Feedback Vertex Set on Geometric Intersection Graphs

Shinwoo An ✉
POSTECH, Pohang, South Korea

Eunjin Oh ✉
POSTECH, Pohang, South Korea

Abstract

In this paper, we present an algorithm for computing a feedback vertex set of a unit disk graph of size k, if it exists, which runs in time $2^{O(\sqrt{k})}(n + m)$, where n and m denote the numbers of vertices and edges, respectively. This improves the $2^{O(\sqrt{k}\log k)}n^{O(1)}$-time algorithm for this problem on unit disk graphs by Fomin et al. [ICALP 2017]. Moreover, our algorithm is optimal assuming the exponential-time hypothesis. Also, our algorithm can be extended to handle geometric intersection graphs of similarly sized fat objects without increasing the running time.

1 Introduction

The FEEDBACK VERTEX SET problem is a classical and fundamental graph problem, which is one of Karp's 21 NP-complete problems. Given an undirected graph $G = (V, E)$ and an integer k, the goal is to find a set S of vertices of size k such that every cycle of G contains at least one vertex of S. In other words, this problem asks to find a set S of vertices of size k whose removal from G results in a forest. This problem has been studied extensively from the viewpoint of exact exponential-time algorithms [13], parameterized algorithms [6, 18], and approximation algorithms [1].

In this paper, we study the FEEDBACK VERTEX SET problem from the viewpoint of parameterized algorithms. When the parameter is the size k of a feedback vertex set, the best known parameterized algorithm takes $3.62^k n^{O(1)}$ time [18]. On the other hand, it is known that no algorithm for FEEDBACK VERTEX SET runs in $2^{o(k)}n^{O(1)}$ time assuming the exponential-time hypothesis (ETH) [8]. For special classes of graphs such as planar graphs and H-minor-free graphs for any fixed H, there are $2^{O(\sqrt{k})}n^{O(1)}$-time algorithms for the FEEDBACK VERTEX SET problem [10]. Moreover, for planar graphs, FEEDBACK VERTEX SET admits a linear kernel [4].

We present a subexponential-time algorithm for FEEDBACK VERTEX SET on *geometric intersection graphs*, which can be considered as a natural generalization of planar graphs. Consider a set F of geometric objects (for example, disks and polygons) in the plane. The *intersection graph* $G[F]$ for F is defined as the undirected graph whose vertices correspond to the objects in F such that two vertices are connected by an edge if and only if the two objects corresponding to them intersect. In the case that F is a set of disks, its intersection graph is called a *disk graph*, which has been studied extensively for various algorithmic problems [5, 7, 17]. It can be used as a model for broadcast networks: The disks of F

represent transmitter-receiver stations with the same transmission power. A planar graph can be represented as a disk graph, and thus the class of disk graphs is a generalization of the class of planar graphs.

Previous Work.   Prior to our work, the best known algorithm for FEEDBACK VERTEX SET parameterized by the size k of a feedback vertex set on unit disk graphs takes $2^{O(\sqrt{k}\log k)}n^{O(1)}$ time [14, 16]. Since the best known lower bound on the computation time is $2^{o(\sqrt{n})}$ assuming the exponential-time hypothesis (ETH) [9], it is a natural question if FEEDBACK VERTEX SET on unit disk graphs can be solved optimally. De Berg et al. [9] presented an (non-parameterized) ETH-tight algorithm for this problem, which runs in $2^{O(\sqrt{n})}$ time. However, it was not known if there is an ETH-tight parameterized algorithm for FEEDBACK VERTEX SET on unit disk graphs.

Recently, several NP-complete problems have been studied for unit disk graphs (and geometric intersection graphs) from the viewpoint of parameterized algorithms, for example, the STEINER TREE, FEEDBACK VERTEX SET, VERTEX COVER, k-PATH and CYCLE PACKING problems [2, 14, 19]. In the case of VERTEX COVER, the work by de Berg et al. [9] implies an ETH-tight parameterized algorithm. Also, in the case of k-PATH problem, Fomin et al. [15] presented an ETH-tight parameterized algorithm which runs in $2^{O(\sqrt{k})}O(n+m)$ time. However, for the other problems, there is a gap between the running time of the best known algorithms and the best known lower bounds.

Our Result.   In this paper, we present an ETH-tight parameterized algorithm for the FEEDBACK VERTEX SET problem on unit disk graphs, which runs in $2^{O(\sqrt{k})}(n+m)$ time, where n and m denote the numbers of vertices and edges, respectively. This improves the $2^{O(\sqrt{k}\log k)}n^{O(1)}$-time algorithm for this problem on unit disk graphs by Fomin et al. [14]. Moreover, unlike the algorithm in [14], our algorithm works on the graph itself and do not require the geometric representation. Also, our algorithm indeed handles geometric intersection graphs of similarly sized fat objects in the plane without increasing the running time, which will be defined in Section 2.1.

2   Preliminaries

For a graph $G = (V, E)$, let $V(G)$ and $E(G)$ denote the sets of vertices and edges, respectively. For a subset S of V, let $G[S]$ be the subgraph of G induced by S. Also, let $G - S$ be the subgraph of G induced by $V - S$.

2.1   Geometric Intersection Graphs

For a constant $0 < \alpha < 1$, an object $o \subseteq \mathbb{R}^2$ is said to be α-*fat* if there are two disks B_1 and B_2 with $B_1 \subseteq o \subseteq B_2$ such that the radius ratio of B_1 to B_2 is at least α.[1] See Figure 1(a). For example, a disk is a 1-fat object, and a square is a $1/\sqrt{2}$-fat object. We call a set F of α-fat objects a *similarly sized set* if the ratio of the largest diameter to the smallest diameter of the objects in F is bounded by a fixed constant γ. For an α-fat object o of F, there are two *concentric* disks B_3 and B_4 with $B_3 \subseteq o \subseteq B_4$ such that the radius ratio of B_3 and B_4 is at least $\alpha/2$. We consider the center of B_3 and B_4 as the *center* of o. For convenience, we assume that the smallest diameter of the objects of F is one.

---

[1] An *object* is a point set in the plane, which is not necessarily connected.
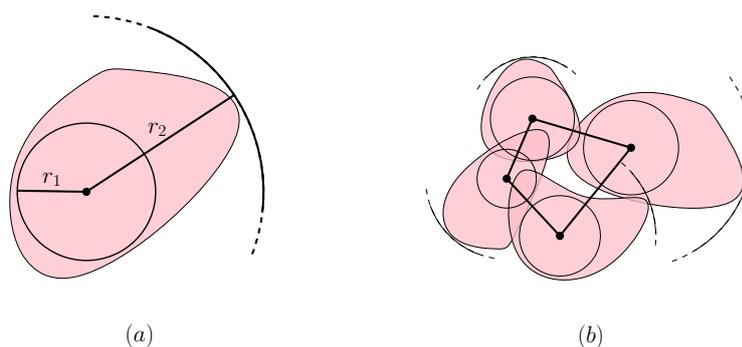
Figure 1 (a) A $\frac{r_1}{r_2}$-fat object. (b) The drawing of the geometric intersection graph $G[F]$.

The *intersection graph* $G[F]$ of a set F of objects in $\mathbb{R}^2$ is defined as the graph whose vertex set is F and two vertices are connected by and edge if and only if their corresponding objects intersect. The *drawing* of the intersection graph $G[F]$ is a representation of $G[F]$ in the plane such that the vertices lie on the centers of the objects of F and the edges are drawn as line segments. See Figure 1(b). We sometimes use an intersection graph $G[F]$ and its drawing interchangeably if it is clear from the context.

In this paper, we focus on geometric intersection graphs of objects in the *plane* only. Because FEEDBACK VERTEX SET on unit ball graphs in $\mathbb{R}^3$ has no subexponential-time algorithm parameterized by k unless ETH fails [16].

2.2 Tree Decomposition and Weighted Width

A *tree decomposition* of a graph $G = (V, E)$ is defined as a pair (T, β), where T is a tree and β is a mapping from nodes of T to subsets of V (called bags) with the following properties. Let $\mathcal{B} := \{\beta(u) : u \in V(T)\}$ be the set of bags of T.

1. For any vertex $u \in V$, there is at least one bag in $\mathcal{B}$ which contains u;
2. For any edge $(u, v) \in E$, there is at least one bag in $\mathcal{B}$ which contains both u and v.
3. For any vertex $u \in V$, the subset of bags of $\mathcal{B}$ containing u forms a subtree of T.

The *width* of a tree decomposition is defined as the size of its largest bag minus one, and the *treewidth* of G is the minimum width among the tree decompositions of G. In this paper, as in [9], we use the notion of *weighted treewidth* introduced by [20]. Here, assume that each vertex v has its weight. The weight of each bag is defined as the sum of the weights of the vertices in the bag, and the *weighted width* of tree decomposition is defined as the maximum weight of the bags. The *weighted treewidth* of a graph G is the minimum weighted width among the tree decompositions of G.

2.3 κ-Partition and $\mathcal{P}$-Contraction

We use the concept of κ-*partitions* and $\mathcal{P}$-*contractions* introduced by de Berg et al. [9]. Let $G[F] = (V, E)$ be a geometric intersection graph of a set F of similarly sized α-fat objects. A κ-*partition* of G is a partition $\mathcal{P} = (P_1, ..., P_t)$ of V such that $G[P_i]$ is connected and is the union of at most κ cliques for every partition class P_i. Given a κ-partition $\mathcal{P}$, we consider a graph obtained by contracting the vertices in the same partition class to a single vertex and removing all loops and parallel edges. We call the resulting graph the $\mathcal{P}$-*contraction* of G and denote it by $G_{\mathcal{P}}$. The weight of each vertex of $G_{\mathcal{P}}$ is defined as $\lceil \log |P_i| \rceil + 1$, where P_i denotes the partition class of $\mathcal{P}$ defining the vertex.

A *greedy partition* is a notion from De berg et al. [9]. For the given geometric intersection graph $G[F]$, they iteratively pick one geometric object, then construct a partition class which consist of the objects adjacent to the picked object. They denote the final partition by the greedy partition. Since all objects are α-fat, the greedy partition is κ-partition of $G[F]$ and moreover has maximum degree δ for constants κ and δ. Given a greedy partition $\mathcal{P}$, they showed that the weighted treewidth of $G_{\mathcal{P}}$ is $O(\sqrt{n})$. Moreover, they presented a $2^{O(\sqrt{n})}$-time algorithm for computing a tree decomposition of $G_{\mathcal{P}}$ of weighted width $O(\sqrt{n})$.

2.4   Overview of Our Algorithm

Our algorithm consists of two steps: computing a (weighted) tree decomposition of $G_{\mathcal{P}}$, and then using a dynamic programming on the tree decomposition. As in [9], we first compute a greedy partition. Then we show that the weighted treewidth of $G_{\mathcal{P}}$ is $O(\sqrt{k})$ if (G, k) is a **yes**-instance. To do this, we first show that the weighted treewidth of $G_{\mathcal{P}}$ is $O(\sqrt{k})$ if G has $O(\sqrt{k})$ vertices of degree at least three in Section 3. Then we show that G has $O(\sqrt{k})$ vertices of degree at least three in Section 4.1.

Using this fact, we compute a constant approximation to the weighted treewidth of $G_{\mathcal{P}}$. If it is $\omega(\sqrt{k})$, we conclude that (G, k) is a **no**-instance immediately. Otherwise, we compute a feedback vertex set of size k, if it exists, using dynamic programming on a tree decomposition of $G_{\mathcal{P}}$ of weighted width $O(\sqrt{k})$. The dynamic programming algorithm is described in Section 4.2.

## 3   Tree Decomposition and Weighted Treewidth

In this section, we present the first step of our algorithm: computing a tree decomposition of $G_{\mathcal{P}}$ of weighted width $O(\sqrt{|U|})$ for a greedy partition $\mathcal{P}$ of a geometric intersection graph G, where U denotes the set of vertices of G of degree at least three in G. More precisely, we prove the following theorem.

▶ **Theorem 1.** *Let G be an intersection graph of similarly sized α-fat objects with n vertices and m edges, and let $\mathcal{P}$ be a greedy partition of G. Then the weighted treewidth of $G_{\mathcal{P}}$ is $O(\sqrt{|U|})$, where U denotes the set of vertices of G of degree at least three in G. Moreover, we can compute a tree decomposition of $G_{\mathcal{P}}$ of weighted width $O(\sqrt{|U|})$ in $2^{O(\sqrt{|U|})}(n + m)$ time without using a geometric representation of G.*

In the following, let G be an intersection graph of similarly sized α-fat objects, $\mathcal{P}$ be a greedy partition of G, and U be the set of vertices of G of degree at least three in G. In Section 3.1, we prove the first part of the theorem, and in Section 3.2, we prove the second part of the theorem.

3.1   Weighted Treewidth of the $\mathcal{P}$-contraction

We first show that the weighted treewidth of $G_{\mathcal{P}}$ is $O(\sqrt{|U|})$. To do this, we transform G into a planar graph H. As we did for G, we compute the $\mathcal{P}$-contraction of H, and remove degree-2 vertices of the $\mathcal{P}$-contractions of H and G in a specific way. In this way, we obtain $\bar{H}_{\mathcal{P}}$ and $\bar{G}_{\mathcal{P}}$ of complexity $O(\sqrt{|U|})$. Also, we assign the weight to each vertex of the resulting graphs. Then we transform a balanced separator of $\bar{H}_{\mathcal{P}}$ of weight $O(\sqrt{|U|})$ into a balanced separator of $\bar{G}_{\mathcal{P}}$ of weight $O(\sqrt{|U|})$. Using a weighted balanced separator, we compute a tree decomposition of $\bar{G}_{\mathcal{P}}$ of weighted width $O(\sqrt{|U|})$, and then transform it into a tree decomposition of $G_{\mathcal{P}}$ of weighted width $O(\sqrt{|U|})$.

(a) (b) (c)
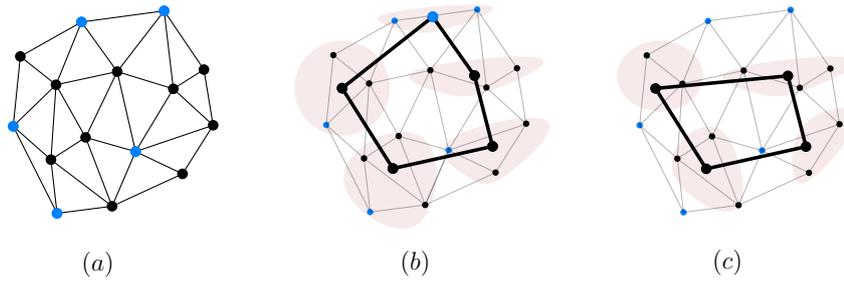
Figure 2 (a) The Delaunay triangulation H of the vertex set of $V(G)$. The vertices of G of degree at most two are colored blue. (b) $\mathcal{P}$ partitions V into five subsets. Each subset of $\mathcal{P}$ is contained in a single pink region. Then the $\mathcal{P}$-contraction of H is the cycle consisting of five edges. (c) Each subset of $\bar{\mathcal{P}}$ consists of the black points contained in a single pink region.

Delaunay triangulation and its contraction: H and $H_{\mathcal{P}}$.   Consider the Delaunay triangulation H of the point set $V(G)$. Here, we consider $V(G)$ as the set of the centers of the objects defining G. We consider the Delaunay triangulation H as an edge-weighted plane graph such that the *length* of each edge is the Euclidean distance between their endpoints. The Delaunay triangulation H is a *5.08-spanner* of the complete Euclidean graph defined by $V(G)$ [12]. That is, for any two points u and v of $V(G)$, the length of the shortest path in H between u and v is at most 5.08 times their Euclidean distance.

Notice that H might contain an edge which is not an edge of G. However, the following still holds as G is a subgraph of the compete Euclidean graph. For any edge (u,v) in G, there is a u-v path in H consisting of $5.08\|uv\|$ edges. Recall that α is the measure for the fatness of the objects defining G, which is a constant. Let $H_{\mathcal{P}}$ be the $\mathcal{P}$-contraction of H. Note that $V(G_{\mathcal{P}}) = V(H_{\mathcal{P}})$. For a vertex v of $G_{\mathcal{P}}$ or $H_{\mathcal{P}}$, we let $P(v)$ to denote the partition class of $\mathcal{P}$ corresponding to v. See Figure 2(b).

Partition $\bar{\mathcal{P}}$ of U: $\bar{G}_{\mathcal{P}}$ and $\bar{H}_{\mathcal{P}}$.   The size of $G_{\mathcal{P}}$ and $H_{\mathcal{P}}$ might be $\Theta(n)$ even if $|U|$ is small. Recall that U is the set of vertices of G of degree at least three in G. To obtain a balanced separator of $G_{\mathcal{P}}$ of small weight, we compute a new graph $\bar{G}_{\mathcal{P}}$ and $\bar{H}_{\mathcal{P}}$ of size $O(|U|)$ as follows.

Let $\mathcal{P} = (P_1, \ldots, P_t)$, which is a partition of $V(G)$. Using this, we consider the partition $\bar{\mathcal{P}} = (\bar{P}_1, \ldots \bar{P}_t)$ of U such that $\bar{P}_i = P_i \cap U$. Note that $\bar{P}_i = \emptyset$ if every vertex of P_i has degree at most two in G. We first ignore the empty partition classes from $\bar{\mathcal{P}}$, and let $\bar{G}_{\mathcal{P}}$ and $\bar{H}_{\mathcal{P}}$ be the $\bar{\mathcal{P}}$-contraction of G and the $\bar{\mathcal{P}}$-contraction of H, respectively. We add several edges to $\bar{G}_{\mathcal{P}}$ and $\bar{H}_{\mathcal{P}}$ by considering the empty partition classes of $\bar{\mathcal{P}}$ as follows. Since $G[P_i]$ is connected and consists of κ cliques, $G[P_i]$ is a simple cycle or a simple path if $\bar{P}_i = \emptyset$. Let $P_\emptyset$ be the union of P_i's for all indices with $\bar{P}_i = \emptyset$. Then each connected component of $G[P_\emptyset]$ is also a simple cycle or a simple path. If the connected component is a simple cycle, no vertex in the component is connected by a vertex of U in G. Otherwise, the endpoints of the simple path, say p and q, are contained in U. In this case, we connect the vertices of $\bar{G}_{\mathcal{P}}$ (and $\bar{H}_{\mathcal{P}}$) by an edge unless this edge forms a loop. See Figure 2(c).

For a vertex $\bar{v}$ of $\bar{G}_{\mathcal{P}}$ or $\bar{H}_{\mathcal{P}}$, we let $\bar{P}(\bar{v})$ be the partition class of $\bar{\mathcal{P}}$ corresponding to $\bar{v}$. By construction, $\bar{P}(\bar{v})$ is not empty for any vertex $\bar{v}$. Each vertex $\bar{v}$ of $\bar{G}_{\mathcal{P}}$ has a *g-weight* $g(\bar{v}) = \lceil \log |\bar{P}(\bar{v})| \rceil + 1$. Also, each vertex $\bar{v}$ of $\bar{H}_{\mathcal{P}}$ has a *h-weight* $h(\bar{v}) = \sum_{\bar{u} \in N(\bar{v})} (\lceil \log |\bar{P}(\bar{u})| \rceil + 1)$, where $N(\bar{v})$ is the set of vertices $\bar{u}$ of $\bar{H}_{\mathcal{P}}$ such that there is a $\bar{u}$-$\bar{v}$ path in $\bar{H}_{\mathcal{P}}$ consisting of at most $t = (\delta + 1)(10.16(\alpha\gamma)^{-1})^2\pi$ vertices.
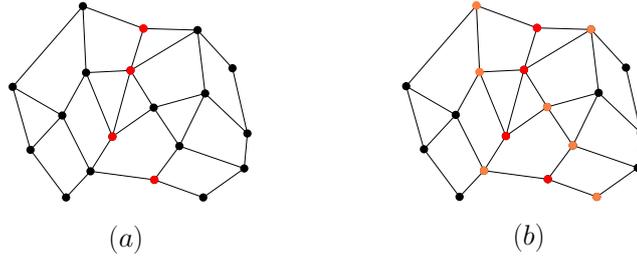
Figure 3 (a) A balanced separator S of $\bar{H}_{\mathcal{P}}$ consists of the red vertices. (b) A balanced separator S' of $\bar{G}_{\mathcal{P}}$ consists of the red vertices and the orange vertices. The vertices whose distance from the red vertices are at most $t = 1$ are colored with orange.

▶ **Observation 2.** *A vertex $\bar{v}$ of $\bar{H}_{\mathcal{P}}$ has a constant degree, and thus the size of $N(\bar{v})$ is $O(1)$.*

Balanced separator of $\bar{H}_{\mathcal{P}}$ of small h-weights. For a vertex-weighted graph G' and a subgraph H', we denote the sum of the weights of the vertices of H' by the *weight* of H'. a subset S of $V(G')$ is called a *balanced separator* of G' if the weight of each connected component of $G' - S$ is at most $2/3$ of the weight of G'. The *weight* of a balanced separator S of G' is the sum of the weights of the vertices of S. Djidjev [11] showed that a planar graph G' has a balanced separator of weight $O(\sqrt{\sum_{v \in V} w(v)^2})$, where $w(v)$ is the weight of a vertex v.

The following lemma implies that $\bar{H}_{\mathcal{P}}$ has a balanced separator of h-weight $O(\sqrt{|U|})$.

▶ **Lemma 3.** *The sum of $h(\bar{v})^2$ for all vertices $\bar{v}$ of $\bar{H}_{\mathcal{P}}$ is $O(|U|)$.*

Proof. Let $\bar{v}$ be a vertex of $\bar{H}_{\mathcal{P}}$. The h-weight of $\bar{v}$ is the sum of $g(\bar{u}) = (\lceil \log |\bar{P}(\bar{u})| \rceil + 1)$ for all vertices $\bar{u}$ of $N(\bar{v})$ by definition. By Cauchy–Schwarz inequality, $(\sum_{\bar{u} \in N(\bar{v})} g(\bar{u}))^2$ is at most $|N(\bar{v})|(\sum_{\bar{u} \in N(\bar{v})} g(\bar{u})^2)$. Since the size of $N(\bar{v})$ is at most a constant, say c, we have the following inequality.

$$\sum_{\bar{v} \in V} h(\bar{v})^2 \leq c \cdot \sum_{\bar{v} \in V} \sum_{\bar{u} \in N(\bar{v})} g(\bar{u})^2 = c \cdot \sum_{\bar{u} \in V} \sum_{\bar{v} \in N(\bar{u})} g(\bar{u})^2,$$

where V denotes the vertex set of $\bar{H}_{\mathcal{P}}$. Since $\sum_{\bar{v} \in N(\bar{u})} g(\bar{u})^2 = |N(\bar{u})| g(\bar{u})^2 \leq c \cdot g(\bar{u})^2$, we finally have the following.

$$\sum_{\bar{v} \in V} h(\bar{v})^2 \leq c^2 \cdot \sum_{\bar{u} \in V} g(\bar{u})^2 = c^2 \cdot \sum_{\bar{u} \in V} (\lceil \log |\bar{P}(\bar{u})| \rceil + 1)^2 = O(\sum_{\bar{u} \in V} |\bar{P}(\bar{u})|) = O(|U|).$$

Here, the second equality holds because $2x \geq (\log x + 1)^2$ for all values $x \geq 1$. Also, the last equality holds because $\bar{\mathcal{P}}$ is a partition of U.   ◀

Balanced separator of $\bar{G}_{\mathcal{P}}$ of small g-weight. Let S be a balanced separator of $\bar{H}_{\mathcal{P}}$ of h-weight $O(\sqrt{|U|})$. Let S' be the union of $N(\bar{v})$ for all $\bar{v} \in S$. Recall that $|V(\bar{G}_{\mathcal{P}})| = |V(\bar{H}_{\mathcal{P}})|$. The sum of the g-weights of the vertices of S' is $O(\sqrt{|U|})$ by definition of the g- and h-weights. Therefore, it suffices to show that the weight of each connected component of $\bar{G}_{\mathcal{P}} - S'$ is at most $2/3$ of the weight of $\bar{G}_{\mathcal{P}}$.

▶ **Lemma 4.** *For any edge (u, v) of $G_{\mathcal{P}}$, there is a u-v path in $H_{\mathcal{P}}$ consisting of t vertices.*[2]

Proof. Consider an edge (u, v) of $G_{\mathcal{P}}$. By construction, there is an edge (p, q) in G such that $p \in P(u)$ and $q \in P(v)$. Recall that Delaunay triangulation is a 5.08-spanner of the complete Euclidean graph. Therefore, there exists a p-q path τ in H such that the sum of the lengths (Euclidean distance between two endpoints) of the edges is at most $5.08\|pq\|$.

We claim that τ intersects at most $(\delta + 1)(10.16(\alpha\gamma)^{-1})^2\pi$ partition classes of $\mathcal{P}$, where δ is the maximum degree of $G_{\mathcal{P}}$, α is the measure for the fatness of the objects, and γ is the ratio of the largest and smallest diameters of the objects. Consider a *grid* of the plane, which is a partition of the plane into axis-parallel squares (called *cells*) with diameter $1/\sqrt{2}$. Recall that the smallest diameter of the objects defining G is one. For a cell σ, we denote the set of vertices of G contained in σ by P_σ. By definition, P_σ forms a clique in G. Since the maximum degree δ of $G_{\mathcal{P}}$ is constant, at most $\delta + 1$ partition classes of $\mathcal{P}$ has their vertices in P_σ. Note that every object is contained in a ball of radius $(\alpha\gamma)^{-1}$. Therefore, $\|pq\| \leq 2(\alpha\gamma)^{-1}$. Then, the Euclidean length of τ is at most $10.16(\alpha\gamma)^{-1}$. This implies that τ is contained in a ball of radius $10.16(\alpha\gamma)^{-1}$ centered at p. In other words, τ intersects at most $(10.16(\alpha\gamma)^{-1})^2\pi$ cells. Since each cell intersects at most $\delta + 1$ partition classes of $\mathcal{P}$, τ intersects $(\delta + 1)(10.16(\alpha\gamma)^{-1})^2\pi$ partition classes. Let $t = (\delta + 1)(10.16(\alpha\gamma)^{-1})^2\pi$.

Then we consider a u-v path in $H_{\mathcal{P}}$ obtained by replacing each point in τ to its contracted point in $H_{\mathcal{P}}$. This path now consists of t different vertices of $H_{\mathcal{P}}$, but it is not necessarily simple. We can obtain a simple path by removing duplicated subpaths that have same endpoints. Then we obtain a simple path of consisting of t vertices. This completes the proof. ◀

▶ **Lemma 5.** *For any edge $(\bar{u}, \bar{v})$ of $\bar{G}_{\mathcal{P}}$, there is a $\bar{u}$-$\bar{v}$ path in $\bar{H}_{\mathcal{P}}$ consisting of t vertices.*

Proof. We consider an edge $(\bar{u}, \bar{v})$ of $\bar{G}_{\mathcal{P}}$. Let u and v be the vertices of $G_{\mathcal{P}}$ with $P(u) \cap U = \bar{P}(\bar{u})$ and $P(v) \cap U = \bar{P}(\bar{v})$. By construction, either there is an edge (u, v) in $G_{\mathcal{P}}$, or there is a u-v path in $G_{\mathcal{P}}$ such that no internal vertex is contained in U. We consider these two cases separately.

- **Case 1. (u, v) is an edge of $G_{\mathcal{P}}$.** In this case, there is a u-v path τ_1 in $H_{\mathcal{P}}$ consisting of t vertices by Lemma 4. Let I be the sequence of the indices of the partition classes of $\mathcal{P}$ corresponding to the vertices of τ_1. Then let I' be the subsequence of I consisting of the indices j with $\bar{P}_j \neq \emptyset$. If $I = I'$, then $\bar{H}_{\mathcal{P}}$ has a $\bar{u}$-$\bar{v}$ path consisting of s vertices. Otherwise, since the internal vertices of τ_1 has degree at least two in $H_{\mathcal{P}}$, the vertices of $\bar{H}_{\mathcal{P}}$ corresponding to $\bar{P}(i)$ and $\bar{P}(j)$ are connected by an edge for every consecutive indices i and j of I'. Therefore, there exists a $\bar{u}$-$\bar{v}$ path in $\bar{H}_{\mathcal{P}}$ consisting of t vertices.
- **Case 2. There is a u-v path τ' in $G_{\mathcal{P}}$ with no internal vertices in U.** Recall that $V(G_{\mathcal{P}}) = V(H_{\mathcal{P}})$. If τ' entirely exists in $H_{\mathcal{P}}$, there is an edge $(\bar{u}, \bar{v})$ in $\bar{H}_{\mathcal{P}}$, and thus we are done. If it is not the case, let (u', v') be an edge in τ' which does not exist in $H_{\mathcal{P}}$. By Lemma 4, there is a u'-v' path in $H_{\mathcal{P}}$ consisting of t vertices. Since the internal vertices of τ' have degree exactly two, the u'-v' path contains both u and v. Therefore, the subgraph of the u'-v' path lying between u and v is a u-v path in $H_{\mathcal{P}}$ consisting of at most t vertices. Then similarly to Case 1, we can show that there is a $\bar{u}$-$\bar{v}$ path in $\bar{H}_{\mathcal{P}}$ consisting of t vertices.

Therefore, for an edge $(\bar{u}, \bar{v})$ in $\bar{G}_{\mathcal{P}}$, there is a $\bar{u}$-$\bar{v}$ path in $\bar{H}_{\mathcal{P}}$ consisting of at most t vertices. This completes the proof. ◀

---

[2] Recall that $t = (\delta + 1)(10.16(\alpha\gamma)^{-1})^2\pi$.

▶ **Lemma 6.** *The weight of each connected component of $\bar{G}_\mathcal{P} - S'$ is at most 2/3 of the weight of $\bar{G}_\mathcal{P}$.*

Proof. We show that each connected component of $\bar{G}_\mathcal{P} - S'$ is a subset of a connected component of $\bar{H}_\mathcal{P} - S$. This implies that each connected component of $\bar{G}_\mathcal{P} - S'$ consists of at most $2|V(\bar{G}_\mathcal{P})|/3$ vertices since S is a balanced separator of $G_\mathcal{P}$, and $V(\bar{G}_\mathcal{P}) = V(\bar{H}_\mathcal{P})$.

To show this, consider an edge (a,b) of $\bar{G}_\mathcal{P} - S'$. We claim that a and b are contained in the same connected component of $\bar{H}_\mathcal{P} - S$. By Lemma 5, there is an a-b path in $\bar{H}_\mathcal{P}$ consisting of t vertices. If this path contains a vertex of S, then a and b are contained in S' by construction, which makes a contradiction. Therefore, an a-b path in $\bar{H}_\mathcal{P}$ does not intersects S, which means that a and b are contained in the same connected component of $\bar{H}_\mathcal{P} - S$.

Therefore, each connected component of $\bar{G}_\mathcal{P} - S'$ is a subset of the connected component of $\bar{H}_\mathcal{P} - S$, and the lemma holds.   ◀

Tree decomposition of $\bar{G}_\mathcal{P}$.   To construct a tree decomposition of $\bar{G}_\mathcal{P}$, we observe that any subgraph of $\bar{G}_\mathcal{P}$ also has a balanced separator of small weight. Let A be a connected component of $\bar{G}_\mathcal{P} - S'$. Then consider the subgraph $\bar{H}_\mathcal{P}[A]$ of $\bar{H}_\mathcal{P}$ induced by the vertices in A. Since $\bar{H}_\mathcal{P}[A]$ is planar, it has a balanced separator with small h-weight, and thus A has a balanced separator with small g-weight. Therefore, we can recursively compute a balanced separator for every connected component of $\bar{G}_\mathcal{P} - S'$.

We compute a tree decomposition of each connected component of $\bar{G}_\mathcal{P} - S'$ recursively, and then we connect those trees by one additional empty bag, and add the separator to all bags of the resulting tree. Since the weight of the connected components decreases geometrically, the maximum weight of the bags is $O(\sqrt{|U|})$. Therefore, we can compute a tree decomposition (T, β) of $\bar{G}_\mathcal{P}$ of weighted width $O(\sqrt{|U|})$.

Tree decomposition of $G_\mathcal{P}$.   The remaining step is computing a tree decomposition of $G_\mathcal{P}$ from (T, β). Then we start with the pair (T, β), and then add several nodes (and bags) to T and add several vertices to the bags of β as follows.

Let Q be the set of vertices u of $G_\mathcal{P}$ with $P(u) \cap U \neq \emptyset$, and let Q^{c} be the set of vertices of $G_\mathcal{P}$ not in Q. For any vertex v of Q, there is a vertex $\bar{v}$ in $\bar{G}_\mathcal{P}$ with $P(v) \cap U = \bar{P}(\bar{v})$, and thus $\bar{v}$ is contained bags of (T, β). We replace $\bar{v}$ with v in the bags of (T, β) containing v.

On the other hand, for a vertex v of Q^{c}, $P(v) \cap U = \emptyset$, and thus no vertex of $\bar{G}_\mathcal{P}$ corresponds to v. Thus we are required to insert $v \in Q$ into T'. Recall that every vertex in Q has degree one or two in G, and thus it also has degree one or two in $G_\mathcal{P}$. Also, $G_\mathcal{P}$ is connected.[3] Therefore, $G_\mathcal{P}[Q]$ consists of a number of simple paths and isolated vertices, all of which are connected to at most two vertices of Q. Thus, there are the following four cases.

▬ **Case 1.** An isolated vertex v is connected to only one vertex $u \in Q$. For a bag $\beta(x)$ of (T, β) containing u, we add a leaf with bag $\{u, v\}$ as a child of x.
▬ **Case 2.** An isolated vertex v is connected to exactly two vertices $u, w \in Q$. By construction, there is an edge (u, w) in $\bar{G}_\mathcal{P}$. Then, there is a bag $\beta(x)$ of T containing both $\bar{u}$ and $\bar{w}$. We add a leaf with bag $\{u, v, w\}$ as a child of x.
▬ **Case 3.** A simple v-v' path $(v_1, ..., v_k)$ is connected to exactly one vertex $u \in Q$. Note that either v or v' is connected to u, but not both. Without loss of generality, we assume

---

[3] We assume that the input intersection graph G is connected, and thus $G_\mathcal{P}$ is connected by construction.

that there is an edge (u, v) in $G_\mathcal{P}$. For a bag $\beta(x)$ of T containing u, we add a node x_1 with bag $\{u, v\}$ as a child of x. For $1 < t < k$, we add a node x_t with bag $\{v_t, v_{t+1}\}$ as a child of x_{t-1}.

- **Case 4.** A simple v-v' path $(v_1, ..., v_k)$ is connected to two vertices $u, w \in Q$. In this case, v is connected to one of u and w, say u, and v' is connected to the other vertex, say w. Since $\bar{G}_\mathcal{P}$ has an edge $(\bar{u}, \bar{w})$, there is a bag $\beta(x)$ of T that contains both u and w, where $\bar{u}$ and $\bar{v}$ are the vertices of $\bar{G}_\mathcal{P}$ corresponding to u and v, respectively. We add a node x_1 with bag (u, w, v, v') as a child of x. For $1 < t < k/2$, we add a node x_t with bag $\{v_t, v_{t+1}, v_{k-t}, v_{k-t+1}\}$ as a child of x_{t-1}.

In this way, (T, β) is a tree decomposition of $G_\mathcal{P}$. Now we analyze the weight of a bag of (T, β). Since we replace $\bar{v}$ with v for a vertex v of Q in the bags of (T, β), the weight of a bag B of (T, β) consisting of vertices of Q increases. Let v be a vertex of Q, and let $\bar{v}_i$ be the vertex of $\bar{G}_\mathcal{P}$ corresponding to v. Since $P(v)$ is composed of at most $O(1)$ cliques, it contains at most $O(1)$ vertices of $V \setminus U$. Note that B contains $O(\sqrt{|U|})$ vertices because the weighted width of the tree decomposition of $\bar{G}_\mathcal{P}$ is $O(\sqrt{|U|})$. Therefore, the weight of B increases by $O(\sqrt{|U|})$ after replacing vertices of $\bar{G}_\mathcal{P}$ with vertices of $G_\mathcal{P}$. Thus, the weight of a bag consisting of vertices of Q is $O(\sqrt{|U|})$.

Now we show that the weight of a bag containing a vertex of Q^{c} is $O(\sqrt{|U|})$. Every vertex $v \in Q^{\mathsf{c}}$ is a contraction of simple paths or simple cycles of length at most $O(1)$ in G. In particular, its weight in $G_\mathcal{P}$ is $O(1)$. Since a bag containing a vertex of Q^{c} consists of at most two vertices of Q^{c} and at most two vertices of Q, the weight of the bag is $O(\sqrt{|U|})$.

3.2 Computing a Tree Decomposition of Small Weighted Width

We can compute a tree decomposition of weighted treewidth $O(\sqrt{|U|})$ of $G_\mathcal{P}$ using the algorithm proposed by de Berg et al. [9], which computes a tree decomposition of weighted width $O(w)$ of $G_\mathcal{P}$ assuming that the weighted treewidth of $G_\mathcal{P}$ is w. This algorithm runs in $2^{O(w)}(n + m)$ time. This algorithm works on the graph itself even if the geometric representation is unknown. In this section, we briefly describe how the algorithm proposed by de Berg et al. [9] for computing such a tree decomposition works.

A *blowup* of a vertex v by an integer t results in a graph where we replace a single vertex v into a clique of size t, in which we connect every vertex to the neighbor vertices of v. We denote the set of vertices in the clique by $\mathcal{B}(v)$. For the graph $G_\mathcal{P}$, we construct an unweighted graph $G_\mathcal{B}$ by blowing up each vertex v by an integer $\lceil g(v) \rceil$ where $g(v) = \log |P(v)| + 1$. We compute a tree decomposition of $G_\mathcal{B}$, denoted by $(T_\mathcal{B}, \sigma_\mathcal{B})$. Then we compute a tree decomposition of $G_\mathcal{P}$ by using the same tree layout of $T_\mathcal{B}$. In particular, we add a vertex $v \in G_\mathcal{P}$ to a bag if and only if the bag in $T_\mathcal{B}$ contains all vertices of $\mathcal{B}(v)$. The Lemma 2.9 and Lemma 2.10 of [9] prove the correctness of this algorithm.

The remaining step is computing a tree decomposition of $G_\mathcal{B}$. We apply a constant-factor approximation algorithm for computing an optimal tree decomposition [8]. By Theorem 1 and Lemma 2.9 of [9], the treewidth of $G_\mathcal{B}$ is $O(\sqrt{|U|})$. Therefore, the algorithm returns a tree decomposition of $G_\mathcal{B}$ of width $O(\sqrt{|U|})$. This algorithm takes a $2^{O(tw(G_\mathcal{B}))}n = 2^{O(\sqrt{|U|})}(n+m)$ time, and therefore, we can compute a tree decomposition of $G_\mathcal{P}$ of weighted width $O(\sqrt{|U|})$ without using the geometric representation of the graph. This completes the proof of Theorem 1.

4 Computing a Feedback Vertex Set

In this section, we present a $2^{O(\sqrt{k})}(n+m)$-time algorithm for finding a feedback vertex set of size k for the intersection graph $G[F] = (V, E)$ of similarly sized fat objects in the plane.

To do this, we first show that G has $O(k)$ vertices of degree at least three if it has a feedback vertex set of size at most k in Section 4.1. Therefore, by Theorem 1, for a greedy partition $\mathcal{P}$ of G, the weighted treewidth of $G_\mathcal{P}$ is $O(\sqrt{k})$ if (G, k) is a **yes**-instance.

To use this fact, we first compute such a greedy partition $\mathcal{P}$ in $O(n + m)$ time using the algorithm in [9]. Then we check if the weighted treewidth of $G_\mathcal{P}$ is $O(\sqrt{k})$ in $2^{O(\sqrt{k})}(n + m)$ time. If it is $\omega(\sqrt{k})$, we conclude that (G, k) is a **no**-instance immediately. Otherwise, we compute a feedback vertex set of size k, if it exists, using dynamic programming on a tree decomposition of $G_\mathcal{P}$ of weighted width $O(\sqrt{k})$. De Berg et al. [9] presented a dynamic programming algorithm for FEEDBACK VERTEX SET which runs in $2^{O(\sqrt{w})}(n + m)$ time, where w is the weighted treewidth of $G_\mathcal{P}$. We can use this algorithm, but for completeness, we briefly describe this algorithm in Section 4.2.

4.1 Weighted Treewidth for a Yes-Instance

We first remove all degree-1 vertices from G and denote it by G'. This reduction is feasible because no cycle encounters a degree-1 vertex. We next show that G' has $O(k)$ vertices of degree at least three if it has a feedback vertex set of size k. We consider a partition of the plane into axis-parallel squares (called *cells*) with diameter $1/\sqrt{2}$, which we call the *grid*. Recall that we let the smallest diameter of the objects defining G be one. For a cell σ, let $P(\sigma)$ be the set of the centers of the objects defining G. By definition, $P(\sigma)$ forms a clique in G. We say a grid cell is a *neighbor* of another grid cell if the smallest distance between two points from the two cells is at most $2\alpha\gamma$, where α is the measure for the fatness of the objects of F, and γ is the ratio of the largest and the smallest diameter of the objects. We say a grid cell σ is *heavy* if $P(\sigma)$ consists of at least three points, and *light*, otherwise. Let P_h denote the set of vertices contained in the heavy cells, and let P_ℓ denote the set of vertices contained in the light cells.

The following lemma implies that $G_\mathcal{P}$ has a weighted treewidth of $O(\sqrt{k})$ if it has a feedback vertex set of size k. The proof of the following lemma is inspired by the proof of [8, Lemma 9.1].

Note that the weighted treewidth of G is at most the weighted treewidth of G' plus one. Also, (G, k) is a **yes**-instance of FEEDBACK VERTEX SET if and only if (G', k) is a **yes**-instance of FEEDBACK VERTEX SET.

▶ **Lemma 7.** G' *has* $O(k)$ *vertices of degree at least three if* (G', k) *is a **yes**-instance of* FEEDBACK VERTEX SET, *where* G' *is the graph obtained from* G *by removing all degree-1 vertices.*

Proof. If G' has a feedback vertex set of size k, the number of heavy cells is at most k, and the size of P_h is at most $3k$. This is because all but two of the vertices contained in the same cell must be contained in a feedback vertex set of G. In the following, we show that the number of vertices of P_ℓ of degree at least three is at most $O(k)$.

To do this, consider the subgraph G_ℓ of G' induced by P_ℓ. We first observe that every vertex of G_ℓ has degree at most $4(\alpha\gamma)^2$ in G_ℓ. (But note that its degree in G' might be larger than $4(\alpha\gamma)^2$.) Moreover, G_ℓ also has a feedback vertex set of size k. Let X be a feedback vertex set of G_ℓ of size k, and let Y be the set of vertices of G_ℓ not contained in X. By definition, the subgraph $G[Y]$ of G_ℓ induced by Y is a forest. The number of vertices

of degree at least three is linear in the number of leaf nodes of the trees. Since G' has no degree-1 vertex, a leaf node of $G[Y]$ is incident to a vertex of $X \cup P_h$ in G'. Each vertex in $X \cup P_h$ is incident to at most $4(\alpha\beta)^2$ vertices of Y. Since $|X \cup P_h|$ is $O(k)$, the total number of vertices of leaf nodes is $O(k)$, and thus, the total number of vertices of degree at least three of G_ℓ (and G') is $O(k)$. ◄

▶ **Lemma 8.** *For a greedy partition $\mathcal{P}$, the weighted treewidth of $G_{\mathcal{P}}$ is $O(\sqrt{k})$ if (G, k) is a* ***yes****-instance of* FEEDBACK VERTEX SET.

4.2 Dynamic Programming

By combining Lemma 7 and Theorem 1, we can obtain a tree decomposition of weighted width $O(\sqrt{k})$ if a geometric intersection graph G has a feedback vertex set of size k. De Berg et al. [9] showed how to compute a feedback vertex set of size k in $2^{O(\sqrt{w})} \operatorname{poly} n$ time, where w denotes the weighted treewidth of $G_{\mathcal{P}}$. In this section, we briefly describe how this algorithm works.

Given a tree decomposition (T, β) of a graph G', algorithm for solving FEEDBACK VERTEX SET computes for each bag $\beta(t)$ and each subset $S \subseteq \beta(t)$ together with the connectivity information η of the vertices of S, the maximum size $c[t, S, \eta]$ of a feedback vertex set $\hat{S}$ of $G[t]$ with $\hat{S} \cap \beta(t) = S$ satisfying the connectivity information η, where $G[t]$ denotes the subgraph of G induced by the vertices appearing in bags in the subtree of T rooted at t.

Let t be a node of a tree decomposition (T, β) of $G_{\mathcal{P}}$. Recall that each vertex v of $\beta(t)$ corresponds to the partition class $P(v)$. Let $X_t = \bigcup_{v \in \beta(t)} P(v)$. A feedback vertex set contains at least $|C| - 2$ vertices of a clique C. Therefore, since from each partition class we can exclude $O(1)$ vertices (at most two vertices from each clique), the number of subsets $\hat{S}$ that need to be considered is at most

$$\prod_{v \in \beta(t)} |P(v)|^2 = \exp\left(\sum_{C \in \beta(t)} 2 \log |P(v)|\right) = 2^{O(\sqrt{k})}.$$

Therefore, we can track connectivity of these subsets by applying the rank-based approach of [3], which allows us to keep the number of equivalence relations considered single-exponential in $O(\sqrt{k})$. Assuming that we have $c[t', \cdot, \cdot]$ for all descendants t' of t in T and their connectivity information, we can compute $c[t, S, \eta]$ in $2^{O(\sqrt{k})}$ time. Since the number of nodes in the tree decomposition is $O(kn)$, the total running time of the dynamic programming algorithm is $2^{O(\sqrt{k})}(n + m)$. Therefore, we have the following theorem.

▶ **Theorem 9.** *Given a intersection graph G of similarly sized fat objects in the plane (without its geometric representation), we can compute a feedback vertex set of size k in $2^{O(\sqrt{k})}(n+m)$ time, if it exists.*

─── **References** ───

1   Ann Becker and Dan Geiger. Optimization of Pearl's method of conditioning and greedy-like approximation algorithms for the vertex feedback set problem. *Artificial Intelligence*, 83(1):167–188, 1996.

2   Sujoy Bhore, Paz Carmi, Sudeshna Kolay, and Meirav Zehavi. Parameterized study of Steiner tree on unit disk graphs. In *Proceedings of the 17th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT 2020)*, volume 162, pages 13:1–13:18, 2020.

3   Hans L. Bodlaender, Marek Cygan, Stefan Kratsch, and Jesper Nederlof. Deterministic single exponential time algorithms for connectivity problems parameterized by treewidth. *Information and Computation*, 243:86–111, 2015. 40th International Colloquium on Automata, Languages and Programming (ICALP 2013).

4   Marthe Bonamy and Łukasz Kowalik. A 13k-kernel for planar feedback vertex set via region decomposition. *Theoretical Computer Science*, 645:25–40, 2016.

5   Sergio Cabello and Miha Jejčič. Shortest paths in intersection graphs of unit disks. *Computational Geometry*, 48(4):360–367, 2015.

6   Yixin Cao, Jianer Chen, and Yang Liu. On feedback vertex set: New measure and new structures. *Algorithmica*, 73(1):63–86, 2015.

7   Timothy M Chan and Dimitrios Skrepetos. Approximate shortest paths and distance oracles in weighted unit-disk graphs. *Journal of Computational Geometry*, 10(2):3–20, 2019.

8   Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Daniel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer Publishing Company, Incorporated, 2015.

9   Mark de Berg, Hans L Bodlaender, Sándor Kisfaludi-Bak, Dániel Marx, and Tom C van der Zanden. A framework for exponential-time-hypothesis–tight algorithms and lower bounds in geometric intersection graphs. *SIAM Journal on Computing*, 49(6):1291–1331, 2020.

10   Erik D Demaine, Fedor V Fomin, Mohammadtaghi Hajiaghayi, and Dimitrios M Thilikos. Subexponential parameterized algorithms on bounded-genus graphs and h-minor-free graphs. *Journal of the ACM (JACM)*, 52(6):866–893, 2005.

11   Hristo N Djidjev. Weighted graph separators and their applications. In *Proceedings of the 5th Annual European Symposium on Algorithms (ESA 1997)*, pages 130–143. Springer, 1997.

12   David P Dobkin, Steven J Friedman, and Kenneth J Supowit. Delaunay graphs are almost as good as complete graphs. *Discrete & Computational Geometry*, 5(4):399–407, 1990.

13   Fedor V. Fomin, Serge Gaspers, and Artem V. Pyatkin. Finding a minimum feedback vertex set in time $O\left(1.7548^n\right)$. In Hans L. Bodlaender and Michael A. Langston, editors, *Proceedings of the 2nd International Workshop on Parameterized and Exact Computation (IWPEC 2006)*, volume 4169, pages 184–191, 2006.

14   Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, Saket Saurabh, and Meirav Zehavi. Finding, hitting and packing cycles in subexponential time on unit disk graphs. In *Proceedings of the 44th International Colloquium on Automata, Languages, and Programming (ICALP 2017)*, volume 80, pages 65:1–65:15, 2017.

15   Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, Saket Saurabh, and Meirav Zehavi. ETH-tight algorithms for long path and cycle on unit disk graphs. In *Proceedings of the 36th International Symposium on Computational Geometry (SoCG 2020)*, volume 164, pages 44:1–44:18, 2020.

16   Fedor V. Fomin, Daniel Lokshtanov, and Saket Saurabh. Bidimensionality and geometric graphs. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2012)*, pages 1563–1575, 2012.

17   Haim Kaplan, Wolfgang Mulzer, Liam Roditty, and Paul Seiferth. Routing in unit disk graphs. *Algorithmica*, 80(3):830–848, 2018.

18   Tomasz Kociumaka and Marcin Pilipczuk. Faster deterministic feedback vertex set. *Information Processing Letters*, 114(10):556–560, 2014.

19   Dániel Marx. Parameterized complexity of independence and domination on geometric graphs. In *Proceedings of the 2nd International Conference on Parameterized and Exact Computation (IWPEC 2006)*, pages 154–165, 2006.

20   Frank van den Eijkhof, Hans L Bodlaender, and MC Arie Koster. Safe reduction rules for weighted treewidth. *Algorithmica*, 47(2):139–158, 2007.