Point Separation and Obstacle Removal by Finding and Hitting Odd Cycles

Neeraj Kumar
Department of Computer Science, University of California, Santa Barbara, CA, USA

Daniel Lokshtanov
Department of Computer Science, University of California, Santa Barbara, CA, USA

Saket Saurabh
Institute of Mathematical Sciences, Chennai, India
University of Bergen, Norway

Subhash Suri
Department of Computer Science, University of California, Santa Barbara, CA, USA

Jie Xue
New York University Shanghai, China

Abstract

Suppose we are given a pair of points s, t and a set $\mathcal{S}$ of n geometric objects in the plane, called obstacles. We show that in polynomial time one can construct an auxiliary (multi-)graph G with vertex set $\mathcal{S}$ and every edge labeled from $\{0, 1\}$, such that a set $\mathcal{S}_d \subseteq \mathcal{S}$ of obstacles separates s from t if and only if $G[\mathcal{S}_d]$ contains a cycle whose sum of labels is odd. Using this structural characterization of separating sets of obstacles we obtain the following algorithmic results.

In the Obstacle-removal problem the task is to find a curve in the plane connecting s to t intersecting at most q obstacles. We give a $2.3146^q n^{O(1)}$ algorithm for Obstacle-removal, significantly improving upon the previously best known $q^{O(q^3)} n^{O(1)}$ algorithm of Eiben and Lokshtanov (SoCG'20). We also obtain an alternative proof of a constant factor approximation algorithm for Obstacle-removal, substantially simplifying the arguments of Kumar et al. (SODA'21).

In the Generalized Points-separation problem input consists of the set $\mathcal{S}$ of obstacles, a point set A of k points and p pairs $(s_1, t_1), \ldots (s_p, t_p)$ of points from A. The task is to find a minimum subset $\mathcal{S}_r \subseteq \mathcal{S}$ such that for every i, every curve from s_i to t_i intersects at least one obstacle in $\mathcal{S}_r$. We obtain $2^{O(p)} n^{O(k)}$-time algorithm for Generalized Points-separation. This resolves an open problem of Cabello and Giannopoulos (SoCG'13), who asked about the existence of such an algorithm for the special case where $(s_1, t_1), \ldots (s_p, t_p)$ contains all the pairs of points in A. Finally, we improve the running time of our algorithm to $f(p, k) \cdot n^{O(\sqrt{k})}$ when the obstacles are unit disks, where $f(p, k) = 2^{O(p)} k^{O(k)}$, and show that, assuming the Exponential Time Hypothesis (ETH), the running time dependence on k of our algorithms is essentially optimal.

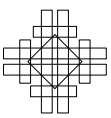
2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases points-separation, min color path, constraint removal, barrier resillience

Digital Object Identifier 10.4230/LIPIcs.SoCG.2022.52

Related Version *Full Version*: https://arxiv.org/abs/2203.08193

Funding *Daniel Lokshtanov*: BSF award 2018302 and NSF award CCF-2008838
Saket Saurabh: European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 819416), and Swarnajayanti Fellowship (No. DST/SJF/MSA01/2017-18).
Subhash Suri: NSF award CCF-1814172



1 Introduction

Suppose we are given a set $\mathcal{S}$ of geometric objects in the plane, and we want to modify $\mathcal{S}$ in order to achieve certain guarantees on coverage of paths between a given set A of points. Such problems have received significant interest in sensor networks [2, 4, 6, 16], robotics [9, 12] and computational geometry [3, 8, 11]. There have been two closely related lines of work on this topic: **(i)** *remove* a smallest number of obstacles from $\mathcal{S}$ to satisfy *reachability* requirements for points in A, and **(ii)** *retain* a smallest number of obstacles to satisfy *separation* requirements for points in A.

In the most basic version of these problems the set A consists of just two points s and t. Specifically, in Obstacle-removal the task is to find a smallest possible set $\mathcal{S}_d \subseteq \mathcal{S}$ such that there is a curve from s to t in the plane avoiding all obstacles in $\mathcal{S} \setminus \mathcal{S}_d$. In 2-Points-separation the task is to find a smallest set $\mathcal{S}_r \subseteq \mathcal{S}$ such that every curve from s to t in the plane intersects at least one obstacle in $\mathcal{S}_r$. It is quite natural to require the obstacles in the set $\mathcal{S}$ to be connected. Indeed, removing the connectivity requirements results in problems that are computationally intractable [8, 10, 21].

When the obstacles are required to be connected Obstacle-removal remains NP-hard, but becomes more tractable from the perspective of approximation algorithms and parameterized algorithms. For approximation algorithms, Bereg and Kirkpatrick [4] designed a constant factor approximation for unit disk obstacles. Chan and Kirkpatrick [6, 7] improved the approximation factor for unit disk obstacles. Korman et al. [14] obtained a $(1+\epsilon)$-approximation algorithm for the case when obstacles are fat, similarly sized, and no point in the plane is contained in more than a constant number of obstacles. Whether a constant factor approximation exists for general obstacles was posed repeatedly as an open problem [3, 6, 7] before it was resolved in the affirmative by a subset of the authors of this article [21].

For parameterized algorithms, Korman et al. [14] designed an algorithm for Obstacle-removal with running time $f(q)n^{O(1)}$ for determining whether there exists a solution $\mathcal{S}_d$ of size at most q, when obstacles are fat, similarly sized, and no point in the plane is contained in more than a constant number of obstacles. Eiben and Kanj [8, 10] generalized the result of Korman et al. [14], and posed as an open problem the existence of a $f(q)n^{O(1)}$ time algorithm for Obstacle-removal with general connected obtacles. Eiben and Lokshtanov [11] resolved this problem in the affirmative, providing an algorithm with running time $q^{O(q^3)}n^{O(1)}$.

Like Obstacle-removal, the 2-Points-separation problem becomes more tractable when the obstacles are connected. Cabello and Giannopoulos [5] showed that 2-Points-separation with connected obstacles is polynomial time solvable. They show that the more general Points-separation problem where we are given a point set A and asked to find a minimum size set $\mathcal{S}_r \subseteq \mathcal{S}$ that separates every pair of points in A, is NP-complete, even when all obstacles are unit disks. They leave as an open problem to determine the existence of $f(k)n^{O(1)}$ and $f(k)n^{g(k)}$ time algorithms for Points-separation, where $k = |A|$.

Our Results and Techniques

Our main result is a structural characterization of separating sets of obstacles in terms of odd cycles in an auxiliary graph.

▶ **Theorem 1.** *There exists a polynomial time algorithm that takes as input a set $\mathcal{S}$ of obstacles in the plane, two points s and t, and outputs a (multi-)graph G with vertex set $\mathcal{S}$ and every edge labeled from $\{0,1\}$, such that a set $\mathcal{S}_d \subseteq \mathcal{S}$ of obstacles separates s from t if and only if $G[\mathcal{S}_d]$ contains a cycle whose sum of labels is odd.*

The proof of Theorem 1 is an application of the well known fact that a closed curve separates s from t if and only if it crosses a curve from s to t an odd number of times. Theorem 1 allows us to re-prove, improve, and generalize a number of results for Obstacle-removal, 2-Points-separation and Points-separation in a remarkably simple way. More concretely, we obtain the following results.

- *There exists a polynomial time algorithm for* 2-Points-separation.

Here is the proof: construct the graph G from Theorem 1 and find the shortest odd cycle, which is easy to do in polynomial time. This re-proves the main result of Cabello and Giannopoulos [5]. Next we turn to Obstacle-removal, and obtain an improved parameterized algorithm and simplified approximation algorithms.

- *There exists an algorithm for* Obstacle-removal *that determines whether there exists a solution size set $\mathcal{S}$ of size at most q in time* $2.3146^q n^{O(1)}$.

Here is a proof sketch: construct the graph G from Theorem 1 and determine whether there exists a subset $\mathcal{S}_d$ of $\mathcal{S}$ of size at most q such that $G - \mathcal{S}_d$ does not have any odd label cycle. This can be done in time $2.3146^q n^{O(1)}$ using the algorithm of Lokshtanov et al. [18] for Odd Cycle Transversal.[1] This parameterized algorithm improves over the previously best known parameterized algorithm for Obstacle-removal of Eiben and Lokshtanov [11] with running time $q^{O(q^3)} n^{O(1)}$.

If we run an approximation algorithm for Odd Cycle Transversal on G instead of a parameterized algorithm, we immediately obtain an approximation algorithm for Obstacle-removal with the same ratio. Thus, the $O(\sqrt{\log n})$-approximation algorithm for Odd Cycle Transversal [1, 15] implies a $O(\sqrt{\log n})$-approximation algorithm for Obstacle-removal as well. Going a little deeper we observe that the structure of G implies that the standard Linear Programming relaxation of Odd Cycle Transversal on G only has a constant integrality gap. This yields a constant factor approximation for Obstacle-removal, substantially simplifying the approximation algorithm of Kumar et al [21].

- *There exists a a constant factor approximation for* Obstacle-removal.

Finally we turn our attention back to a generalization of Points-separation, called Generalized Points-separation. Here, instead of separating all k points in A from each other, we are only required to separate p specific pairs $(s_1, t_1), \ldots, (s_p, t_p)$ of points in A (which are specified in the input). We apply Theorem 1 several times, each time with the same obstacle set $\mathcal{S}$, but with a different pair (s_i, t_i). Let G_i be the graph resulting from the construction with the pair (s_i, t_i). Finding a minimum size set $\mathcal{S}_r$ of obstacles that separates s_i from t_i for every i now amounts to finding a minimum size set $\mathcal{S}_r$ such that $G_i[\mathcal{S}_r]$ contains an odd label cycle for every i. The graph in the construction of Theorem 1 does not depend on the points (s_i, t_i) - only the labels of the edges do. Thus $G_1, \ldots, G_p$ are copies of the same graph G, but with p different edge labelings. Our task now is to find a subgraph of G on the minimum number of vertices, such that the subgraph contains an odd labeled cycle with respect to each one of the p labels. We show that such a subgraph has at most $O(p)$ vertices of degree at least 3 and use this to obtain a $2^{O(p^2)} n^{O(p)}$ time algorithm

[1] The only reason this is a proof sketch rather than a proof is that the algorithm of Lokshtanov et al. [18] works for unlabeled graphs, while G has edges with labels 0 or 1. This difference can be worked out using a well-known and simple trick of subdividing every edge with label 0 (see Section 4).

for Generalized Points-separation. This implies a $2^{O(k^4)}n^{O(k^2)}$ time algorithm for Points-separation, resolving the open problem of Cabello and Giannopoulos [5]. With additional technical effort we are able to bring down the running time of our algorithm for Generalized Points-separation to $2^{O(p)}n^{O(k)}$. This turns out to be close to the best one can do. On the other hand, for *pseudo-disk* obstacles we can get a faster algorithm.

- *There exists a $2^{O(p)}n^{O(k)}$ time algorithm for Generalized Points-separation, and a $n^{O(\sqrt{k})}$ time algorithm for Generalized Points-separation with pseudo-disk obstacles.*
- *A $f(k)n^{o(k/\log k)}$ time algorithm for Points-separation, or a $f(k)n^{o(\sqrt{k})}$ time algorithm for Points-separation with pseudo-disk obstacles would violate the ETH [13].*

2 Preliminaries

All graphs used in this paper are undirected. It will also be more convenient to sometimes consider multi-graphs, in which self-loops and parallel edges are allowed. The *degree* of a vertex is the number of adjacent edges.

The *arrangement* $\mathsf{Arr}(\mathcal{S})$ of a set of obstacles $\mathcal{S}$ is a subdivision of the plane induced by the boundaries of the obstacles in $\mathcal{S}$. The faces of $\mathsf{Arr}(\mathcal{S})$ are connected regions and edges are parts of obstacle boundaries. The *arrangement graph* $G_{\mathsf{Arr}} = (V, E)$ is the dual graph of the arrangement whose vertices are faces of $\mathsf{Arr}(\mathcal{S})$ and edges connect neighboring faces. The complexity of the arrangement is the size of its arrangement graph which we denote by $|\mathsf{Arr}(\mathcal{S})|$. We assume that the size of the arrangement is polynomial in the number of obstacles, that is $|\mathsf{Arr}(\mathcal{S})| = |G_{\mathsf{Arr}}| = n^{O(1)}$. This is indeed true for most reasonable obstacle models such as polygons or low-degree splines.

Plane curves and Crossings. A *plane curve* (or simply *curve*) is specified by a continuous function $\pi : [0, 1] \to \mathbb{R}^2$, where the points $\pi(0)$ and $\pi(1)$ are called the *endpoints* (we also use the notation π to denote the image of the path function π). A curve is *simple* if it is injective, and is *closed* if its two endpoints are the same. We say a curve π *separates* a pair (a, b) of two points in $\mathbb{R}^2$ if a and b belong to different connected components of $\mathbb{R}^2 \backslash \pi$.

A *crossing* of π with π' is an element of the set $\{t \in [0, 1] \mid \pi(t) \in \pi'\}$. We will often be concerned with the *number* of times π crosses π'. This is defined as $|\{t \in [0, 1] \mid \pi(t) \in \pi'\}|$. Whenever we count the number of times a curve π crosses another curve π' we shall assume that (and ensure that) $|\{t \in [0, 1] \mid \pi(t) \in \pi'\}|$ is finite and that π and π' are *transverse*. That is for every $t \in [0, 1]$ such that $\pi(t) \in \pi'$ there exists an $\epsilon > 0$ such that the intersection of $\pi \cup \pi'$ with an ϵ radius ball around $\pi(t)$ is homotopic with two orthogonal lines. We will make frequent use of the following basic topological fact.

▶ **Fact 2.** *Let π be a curve with endpoints $a, b \in \mathbb{R}^2$. We have that* (i) *A* simple closed curve *γ separates (a, b) iff π crosses γ an odd number of times.* (ii) *If π crosses a* closed curve *γ an odd number of times, then γ separates (a, b).*

3 Labeled Intersection Graph of Obstacles

We begin by describing the construction of the labeled intersection graph $G_{\mathcal{S}} = (\mathcal{S}, X)$ of the obstacles $\mathcal{S}$. For the ease of exposition, we will use S to refer to the obstacle $S \in \mathcal{S}$ as well as the vertex for S in $G_{\mathcal{S}}$ interchangeably.

Constructing the graph $G_{\mathcal{S}}$. For every obstacle $S \in \mathcal{S}$ we first select an arbitrary point $\mathsf{ref}(S) \in S$ and designate it to be the *reference point* of the obstacle. Next, we select the *reference curve* π to be a simple curve in the plane connecting s and t such that including it to the arrangement $\mathsf{Arr}(\mathcal{S})$ does not significantly increase its complexity. That is, we want to ensure that $|\mathsf{Arr}(\mathcal{S} \cup \pi)| = O(|\mathsf{Arr}(\mathcal{S})|)$. Additionally, the reference curve π is chosen such that there exists an $\epsilon > 0$ and π is disjoint from an ϵ ball around every intersection point of two obstacles in $\mathsf{Arr}(\mathcal{S})$ and from an ϵ ball around every reference point $\mathsf{ref}(S)$ for $S \in \mathcal{S}$.

As long as the intersection of every pair of obstacles is finite and their arrangement has bounded size, a suitable choice for π always exists (and can be efficiently computed). For example one can choose π to be the plane curve corresponding to an s–t path in G_{Arr}.

We will now add edges to $G_{\mathcal{S}}$ as follows. (See also Figure 1(c) for an example.)

- For every obstacle $S \in \mathcal{S}$ that contains s or t, add a self-loop $e = (S, S)$ with $\mathsf{lab}(e) = 1$.
- For every pair of obstacles $S, S' \in \mathcal{S}$ that intersect, we add edges to G as follows.
  - Add an edge $e_0 = (S, S')$ with $\mathsf{lab}(e_0) = 0$ if there exists a curve connecting $\mathsf{ref}(S)$ and $\mathsf{ref}(S')$ contained in the region $S \cup S'$ that crosses π an *even* number of times.
  - Add an edge $e_1 = (S, S')$ with $\mathsf{lab}(e_1) = 1$ if there exists a curve connecting $\mathsf{ref}(S)$ and $\mathsf{ref}(S')$ contained in the region $S \cup S'$ that crosses π an *odd* number of times.

Checking whether there exists a curve contained in the region $S \cup S'$ with endpoints $\mathsf{ref}(S)$ and $\mathsf{ref}(S')$ that crosses π an odd (resp. even) number of times can be done in time linear in the size of arrangement $\mathsf{Arr}' = \mathsf{Arr}(S \cup S' \cup \pi)$. Specifically, we build the arrangement graph $G_{\mathsf{Arr}'}$ and only retain edges (f_i, f_j) such that the faces $f_i, f_j \in S \cup S'$. If the common boundary of faces f_i, f_j is a portion of π, we assign a label 1 to the edge (f_i, f_j), otherwise we assign it a label 0. An odd (resp. even) labeled walk in $G_{\mathsf{Arr}'}$ connecting the faces containing $\mathsf{ref}(S)$ and $\mathsf{ref}(S')$ gives us the desired plane curve π_{ij}. Since edges of $G_{\mathsf{Arr}'}$ connect adjacent faces of Arr', we can ensure that the intersections between curve π_{ij} and the edges of arrangement (including parts of reference curve π) are all transverse.

We are now ready to prove the following important structural property of the graph $G_{\mathcal{S}}$.

▶ **Lemma 3.** *A set of obstacles $\mathcal{S}' \subseteq S$ in the graph $G_{\mathcal{S}}$ separates the points s and t if and only if the induced graph $H = G_{\mathcal{S}}[\mathcal{S}']$ contains an odd labeled cycle.*

Proof. ($\Rightarrow$) For the forward direction, suppose we are given a set of obstacles $\mathcal{S}'$ that separate s from t. If s or t are contained in some obstacle, then we must have an odd self-loop in $G_{\mathcal{S}}$ and we will be done. Otherwise, assume that s, t lie in the exterior of all obstacles, so we have $s, t \notin \mathcal{R}(\mathcal{S}')$ where $\mathcal{R}(\mathcal{S}') = \bigcup_{S \in \mathcal{S}'} S$ is the region bounded by obstacles in $\mathcal{S}'$. Observe that s, t must lie in different connected regions R_s, R_t of $\mathbb{R}^2 \setminus \mathcal{R}(\mathcal{S}')$ or else the set $\mathcal{S}'$ would not separate them. At least one of R_s or R_t must be bounded, wlog assume it is R_s. Let γ' be the simple closed curve that is the common boundary of $\mathcal{R}(\mathcal{S}')$ and R_s. We have that γ' encloses s but not t and therefore separates s from t. Using first statement of Fact 2, we obtain that γ' crosses the reference curve π an odd number of times. Observe that the curve γ' consists of multiple *sections* $\alpha'_1 \to \alpha'_2 \cdots \to \alpha'_r$ where each curve α'_i is part of the boundary of some obstacle S_i. For each of these curves α'_i, we add a *detour* to and back from the reference point $\mathsf{ref}(S_i)$ of the obstacle it belongs. Specifically, let q_i be an arbitrary point on the curve α'_i and let $\alpha'_{i\ell}, \alpha'_{ir}$ be the portion of α'_i before and after q_i respectively. We add the *detour curve* $\delta_i = q_i \to \mathsf{ref}(S_i) \to q_i$ ensuring that it always stays within the obstacle S_i which is possible because the obstacles are connected. (Same as before the curve δ_i can be chosen to be transverse with π by considering the corresponding walk in graph of $\mathsf{Arr}(S_i \cup \pi)$.) Let $\alpha_i = \alpha'_{i\ell} \to \delta_i \to \alpha'_{ir}$ be the curve obtained by adding detour δ_i to α'_i. Let

$\gamma = \alpha_1 \to \alpha_2 \cdots \to \alpha_r$ be the closed curve obtained by adding these detours to γ'. Note that γ is not necessarily simple as the detour curves may intersect each other. Every detour δ_i consists of identical copies of two curves, so it crosses the reference curve π an even number of times. Since γ' crosses π an odd number of times, the curve γ also crosses π an odd number of times. (See also Figure 1.) Observe that γ and γ' are transverse with π because intersections of π and obstacle boundaries are transverse and the detour curves δ_i are chosen to be transverse with π.

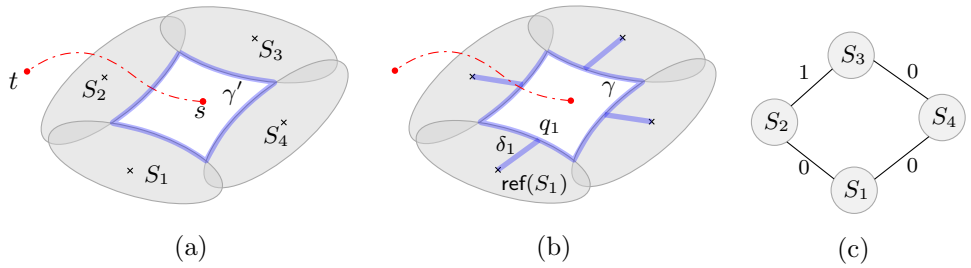


Figure 1 (a) The curve γ' shown shaded in blue is the common boundary of $\mathcal{R}(\mathcal{S}')$ and region R_s (b) Adding detours δ_i to obtain curve γ (c) Labeled Intersection graph $G_{\mathcal{S}}$ ob obstacles.

We will now translate the curve γ to a *walk* in the labeled intersection graph $G_{\mathcal{S}}$. Specifically, consider the section of γ between two consecutive detours: $\gamma_{i,i+1} = \mathsf{ref}(S_i) \to q_i \to q_{i+1} \to \mathsf{ref}(S_{i+1})$. Therefore the obstacles S_i, S_{i+1} must intersect and we have a curve $\gamma_{i,i+1}$ connecting their reference points contained in the region $S_i \cup S_{i+1}$ that also intersects the reference curve π an odd (resp. even) number of times. By construction, $G_{\mathcal{S}}$ must contain an edge $e_{i,i+1}$ with label 1 (resp. 0). By replacing all these sections of γ with the corresponding edges of $G_{\mathcal{S}}$, we obtain an odd-labeled closed walk W in $G_{\mathcal{S}}$. Of all the odd-labeled closed sub-walks of W, we select one that is inclusion minimal. This gives a simple odd-labeled cycle in $G_{\mathcal{S}}[\mathcal{S}']$.

($\Leftarrow$) The reverse direction is relatively simpler. Given an odd-labeled cycle in $G_{\mathcal{S}}[\mathcal{S}']$, we obtain a closed curve γ in the plane contained in region $\mathcal{R}(\mathcal{S}')$ as follows. For every edge $e_i = (S, S')$ of the cycle with label $\mathsf{lab}(e_i)$, we consider the curve γ_i that connects the reference points $\mathsf{ref}(S)$ and $\mathsf{ref}(S')$ contained in $S \cup S'$ and crosses the reference curve π consistent with $\mathsf{lab}(e_i)$. Moreover γ_i needs to be transverse with π. Such a curve exists by construction of $G_{\mathcal{S}}$. Combining these curves γ_i in order gives us a closed curve γ in the plane that crosses π an odd number of times. Although this curve may be self intersecting, from second statement of Fact 2, we have that γ separates s and t. ◀

The construction of the graph $G_{\mathcal{S}}$, together with Lemma 3 prove Theorem 1.

2-Points-separation as Shortest Odd Cycle in $G_{\mathcal{S}}$. From Lemma 3, it follows that a minimum set of obstacles that separates s from t corresponds to an odd-labeled cycle in $G_{\mathcal{S}}$ with fewest vertices. This readily gives a polytime algorithm for 2-Points-separation. In particular, for a fixed starting vertex, we can compute the shortest odd cycle in $G_{\mathcal{S}}$ in $O(|\mathcal{S}|^2)$ time by the following well-known technique. Consider an unlabeled auxiliary graph G' with vertex set is $\mathcal{S} \times \{0, 1\}$. For every edge $e = (S, S')$ of $G_{\mathcal{S}}$, we add edges $\{(S, 0), (S', 0)\}$ and $\{(S, 1), (S', 1)\}$ if $\mathsf{lab}(e) = 0$. Otherwise, we add the edges $\{(S, 0), (S', 1)\}$ and $\{(S, 1), (S', 0)\}$. The shortest odd cycle containing a fixed vertex S is the shortest path in G' between vertices $(S, 0)$ and $(S, 1)$. Repeating over all starting vertices gives the shortest odd cycle in $G_{\mathcal{S}}$. This

can be easily extended for the node-weighted case which gives us the following useful lemma that also yields a polynomial time algorithm for 2-Points-separation, reproving a result of Cabello and Giannopoulos [5].

▶ **Lemma 4.** *There exists a polynomial time algorithm for computing a minimum weight labeled odd cycle in the graph $G_{\mathcal{S}}$.*

Next we prove one more structural property of labeled intersection graph $G_{\mathcal{S}}$ that will be useful later. We define a (labeled) *spanning tree* T of a connected labeled multi-graph $G_{\mathcal{S}}$ to be a subgraph of $G_{\mathcal{S}}$ that is a tree and connects all vertices in $\mathcal{S}$. An edge $e = (u, v) \in G_{\mathcal{S}}$ is a *tree edge* if $(u, v) \in T$, otherwise it is called a *non-tree* edge.

▶ **Lemma 5.** *Let $G_{\mathcal{S}}$ be a connected labeled intersection graph and T be a spanning tree of $G_{\mathcal{S}}$. If $G_{\mathcal{S}}$ contains an odd labeled cycle, then it also contains an odd labeled cycle with exactly one non-tree edge.*

Proof. Let C be an odd cycle in $G_{\mathcal{S}}$ that contains fewest non-tree edges. If C consists of exactly one non-tree edge, we are done. Otherwise, C contains more than one non-tree edge. Let $e = (u, v) \in C$ be a non-tree edge and $C' \subset C$ be the remainder of C without the edge e. Since C is odd labeled, we must have $\mathsf{lab}(C') \neq \mathsf{lab}(e)$.

Let π_{uv} be the unique path connecting u, v in T. This gives us a path π_{uv} with label $\mathsf{lab}(\pi_{uv})$. Recall that $\mathsf{lab}(C') \neq \mathsf{lab}(e)$. We have two cases. *(i)* If $\mathsf{lab}(\pi_{uv}) \neq \mathsf{lab}(e)$, then we obtain an odd labeled cycle $\pi_{uv} \oplus e$ that has one non-tree edge, namely e, and we are done. *(ii)* Otherwise, $\mathsf{lab}(\pi_{uv}) = \mathsf{lab}(e) \neq \mathsf{lab}(C')$. This gives us an odd labeled closed walk $W^* = \pi_{uv} \oplus C'$ which contains one less non-tree edge than C. Let $C^* \subseteq W^*$ be an odd-labeled inclusion minimal closed sub-walk of W^* (one such C^* always exists). Therefore, C^* is an odd-labeled cycle in $G_{\mathcal{S}}$ that has fewer non-tree edges than C. But C was chosen to be an odd labeled cycle with fewest non-tree edges, a contradiction. ◀

The above lemma also gives a simple $O(|S^2|)$ algorithm to *detect* whether there exists an odd label cycle in $G_{\mathcal{S}}$. Specifically, consider an arbitrary spanning tree of T of $G_{\mathcal{S}}$ and for each edge not in T, compare its label with the label of the path connecting its endpoints in T.

▶ **Lemma 6.** *Given a labeled graph $G_{\mathcal{S}}$, there exists an $O(|S^2|)$ time algorithm to detect whether $G_{\mathcal{S}}$ contains an odd labeled cycle.*

4 Application to Obstacle-removal

We will show how to cast Obstacle-removal as a Labeled Odd Cycle Transversal problem on the graph $G_{\mathcal{S}}$. Recall that in Obstacle-removal problem, we want to remove a set $\mathcal{S}_d \subseteq \mathcal{S}$ of obstacles from the input so that s and t are connected in $\mathcal{S} \setminus \mathcal{S}_d$. Equivalently, we want to select a subset $\mathcal{S}_d$ of obstacles such that the complement set $\mathcal{S} \setminus \mathcal{S}_d$ *does not separate* s and t. From Lemma 3, it follows that the obstacles $\mathcal{S} \setminus \mathcal{S}_d$ do not separate s and t if and only if $G_{\mathcal{S}}[\mathcal{S} \setminus \mathcal{S}_d]$ does not contain an odd labeled cycle. This gives us the following important lemma.

▶ **Lemma 7.** *A set of obstacles $\mathcal{S}_d \subseteq \mathcal{S}$ is a solution to Obstacle-removal if and only if the set of vertices $\mathcal{S}_d$ is a solution to Odd Cycle Transversal of $G_{\mathcal{S}}$.*

This allows us to apply the set of existing results for Odd Cycle Transversal to obstacle removal problems. In particular, this readily gives an improved algorithm for Obstacle-removal when parameterized by the solution size (number of removed obstacles). Let

$G^+_{\mathcal{S}}$ denote the graph $G_{\mathcal{S}}$ where every edge e with $\mathsf{lab}(e) = 0$ is subdivided. Clearly an odd-labeled cycle in $G_{\mathcal{S}}$ has odd length in $G^+_{\mathcal{S}}$ and vice versa. Applying the FPT algorithm for Odd Cycle Transversal from [18] on the graph $G^+_{\mathcal{S}}$ gives us the following result.

▶ **Theorem 8.** *There exists a* $2.3146^k n^{O(1)}$ *algorithm for* Obstacle-removal *parameterized by* k*, the number of removed obstacles.*

This also immediately gives us an $O(\sqrt{\log \mathsf{OPT}})$ approximation for Obstacle-removal by using the best known $O(\sqrt{\log \mathsf{OPT}})$-approximation [15] for Odd Cycle Transversal on the graph $G^+_{\mathcal{S}}$. Observe that instances of obstacle removal are special cases of odd cycle transversal, specifically where the graph $G_{\mathcal{S}}$ is an intersection graph of obstacles. By applying known results on *small diameter decomposition* of *region intersection graphs*, Kumar et al. [21] obtained a constant factor approximation for Obstacle-removal. In the next section we present an alternative constant factor approximation algorithm. Although our algorithm follows a similar high level approach of using small diameter decomposition of $G_{\mathcal{S}}$, we give an alternative proof which significantly simplifies the arguments of [21].

Constant Approximation for Obstacle-removal

Our algorithm is based on formulating and rounding a standard LP for labeled odd cycle transversal on a labeled intersection graph $G_{\mathcal{S}}$. Let $0 \leq x_i \leq 1$ be an indicator variable that denotes whether obstacle S_i is included to the solution or not. The LP formulation which will be referred as Hit-odd-cycles-LP can be written as follows:

$$\min \sum_{S_i \in \mathcal{S}} x_i \text{ subject to:}$$
$$\sum_{S_j \in C} x_j \geq 1 \qquad \text{for all odd-labeled cycles } C \in G_{\mathcal{S}}$$

Although this LP has exponentially many constraints, it can be solved in polynomial time using the ellipsoid method with the polynomial time algorithm for minimum weight odd labeled cycle in $G_{\mathcal{S}}$ (Lemma 4) as separation oracle. The next step is to round the fractional solution $\hat{x} = x_1, x_2, \ldots, x_n$ obtained from solving the Hit-odd-cycles-LP. We will need some background on small diameter decomposition of graphs.

Small Diameter Decomposition. Given a graph $G = (V, E)$ and a distance function $d : V \to \mathbb{R}^+$ associated with each vertex, we can define the distance of each edge as $d(e) = d(v) + d(w)$ for every edge $e = (v, w) \in E$. We can then extend the distance function to any pair of vertices $d(u, v)$ as the shortest path distance between u and v in the edge-weighted graph with distance values of edges as edge weights. We use the following result of Lee [17] for the special case of *region intersection graph* over planar graphs.

▶ **Lemma 9.** *Let* $G = (V, E)$ *be a node-weighted intersection graph of connected regions in the plane, then for every* $\Delta > 0$ *there exists a set* $X \subseteq V$ *of* $|X| = O(1/\Delta) \cdot \sum d(v)$ *vertices such that the diameter of* $G - X$ *is at most* Δ *in the metric* d*. Moreover, such a set* X *can be computed in polynomial time.*

For the sake of convenience, we assume that $G_{\mathcal{S}}$ does not contain an obstacle S_i with a self-loop, because if so, we must always include S_i to the solution. Let $G^*_{\mathcal{S}}$ be the underlying unlabeled graph obtained by removing labels and multi-edges from $G_{\mathcal{S}}$. Since $G^*_{\mathcal{S}}$ is simply the intersection graph of connected regions in the plane, it is easy to show that $G^*_{\mathcal{S}}$ is a region intersection graph over a planar graph (See also Lemma 4.1 [21] for more details.)

Algorithm: Hit-Odd-Cycles. With small diameter decomposition for $G^*_{\mathcal{S}}$ in place, the rounding algorithm is really simple.

- Assign distance values to vertices of $G^*_{\mathcal{S}} = (\mathcal{S}, E)$ as $d(S_i) = x_i$, where x_i is the fractional solution obtained from solving Hit-Odd-Cycle-LP.
- Apply Lemma 9 on graph $G^*_{\mathcal{S}}$ with diameter $\Delta = 1/2$. Return the set of vertices X obtained from applying the lemma as solution.

It remains to show that the set $X \subseteq \mathcal{S}$ returned above indeed hits all the odd labeled cycles in $G_{\mathcal{S}}$. Define a ball $\mathcal{B}(c, R) = \{v \in V : d(c, v) < R - d(v)/2\}$ with center c, radius R and distance metric d defined before. Intuitively, $\mathcal{B}(c, R)$ consists of the vertices that lie strictly inside the radius R ball drawn with c as center.

▶ **Lemma 10.** *X hits all odd labeled cycles in $G_{\mathcal{S}}$.*

Proof. The proof is by contradiction. Let C be an odd labeled cycle such that $C \cap X = \emptyset$. Then C must be contained in a single connected component κ of $G_{\mathcal{S}} - X$. Let v_1 be an arbitrary vertex of C and consider a ball $B = \mathcal{B}(v_1, 1/2)$ of radius $1/2$ centered at v_1. We have $\kappa \subseteq B$ due to the choice of diameter Δ. Consider the shortest path tree T of ball B rooted at v_1 using the distance function $d(e)$ in the unlabeled graph $G^*_{\mathcal{S}}$. For every edge $(u, v) \in T$ assign the label $\mathsf{lab}(e)$ of $e = (u, v) \in G_{\mathcal{S}}$. If multiple labeled edges exist between u and v, choose one arbitrarily.

Now consider the induced subgraph $G'_{\mathcal{S}} = G_{\mathcal{S}}[B]$ which is a connected labeled intersection graph of obstacles in the ball B. Moreover, T is a spanning tree of $G'_{\mathcal{S}}$, and $G'_{\mathcal{S}}$ contains an odd-labeled cycle because $\kappa \subseteq G'_{\mathcal{S}}$. Applying Lemma 5 gives us an odd-labeled cycle $C \in G'_{\mathcal{S}}$ that contains exactly one edge $e \notin T$. The cost of this cycle is $\mathsf{cost}(C) < 1/2 + 1/2 = 1$. This contradicts the constraint of Hit-Odd-Cycle-LP corresponding to C. ◀

We conclude with the main result for this section.

▶ **Theorem 11.** *There exists a polynomial time constant factor approximation algorithm for Obstacle-removal.*

5 Generalized Points-separation

So far, we have focused on separating a pair of points s, t in the plane. In this section, we consider the more general problem where we are given a set $\mathcal{S}$ of n obstacles, a set of points A and a set $P = \{(s_1, t_1), \ldots, (s_p, t_p)\}$ of p pairs of points in A which we want to separate. First we show how to extend the labeled intersecting graph $G_{\mathcal{S}}$ to p source-destination pairs and that the optimal solution subgraph $G_{\mathcal{S}}[\mathcal{S}_{OPT}]$ exhibits a "nice" structure. Then we exploit this structure to obtain an $2^{O(p^2)}n^{O(p)}$ exact algorithm for Generalized Points-separation. Since $p = O(k^2)$, this algorithm runs in polynomial time for any fixed k, resolving an open question of [5]. Using a more sophisticated approach, we later show how to improve the running time to $2^{O(p)}n^{O(k)}$.

5.1 A $2^{O(p^2)}n^{O(p)}$ Algorithm

Recall the construction of the labeled intersection graph $G_{\mathcal{S}}$ for a single point pair (s, t) from Section 3. The *label* $\mathsf{lab}(e) \in \{0, 1\}$ of each edge $e \in G_{\mathcal{S}}$ denotes the *parity* of edge e with respect to *reference curve* π connecting s and t. As we generalize the graph $G_{\mathcal{S}} = (\mathcal{S}, E)$ to p point pairs, we extend the label function $\mathsf{lab} : E \to \{0, 1\}^p$ as a p-bit binary string that denotes the parity with respect to reference curve π_i connecting s_i and t_i for all $i \in [p]$. We will use $\mathsf{lab}_i(e)$ to denote the i-th bit of $\mathsf{lab}(e)$.

Generalized Label Intersection Graph.

- For each $(s_i, t_i) \in P$ and each $S \in \mathcal{S}$ that contains at least one of s_i or t_i, we add a self loop e on S with $\mathsf{lab}_i(e) = 1$ and $\mathsf{lab}_j(e) = 0$ for all $j \neq i$.
- For every pair of intersecting obstacles S, S' and a p-bit string $\ell \in \{0,1\}^p$:
  - Let $\Pi = \{\pi_i \mid s_i, t_i \notin S \cup S'\}$ be the set of reference curves that do not have endpoints in $S \cup S'$.
  - We add an edge $e = (S, S')$ with $\mathsf{lab}(e) = \ell$ if there exists a plane curve connecting $\mathsf{ref}(S)$ and $\mathsf{ref}(S')$ contained in $S \cup S'$ that crosses all reference curves $\pi_i \in \Pi$ with parity consistent with label ℓ. That is, the curve crosses π_i and odd (resp. even) number of times if i-th bit of ℓ is 1 (resp. 0).

Similar to the one pair case, we can build an unlabeled graph G' with vertex set $\mathcal{S} \times \{0,1\}^p$ and edges between them based on the arrangement $\mathsf{Arr}(S \cup S' \cup \pi_1 \cup \ldots \pi_p)$. Using this graph, we can obtain the following lemma.

▶ **Lemma 12.** *The generalized label intersection graph $G_{\mathcal{S}}$ with p-bit labels can be constructed in $2^{O(p)}n^{O(1)}$ time.*

Suppose we define $G_{\mathcal{S}}(i)$ to be the image of $G_{\mathcal{S}}$ induced by the labeling $\mathsf{lab}_i : E \to \{0,1\}$. Specifically, we obtain $G_{\mathcal{S}}(i)$ from $G_{\mathcal{S}}$ by replacing label of each edge by the i-th bit $\mathsf{lab}_i(e)$, followed by removing parallel edges that have the same label. Observe that $G_{\mathcal{S}}(i)$ is precisely the graph obtained by applying algorithm from Section 3 with reference curve π_i. We say that a subgraph $G'_{\mathcal{S}} \subseteq G_{\mathcal{S}}$ is *well-behaved* if $G'_{\mathcal{S}}(i)$ contains an odd labeled cycle for all $i \in [p]$. The following lemma can be obtained by applying Lemma 3 for every pair $(s_i, t_i) \in P$.

▶ **Lemma 13.** *A set of obstacles $\mathcal{S}' \subseteq \mathcal{S}$ separate all point pairs in P iff $G_{\mathcal{S}}[\mathcal{S}']$ is well-behaved.*

We will prove the following important property of well-behaved subgraphs of $G_{\mathcal{S}}$.

▶ **Lemma 14.** *Let $G \subseteq G_{\mathcal{S}}$ be an inclusion minimal well-behaved subgraph of $G_{\mathcal{S}}$. Then there exists a set $V_c \subseteq V(G)$ of* connector vertices *such that G consists of the vertex set V_c and a set of K* chains *(path of degree 2 vertices) with endpoints in V_c. Moreover, $|V_c| \leq 4p$ and $|K| \leq 5p$.*

Proof. Since G is an inclusion minimal well-behaved subgraph, it does not contain a proper subgraph that is also well-behaved. Therefore, G does not contain a vertex of degree at most 1 because such vertices and edges adjacent to them cannot be part of any cycle. Suppose G has r connected components $C_1, \ldots, C_r$. We fix a spanning tree T_j of C_j for each $j \in [r]$. We construct the set V_c by including every vertex of degree three or more to V_c. The components C_j that do not contain a vertex of degree three must be a simple cycle because G does not have degree-1 vertices. For every such C_j, we include vertices adjacent to the only non-tree edge of C_j. It is easy to verify that G consists of K chains connecting vertices in V_c.

Let E_0 be the set of non-tree edges, that are edges not in T_j for some $j \in [r]$. We claim that $|E_0| \leq p$. Since G is well-behaved, $G(i)$ consists an odd-labeled cycle for all $i \in [p]$. Using Lemma 5, and the spanning tree T_j of the component containing that odd labeled cycle, we can transform into an odd-labeled cycle that uses at most one non-tree edge. Repeating this for all pairs, we can use at most p edges from E_0. If $|E_0| > p$, then we would have a proper subgraph of G with at most p edges that is also well-behaved, which is not possible because G was chosen to be inclusion minimal. Therefore $|E_0| \leq p$.

The graph G only contains vertices of degree 2 or higher, hence each leaf node of the trees $T_1, \ldots, T_r$ must be adjacent to some edge in E_0. Therefore, the number of leaf nodes is at most $2p$, and so the number of nodes of degree three or above in $T_1, \ldots, T_r$ is also at most

$2p$. Observe that the vertices in V_c are either adjacent to some edge in E_0 or have degree three or more in some tree T_j. The number of both these type of vertices is at most $2p$, which gives us $|V_c| \leq 4p$. Finally, we bound $|K|$, the number of chains. Note that each edge of G belongs to exactly one chain in K. Therefore, the number of chains containing at least one edge in E_0 is at most p, because $|E_0| \leq p$. All the other chains that do not have any edge in E_0, are contained in the trees $T_1, \ldots, T_r$. It follows that these chains do not form any cycle, and thus their number is less than $|V_c|$. This gives us $|K| \leq 5p$. ◀

It is easy to see that if $\mathcal{S}' \subseteq \mathcal{S}$ is an optimal set of obstacles separating all pairs in P, then there exists an inclusion minimal well-behaved subgraph G of $G_{\mathcal{S}}[\mathcal{S}']$ that satisfies the property of Lemma 14. Observe that the K chains of graph G are vertex disjoint, so for every chain K_t connecting vertices $S_i, S_j \in V_c$ that has $\mathsf{lab}(K_t) = \ell$, an optimal solution will always choose the walk in $G_{\mathcal{S}}$ that has label ℓ and has fewest vertices. To that end, we will need the following simple lemma which is a generalization of the algorithm to compute shortest odd cycle in $G_{\mathcal{S}}$ with 1-bit labels.

▶ **Lemma 15.** *Given a labeled graph $G_{\mathcal{S}} = (\mathcal{S}, E)$ with labeling $\mathsf{lab} : E \to \{0,1\}^p$, the shortest walk between any pair of vertices S_i, S_j with a fixed label $\ell \in \{0,1\}^p$ can be computed in $2^{O(p)}n^{O(1)}$ time.*

Algorithm: Separate-Point-Pairs.

1. For every pair of vertices $S_i, S_j \in \mathcal{S}$ and every label $\ell \in \{0,1\}^p$, precompute the shortest walk connecting S_i, S_j with label ℓ in $G_{\mathcal{S}}$ using Lemma 15.
2. For all possible sets $V_c \subseteq \mathcal{S}$ and ways of connecting V_c by K chains:
   - For all $(2^p)^{5p} = 2^{O(p^2)}$ possible labeling of K chains:
     a. Let $G \subseteq G_{\mathcal{S}}$ be the labeled graph consisting of vertices V_c and chains $K_t \in K$ replaced by shortest walk between endpoints of K_t with label $\mathsf{lab}(K_t)$, already computed in Step 1.
     b. Check if the graph G is well-behaved. If so, add its vertices as one candidate solution.
3. Return the candidate vertex set with smallest size as solution.

Precomputing labeled shortest walks in Step 1 takes at most $2^{O(p)}n^{O(p)}$ time. The total number of candidate graphs G is $n^{O(p)} \cdot p^{O(p)} \cdot 2^{O(p^2)}$, and checking if it is well behaved can be done in $n^{O(1)}$ time. We have the following result.

▶ **Theorem 16.** Generalized Points-separation *for connected obstacles in the plane can be solved in $2^{O(p^2)}n^{O(p)}$ time, where n is the number of obstacle and p is the number of point-pairs to be separated.*

▶ **Corollary 17.** Point-Separation *for connected obstacles in the plane can be solved in $2^{O(k^4)}n^{O(k^2)}$ time, where n is the number of obstacles and k is the number of points. This is polynomial in n for every fixed k.*

5.2 Faster Algorithms for Points-separation

Recall that the labeled graph $G_{\mathcal{S}}$ constructed in the previous section consisted of labels that are p-bit binary strings. As a result, the running time has a dependence of $n^{O(p)}$ which in worst case could be $n^{O(k^2)}$, for example, in the case of Points-separation when P consists of all point pairs. In this section, we describe an alternative approach that builds a labeled intersection graph whose labels are k-bit strings. Using this graph and the notion of *parity*

partitions, we obtain an $2^{O(p)}n^{O(k)}$ algorithm for Generalized Points-separation which gets rid of the $n^{O(k^2)}$ dependence for Points-separation. Due to lack of space, we describe our approach at a high level and defer the details to the full paper.

The construction of graph $G_{\mathcal{S}}$ is almost the same as before, except that now we choose the reference curves π_i differently. In particular, let $A = \{a_1, a_2, \ldots, a_k\}$ be the set of points and P be a set of pairs (a_i, a_j) of points we want to separate. We pick an arbitrary point o in the plane, and for each $i \in [k]$, we fix a plane curve with endpoints a_i and o as the reference curve π_i. For an edge e, the parity of crossing with respect to π_i defines the i-th bit of $\mathsf{lab}(e)$. The graph $G_{\mathcal{S}}$ constructed in this fashion has k-bit labels and will be referred as *k-labeled graph*.

Let G be a k-labeled graph. For a cycle (or a path) γ in G with edge sequence $(e_1, \ldots, e_r)$, we define $\mathsf{parity}(\gamma) = \bigoplus_{t=1}^{r} \mathsf{lab}(e_t)$ and denote by $\mathsf{parity}_i(\gamma)$ the i-th bit of $\mathsf{parity}(\gamma)$ for $i \in [k]$. Here the notation "$\oplus$" denotes the bitwise XOR operation for binary strings. Also, we define $\Phi(\gamma)$ as the partition of $[k]$ consisting of two parts $I_0 = \{i : \mathsf{parity}_i(\gamma) = 0\}$ and $I_1 = \{i : \mathsf{parity}_i(\gamma) = 1\}$. Next, we define an important notion called *parity partition*.

▶ **Definition 18** (parity partition)**.** *Let G be a k-labeled graph. The **parity partition** induced by G, denoted by Φ_G, is the partition of $[k]$ such that $i, j \in [k]$ belong to the same part of Φ_G iff $\mathsf{parity}_i(\gamma) = \mathsf{parity}_j(\gamma)$ for every cycle γ in G.*

We say a k-labeled graph G is *P-good* if for all $(i, j) \in P$, i and j belong to different parts in Φ_G. The notion of P-goodness in k-labeled graphs is similar to *well-behaved* property of subgraphs $G'_{\mathcal{S}}$ that we defined in Lemma 13 except that the latter is defined using p reference curves. We prove the following lemma that establishes a characterization of obstacles that separate all point pairs in P called *P-separators* using P-goodness.

▶ **Lemma 19.** *A subset $\mathcal{S}' \subseteq \mathcal{S}$ is a P-separator iff the induced subgraph $G_{\mathcal{S}}[\mathcal{S}']$ is P-good.*

Similar to Lemma 14, one can show that there exists a P-good subgraph with $4k$ vertices and $5k$ edges. Applying the algorithm Separate-Point-Pairs from previous section gives an improved bound of $2^{k^2}n^{O(k)}$. Improving the running time to $2^{O(p)}n^{O(k)}$ require further nontrivial efforts. We defer the details to full version and state our main results.

▶ **Theorem 20.** Generalized Point-Separation *for connected obstacles in the plane can be solved in $2^{O(p)}n^{O(k)}$ time, where n is the number of obstacles, k is the number of points, and p is the number of point-pairs to be separated.*

▶ **Corollary 21.** Point-Separation *for connected obstacles in the plane can be solved in $2^{O(k^2)}n^{O(k)}$ time, where n is the number of obstacles and k is the number of points.*

Even Faster Algorithm for Pseudo-Disk Obstacles. If the obstacles in $\mathcal{S}$ are pseudo-disks then we can further improve the dependence on n to be $n^{O(\sqrt{k})}$. To this end, the key observation is the following analog of Lemma 19 for pseudo-disk obstacles.

▶ **Lemma 22.** *Suppose $\mathcal{S}$ consists of pseudo-disk obstacles. Then a subset $\mathcal{S}' \subseteq \mathcal{S}$ is a P-separator iff there is a subgraph of the induced subgraph $G_{\mathcal{S}}[\mathcal{S}']$ that is planar and P-good.*

The planarity of subgraph $G_{\mathcal{S}}[\mathcal{S}']$ allows us to efficiently enumerate the candidate sets using the *planar separator theorem*. We state our main result for such obstacles.

▶ **Theorem 23.** Generalized Point-Separation *for pseudo-disk obstacles in the plane can be solved in $2^{O(p)}k^{O(k)}n^{O(\sqrt{k})}$ time, where n is the number of obstacles, k is the number of points, and p is the number of point-pairs to be separated.*

▶ **Corollary 24.** Point-Separation *for pseudo-disk obstacles in the plane can be solved in* $2^{O(k^2)}n^{O(\sqrt{k})}$ *time, where* n *is the number of obstacles and* k *is the number of points.*

5.3 Hardness of Points-separation

We complement our algorithmic results for Points-separation with almost matching hardness bounds assuming the Exponential Time Hypothesis (ETH). We obtain the following results by reductions from Partitioned Subgraph Isomorphism [19] and Planar Multiway Cut [20] respectively.

▶ **Theorem 25.** *Unless ETH fails, a* Points-separation *instance* $(\mathcal{S}, A)$ *for general obstacles cannot be solved in* $f(k)n^{o(k/\log k)}$ *time where* $n = |\mathcal{S}|$ *and* $k = |A|$.

▶ **Theorem 26.** *Unless ETH fails, a* Points-separation *instance* $(\mathcal{S}, A)$ *with pseudodisk obstacles cannot be solved in* $f(k)n^{o(\sqrt{k})}$ *time where* $n = |\mathcal{S}|$ *and* $k = |A|$.

References

1. Amit Agarwal, Moses Charikar, Konstantin Makarychev, and Yury Makarychev. O(sqrt(log n)) approximation algorithms for min uncut, min 2cnf deletion, and directed cut problems. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*, pages 573–581, 2005.
2. Paul Balister, Zizhan Zheng, Santosh Kumar, and Prasun Sinha. Trap coverage: Allowing coverage holes of bounded diameter in wireless sensor networks. In *IEEE INFOCOM 2009*, pages 136–144. IEEE, 2009.
3. Sayan Bandyapadhyay, Neeraj Kumar, Subhash Suri, and Kasturi Varadarajan. Improved approximation bounds for the minimum constraint removal problem. *Computational Geometry*, 90:101650, 2020.
4. Sergey Bereg and David G. Kirkpatrick. Approximating barrier resilience in wireless sensor networks. In *Proc. of 5th ALGOSENSORS*, volume 5804, pages 29–40, 2009.
5. S. Cabello and P. Giannopoulos. The complexity of separating points in the plane. *Algorithmica*, 74(2):643–663, 2016.
6. David Yu Cheng Chan and David G. Kirkpatrick. Approximating barrier resilience for arrangements of non-identical disk sensors. In *Proc. of 8th ALGOSENSORS*, pages 42–53, 2012.
7. David Yu Cheng Chan and David G. Kirkpatrick. Multi-path algorithms for minimum-colour path problems with applications to approximating barrier resilience. *Theor. Comput. Sci.*, 553:74–90, 2014.
8. E. Eiben and I. Kanj. How to navigate through obstacles? In *Proc. of 45th ICALP*, 2018.
9. Eduard Eiben, Jonathan Gemmell, Iyad A. Kanj, and Andrew Youngdahl. Improved results for minimum constraint removal. In *Proc. of 32nd AAAI*, pages 6477–6484, 2018.
10. Eduard Eiben and Iyad Kanj. A colored path problem and its applications. *ACM Trans. Algorithms*, 16(4):47:1–47:48, 2020.
11. Eduard Eiben and Daniel Lokshtanov. Removing connected obstacles in the plane is FPT. In *Proc. of 36th SoCG*, volume 164, pages 39:1–39:14, 2020.
12. Lawrence H. Erickson and Steven M. LaValle. A simple, but NP-Hard, motion planning problem. In *Proc. of 27th AAAI*, 2013.
13. Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001.
14. Matias Korman, Maarten Löffler, Rodrigo I. Silveira, and Darren Strash. On the complexity of barrier resilience for fat regions and bounded ply. *Comput. Geom.*, 72:34–51, 2018.
15. Stefan Kratsch and Magnus Wahlström. Representative sets and irrelevant vertices: New tools for kernelization. *Journal of the ACM (JACM)*, 67(3):1–50, 2020.

16 Santosh Kumar, Ten-Hwang Lai, and Anish Arora. Barrier coverage with wireless sensors. *Wirel. Networks*, 13(6):817–834, 2007.

17 James R. Lee. Separators in region intersection graphs. In *Proc. of 8th ITCS*, volume 67, pages 1–8, 2017.

18 Daniel Lokshtanov, NS Narayanaswamy, Venkatesh Raman, MS Ramanujan, and Saket Saurabh. Faster parameterized algorithms using linear programming. *ACM Transactions on Algorithms (TALG)*, 11(2):1–31, 2014.

19 Dániel Marx. Can you beat treewidth? In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*, pages 169–179. IEEE, 2007.

20 Dániel Marx. A tight lower bound for planar multiway cut with fixed number of terminals. In *International Colloquium on Automata, Languages, and Programming*, pages 677–688. Springer, 2012.

21 Saket Saurabh Neeraj Kumar, Daniel Lokshtanov and Subhash Suri. A constant factor approximation for navigating through connected obstacles in the plane. In *Proc. 32nd SODA*, 2021.