Minimum Cuts in Geometric Intersection Graphs

Sergio Cabello* Wolfgang Mulzer†

October 29, 2020

Abstract

Let $\mathscr{D}$ be a set of n disks in the plane. The *disk graph* $G_{\mathscr{D}}$ for $\mathscr{D}$ is the undirected graph with vertex set $\mathscr{D}$ in which two disks are joined by an edge if and only if they intersect. The *directed transmission graph* $G^{\rightarrow}_{\mathscr{D}}$ for $\mathscr{D}$ is the directed graph with vertex set $\mathscr{D}$ in which there is an edge from a disk $D_1 \in \mathscr{D}$ to a disk $D_2 \in \mathscr{D}$ if and only if D_1 contains the center of D_2.

Given $\mathscr{D}$ and two non-intersecting disks $s, t \in \mathscr{D}$, we show that a minimum s-t vertex cut in $G_{\mathscr{D}}$ or in $G^{\rightarrow}_{\mathscr{D}}$ can be found in $O(n^{3/2} \operatorname{polylog} n)$ expected time. To obtain our result, we combine an algorithm for the maximum flow problem in general graphs with dynamic geometric data structures to manipulate the disks.

As an application, we consider the *barrier resilience problem* in a rectangular domain. In this problem, we have a vertical strip S bounded by two vertical lines, L_ℓ and L_r, and a collection $\mathscr{D}$ of disks. Let a be a point in S above all disks of $\mathscr{D}$, and let b a point in S below all disks of $\mathscr{D}$. The task is to find a curve from a to b that lies in S and that intersects as few disks of $\mathscr{D}$ as possible. Using our improved algorithm for minimum cuts in disk graphs, we can solve the barrier resilience problem in $O(n^{3/2} \operatorname{polylog} n)$ expected time.

Keywords: computational geometry, geometric intersection graph, disk graph, unit-disk graph, vertex-disjoint paths, barrier resilience.

Acknowledgments. Parts of this work were initiated at the Fifth Annual Workshop on Geometry and Graphs that took place March 5–10, 2017, at the Bellairs Research Institute. We thank the organizers and all participants for the productive and positive atmosphere.

*Faculty of Mathematics and Physics, University of Ljubljana, Slovenia, and Institute of Mathematics, Physics and Mechanics, Slovenia. Supported by the Slovenian Research Agency (P1-0297, J1-9109, J1-8130, J1-8155, J1-1693, J1-2452.). Email address: sergio.cabello@fmf.uni-lj.si

†Institut für Informatik, Freie Universität Berlin, Germany. Supported in part by ERC StG 757609. Email address: mulzer@inf.fu-berlin.de

1 Introduction

Let $\mathscr{D}$ be a family of n (closed) disks in the plane. The ***disk graph*** $G_{\mathscr{D}}$ for $\mathscr{D}$ is the undirected graph with vertex set $\mathscr{D}$ and edge set

$$E(G_{\mathscr{D}}) = \{D_1D_2 \mid D_1, D_2 \in \mathscr{D}, D_1 \cap D_2 \neq \emptyset\}.$$

If the disks in $\mathscr{D}$ are partitioned into two sets $\mathscr{D}_A$ and $\mathscr{D}_B$, one can also define a *bipartite* intersection graph by considering only the edges that come from an intersection between a disk in $\mathscr{D}_A$ and a disk in $\mathscr{D}_B$. If all disks in $\mathscr{D}$ have the same radius, we call $G_{\mathscr{D}}$ a ***unit-disk graph***. A directed version of disk graphs can be defined as follows: for $D \in \mathscr{D}$, let $c_D \in D$ denote the center of D. The ***directed transmission graph*** $G^{\rightarrow}_{\mathscr{D}}$ is the directed graph with vertex set $\mathscr{D}$ and edge set

$$E\left(G^{\rightarrow}_{\mathscr{D}}\right) = \{D_1 \rightarrow D_2 \mid D_1, D_2 \in \mathscr{D}, c_{D_2} \in D_1\}.$$

If we ignore the direction of the edges in $G^{\rightarrow}_{\mathscr{D}}$, we obtain a subgraph of $G_{\mathscr{D}}$.

Unit disk graphs are often used to model ad-hoc wireless communication networks and sensor networks [GG11, ZG04, HS95]. Disks of varying sizes become relevant when different sensors cover different areas. Moreover, general disk graphs may serve as a tool to approach other problems; for example, an application to the barrier resilience problem [KLA07] is discussed below. Directed transmission graphs model ad-hoc networks where different entities have different power ranges [PR10].

Minimum s-t cut in disk graphs. Consider a graph $G = (V, E)$ with n vertices and m edges, and two non-adjacent vertices $s, t \in V$. A set $X \subseteq V \setminus \{s, t\}$ of vertices is called an s-t ***(vertex) cut*** if $G - X$ contains no path from s to t. Two paths from s to t are ***(interior-)vertex-disjoint*** if their only common vertices are s and t. By Menger's theorem (see, for example, [KV10, Section 8.2]), the minimum size of an s-t cut equals the maximum number of vertex-disjoint s-t paths, both in directed and in undirected graphs. Using blocking flows, Even and Tarjan, as well as Karzanov [ET75, Kar73] showed that an s-t minimum-cut can be computed in time $O(\sqrt{n}m)$. In the worst case, if $m = \Theta(n^2)$, this is $O(n^{5/2})$. This was an improvement over the previous algorithm by Dinitz [Din70]; see [Din06] for a great historical account of the algorithms. In particular, the use of DFS did not appear in his original description [Din70], but it was developed by Shimon Even and Alon Itai and included in Even's textbook [Eve79]. The more recent $O(m^{10/7})$-time algorithm of Mądry [Mąd13] gives a better running time for sparse graphs, i.e., for $m = o(n^{7/4})$.

The size of a minimum s-t vertex cut in a network G is a key estimator for its vulnerability. Since such networks often arise from geometric settings, it is natural to consider the case where G is a disk graph. A particularly interesting scenario of this kind is the ***barrier resilience problem***, an optimization problem introduced by Kumar, Lai, and Arora [KLA07]. In one variant of the problem, we are given a vertical strip S bounded by two vertical lines, L_ℓ and L_r, and a collection $\mathscr{D}$ of disks. Each disk $D \in \mathscr{D}$ represents a region monitored by a sensor. Let a be a point in S above all disks of $\mathscr{D}$, and let b a point in S below all disks of $\mathscr{D}$. The task is to find a curve from a to b that lies in the strip S and that intersects as few disks of $\mathscr{D}$ as possible (the disks do not need to lie inside S). This models the resilience of monitoring a boundary region with respect to (total) failures of the sensors. Kumar, Lai, and Arora show that the problem reduces to an L_ℓ-L_r minimum-cut problem in the intersection graph of $\mathscr{D} \cup \{L_\ell, L_r\}$. We mention that for another variant of the problem, where the endpoints a and b can lie in arbitrary locations, the complexity status is still unknown, despite many efforts by several researchers [ACGK17, BK09, CK14, EL20, KLSS18, TK11].

A variant of the problem, called ***minimum shrinkage***, was recently introduced by Cabello et al. [CJLM20]. Here, the task is to shrink some of the disks, potentially by different amounts, such that there is an a-b curve that is disjoint from the interiors of all disks. The objective is to minimize the total amount of shrinkage. Cabello et al. provide an FPTAS for the version where the path is

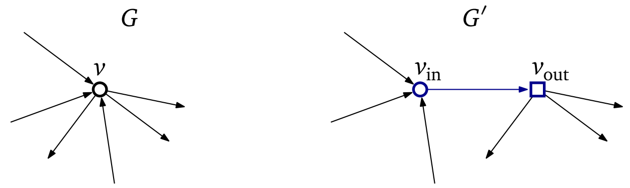

Figure 1: Transforming a vertex $v \in V(G) \setminus \{s,t\}$ to get from G to G'.

restricted to lie inside a vertical strip and the endpoints a and b are above and below all the disks. This result is achieved by reducing the problem to a barrier resilience instance with $O(n^2/\varepsilon)$ disks of different radii. In contrast, when the endpoints a and b can lie in arbitrary locations, the problem is weakly NP-hard [CC20].

Our Results. We exploit the geometric structure to provide a new algorithm to find the minimum s-t cut in disk graphs and directed transmission graphs in $O(n^{3/2} \operatorname{polylog} n)$ expected time. For this, we adapt the approach of Even and Tarjan [ET75], extending it with suitable geometric data structures. Our method is similar in spirit to the algorithm by Efrat, Itai, and Katz [EIK01] for maximum *bipartite* matching in (unit) disk graphs. However, since our graph is not bipartite, the structure of the graph is more complex and additional care is needed.

2 Minimum s-t Cut in Disk Graphs

Let $\mathcal{D}$ be a set of n disks in the plane, and let $s,t \in \mathcal{D}$ be two non-intersecting disks. We show how to compute the maximum number of vertex disjoint paths between s and t in $G_{\mathcal{D}}$ and in $G_{\vec{\mathcal{D}}}$. This also provides a way to find a minimum s-t (vertex) cut. For this, we adapt the algorithm of Even and Tarjan [ET75] to our geometric setting. First, we suppose that certain geometric primitives are available as a black box, and we analyze the running time under this assumption. Then, we instantiate these primitives with appropriate data structures to obtain the desired result.

2.1 Generic algorithm

Let G be a graph with n vertices and m edges, and let s and t be two non-adjacent vertices of G. We want to find the maximum number of paths from s to t in G that are pairwise vertex disjoint. The graph G is assumed to be directed.[1]

First, we transform the graph G into another graph G' in which every vertex other than s and t has in-degree or out-degree 1. More precisely, for each vertex $v \in V(G) \setminus \{s,t\}$, we perform the following operation: we replace v with two new vertices v_{in} and v_{out}, add the directed edge $v_{\text{in}} \to v_{\text{out}}$, replace every directed edge $u \to v$ with $u \to v_{\text{in}}$, and replace every directed edge $v \to w$ with $v_{\text{out}} \to w$; see Figure 1. The vertices s and t remain untouched. The transformed graph G' has $2n-2$ vertices and $m+n-2$ edges. It is bipartite, as can be seen by partitioning the vertices into the sets $\{s\} \cup \{v_{\text{out}} \mid v \in V(G) \setminus \{s,t\}\}$ and $\{t\} \cup \{v_{\text{in}} \mid v \in V(G) \setminus \{s,t\}\}$. Vertex-disjoint s-t paths in G directly correspond to vertex disjoint s-t paths in G'. Furthermore, in G' we have that edge-disjoint and vertex-disjoint s-t paths are equivalent, because every vertex (other than s and t) has in-degree or out-degree 1. Thus, it suffices to find the maximum number of edge-disjoint s-t paths in G'.

Assume we have a family $\Pi = \{\pi_1, \dots, \pi_k\}$ of k edge-disjoint s-t paths in G'. Let $E(\Pi) = \bigcup_{\pi \in \Pi} E(\pi)$ denote the set of all the directed edges on the paths of Π. See Figure 2 for an illustration of the following concepts and discussion. The ***residual graph*** $R = R(G', \Pi)$ is the directed graph

[1]Otherwise, we replace each undirected edge uv by two directed edges $u \to v$ and $v \to u$. An optimal solution to the directed instance directly gives an optimal solution to the undirected case.

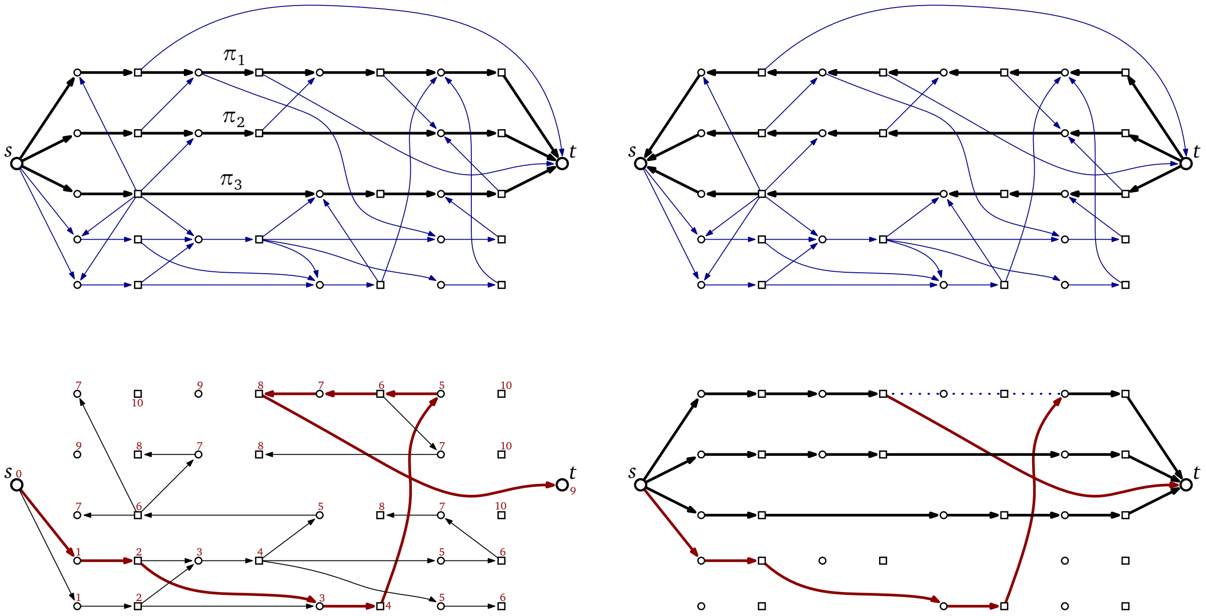

Figure 2: Top left: a graph G' with 3 vertex-disjoint s-t paths $\Pi = \{\pi_1, \pi_2, \pi_3\}$ in bold. Top right: the residual graph $R(G', \Pi)$. Bottom left: the layered residual graph $L(G', \Pi)$. We keep all vertices and each vertex has its distance from the vertex s annotated. An s-t path γ in L is marked in thick red. Bottom right: the paths obtained from $E(\Pi) \oplus E(\gamma)$.

with vertex set $V(G')$ and edge set

$$E(R) = \{u \to v \mid u \to v \in E(G') \setminus E(\Pi) \text{ or } v \to u \in E(\Pi)\}.$$

The residual graph R is bipartite with the same bipartition as G'. As in G', every vertex in $V(R) \setminus \{s, t\}$ has in-degree or out-degree at most 1.

For a vertex v of G', the ***level*** $\lambda(v)$ (with respect to R) of v is the BFS-distance from s to v in R, i.e., the minimum number of edges on a path from s to v in R.[2] If v is not reachable from s in R, we set $\lambda(v) = +\infty$. For every integer $i \geq 0$, the ***layer*** $L[i]$ is the set of vertices at level i, i.e., $L[i] = \{v \in V(G') \mid \lambda(v) = i\}$. The ***layered residual graph*** $L(G', \Pi)$ for G' and Π is the subgraph of the residual graph $R(G', \Pi)$ where only the directed edges from $L[i-1]$ to $L[i]$, for $i = 1, \ldots, \lambda(t)-1$, and the directed edges from $L[\lambda(t)-1]$ to t are kept. More precisely, this means that $L = L(G', \Pi)$ has vertex set $V(G')$ and directed edge set

$$E_t \cup \{u \to v \in E(R) \mid \lambda(u) + 1 = \lambda(v) < \lambda(t)\},$$

where

$$E_t \quad = \quad \{u \to t \in E(R) \mid \lambda(u) + 1 = \lambda(t)\}.$$

Let $\Gamma = \{\gamma_1, \ldots, \gamma_\ell\}$ be a family of edge-disjoint s-t paths in the layered residual graph $L = L(G', \Pi)$. By construction, all paths of Γ have exactly $\lambda(t)$ edges. Using the k paths of Π in G' and the ℓ paths of Γ in L, we can obtain $k + \ell$ edge-disjoint s-t paths in G'. For this, consider the edges

$$E(\Pi) \oplus E(\Gamma) \quad = \quad \{u \to v \mid u \to v \in E(\Pi) \text{ and } v \to u \notin E(\Gamma)\} \cup \{u \to v \mid u \to v \in E(\Gamma) \text{ and } v \to u \notin E(\Pi)\}$$

that are obtained from $E(\Pi) \cup E(\Gamma)$ by canceling out directed edges that appear in both directions. The following observation is simple:

Lemma 1. *The set $E(\Pi) \oplus E(\Gamma)$ consists of $k + \ell$ edge-disjoint s-t paths in G'. Given Π and Γ, we can construct $E(\Pi) \oplus E(\Gamma)$ and the corresponding $k + \ell$ edge-disjoint s-t paths in G' in $O(|E(\Pi)| + |E(\Gamma)|)$ total time.*

[2] Recall that R depends on both G' and Π.

Proof. The definition of R ensures that the edges $E(\Pi) \oplus E(\Gamma)$ all lie in G', since for $u \to v \in E(\Gamma) \setminus E(\Pi)$, we must have $v \to u \in E(\Pi)$. Furthermore, every vertex v of $V(G') \setminus \{s,t\}$ has in-degree and out-degree both 0 or both 1 in $E(\Pi) \oplus E(\Gamma)$. This is clear if v appears on at most one path in $\Pi \cup \Gamma$. If v appears on both a path from Π and from Γ, then one incoming edge and one outgoing edge of v must cancel, since v has at most one incoming or outgoing edge in L and the corresponding reverse edge must have appeared on a path in Γ. The in-degree of s is 0 and the out-degree of t is 0. Moreover, the out-degree of s is $k + \ell$, because the outgoing edges from s never cancel out. This means that $E(\Pi) \oplus E(\Gamma)$ defines $k + \ell$ paths from s to t. These paths can be found in $O(|E(\Pi)| + |E(\Gamma)|)$ time by constructing the graph $(V(\Pi \cup \Gamma), E(\Pi) \oplus E(\Gamma))$ explicitly. □

A family Γ of s-t paths in the layered residual graph L is ***blocking*** if $L - E(\Gamma)$ contains no s-t path, i.e., every s-t path in L contains at least one edge from $E(\Gamma)$. Even and Tarjan [ET75] describe the following algorithm for finding a blocking family Γ of s-t paths in a layered residual graph L: we start with $\Gamma = \emptyset$, $D_0 = L$, and $j = 1$. The algorithm proceeds in rounds. In round j, we perform a DFS traversal from s in D_{j-1}. When we reach t, the DFS stack contains a path γ_j from s to t in $D_{j-1} \subseteq L$. We add the path γ_j to Γ, and we obtain D_j by removing from D_{j-1} all the vertices (other than s and t) that have been explored during the partial DFS traversal of D_{j-1}. We finish when the graph D_{j-1} of the current round j does not contain any s-t path. This is detected during the DFS traversal of D_{j-1}. We refer to the paper of Even and Tarjan [ET75] for the running time analysis and the proof of correctness. The following lemma summarizes the result.

Lemma 2 (Even and Tarjan [ET75])**.** *Let L be a layered residual graph. In $O(|E(L)|)$ time, we can find a blocking family Γ of s-t paths in L.*

The algorithm to find the maximum number of edge-disjoint s-t paths in G' is the following: we start with $\Pi_0 = \emptyset$. Then, for $j = 1, \dots,$ we construct the residual graph $R_j = R(G, \Pi_{j-1})$, the layered residual graph $L_j = L(G, \Pi_{j-1})$, a blocking family Γ_j of s-t paths in L_j, and we set Π_j to the set of (edge-disjoint) s-t paths defined by $E(\Pi_{j-1}) \oplus E(\Gamma_j)$. We finish when L_j contains no s-t path. The work performed for a single value of j (constructing L_j, R_j, Γ_j and Π_j), is called a ***phase***. Let $\lambda_j(\cdot)$ denote the level of a vertex in the residual graph R_j. Even and Tarjan [ET75] show that $\lambda_j(t)$ increases monotonically as a function of j. Thus, using that the paths Γ_j are vertex-disjoint and have length $\lambda_j(t)$ (whenever L_j contains some s-t path), one obtains the following.

Theorem 3 (Even and Tarjan [ET75])**.** *The algorithm performs at most $O(\sqrt{n})$ phases. When the algorithm finishes, Π_{j-1} contains the maximum possible number of vertex-disjoint s-t paths in G'.*

2.2 Adaptation for neighbor queries

We want to adapt the algorithm from Section 2.1 to our geometric setting. For this, we extend the approach by Efrat, Itai, and Katz [EIK01] for finding maximum matchings in bipartite geometric intersection graphs. The idea is to avoid the explicit construction of the layered residual graphs $L_j = L(G, \Pi_{j-1})$, and to use instead an implicit representation that allows for an efficient DFS traversal of the current L_j. For this, we identify which vertices belong to each layer of the current L_j, and we use dynamic nearest-neighbor data structures to find the directed edges between the layers. In order to encapsulate the geometric primitives, we assume that we have a certain geometric data structure to access the directed edges of G. Note that the assumption is on the original graph G, not in the transformed graph G'. Later, we will describe how such a data structure can be derived from known results about (semi-)dynamic nearest neighbor searching.

Graph Encoding A. *Let G be a directed graph with n vertices. We assume that we have a data structure* $\mathrm{DS} = \mathrm{DS}(U)$ *that semi-dynamically maintains a subset $U \subseteq V(G)$ with the following operations:*

- *construct the data structure* $\mathrm{DS}(U)$ *for an initial subset $U \subseteq V(G)$ of vertices from G. The construction time is denoted by $T_c(m)$, where m is the number of vertices in U, and we require that $T_c(\cdot)$ satisfies $T_c(m) + T_c(m') \le T_c(m + m')$, for all $m, m' \in \mathbb{N}$;*

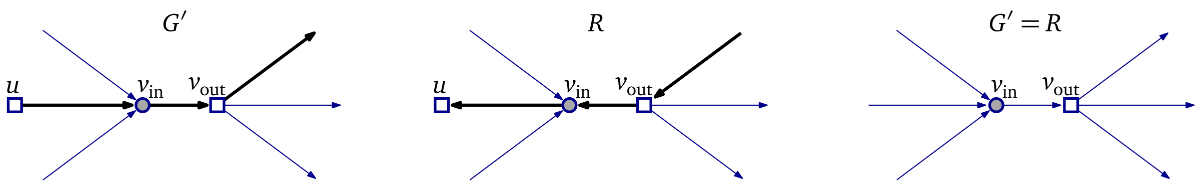

Figure 3: Case in the proof of Lemma 4: $v_{\text{in}} \in L[i-1]$, for i even. The left and center figure show G' and R when v_{in} belongs to some path of Π (bold). The right figure shows $G' = R$ when v_{in} does not belong to any path of Π.

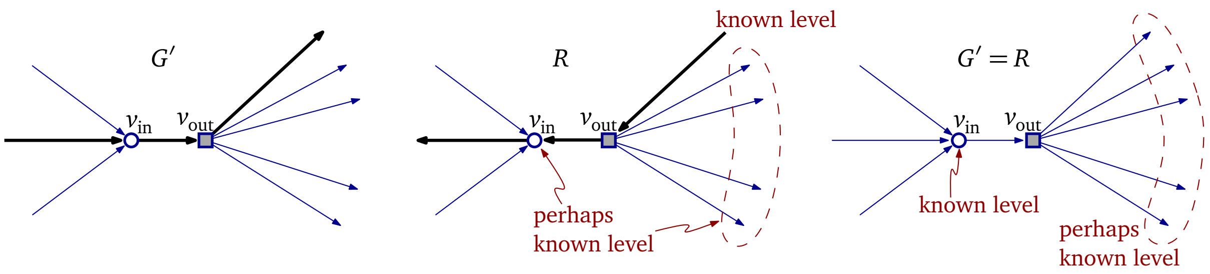

Figure 4: Case in the proof of Lemma 4: $v_{\text{out}} \in L[i-1]$, for i odd. The left and center figure show G' and R when v_{out} belongs to some path of Π (bold). The right figure shows $G' = R$ when v_{out} does not belong to any path of Π.

- *delete of a vertex $u \in U$ from* $\mathrm{DS}(U)$*. The deletion time is denoted by $T_d(n)$, where n refers to the number of vertices in G; and*
- *given a query vertex $v \in V(G)$ from G, find an outgoing edge $v \to u$ with $u \in U$, or report that no such vertex exists in the current set U. The query time is denoted by $T_q(n)$, where n refers to the number of vertices in G.*

Henceforth, we assume our n-vertex graph G can be accessed as in Graph Encoding A. As before, we denote the corresponding transformed graph by G'. First, we show how to find the levels in the layered residual graph.

Lemma 4. *Let Π be a set of edge-disjoint paths in the transformed graph G'. In time $O(T_c(n) + nT_q(n) + nT_d(n))$, we can find the level $\lambda(v)$ of each vertex $v \in V(G')$ in the layered residual graph $L = L(G', \Pi)$.*

Proof. Our goal is to perform a BFS in the residual graph $R = R(G', \Pi)$ without explicitly constructing the edge set of R. In a preprocessing phase, for every vertex v in $V(G') \setminus \{s, t\}$ that appears in some path of Π, we mark v and store the unique vertices $\mathrm{prev}(v)$ and $\mathrm{next}(v)$ such that $\mathrm{prev}(v) \to v$ and $v \to \mathrm{next}(v)$ are directed edges in $E(\Pi)$. This takes time $O(|E(\Pi)|) = O(n)$.

Next, we set $L[0] = \{s\}$, construct the data structure DS of Graph Encoding A for $V(G) \setminus \{s\}$. Thus, the current vertex set U in DS is initially $U = V(G) \setminus \{s\}$. In our algorithm, we iteratively compute the layers $L[i]$, for $i = 1, 2, \ldots$. In the process, we maintain the invariant that, after computing $L[i]$, the structure DS contains t and the vertices u in $V(G)$ for which we do not yet know the level $\lambda(u_{\text{in}})$ in $L(G', \Pi)$.

To find $L[1]$, we repeatedly query DS with s and remove from DS the reported item, until DS contains no further out-neighbors of s. This gives the set

$$U' = \{u \in V(G) \setminus \{s\} \mid s \to u \in E(G)\}$$

of all out-neighbors of s in G. Let $U'_{\text{in}} = \{u_{\text{in}} \mid u \in U\}$ be the set of corresponding out-neighbors of s in G'. We filter U'_{in} and remove those vertices v that are in some path of Π and have $\mathrm{prev}(v) = s$. This gives a set U''_{in} with $L[1] = U''_{\text{in}}$. For each vertex $u_{\text{in}} \in U''_{\text{in}}$, we set $\lambda(u_{\text{in}}) = 1$. For each vertex

$u_{\text{in}} \in U'_{\text{in}} \setminus U''_{\text{in}}$, the level of u_{in} in L is not yet known. If DS supported insertions, we would insert the vertices u with $u_{\text{in}} \in U'_{\text{in}} \setminus U''_{\text{in}}$ back into DS. Instead, we just construct the data structure DS *anew* for $V(G) \setminus (\{s\} \cup \{u \mid u_{\text{in}} \in U''_{\text{in}}\})$.

Then, for $i = 2, \ldots,$ while $L[i-1]$ is not empty and $L[i-1]$ does not contain t, we compute $L[i]$. If i is even, we iterate over the vertices v_{in} of $L[i-1]$; see Figure 3. The vertex v_{in} has one outgoing edge in L: if v_{in} does not lie on some path of Π, then L contains only the outgoing edge $v_{\text{in}} \to v_{\text{out}}$; if v_{in} lies on some path of Π, then L contains only the outgoing edge $v_{\text{in}} \to \text{prev}(v_{\text{in}})$. If v_{in} does not belong to any path of Π, we set $\lambda(v_{\text{out}}) = i$ and add v_{out} to $L[i]$. (In this case, the only incoming edge to v_{out} in the residual graph is from v_{in}, so we know that $\lambda(v_{\text{out}})$ was not yet determined.) If v_{in} belongs to some path of Π, we set $u = \text{prev}(v_{\text{in}})$ and distinguish two cases. If $u = s$, we do not need to do anything because $\lambda(s)$ is already set. If $u \neq s$, we set $\lambda(u) = i$ and add u to $L[i]$. (In this case, $u = w_{\text{out}}$ for some vertex $w \in V(G) \setminus \{s, t\}$ and $\lambda(u)$ was not yet determined because $v_{\text{in}} \to w_{\text{out}}$ is the only incoming edge to w_{out} in the residual graph.)

If i is odd, we iterate over the vertices v_{out} of $L[i-1]$; see Figure 4. If the vertex v_{out} does not lie on some path of Π, the outgoing edges of v_{out} in L correspond to the outgoing edges of v_{out} in G'; if v_{out} lies on some path of Π, then the outgoing edge $v_{\text{out}} \to \text{next}(v_{\text{out}})$ in G' is replaced with the outgoing edge $v_{\text{out}} \to v_{\text{in}}$ in R. We proceed as follows: we query DS repeatedly with v and delete the reported items. This gives the set U' of vertices $u \in V(G)$ that are stored in DS and have $v \to u \in E(G)$. Due to the invariant, the set U' contains exactly those out-neighbors u of v in G such that $\lambda(u_{\text{in}})$ was not known before processing v_{out}. If v_{out} lies on some path of Π, then we already know the level of $w_{\text{in}} = \text{next}(v_{\text{out}})$ (it is $i-2$) because $w_{\text{in}} \to v_{\text{out}}$ is the only incoming edge to v_{out} in the residual graph, and therefore $w \notin U'$. For each $u \in U'$, we set $\lambda(u_{\text{in}}) = i$ and add u_{in} to $L[i]$. If v_{out} belongs to some path of Π, we check if v_{in} still has no level assigned, and if so, we set $\lambda(v_{\text{in}}) = i$, add v_{in} to $L[i]$, and delete v from DS.

We finish when $t \in L[i]$ or when $L[i]$ is empty. In the latter case, t cannot be reached from s in R, and therefore Π already contains a maximum number of vertex-disjoint s-t paths. In the former case, we remove all elements from $L[\lambda(t)]$ except for t.

To bound the running time, we note first that it takes $O(T_c(n))$ time to construct the data structure DS, and this is done twice. Next, we observe that every node u of G is deleted at most once from DS. Additionally, each query with a vertex of G in DS leads either to a deletion in DS or does not yield an out-neighbor of the vertex, but the latter happens at most once per vertex of G. Thus, in total we are making $O(n)$ queries and deletions in the data structure DS. The time bound follows. □

The next lemma shows how to find an actual blocking family in L.

Lemma 5. *Consider a set Π of edge-disjoint paths in G'. In $O(T_c(n) + nT_q(n) + nT_d(n))$ time, we can find a blocking family of s-t paths in the layered residual graph L.*

Proof. Using Lemma 4, we compute the level $\lambda(v)$ of each vertex v of G'. Recall the notation prev(v) and next(v) from the proof of Lemma 4 to denote the predecessor and successor of a vertex v on a path of Π. We adapt the algorithm in the proof of Lemma 2, which is based on a DFS traversal of L.

For each odd i with $1 \leq i \leq \lambda(t)$, we build a data structure DS$[i]$ as in Graph Encoding A for the set $V[i] = \{v \in V(G) \mid v_{\text{in}} \in L[i]\}$. This takes $\sum_i T_c(|V[i]|) \leq O(T_c(n))$ time because the sets $V[i]$ are pairwise disjoint. During the algorithm, the data structure DS$[i]$ will contain the vertices $v \in V[i]$ such that v_{in} has not yet been explored by the DFS traversal. Thus, in contrast to the approach in Lemma 2, we delete vertices as we explore them with the DFS traversal.

When we explore a vertex v_{in} (at odd level i), there are two options; see Figure 3. If v_{in} lies on some path of Π, we look at $u = \text{prev}(v_{\text{in}})$. If u has been explored already, we return[3]. Otherwise, we continue the DFS traversal at u. If v_{in} does not belong to any path of Π, then v_{out} has not been

[3]This happens only if $u = s$, as in any other case $u = w_{\text{out}}$ for some vertex $w \in V(G) \setminus \{s, t\}$ and $v_{\text{in}} \to w_{\text{out}}$ is the only incoming edge to w_{out} in the residual graph and thus in the layered residual graph.

explored yet, as $v_{\rm in}{\to}v_{\rm out}$ is the only incoming edge of $v_{\rm out}$, so we continue the DFS at $v_{\rm out}$. For each such vertex, we spend $O(1)$ time plus the time for the recursive calls, if they occur.

Consider now the case that we explore a vertex $v_{\rm out}$, at even level i; see Figure 4. If $i = \lambda(t) - 1$, we check whether the edge $v{\to}t$ belongs to $G \setminus E(\Pi)$. If so, we have found an s-t path γ in L. We add γ to the output, and restart the DFS traversal from s. If not, we return from the recursive call.

Consider the remaining case: we explore a vertex $v_{\rm out}$ at even level i and $i < \lambda(t) - 1$. If $v_{\rm out}$ belongs to some path of Π, and $v_{\rm in}$ has not been explored yet, we recursively explore $v_{\rm in}$ and remove v from $\mathrm{DS}[i+1]$. If $v_{\rm out}$ does not belong to any path of Π or we have returned from the exploration of $v_{\rm in}$, we explore the outgoing edges from $v_{\rm out}$ to $L[i+1]$ by repeating the following procedure. We query $\mathrm{DS}[i+1]$ with v to obtain an edge $v{\to}u$ of G such that $u_{\rm in} \in L[i+1]$, we remove u from $\mathrm{DS}[i+1]$, and we continue the DFS traversal from $u_{\rm in}$. The recursive call is correctly made along an edge of the layered residual graph because it *cannot* happen that $v_{\rm out}{\to}u_{\rm in}$ is an edge of Π; indeed, if $v_{\rm out}{\to}u_{\rm in}$ were an edge in Π, then in the residual graph the edge $u_{\rm in}{\to}v_{\rm out}$ would be the only edge incoming into $v_{\rm out}$, which would mean that in the DFS traversal we arrived to $v_{\rm out}$ from $u_{\rm in}$, and u would belong to $V[i-1]$ instead of $V[i+1]$. When the query to $\mathrm{DS}[i+1]$ with v returns an empty answer, we return from the recursive call at $v_{\rm out}$.

Every vertex u of $V[i]$, for i odd, is returned and removed from $\mathrm{DS}[i]$ at most once. Thus, each vertex of $V(G)$ is deleted exactly once from exactly one data structure $\mathrm{DS}[i]$. Furthermore, for every vertex v of $V(G)$, we make at most one query to the corresponding data structure $\mathrm{DS}[\cdot]$ that returns an empty answer. Thus, the running time is $O(n + T_c(n) + nT_q(n) + nT_d(n))$. □

The following lemma discusses how to find a minimum cut from a maximum family of s-t vertex disjoint paths.

Lemma 6. *Let Π be a maximum family of s-t vertex disjoint paths (in G or in G'). Given Π, we can obtain a minimum s-t cut in $O(T_c(n) + nT_q(n) + nT_d(n))$ time.*

Proof. Consider the residual graph $R = R(G', \Pi)$. Let A be the set of vertices in $V(G)$ that in the residual graph R are reachable from s. A standard result from the theory of maximum flows tells that the edges from A to $V(G) \setminus A$, denoted by $\delta_R(A)$, form a minimum edge s-t cut in G' and there are $|\Pi|$ edges in such a cut $\delta_R(A)$.

Let U be the set of vertices $u \in V(\Pi)$ such that $u_{\rm out} \notin A$ but $u_{\rm in} \in A$ or such that $u_{\rm in} \notin A$ but $\mathrm{prev}(u_{\rm in}) \in A$. (Here, like in previous proofs, we use $\mathrm{prev}(u)$ to denote the vertex such that $\mathrm{prev}(u){\to}u$ belongs to $E(\Pi)$.) Each edge of the cut $\delta_R(A)$ contributes one vertex to U. Then U is a minimum s-t cut in G.

If t is not reachable from s in R, then a vertex u is reachable from s in the residual graph R if and only if u is reachable from s in layered residual graph L. Thus, to compute U, we apply Lemma 4 to find the level $\lambda(v)$ of every vertex v in L. Then, the set U is

$$\{u \in V(G) \mid \lambda(u_{\rm in}) < +\infty, \lambda(u_{\rm out}) = +\infty\} \cup \{u \in V(G) \mid \lambda(\mathrm{prev}(u_{\rm in})) < +\infty, \lambda(u_{\rm in}) = +\infty\},$$

as desired. □

Now, we put everything together. By Theorem 3, we have $O(\sqrt{n})$ phases, and each phase can be implemented in $O(T_c(n) + nT_q(n) + nT_d(n))$ time because of Lemma 1 and Lemma 5.

Theorem 7. *Let G be a directed graph with n vertices and assume that a representation of its edges as given in Graph Encoding A exists. Then, we can find in $O(n^{1/2}(T_c(n) + nT_q(n) + nT_d(n)))$ time the maximum number of vertex-disjoint s-t paths for any given $s, t \in V(G)$. Similarly, we can find a minimum s-t cut.*

Proof. We use the algorithm described in Section 2.1, before Theorem 3. Because of Theorem 3, we have $O(\sqrt{n})$ phases. At phase j, we have a set Π_{j-1} of vertex-disjoint paths in G', and we use Lemma 5 to find a blocking family Γ_j of s-t paths in the layered residual graph $L_j = L(G', \Pi_{j-1})$. This takes $O(T_c(n) + nT_d(n) + nT_q(n))$ time per phase. Because of Lemma 1, we can then obtain the new family of s-t paths Π_j in $O(n)$ time per phase. The result for maximum number of vertex-disjoint s-t paths follows. For the minimum s-t cut, we use Lemma 6. □

3 Geometric Applications

Theorem 7 leads to several consequences for geometrically defined graphs, as we can use geometric data structures to realize Graph Encoding A efficiently. For unit disk graphs, there is the semi-dynamic data structure of Efrat, Itai, and Katz [EIK01]. The construction takes $O(n \log n)$ time, while each deletion and neighbor query takes $O(\log n)$ amortized time. For arbitrary disks, we can use the structure of Kaplan et al. [KMR+20].

Corollary 8. *Let $\mathcal{U}$ be a set of n unit disks in the plane and let s and t be two of the disks. We can find in $O(n^{3/2} \log n)$ time the minimum s-t cut in the intersection graph $G_{\mathcal{U}}$. For arbitrary disks, the running time becomes $O(n^{3/2} \log^{11} n)$ in expectation.*

We can easily adapt the algorithm to the case where s and t are arbitrary shapes (and the other vertices are still represented as disks), by precomputing the disks that intersect s and the disks that intersect t. We get the following consequence.

Corollary 9. *The barrier resilience problem with n unit disks can be solved in $O(n^{3/2} \log n)$ time. For arbitrary disks, the running time becomes $O(n^{3/2} \log^{11} n)$.*

For directed transmission graphs, we can use the data structure of Chan [Cha19] to report a disk center contained in a query disk. It takes $O(\log^4 n)$ amortized time per edition and query. (See [AM95, Cha10, CT16, KMR+20, Liu20] for related bounds and for an alternative presentation of Chan's data structure.)

Corollary 10. *Let $\mathcal{U}$ be a set of n disks of arbitrary radii in the plane and let s and t be two of the disks. We can find in $O(n^{3/2} \log^4 n)$ time the minimum s-t cut in the directed transmission graph $G_{\vec{\mathcal{U}}}$.*

Similar results can be obtained for squares and rectangles using data structures for orthogonal range searching. We next provide concrete running times for future reference. For intersection graphs of unit squares, we can again use the semi-dynamic data structure of Efrat, Itai, and Katz, that also applies for the L_1-metric [EIK01, Remark 5.5]. As before, the construction takes $O(n \log n)$ time, while each deletion and neighbor query takes $O(\log n)$ amortized time.

Corollary 11. *Let $\mathcal{U}$ be a set of n unit axis-parallel squares in the plane, and let s and t be two of the squares. We can find in $O(n^{3/2} \log n)$ time the minimum s-t cut in the intersection graph $G_{\mathcal{U}}$.*

The following corollary covers also the case of squares of different sizes.

Corollary 12. *Let $\mathcal{U}$ be a set of n axis-parallel rectangles in the plane and let s and t be two of the rectangles. We can find in $O(n^{3/2} \log^2 n)$ time the minimum s-t cut in the intersection graph $G_{\mathcal{U}}$.*

Proof. To apply Theorem 7, we need a data structure that maintains a set $\mathcal{U}$ of n axis-parallel rectangles under deletions and answers the following type of queries: given an axis-parallel rectangle R, report some rectangle $\mathcal{U}$ that intersects R.

For this task, Edelsbrunner [Ede80] provides a data structure with $T_c(n) = O(n \log^2 n)$, $T_q(n) = O(\log^2 n)$ amortized, and $T_d(n) = O(\log^2 n)$ amortized.

An alternative approach to get the desired data structure is the following. Two rectangles R and R' intersect if and only if some edge of R intersects some edge of R', or some vertex of R is contained in R', or some vertex of R' is contained in R. Each one of this conditions can be checked using orthogonal range searching techniques, namely using range trees [dBCvKO08, Section 5.3], interval trees [dBCvKO08, Section 10.1] and segment trees [dBCvKO08, Section 10.3, Exercise 10.8]. In the static setting, this readily gives a data structure with construction time $O(n \log^2 n)$ and worst-case query time $O(\log^2 n)$. These data structures can handle deletions by marking certain information as deleted and updating the pointers locally, without rebalancing. Together, this gives a data structure with $T_c(n) = O(n \log^2 n)$, $T_q(n) = O(\log^2 n)$ in the worst case, and $T_d(n) = O(\log^2 n)$ in the worst case; here n denoted the original number of rectangles.

Using any of the two data structures and Theorem 7, the result follows. □

The barrier problem with axis-parallel squares or rectangles can now be solved similarly. For the case of unit squares, it pays off to precompute the squares intersected by each boundary of the strip. For arbitrary squares or rectangles, we could treat each boundary as a rectangle.

Corollary 13. *The barrier resilience problem with n axis-parallel squares of unit side length can be solved in $O(n^{3/2} \log n)$ time. For arbitrary squares or rectangles, the running time becomes $O(n^{3/2} \log^2 n)$.*

The reachability graph $G_{\mathcal{U}}^{\rightarrow}$ can be defined also for sets $\mathcal{U}$ of axis-parallel squares: there is a directed edge from square S to square S' if S contains the center of S'.

Corollary 14. *Let $\mathcal{U}$ be a set of n axis-parallel squares in the plane and let s and t be two of the squres. We can find in $O(n^{3/2} \log^2 n)$ time the minimum s-t cut in the directed transmission graph $G_{\mathcal{U}}^{\rightarrow}$.*

Proof. Use the semi-dynamic data structure to report a point contained in a query square that is based on range trees [dBCvKO08, Section 5.3], as sketched in the proof of Corollary 12. □

4 Conclusion

We have shown how to combine the classic maximum-flow algorithm of Even and Tarjan [ET75] with recent geometric data structures in order to find a minimum s-t cut in geometric intersection graphs. Even though we follow along the lines of the classic algorithms, the details for an efficient implementation in the geometric setting are quite subtle and show an interesting interplay between geometric and combinatorial algorithms.

Our work raises the question whether similar "geometric" versions are possible for other, more advanced, network flow algorithms such as the one by Goldberg and Rao [GR99]. Similarly, it is an interesting challenge to adapt algorithms for other combinatorial graph optimization problems to the geometric setting. For a recent example that considers the maximum matching problem, see [BCM20].

Finally, we cannot resist mentioning the tantalizing open problem of settling the complexity status of the general barrier resilience problem [KLA07]. Unlike for the strip version, we do not know any polynomial time algorithm for it. On the other hand, up to now, all attempts at a proof of NP-hardness have failed. An answer to this question would be most welcome.

References

[ACGK17] Helmut Alt, Sergio Cabello, Panos Giannopoulos, and Christian Knauer. Minimum cell connection in line segment arrangements. *Int. J. Comput. Geom. Appl.*, 27(3):159–176, 2017.

[AM95] Pankaj K. Agarwal and Jiří Matoušek. Dynamic half-space range reporting and its applications. *Algorithmica*, 13(4):325–345, 1995.

[BCM20] Édouard Bonnet, Sergio Cabello, and Wolfgang Mulzer. Maximum matchings in geometric intersection graphs. In *37th International Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 154 of *LIPIcs*, pages 31:1–31:17, 2020.

[BK09] Sergey Bereg and David G. Kirkpatrick. Approximating barrier resilience in wireless sensor networks. In *5th Int. Workshop on Algorithmic Aspects of Wireless Sensor Networks Workshop, (ALGOSENSORS)*, pages 29–40, 2009.

[CC20] Sergio Cabello and Éric Colin de Verdière. Hardness of minimum barrier shrinkage and minimum installation path. *Theor. Comput. Sci.*, 835:120–133, 2020.

[Cha10] Timothy M. Chan. A dynamic data structure for 3-d convex hulls and 2-d nearest neighbor queries. *J. ACM*, 57(3):16:1–16:15, 2010.

[Cha19] Timothy M. Chan. Dynamic geometric data structures via shallow cuttings. In *35th International Symposium on Computational Geometry (SoCG)*, volume 129 of *LIPIcs*, pages 24:1–24:13, 2019.

[CJLM20] Sergio Cabello, Kshitij Jain, Anna Lubiw, and Debajyoti Mondal. Minimum shared-power edge cut. *Networks*, 75(3):321–333, 2020.

[CK14] 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.

[CT16] Timothy M. Chan and Konstantinos Tsakalidis. Optimal deterministic algorithms for 2-d and 3-d shallow cuttings. *Discrete & Computational Geometry*, 56(4):866–881, 2016.

[dBCvKO08] Mark de Berg, Otfried Cheong, Marc J. van Kreveld, and Mark H. Overmars. *Computational geometry: algorithms and applications, 3rd Edition*. Springer, 2008.

[Din70] Yefim A. Dinitz. Algorithm for solution of a problem of maximum flow in a network with power estimation. *Soviet Mathematics Doklady*, 11:1277–1280, 1970.

[Din06] Yefim A. Dinitz. Dinitz' algorithm: The original version and Even's version. In *Theoretical Computer Science, Essays in Memory of Shimon Even*, volume 3895 of *Lecture Notes in Computer Science*, pages 218–240. Springer, 2006.

[Ede80] Herbert Edelsbrunner. Dynamic data structures for orthogonal intersection queries. Technical Report F59, Graz Univ. Technology, Austria, 1980. Available at `http://pub.ist.ac.at/~edels/Papers/1980-01-R-OrthogonalIntersectionQueries.pdf`.

[EIK01] Alon Efrat, Alon Itai, and Matthew J. Katz. Geometry helps in bottleneck matching and related problems. *Algorithmica*, 31(1):1–28, 2001.

[EL20] Eduard Eiben and Daniel Lokshtanov. Removing connected obstacles in the plane is FPT. In *36th International Symposium on Computational Geometry (SoCG)*, volume 164 of *LIPIcs*, pages 39:1–39:14, 2020.

[ET75] Shimon Even and Robert E. Tarjan. Network flow and testing graph connectivity. *SIAM J. Comput.*, 4(4):507–518, 1975.

[Eve79] Shimon Even. *Graph Algorithms*. W. H. Freeman & Co., New York, NY, USA, 1979.

[GG11] Jie Gao and Leonidas Guibas. Geometric algorithms for sensor networks. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 370(1958):27–51, 2011.

[GR99] Andrew V. Goldberg and Satish Rao. Flows in undirected unit capacity networks. *SIAM J. Discret. Math.*, 12(1):1–5, 1999.

[HS95] Mark L. Huson and Arunabha Sen. Broadcast scheduling algorithms for radio networks. In *IEEE MILCOM '95*, volume 2, pages 647–651 vol.2, 1995.

[Kar73] Alexander V. Karzanov. O nakhozhdenii maksimal'nogo potoka v setyakh spetsial'nogo vida i nekotorykh prilozheniyakh. *Matematicheskie Voprosy Upravleniya Proizvodstvom (Mathematical Issues of Production Control)*, pages 81–94, 1973. A translation by the author with the title "On finding a maximum flow in a network with special structure and some applications" is available at http://alexander-karzanov.net/ScannedOld/73_spec-net-flow_transl.pdf.

[KLA07] Santosh Kumar, Ten H. Lai, and Anish Arora. Barrier coverage with wireless sensors. *Wireless Networks*, 13(6):817–834, 2007.

[KLSS18] 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.

[KMR+20] Haim Kaplan, Wolfgang Mulzer, Liam Roditty, Paul Seiferth, and Micha Sharir. Dynamic planar Voronoi diagrams for general distance functions and their algorithmic applications. *Discrete Comput. Geom.*, 64(3):838–904, 2020.

[KV10] Bernhard Korte and Jens Vygen. *Combinatorial Optimization: Theory and Algorithms*, volume 21 of *Algorithms and Combinatorics*. Springer, 4th edition, 2010.

[Liu20] Chih-Hung Liu. Nearly optimal planar k-nearest neighbors queries under general distance functions. In *31st ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2842–2859, 2020.

[Mąd13] Aleksander Mądry. Navigating central path with electrical flows: From flows to matchings, and back. In *54th IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 253–262. IEEE Computer Society, 2013.

[PR10] David Peleg and Liam Roditty. Localized spanner construction for ad hoc networks with variable transmission range. *ACM Trans. Sen. Netw.*, 7(3):25:1–25:14, 2010.

[TK11] Kuan-Chieh Robert Tseng and David G. Kirkpatrick. On barrier resilience of sensor networks. In *7th Int. Workshop on Algorithmic Aspects of Wireless Sensor Networks Workshop, (ALGOSENSORS)*, pages 130–144, 2011.

[ZG04] Feng Zhao and Leonidas Guibas. *Wireless Sensor Networks: An Information Processing Approach*. Elsevier/Morgan-Kaufmann, 2004.