1995.1078.

17 Jack Spalding-Jamieson and Anurag Murty Naredla. Separating two points with obstacles in the plane: Improved upper and lower bounds, 2025. doi:10.48550/arXiv.2504.17289.