ETH Tight Algorithms for Geometric Intersection Graphs: Now in Polynomial Space

Fedor V. Fomin
University of Bergen, Norway

Petr A. Golovach
University of Bergen, Norway

Tanmay Inamdar
University of Bergen, Norway

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

Abstract

De Berg et al. in [SICOMP 2020] gave an algorithmic framework for subexponential algorithms on geometric graphs with tight (up to ETH) running times. This framework is based on dynamic programming on graphs of weighted treewidth resulting in algorithms that use super-polynomial space. We introduce the notion of weighted treedepth and use it to refine the framework of de Berg et al. for obtaining polynomial space (with tight running times) on geometric graphs. As a result, we prove that for any fixed dimension $d \geq 2$ on intersection graphs of similarly-sized fat objects many well-known graph problems including Independent Set, r-Dominating Set for constant r, Cycle Cover, Hamiltonian Cycle, Hamiltonian Path, Steiner Tree, Connected Vertex Cover, Feedback Vertex Set, and (Connected) Odd Cycle Transversal are solvable in time $2^{\mathcal{O}(n^{1-1/d})}$ and within polynomial space.

2012 ACM Subject Classification Mathematics of computing → Graph algorithms; Theory of computation → Parameterized complexity and exact algorithms

Keywords and phrases Subexponential Algorithms, Geometric Intersection Graphs, Treedepth, Treewidth

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2021.21

Related Version *Full Version*: `https://arxiv.org/abs/2107.06715`

Funding The research leading to these results has received funding from the Research Council of Norway via the project "'MULTIVAL" (grant no. 263317) and the European Research Council (ERC) via grant LOPPRE, reference 819416.

1 Introduction

Most of the fundamental NP-complete problems on graphs like Independent Set, Feedback Vertex Set, or Hamiltonian Cycle do not admit algorithms of running times $2^{o(n)}$ on general graphs unless the Exponential Time Hypothesis (ETH) fails. However, on planar graphs, H-minor-free graphs, and several classes of geometric graphs, such problems admit *subexponential* time algorithms. There are several general frameworks for obtaining subexponential algorithms [4, 6, 8]. The majority of these frameworks utilize dynamic programming algorithms over graphs of bounded treewidth. Consequently, the subexponential algorithms derived within these frameworks use prohibitively large (exponential) space.

We consider another related graph parameter, namely, *treedepth*. Given a graph $G = (V, E)$, a pair (F, φ) is a treedepth decomposition of G, if F is a rooted forest, and $\varphi : V(F) \to V(G)$ is a mapping such that the neighbors in G are mapped to vertices in F that have an ancestor-descendant relationship. Then, the treedepth of G is the minimum height of the forest over all treedepth decompositions. Alternatively, the treedepth of G can be thought of as the elimination distance to the family of edgeless graphs (see the book of Nesetril and de Mendez [17] for more details). Recently, algorithms on graphs of bounded treedepth attracted significant attention [9, 10, 15]. The advantage of these algorithms over dynamic programming used for treewidth is that they use polynomial space. Our work is motivated by the following natural question

Could the treedepth find applications in the design of (polynomial space) subexponential algorithms?

The problem is that the treedepth of a graph could be significantly larger than its treewidth. For example, the treewidth of an n-vertex path is one, while the treedepth is of order $\log n$. It creates problems in using treedepth in frameworks like bidimensionality that strongly exploit the existence of large grid minors in graphs of large treewidth. Despite that, we show the usefulness of treedepth for obtaining polynomial space subexponential algorithms on intersection graphs of some geometrical objects.

In [4], de Berg et al. developed a generic framework facilitating the construction of subexponential algorithms on large classes of geometric graphs. By applying their framework on intersection graphs of similarly-sized fat objects in dimension $d \geq 2$, de Berg et al. obtained algorithms with running time $2^{\mathcal{O}(n^{1-1/d})}$ for many well-known graph problems, including Independent Set, r-Dominating Set for constant r, Hamiltonian Cycle, Hamiltonian Path, Feedback Vertex Set, Connected Dominating Set, and Steiner Tree.

The primary tool introduced by de Berg et al. is the weighted treewidth. They show that solving many optimization problems on intersection graph of n similarly-sized fat objects can be reduced to solving these problems on graphs of weighted treewidth of order $\mathcal{O}(n^{1-1/d})$. Combined with single-exponential algorithms on graphs of bounded weighted treewidth, this yields subexponential algorithms for several problems.

The running times $2^{\mathcal{O}(n^{1-1/d})}$ are tight – de Berg et al. accompanied their algorithmic upper bounds with matching conditional complexity (under ETH) bounds. However, as most of the treewidth-based algorithms, the algorithms of Berg et al. are dynamic programming over tree decompositions. As a result, they require super-polynomial space. Thus a concrete question here is *whether running times $2^{\mathcal{O}(n^{1-1/d})}$ could be achieved using polynomial space.*

We answer this question affirmatively by developing polynomial space algorithms that in time $2^{\mathcal{O}(n^{1-1/d})}$ solve all problems on intersection graphs of similarly-sized fat objects from the paper of de Berg et al. except for Connected Dominating Set. The primary tool in our work is the *weighted treedepth*. To the best of our knowledge, this notion is new.

The Cut&Count technique was introduced by Cygan et al. [3], who gave the first single-exponential (randomized) algorithms parameterized by the treewidth for many problems using this technique. We note that at the heart of these algorithms is a dynamic programming over the tree decomposition, and thus require exponential space. However, unweighted treedepth was recently used by several authors in the design of parameterized algorithms using polynomial space [9, 10, 15]. Some of these works adapt the Cut&Count technique for the treedepth decomposition.

Our main insight is that in the framework of de Berg et al. [4] for most of the problems the weighted treedepth can replace the weighted treewidth. Pipelined with branching algorithms over graphs of small weighted treedepth, this new insight brings us to many tight (up to ETH) polynomial space algorithms on geometric graphs.

Our results. To explain our strategy of "replacing" the weighted treewidth with the weighted treedepth, we need to provide an overview of the framework of de Berg et al. [4]. It has two main ingredients. First, for an intersection graph of n similarly-sized fat objects (we postpone technical definitions to the next section), we construct an auxiliary weighted graph $G_{\mathcal{P}}$. (Roughly speaking, to create $G_{\mathcal{P}}$, we contract some cliques of G and assign weights to the new vertices.) Then the combinatorial theorem of de Berg et al. states that the weighted treewidth of $G_{\mathcal{P}}$ is $\mathcal{O}(n^{1-1/d})$. Second, to solve problems on G in time $2^{\mathcal{O}(n^{1-1/d})}$, one uses a tree decomposition of $G_{\mathcal{P}}$. This part is problem-dependent and, for some problems, could be pretty non-trivial.

To plug in the treedepth into this framework, we first prove that the weighted treedepth of $G_{\mathcal{P}}$ is $\mathcal{O}(n^{1-1/d})$. Moreover, we give an algorithm computing a treedepth decomposition in time $2^{\mathcal{O}(n^{1-1/d})}$ and polynomial space. For Independent Set, a simple branching algorithm over the treedepth decomposition can solve the problem in time $2^{\mathcal{O}(n^{1-1/d})}$ and polynomial space. We also get a similar time and space bounds for Dominating Set, and more generally, r-Dominating Set for constant r; however, we need to use a slightly different kind of recursive algorithm.

Next, we consider connectivity problems like Steiner Tree, Connected Vertex Cover, Feedback Vertex Set, and (Connected) Odd Cycle Transversal. For these problems, we are able to adapt the single exponential FPT algorithms parameterized by (unweighted) treedepth given by Hegerfeld and Kratsch [10], into the framework of weighted treedepth decomposition. Thus, we get $2^{\mathcal{O}(n^{1-1/d})}$ time, polynomial space algorithms for these problems.

Finally, we consider Cycle Cover, which is a generalization of Hamiltonian Cycle. Here, we are able to "compress" the given graph into a new graph, such that the (unweighted) treedepth of the new graph is $\mathcal{O}(n^{1-1/d})$. We can also compute the corresponding treedepth decomposition in $2^{\mathcal{O}(n^{1-1/d})}$ time, and polynomial space. Then, we can use a result by Nederlof et al. [15] as a black box, which is a Cut&Count based a single exponential FPT algorithms parameterized by treedepth, that uses polynomial space. Thus, we get $2^{\mathcal{O}(n^{1-1/d})}$ time, polynomial space algorithms for Cycle Cover, Hamiltonian Cycle, and also for Hamiltonian Path.

We note that the results in the previous two paragraphs are based on the Cut&Count technique, and are randomized. We also note that all of our algorithms, except for Cycle Cover and related problems can work even without the geometric representation of the similarly-sized fat objects. For Cycle Cover and related problems, however, we require the geometric representation. This is in line with similar requirements for these problems from [4].

Organization. In Section 2, we define some of the basic concepts including the weighted treedepth, and then prove our main result about the same. At the end of the section, we give a warm-up example of an algorithm for Independent Set using this framework. Then, in Section 3 we describe the Cut&Count algorithm, and its application for Steiner Tree using the notion of weighted treedepth. Finally, we give a rough sketch of our approach for Cycle Cover in Section 4. The algorithms for the remaining problems are omitted from the main version due to page limit, but they can be found in the full version of the paper.

2 Geometric Graphs and Weighted Treedepth

In this section we define the weighted treedepth, prove a combinatorial bound on the treedepth of certain geometric graphs and provide a generic algorithm and provide an abstract theorem modeling at a high level our subexponential time and polynomial space algorithms. But first, we need some definitions.

Graphs. We consider only undirected simple graphs and use the standard graph theoretic terminology; we refer to the book of Diestel [7] for basic notions. We write $|G|$ to denote $|V(G)|$, and throughout the paper we use n for the number of vertices if it does not create confusion. For a set of vertices $S \subseteq V(G)$, we denote by $G[S]$ the subgraph of G induced by the vertices from S and write $G - S$ to denote the graph obtained by deleting the vertices of S. For a vertex v, $N_G(v)$ denotes the *open neighborhood* of v, that is, the set of vertices adjacent to v, and $N_G[v] = \{v\} \cup N_G(v)$ is the *closed neighborhood.* For a vertex v, $d_G(v) = |N_G(v)|$ denotes the *degree* of v. We may omit subscripts if it does not create confusion. For two distinct vertices u and v of a graph G, a set $S \subseteq V(G)$ is a *(u, v)-separator* if $G - S$ has no (u, v)-path and S is a *separator* if S is a (u, v)-separator for some vertices u and v. A pair of vertex subsets (A, B) is called a separation if $A \cup B = V(G)$, and there are no edges between $A \setminus B$ and $B \setminus A$, that is, $S = A \cap B$ is a (u, v)-separator for $u \in A \setminus B$ and $v \in B \setminus A$. We say that a subset $S \subseteq V(G)$ is an α-balanced separator for a constant $\alpha \in (0, 1)$ if there exists a separation (A, B) such that $A \cap B = S$, and $\max\{|A|, |B|\} \leq \alpha n$.

κ-partition Let $\mathcal{P} = \{V_1, V_2, \ldots, V_t\}$ be a partition of $V(G)$ for some $t \geq 1$, such that any $V_i \in \mathcal{P}$ satisfies the following properties: (1) $G[V_i]$ is connected, and (2) V_i is a union of at most κ cliques in G (not necessarily disjoint). Then, we say that $\mathcal{P}$ is a κ-partition of G. Furthermore, given a κ-partition $\mathcal{P} = \{V_1, V_2, \ldots, V_t\}$ of G, we define the graph $G_{\mathcal{P}}$, the graph induced by $\mathcal{P}$, as the undirected graph obtained by contracting each V_i to a vertex, and removing self-loops and multiple edges.

Treedepth and Weighted Treedepth. We introduce *weighted treedepth* of a graph as a generalization of the well-known notion of treedepth (see e.g. the book of Nesetril and de Mendez [17]). There are different ways to define treedepth but it is convenient for us to deal with the definition via *treedepth decompositions* or *elimination forests.* We say that a forest F supplied with one selected node (it is convenient for us to use the term "node" instead of "vertex" in such a forest) in each connected component, called a *root*, a *rooted forest.* The choice of roots defines the natural parent–child relation on the nodes of a rooted forest. Let G be a graph and let $\omega \colon V(G) \to \mathbb{R}$ be a weight function. A *treedepth decomposition* of G is a pair (F, φ), where F is a rooted forest and $\varphi \colon V(F) \to V(G)$ is a bijective mapping such that for every edge $uv \in E(G)$, either $\varphi^{-1}(u)$ is an ancestor of $\varphi^{-1}(v)$ in F or $\varphi^{-1}(v)$ is an ancestor of $\varphi^{-1}(u)$. Then the *depth* of the decomposition is the depth of F, that is, the maximum number of nodes in a path from a root to a leaf. The *treedepth* of G, denoted $\mathtt{td}(G)$, is the minimum depth of a treedepth decomposition of G. We define the *weighted depth* of a treedepth decomposition as the maximum $\sum_{v \in V(P)} \omega(\varphi(v))$ taken over all paths P between roots and leaves. Respectively, the *weighted treedepth* $\mathtt{wtd}(G)$ is the minimum weighted depth of a treedepth decomposition. For our applications, we assume without loss of generality that G is connected, which implies that the forest F in a (weighted) treedepth decomposition is actually a tree.

Weighted Treewidth. We assume basic familiarity with the notion of treewidth and tree decomposition of a graph – see a textbook such as [2], for example. Similar to the previous paragraph, de Berg et al. [4] define the *weighted treewidth* of a graph. Given an undirected graph $G = (V, E)$ with weights $\omega \colon V(G) \to \mathbb{R}$, the weighted width of a tree decomposition (T, β), is defined to be the maximum over bags, the sum of the weights of vertices in the bag. The weighted treewidth of a graph is the minimum weighted width over all tree decompositions of the graph.

It is useful to observe that we consider treedepth and tree decompositions of the graphs $G_{\mathcal{P}}$ constructed for graphs G with given κ-partition $\mathcal{P} = \{V_1, V_2, \dots, V_t\}$. Then the treedeph decomposition of $G_{\mathcal{P}}$ can be seen as a pair (F, φ), where F is a rooted forest and φ is a bijective mapping of $V(F)$ to $\mathcal{P}$. Similarly, in a tree decomposition (T, β) of $G_{\mathcal{P}}$, corresponding to every node $t \in V(T)$, the bag $\beta(t)$ is a subset of $\mathcal{P}$. Finally, we observe that the results of [4] regarding weighted treewidth – thus our results for weighted treedepth – hold for any weight functions $\omega : \mathcal{P} \to \mathbb{R}^+$, provided that $\omega(\ell) = \mathcal{O}(\ell^{1-1/d-\epsilon})$, for any $\epsilon > 0$. However, as in [4], we will fix the weight function to be $\omega(\ell) \coloneqq \log(1 + \ell)$ throughout the rest of the paper. For the simplicity of notation, we use the shorthand $\omega(u_i) \coloneqq \omega(|V_i|)$, where $\varphi(u_i) = V_i$, and for any $u_i \in V(F)$, and $\omega(S) \coloneqq \sum_{u_i \in S} \omega(u_i)$ for any subset $S \subseteq V(F)$.

Geometric Definitions. Given a set F of objects in $\mathbb{R}^d$, we define the corresponding intersection graph $G[F] = (V, E)$, where there is a bijection between an object in F and $V(G)$, and $uv \in E(G)$ iff the corresponding objects in F have a non-empty intersection. It is sometimes convenient to erase the distinction between F with $V(G)$, and to say that each vertex is a geometric object from F.

We consider the geometric intersection graphs of *fat objects*. A geometric object $g \subset \mathbb{R}^d$ is said to be α-fat for some $\alpha \geq 1$, if there exist balls $B_{\text{in}}, B_{\text{out}}$ such that $B_{\text{in}} \subseteq g \subseteq B_{\text{out}}$, such that the ratio of the radius of B_{out} to that of B_{in} is at most α. We say that a set F of objects is *fat* if there exists a constant $\alpha \geq 1$ such that every geometric object in F is α-fat. Furthermore, we say that F is a set of *similarly-sized* fat objects, if the ratio of the largest diameter of an object in F, to the smallest diameter of an object in F is at most a fixed constant. Finally, observe that if F is a set of *similarly-sized fat objects*, then the ratio of the largest out-radius to the smallest in-radius of an object is also upper bounded by a constant. de Berg et al. [4] prove the following two results regarding the intersection graphs of similarly sized fat objects.

▶ **Lemma 1** ([4]). *Fix dimension $d \geq 2$. There exist constants κ and Δ, such that for any intersection graph $G = (V, E)$ of an (unknown) set of n similarly-sized fat objects in $\mathbb{R}^d$, a κ-partition $\mathcal{P}$ for which $G_{\mathcal{P}}$ has maximum degree Δ can be computed in time polynomial in n.*

In the following, we will use the tuple $(G, d, \mathcal{P}, G_{\mathcal{P}})$ to indicate that $G = (V, E)$ is the intersection graph of n similarly-sized fat objects in $\mathbb{R}^d$, $\mathcal{P}$ is a κ-partition of G such that $G_{\mathcal{P}}$ has maximum degree Δ, where κ, Δ are constants, as guaranteed by Lemma 1.

▶ **Lemma 2** ([4]). *For any $(G, d, \mathcal{P}, G_{\mathcal{P}})$, the weighted treewidth of $G_{\mathcal{P}}$ is $\mathcal{O}(n^{1-1/d})$.*

Now we are ready to prove the following result about the intersection graphs of similarly sized fat objects. This result is at the heart of the subexponential algorithms designed in the following sections.

▶ **Theorem 3.** *There is a polynomial space algorithm that for a given $(G, d, \mathcal{P}, G_{\mathcal{P}})$, computes in time $2^{\mathcal{O}(n^{1-1/d})}$ a weighted treedepth decomposition (F, φ) of $G_{\mathcal{P}}$ of weighted treedepth $\mathcal{O}(n^{1-1/d})$.*

Proof. We use the approximation algorithm from [18] to compute a weighted tree decomposition (T, β) of $G_{\mathcal{P}}$ (see the later part of the proof for a detailed explanation). Using the standard properties of the tree decomposition (e.g., see [2]), there exists a node $t \in V(T)$, such that $V_B \coloneqq \bigcup_{V_i \in \beta(t)} V_i$ is an α-balanced separator for G, for some $\alpha \leq 2/3$. Let $B \coloneqq \beta(t)$. Note that $B \subseteq \mathcal{P}$.

Now we construct a part of the forest F, and the associated bijection φ in the weighted treedepth decomposition (F, φ) of $G_{\mathcal{P}}$. We create a path $\pi = (u_1, u_2, \dots, u_{|B|})$, and arbitrarily assign $\varphi(u_i)$ to some $V_i \in B$ such that it is a bijection. We set u_1, the first vertex on π, to be the root of a tree in F. We also set the weight $\omega(u_i) = \log(1 + |V_i|)$, where $\varphi(u_i) = V_i$. Note that the $\omega(\pi) = \omega(B) = \mathcal{O}(n^{1-1/d})$.

Let (Y_1, Y_2) be the separation of G, corresponding to the separator $\bigcup_{V_i \in \beta(t)} V_i$. Analogously, let $(\mathcal{P}'_1, \mathcal{P}'_2)$ denote the separation of $G_{\mathcal{P}}$, corresponding to the separator B. Furthermore, let $X_i \coloneqq Y_i \setminus V_B$, and $\mathcal{P}_i \coloneqq \mathcal{P}'_i \setminus B$ for $i = 1, 2$. Note that $X_1 \setminus V_B, X_2 \subseteq V(G)$ are disjoint, $\max\{|X_1|, |X_2|\} \leq \alpha n$, and there is no edge from a vertex in X_1, to a vertex in X_2. Furthermore, $\mathcal{P}_1$ is a κ-partition of $G[X_1]$, and $\mathcal{P}_2$ is a κ-partition of $G[X_2]$.

Now, we recursively construct weighted treedepth decomposition (F_1, φ_1) of $G_{\mathcal{P}}[\mathcal{P}_1]$. Note that φ_1 is a bijection between $V(F_1)$ and $\mathcal{P}_1 \subseteq \mathcal{P}$. Let R_1 denote the set of roots of the trees in forest F_1. We add an edge from the last vertex $u_{|B|}$ on the path π, to each root in R_1. In other words, we attach every tree in F_1 as a subtree below $u_{|B|}$. The bijection φ is extended to $\mathcal{P}_1$ using φ_1. Now we consider a weighted treedepth decomposition (F_2, φ_2) of $G_{\mathcal{P}}[\mathcal{P}_2]$, and use it to extend (F, φ) in a similar manner. This completes the construction of (F, φ).

Let us first analyze the weighted treedepth of (F, φ). Let us use $q \coloneqq 1 - 1/d$ for simplicity. For a path π in F, let $\omega(\pi)$ denote the sum of weights of vertices along the path π. Recall that the weight of any root-leaf path π in F is at most $\mathcal{O}(n^q)$. More generally, let $c' \geq 0$ be a universal constant (independent of the path π, or its level in F) such that the weight of a path corresponding to a separator computed at level j, is at most $c' \cdot (\alpha^{j-1} n)^q$. Since $\max\{|X_1|, |X_2|\} \leq \alpha n$, we inductively assume that the weighted treedepth of (F_1, φ_1), and that of (F_2, φ_2) is at most $\mathcal{O}(\alpha^q \cdot n^q)$. More specifically, we assume that there exists a universal constant $c \geq c'$, such that the sum of the weights along any root-leaf path in F_1 is upper bounded by $c \cdot \frac{(\alpha n)^q}{1-\alpha^q}$. The same inductive assumption holds for any root-leaf path in F_2. Therefore, the weight of any root-leaf path in F is upper bounded by

$$\omega(\pi) + c \cdot \frac{\alpha^q n^q}{1 - \alpha^q} \leq c n^q \left(1 + \frac{\alpha^q}{1 - \alpha^q}\right) = \frac{c n^q}{1 - \alpha^q}.$$

Therefore, we have the desired bound on the weighted treedepth by induction.

Now we look the treewidth construction part of the algorithm in order to sketch the claims about bounds on time and space. Given the graph $G_{\mathcal{P}}$, we construct a graph H by replacing every vertex V_i with a (new) clique C_i of size $\log(1+|V_i|)$. If $V_i V_j \in E(G_{\mathcal{P}})$, we also add edges from every vertex in C_i to every vertex in C_j. As shown in [4], the weighted width of $G_{\mathcal{P}}$ is equal to the treewidth of H, plus 1. Note that $|V(H)| = \sum_{V_i \in \mathcal{P}} \log(1 + |V_i|) \leq n$, since $\mathcal{P}$ is a partition of $V(G)$.

The algorithm from [18] (see also Section 7.6.2 in the Parameterized Algorithms book [2]) for approximating treewidth of a graph H works as follows. Suppose the treewidth of a graph is k, which is known. At the heart of this algorithm is a procedure $\texttt{decompose}(W, S)$, where $S \subsetneq W \subseteq V(H)$, and $|S| \leq 3k + 4$. This procedure tries to decompose the subgraph $H[W]$ in such a way that S is completely contained in one bag of the tree decomposition. The first step is to compute a partition (S_A, S_B) of S, such that the size of the separator separating S_A and S_B in $H[W]$ is at most $k+1$. This is done by exhaustively guessing all partitions, which takes $2^{\mathcal{O}(k)}$ time. For each such guess of (S_A, S_B), we run a polynomial time algorithm to check whether the bound on the separator size holds. Once such a partition is found, a set $\hat{S} \supsetneq S$ is found by augmenting S in a particular way. Finally, we recursively run the procedure $\texttt{decompose}(N_H[D], N_H(D))$, for each connected component D in $H[W \setminus \hat{S}]$. Finally, the tree decomposition of $H[W]$ is computed by augmenting the tree decompositions computed by the recursive procedure for its children, with the root bag containing $\hat{S}$. It is shown that this algorithm computes a tree decomposition of width $\mathcal{O}(\texttt{tw})$ in time $2^{\mathcal{O}(\texttt{tw})} \cdot n^{\mathcal{O}(1)}$. Furthermore, it can also be observed that it only uses polynomial space.

Therefore, computing a tree decomposition of $G_{\mathcal{P}}$ of weighted treewidth $\mathcal{O}(n^{1-1/d})$ takes $2^{\mathcal{O}(n^{1-1/d})}$ time and polynomial space, corresponding to the original graph G with n vertices. The treewidth computation algorithm is called at most n times, and there is additional polynomial processing at every step. This implies the time and space bounds as claimed. ◀

We note that de Berg et al. [4] show the existence of a balanced separator of weight $\mathcal{O}(n^{1-1/d})$, which is then used to show the same bound on weighted treewidth (Theorem 2). This separator can be computed in $\mathcal{O}(n^{d+2})$ time if we are also given the geometric representation of the underlying objects in $\mathbb{R}^d$. However, without geometric representation it is not clear whether this separator can be directly computed. Therefore, we first compute an approximate weighted treewidth decomposition, and then retrieve the separator bag in the proof of Theorem 3. We state the following abstract theorem that models at a high level our subexponential algorithms that use polynomial space. The proof of this theorem follows from Theorem 3, and can be found the full version.

▶ **Theorem 4.** *Let $\mathcal{A}$ be an algorithm for solving a problem on graph G, that takes input $(G, d, \mathcal{P}, G_{\mathcal{P}})$, and a weighted treedepth decomposition (F, φ) of $G_{\mathcal{P}}$ of weighted depth $\mathcal{O}(n^{1-1/d})$ (and optionally additional inputs of polynomial size). Suppose $\mathcal{A}$ is a recursive algorithm, that at every node $u \in V(F)$, spends time proportional to $2^{\mathcal{O}(\omega(u))} \cdot n^{\mathcal{O}(1)}$, uses polynomial space, and makes at most $2^{\mathcal{O}(\omega(u))}$ recursive calls on the children of u. Then, the algorithm $\mathcal{A}$ runs in time $2^{\mathcal{O}(n^{1-1/d})}$, and uses polynomial space.*

Independent Set. As a warm-up example for using the weighted treedepth decomposition, we describe an application for Independent Set. Given $(G, d, \mathcal{P}, G_{\mathcal{P}})$, we first observe that every $V_i \in \mathcal{P}$ is a union of at most κ cliques, which implies that the intersection of an independent set with any V_i is bounded by κ. A recursive algorithm for Independent Set works with the weighted treedepth decomposition (F, φ) computed via Theorem 3. When the algorithm is at a node $u_i \in V(F)$, we make a recursive call to the children of u_i, corresponding to each independent subset $U_i \subseteq \varphi(u_i) = V_i$ of size at most κ, that is independent. We recursively compute a Maximum Independent Set in the subgraph of G, corresponding to the subtree rooted at each children of u_i, with the vertices in $N(U_i)$ removed. We return the maximum independent set found over all choices of the subset U_i. Finally, we observe that the number of subsets of V_i of size at most κ is at most $(1 + |V_i|)^\kappa = 2^{\mathcal{O}(\log(1+|V_i|))} = 2^{\mathcal{O}(\omega(u_i))}$, where we use the fact that $\kappa = \mathcal{O}(1)$. A more formal description of the algorithm can be found in the full version.

▶ **Theorem 5.** *There exists a $2^{\mathcal{O}(n^{1-1/d})}$ time, polynomial space algorithm to compute a maximum (weight) independent set in the intersection graphs of similarly sized fat objects in $\mathbb{R}^d$.*

***r*-Dominating Set.** For a fixed $r \geq 1$, r-Dominating Set asks for a minimum-size vertex subset $D \subseteq V(G)$, such that for every $v \in V(G)$, there exists some $u \in D$ such that $dist_G(u, v) \leq r$, where $dist_G(u, v)$ is the number of edges on the shortest path in G between

Algorithm	Space	Similarly-sized	Convex	Robust
Corollary 2.4 in [4]	Poly	Yes	Yes	No
Theorem 2.13 in [4]	$2^{\mathcal{O}(n^{1-1/d})}$	Yes	No	Yes
Theorem 5 (this paper)	Poly	Yes	No	Yes

Figure 1 Comparison of three $2^{\mathcal{O}(n^{1-1/d})}$-time algorithms for Independent Set on the intersection graphs of fat objects. "Robust" in the last column means that the algorithm does not require the geometric representation of the objects. In terms of technique, our algorithm (third row) is closely related to the one in the first row, albeit we use the framework of weighted treedepth.

u and v. Following [4], there are the two important ingredients in our algorithm for r-Dominating Set. First, de Berg et al. [4] show that it can be assumed that $|V_i \cap D| \leq \kappa^2(1+\Delta)$ for any $V_i \in \mathcal{P}$. However, this property alone is not sufficient to obtain a recursive algorithm that runs in subexponential time and polynomial space.

Consider a similar recursive algorithm that is processing a vertex $V_i \in \mathcal{P}$ using a weighted treedepth decomposition (F, φ) of $G_{\mathcal{P}}$. There are three possibilities for a vertex $u \in V_i$ – (i) it is in the dominating set, (ii) it is already being dominated by a vertex that was added to the dominating set at an earlier stage of recursion, or (iii) it will be dominated by a vertex $v \in V_j$ with $dist_G(u,v) \leq r$, where V_j belongs to the subtree of F rooted at V_i. To handle case (iii), we need to enumerate partial solutions from *all* V_j's such that $dist_{G_{\mathcal{P}}}(V_i, V_j) \leq r$. However, the weighted treedepth bound given in Theorem 3 is not sufficient, and we need a strengthened version of the theorem. Such a result appears in [4], and we reprove it in the full version for completeness. Loosely speaking, this result bounds the total weight of *all* the bags that appear within the r-neighborhood of the α-balanced separator obtained via the weighted treewidth decomposition. Armed with this result, the recursive algorithm "guesses" the set of vertices from the balanced separator bag, and for each vertex u type (iii), it also guesses a vertex v from the subtree that dominates u. The stronger theorem implies that the number of recursive calls made from the i-th level of recursion can still be bounded by $2^{\mathcal{O}((\alpha^{i-1}n)^{1-1/d})}$. A formal description and analysis of this algorithm can be found in the full version of the paper. We summarize our result in the following theorem.

▶ **Theorem 6.** *For any fixed $r \geq 1$, there exists a $2^{\mathcal{O}(n^{1-1/d})}$ time, polynomial space algorithm to compute a minimum r-dominating set in the intersection graphs of similarly sized fat objects in $\mathbb{R}^d$.*

3 Cut&Count Algorithms

Hegerfeld and Kratsch [10] adapt the Cut&Count technique to give FPT algorithms for various connectivity based subset problems, parameterized by (unweighted) treedepth. In particular, these algorithms are randomized, have running times of the form $2^{\mathcal{O}(\mathtt{td})} \cdot n^{\mathcal{O}(1)}$, and use polynomial space. In their work, they consider Connected Vertex Cover, Feedback Vertex Set, Connected Dominating Set, Steiner Tree, and Connected Odd Cycle Transversal problems. We are able to adapt their technique for all of these problems, except for Connected Dominating Set. For the rest of the problems, we will extend their ideas to the more general case of *weighted treedepth*, and use it to give $2^{\mathcal{O}(n^{1-1/d})}$ time, polynomial space, randomized algorithms. In the following, we select Steiner Tree as a representative problem, which is explained in detail. For the remaining problems, we give only a brief sketch highlighting the differences from the Steiner Tree algorithm, and defer the formal details to the full version.

3.1 Setup

We adopt the following notation from Hegerfeld and Kratsch [10]. Let $\mathtt{cc}(G)$ denote the number of connected components in G. A cut of $X \subseteq V(G)$ is a pair (X_L, X_R), where $X_L \cap X_R = \emptyset$, $X_L \cup X_R = X$. We refer to X_L, X_R as the left and the right side of the cut (X_L, X_R) respectively. A cut (X_L, X_R) of $G[X]$ is consistent, if for any $u \in X_L$ and $v \in X_R$, $uv \notin E(G[X])$. A *consistently cut subgraph* of G is a pair $(X, (X_L, X_R))$, such that $X \subseteq V(G)$, and (X_L, X_R) is a consistent cut of $G[X]$. Finally, for $X \subseteq V(G)$, we denote the set of consistently cut subgraphs of $G[X]$ by $\mathcal{C}(X)$.

For $n \in \mathbb{N}$, let $[n]$ denote the set of integers from 1 to n. For integers a, b, we write $a \equiv b$ to indicate equality modulo 2. We use Iverson's bracket notation: for a boolean predicate p, $[p]$ is equal to 1 if p is true, otherwise $[p]$ is equal to 0.

Consider a function $f : A \to S$. For every $s \in S$ and a set X, we define the set $X(f, s) := X \cap f^{-1}(s)$ – note that $X(f, s)$ may be empty for some or all $s \in S$. Furthermore, observe that the sets $\{A(f, s)\}_{s \in S}$ define a partition of A. For two functions $g : A \to S$, $f : B \to S$, we define the new function $g \oplus f : (A \cup B) \to S$ as follows. $(g \oplus f)(e) = f(e)$ for $e \in B$, and $(g \oplus f)(e) = g(e)$ for $e \in (A \setminus B)$. That is, $(g \oplus f)$ behaves like g and f on the exclusive domains, but in case of a conflict, the function f takes the priority.

Recall that we work with $(G, d, \mathcal{P}, G_{\mathcal{P}})$, and the corresponding weighted treedepth decomposition (F, φ) of G. Here, φ is a bijection between $V(F)$ and $\mathcal{P}$. For a node u_i, we will use $V_i := \varphi(u_i)$, i.e., we use the same indices in the subscript to identify a node of F and the corresponding part in $\mathcal{P}$. We denote the set of children of u_i by $\mathtt{child}(u_i)$. Additionally,

$$\mathtt{tail}[u_i] = \bigcup_{u_j \text{ is an ancestsor of } u_i} V_j \;; \qquad \mathtt{tail}(u_i) = \mathtt{tail}[u_i] \setminus V_i$$

$$\mathtt{tree}[u_i] = \bigcup_{u_j \text{ is a descendant of } u_i} V_j \;; \qquad \mathtt{tree}(u_i) = \mathtt{tree}[u_i] \setminus V_i$$

$$\mathtt{broom}[u_i] = \mathtt{tail}[u_i] \cup \mathtt{tree}(u_i)$$

Isolation Lemma

▶ **Definition 7.** *Let U be a finite set, and $\mathcal{F} \subseteq 2^U$ be a family of subsets of U. We say that a weight function $\mathbf{w} : U \to \mathbb{Z}$* isolates *the family $\mathcal{F}$ if there exists a unique set $S' \in \mathcal{F}$ such that $\mathbf{w}(S') = \min_{S \in \mathcal{F}} \mathbf{w}(S)$, where $\mathbf{w}(X) := \sum_{x \in X} \mathbf{w}(x)$ for any subset $X \subseteq U$.*

The following isolation lemma due to Mulmuley et al. [14] is at the heart of all Cut&Count algorithms.

▶ **Lemma 8** ([14]). *Let $\mathcal{F} \subseteq 2^U$ be a non-empty family of subsets of a finite ground set U. Let $N \in \mathbb{N}$, and suppose $\mathbf{w}(u)$ is chosen uniformly and independently at random from $[N]$ for every $u \in U$. Then, $\Pr(\mathbf{w}$ isolates $\mathcal{F}) \geq 1 - |U|/N$.*

General Idea

Fix a problem involving connectivity constraints. Let U be the ground set that is related to the graph G, such that $\mathcal{S} \subseteq 2^U$, where $\mathcal{S}$ denotes the set of solutions to the problem. At a high level, a Cut&Count based algorithm contains the following two parts.

- **The Cut part:** We obtain a set $\mathcal{R}$ by relaxing the connectivity requirements on the solutions, such that $\mathcal{S} \subseteq \mathcal{R} \subseteq 2^U$. The set $\mathcal{Q}$ will contain pairs (X, C), where $X \in \mathcal{R}$ is a candidate solution, and C is a consistent cut of X. Note that since $X \in \mathcal{R}$, X may be possibly disconnected.

- **The Count part:** We compute $|\mathcal{Q}| \mod 2$ using an algorithm. The consistent cuts are defined carefully, in order that the non-connected solutions from $\mathcal{R} \setminus \mathcal{S}$ cancel while counting modulo 2, since they are consistent with an even number of cuts.

Note that if $|\mathcal{S}|$ is even, then the procedure counting $|\mathcal{Q}| \mod 2$ will return 0, which will be inconclusive. Therefore, we initially sample a random weight function $\mathbf{w} : U \to [N]$ for some large integer $N \geq 2|U|$, and count $|\mathcal{Q}_w| \mod 2$ (where $\mathcal{Q}_w$ is the subset of $\mathcal{Q}$ such that the corresponding X has weight *exactly* w), for all values of $w \in [2|U|^2]$. Using Lemma 8, it can be argued that with at least probability 1/2, if $\mathcal{S} \neq \emptyset$, then for some weight $w \in [2|U|^2]$, the procedure counting $|\mathcal{Q}_w| \mod 2$ outputs 1. Finally, we guess an arbitrary vertex $v_1 \in V(G)$ in the solution, and force it to be on the left side of the consistent cuts. That is, we count the number of consistent cuts in which v_1 is forced to belong to the left side. This breaks the left-right symmetry. We first have the following two results from [10, 3].

▶ **Lemma 9** ([10, 3]). *Let $X \subseteq V(G)$ such that $v_1 \in X$. The number of consistently cut subgraphs $(X, (X_L, X_R))$ such that $v_1 \in X_L$ is equal to $2^{\mathrm{cc}(G[X])-1}$.*

▶ **Corollary 10** ([10, 3]). *Let $\mathcal{S} \subseteq 2^U$, and $\mathcal{Q} \subseteq 2^{U\times(V\times V)}$, such that for every $\mathbf{w} : U \to [2|U|]$, and a target weight $w \in [2|U|^2]$, the following two properties hold.*

1. *$|\{(X, C) \in \mathcal{Q} : \mathbf{w}(X) = w\}| = |\{X \in \mathcal{S} : \mathbf{w}(X) = w\}|$, and*
2. *There is an algorithm $\mathtt{CountC}(\mathbf{w}, w, (G, d, \mathcal{P}, G_\mathcal{P}), (F, \varphi))$, where (F, φ) is a weighted treedepth decomposition of $(G, d, \mathcal{P}, G_\mathcal{P})$, such that: $\mathtt{CountC}(\mathbf{w}, w, (G, d, \mathcal{P}, G_\mathcal{P}), (F, \varphi)) \equiv |\{(X, C \in \mathcal{Q} : \mathbf{w}(X) = w)\}|$.*

*Then, Algorithm 1 returns **false** if $\mathcal{S} = \emptyset$, and returns **true** with probability at least $\frac{1}{2}$ otherwise.*

Proof. Plugging in $\mathcal{F} = \mathcal{S}$ and $N = 2|U|$ in Lemma 8, we know that if $\mathcal{S} \neq \emptyset$, then with probability at least 1/2, there exists a weight $w \in [2|U|^2]$ such that $|\{X \in \mathcal{S} : \mathbf{w}(X) = w\}| = 1$. Then, Algorithm 1 returns **true** with probability at least 1/2.

On the other hand, if $\mathcal{S} = \emptyset$, then by the first property, and the definition of `CountC`, for any choice of $\mathbf{w}$ and w, the procedure `CountC` returns **false**. Therefore, Algorithm 1 returns **false**. ◀

Algorithm 1 Cut&Count$(U, (G, d, \mathcal{P}, G_\mathcal{P}), (F, \varphi), \mathtt{CountC})$.

```
Input: A set $U$, $(G, d, \mathcal{P}, G_\mathcal{P})$, associated weighted treedepth decomposition $(F, \varphi)$, a
procedure CountC that takes $\mathbf{w} : U \to [N], w \in \mathbb{N}$
1: Choose $\mathbf{w}(u)$ independently and uniformly at random from $[2|U|]$ for each $u \in U$
2: for $w = 1, 2, \ldots, 2|U|^2$ do
3:     if CountC$((G, d, \mathcal{P}, G_\mathcal{P}), (F, \varphi), \mathbf{w}, w) \equiv 1$ return true
4: end for
5: return false
```

3.2 Steiner Tree

▶ **Definition 11** (Steiner Tree).
Input: An undirected graph $G = (V, E)$, a set of terminals $K \subseteq V(G)$, and an integer k.
Question: Is there a subset $X \subseteq V(G)$, with $|X| \leq k$, such that $G[X]$ is connected, and $K \subseteq X$?

Fix $(G, d, \mathcal{P}, G_{\mathcal{P}})$ via Lemma 1. Recall that $\mathcal{P}$ is a κ-partition of G, such that the corresponding graph $G_{\mathcal{P}}$ has maximum degree $\Delta = \mathcal{O}(1)$. We first have the following lemma.

▶ **Lemma 12** ([4]). *Suppose X is a minimal solution for* Steiner Tree *(i.e., no proper subset of X is also a solution) for a given $(G, d, \mathcal{P}, G_{\mathcal{P}})$, and a set of terminals K. Then $|X \cap (V_i \setminus K)| \leq \kappa^2(\Delta + 1)$ for any $V_i \in \mathcal{P}$.*

Let $k' = |K| + \kappa^2(\Delta + 1) \cdot |\mathcal{P}|$. Note that using Lemma 12, we may assume that $k \leq k'$ – if $k \geq k'$, then (G, k) is a "yes-instance" iff (G, k') is a "yes-instance". For any $X \subseteq V(G)$, we say that X is $\mathcal{P}$-restricted if for any $V_i \in \mathcal{P}$, $|X \cap (V_i \setminus K)| \leq \kappa^2(\Delta + 1)$. Note that this definition of a $\mathcal{P}$-restricted set (and later, that of a $\mathcal{P}$-restricted function) is specific to the Steiner Tree problem. For different problems, we need to define this notion differently, albeit the main idea is to use a problem-specific version of Lemma 12.

We will run the following algorithm for all values of $k \leq k'$. Let $t_1 \in K$ be an arbitrary terminal that we will fix to be on the left side of consistent cuts, as discussed previously. Now we give the formal definitions of the sets $\mathcal{R}, \mathcal{S}, \mathcal{Q}$ that were abstractly defined in the setup. We also define weight-restricted versions $\mathcal{R}_w, \mathcal{S}_w, \mathcal{Q}_w$ of these sets, where $w \in \mathbb{N}$.

$$\begin{aligned} \mathcal{R} &= \{X \subseteq V(G) : X \text{ is } \mathcal{P}\text{-restricted}, K \subseteq X, |X| = k\}; & \mathcal{R}_w &= \{X \in \mathcal{R} : \mathbf{w}(X) = w\} \\ \mathcal{S} &= \{X \in \mathcal{R} : G[X] \text{ is connected}\}; & \mathcal{S}_w &= \{X \in \mathcal{S} : \mathbf{w}(X) = w\} \\ \mathcal{Q} &= \{(X, (X_L, X_R)) \in \mathcal{C}(V) : X \in \mathcal{R} \text{ and } t_1 \in X_L\}; & \mathcal{Q}_w &= \{(X, (X_L, X_R)) \in \mathcal{Q} : \mathbf{w}(X) = w\} \end{aligned}$$

▶ **Lemma 13.** *Let $\mathbf{w} : V(G) \to [N]$ be a weight function. Then, for every $w \in \mathbb{N}$, $|\mathcal{S}_w| \equiv |\mathcal{Q}_w|$.*

Proof. From Lemma 9, $|\mathcal{Q}_w| = \sum_{X \in \mathcal{R}_w} 2^{\mathtt{cc}(G[X])-1}$.
Thus, $|\mathcal{Q}_w| \equiv |\{X \in \mathcal{R}_w : \mathtt{cc}(G[X]) = 1\}| = |\mathcal{S}_w|$. Recall that $\equiv$ is equality modulo 2. ◀

The goal of the rest of this subsection is to explain how the procedure `CountC` works.

First, we drop the cardinality constraints and define the following candidates and candidate cut-pairs for induced subgraphs $G[V']$, where $V' \subseteq V(G)$.

$$\hat{\mathcal{R}}(V') = \{X \subseteq V' : X \text{ is } \mathcal{P}\text{-restricted, and } K \cap V' \subseteq X\}$$
$$\hat{Q}(V') = \{(X, (X_L, X_R)) \in \mathcal{C}(V') : X \in \mathcal{R}(V') \text{ and } t_1 \in V' \Longrightarrow t_1 \in X_L\}$$

Recall that each node $u_i \in V(F)$ is bijectively mapped to a $V_i \in \mathcal{P}$. The algorithm will assign a value to every vertex $v \in V_i$ from the set $\mathtt{states} := \{\mathbf{1}_L, \mathbf{1}_R, \mathbf{0}\}$, with the condition that if $v \in K \cap V_i$, then it cannot be assigned $\mathbf{0}$. The interpretation of the states $\mathbf{1}_L$ and $\mathbf{1}_R$ for a vertex $v \in V_i$ is that v is part of a candidate Steiner Tree solution, and is part of the left and the right side of the consistent cut, respectively. On the other hand, the vertices that are not part of a candidate Steiner Tree solution have the state $\mathbf{0}$. Next, we define an important notion of $\mathcal{P}$-restricted functions, which will be crucial for pruning the number of recursive calls.

▶ **Definition 14.** *Let $f : X \to \mathtt{states}$ be a function, where $X \subseteq V(G)$. We say that f is $\mathcal{P}$-restricted, if the following properties hold:*

- *$f^{-1}(\{\mathbf{1}_L, \mathbf{1}_R\})$ is $\mathcal{P}$-restricted, and*
- *$(X \cap K) \subseteq f^{-1}(\{\mathbf{1}_L, \mathbf{1}_R\})$, and if $t_1 \in X$, then $f(t_1) = \mathbf{1}_L$.*

The algorithm will be recursive, and it will compute a multivariate polynomial in the variables Z_W and Z_X, where the coefficient of the term $Z_W^w Z_X^i$ is equal to the cardinality of $\hat{Q}_w^i(V') := \left\{(X, C) \in \hat{Q}(V') : \mathbf{w}(X) = w, |X| = i\right\}$, modulo 2. That is, the formal variables will keep track of the weight and the size of the solutions. The polynomial is computed by using a recursive algorithm that uses the weighted treedepth decomposition to guide recursion. The algorithm starts at the root r and proceeds towards the leaves.

Consider a node $u_i \in V(F)$, and a $\mathcal{P}$-restricted function $f : \texttt{tail}[u_i] \to \texttt{states}$, we define the set of partial solutions at u_i, but excluding any subset of V_i, that respect f by

$$\begin{aligned}\mathcal{C}_{(u_i)}(f) \coloneqq \Big\{(X,(X_L,X_R)) \in \hat{Q}(\texttt{tree}(u_i)) :& X' = X \cup f^{-1}(\{\mathbf{1}_L,\mathbf{1}_R\}),\\ & C' = (X_L \cup f^{-1}(\mathbf{1}_L), X_R \cup f^{-1}(\mathbf{1}_R)),\\ & (X',C') \in \hat{Q}(\texttt{broom}[u_i])\Big\}\end{aligned} \tag{1}$$

That is, the partial solutions in $\mathcal{C}_{(u_i)}(f)$ are given by consistently cut subgraphs of $G[\texttt{tree}(u_i)]$, that are extended to the candidate-cut-pairs for $G[\texttt{broom}[u_i]]$ by f, i.e., consistently cut subgraphs of $G[\texttt{broom}[u_i]]$ that contain all terminals in $\texttt{broom}[u_i]$.

Similarly, for a node $u_i \in V(F)$, and a $\mathcal{P}$-restricted function $g : \texttt{tail}(u_i) \to \texttt{states}$, we define the set of partial solutions at u_i, but possibly including a subset of V_i, that respect g by $\mathcal{C}_{[u_i]}(g)$, whose definition is identical to (1) (after replacing f by g everywhere), except that the candidate consistently cut subgraph $(X,(X_L,X_R))$ is from the set $\hat{Q}[u_i]$.

With these definitions, the coefficients of the terms $Z_W^w Z_X^k$, for $0 \le w \le 2n^2$ in the polynomial $P_{[r]}(\emptyset)$ at the root node $r \in V(F)$ will give the desired quantities.

Recursively Computing Polynomials. Let $u_i \in V(F)$, and let $f : \texttt{tail}[u] \to \texttt{states}$ be a $\mathcal{P}$-restricted function. If u_i is a leaf in F, then

$$\begin{aligned}P_{(u_i)}(f) = &\big[(f^{-1}(\mathbf{1}_L), f^{-1}(\mathbf{1}_R)) \text{ is a consistent cut of } G[f^{-1}(\{\mathbf{1}_L,\mathbf{1}_R\})]\big]\\ \cdot\ &\big[K \cap \texttt{tail}[u_i] \subseteq f^{-1}(\{\mathbf{1}_L,\mathbf{1}_R\})\big] \cdot [t_1 \in \texttt{tail}[u_i] \Longrightarrow f(t_1) = \mathbf{1}_L]\end{aligned} \tag{2}$$

If $u_i \in V(F)$ is not a leaf, then $\quad P_{(u_i)}(f) = \displaystyle\prod_{u_j \in \texttt{child}(u_i)} P_{[u_j]}(f)$ (3)

To define the computation of $P_{[u_i]}(g)$ for a $\mathcal{P}$-restricted function $g : \texttt{tail}(u_i) \to \texttt{states}$, we need the following notation. Let $\mathcal{F}(V_i)$ be a set of $\mathcal{P}$-restricted functions (see Definition 14) from $V_i \to \texttt{states}$ with the following additional property: for all $h \in \mathcal{F}(V_i)$, if $u,v \in h^{-1}(\{\mathbf{1}_L,\mathbf{1}_R\})$ with $uv \in E(G)$, then $h(u) = h(v)$. We refer to this additional property as the function being *cut-respecting.*

Note that g and any $h \in \mathcal{F}(V_i)$ have disjoint domains, and both are $\mathcal{P}$-restricted. Therefore, $g \oplus h$ is also $\mathcal{P}$-restricted for any $h \in \mathcal{F}(V_i)$. We have the following recurrence:

$$P_{[u_i]}(g) = \sum_{h \in \mathcal{F}(V_i)} P_{(u_i)}(g \oplus h) \cdot Z_W^{\mathbf{w}(V_i(h,\mathbf{1}))} Z_X^{|V_i(h,\mathbf{1})|} \tag{4}$$

Where, we use the shorthand $V_i(h,\mathbf{1})$ for the set $V_i(h,\mathbf{1}_L) \cup V_i(h,\mathbf{1}_R)$.

At a high level, the correctness of the equations (2)-(4) essentially follows from the same arguments as in [10]. However, the details are rather technical because a recursive call made at a vertex $u_i \in V(F)$ corresponds to a function from $\mathcal{F}(V_i)$ that simultaneously assigns $\texttt{states}$ to all the vertices in V_i. We defer the formal proof of correctness to the appendix.

Given recurrences (2-4), it is straightforward to compute polynomials $P_{(u_i)}(f)$ and $P_{[u_i]}(g)$ using a recursive algorithm. Finally, we return the coefficient of the term $Z_W^w Z_X^k$ in the polynomial $P_{[u_i]}(\emptyset)$ thus computed. The actual description of the algorithm can be found in the appendix.

▶ **Lemma 15.** *For any $V_i \in \mathcal{P}$, $|\mathcal{F}(V_i)| \leq (1+|V_i|)^{\mathcal{O}(1)} = 2^{\mathcal{O}(\omega(V_i))}$. Furthermore, the set $\mathcal{F}(V_i)$ can be computed in poly$(|\mathcal{F}(V_i)|, n)$ time.*

Proof. Let $K_i = V_i \cap K$. Because of the first property from the definition of $\mathcal{P}$-restricted functions, there are at most $(1+|V_i|)^{\kappa^2(1+\Delta)}$ choices for selecting a subset $U_i \subseteq V_i \setminus K$ of size at most $\kappa^2(1+\Delta)$, to be mapped to $\{\mathbf{1}_L, \mathbf{1}_R\}$. Let us fix such a choice U_i. Note that every terminal in $K_i \coloneqq K \cap V_i$ must be assigned to $\{\mathbf{1}_L, \mathbf{1}_R\}$.

Due to the cut-respecting property, if there are two vertices $u, v \in U_i \cup K_i$ that belong to the same clique, then they must belong to the same side of the consistent cut. Since each V_i is a union of at most κ cliques, there are at most 2^κ choices for assigning vertices in $U_i \cup K_i$ to either side of a consistent cut. Therefore, since $\kappa, \Delta = \mathcal{O}(1)$, and $\omega(V_i) = \log(1+|V_i|)$, we have the following:

$$|\mathcal{F}(V_i)| \leq (1+|V_i|)^{\kappa^2(1+\Delta)} \cdot 2^\kappa = 2^{\mathcal{O}(\omega(V_i))}.$$

Here we would like to highlight the distinction between the weights $\omega : \mathcal{P} \to \mathbb{R}^+$ from the weighted treedepth decomposition, the weights $\mathbf{w} : V(G) \to \mathbb{N}$ from the Isolation Lemma, and the target weight w for $\mathbf{w}$.

It is relatively straightforward to convert this proof into an algorithm for computing $\mathcal{F}(V_i)$. First, we can use a standard algorithm (e.g., [13]) to generate subsets U_i of size at most $\kappa^2(1+\Delta)$. It is known that this can be done in $|V_i|^{\mathcal{O}(\kappa^2(1+\Delta))}$ time.

Now, fix a particular choice of U_i, and consider the set $U_i \cup K_i$ as defined above. Now we compute an inclusion-wise maximal independent set S_i of $U_i \cup K_i$, e.g., by a greedy algorithm. Since V_i is a union of at most κ cliques, $|S_i| \leq \kappa$. Now we consider at most 2^κ choices for assigning $\{\mathbf{1}_L, \mathbf{1}_R\}$ to each vertex in S_i. For any vertex $v \in (V_i \cup K_i) \setminus S_i$, there is a vertex $v' \in S_i$ such that vv' is an edge. Therefore, we set $f(v) = f(v')$. Note that if v has more than one neighbor in S_i, and if a particular choice assigns them different values, then this corresponds to a function that is *not* cut-respecting. In this case, we may move to the next assignment to S_i. Finally, since S_i is a maximal independent set, each cut-respecting function for the fixed choice of V_i will be considered in this manner. Finally, iterating over all choices of V_i, we can compute the set $\mathcal{F}(V_i)$ as claimed. ◀

▶ **Theorem 16.** *There exists a $2^{\mathcal{O}(n^{1-1/d})}$ time, polynomial space, randomized algorithm to solve* Steiner Tree *in the intersection graphs of similarly sized fat objects in $\mathbb{R}^d$.*

Proof. From Lemma 15, it follows that at every $u_i \in F$, the procedure `CountC` spends $2^{\mathcal{O}(\omega(u_i))} \cdot n^{\mathcal{O}(1)}$ time, and makes $2^{\mathcal{O}(\omega(u_i))}$ recursive calls to its children, corresponding to each function in $\mathcal{F}(V_i)$. Furthermore, since the weights defined by $\mathbf{w} : V(G) \to [2n]$ are polynomially bounded, at every node in F the algorithm uses space polynomial in n. We finally observe that the Cut&Count algorithm is a randomized procedure that makes polynomially many calls to `CountC`. The correctness of `CountC` follows from the correctness of recurrence relations, and the bounds on probability follow from Corollary 10. ◀

3.3 Other Problems

We design algorithms for Connected Vertex Cover, Feedback Vertex Set, and (Connected) Odd Cycle Transversal using Cut&Count technique. At a high level, the ideas that are similar to that for Steiner Tree from the previous section. However, there are a few crucial differences that are specific to the problem at hand. Here, we give a very brief sketch of how these differences are handled. A more formal description and analysis can be found in the full version.

Connected Vertex Cover. Recall that the goal is to find a vertex cover C of the smallest size for the given geometric intersection graph G that induces a connected subgraph of G. Since C is a vertex cover, it may leave out at most one vertex from any clique. Thus, $|V_i \setminus C| \leq \kappa$. This is the crucial observation (analogous to 12) that helps us prune the number of recursive calls, via bounding the number of $\mathcal{P}$-restricted functions as in Lemma 15. The details of the Cut&Count computation are very similar to that for Steiner Tree, with appropriate modifications. Indeed, there is a reduction from Connected Vertex Cover to Steiner Tree that increases the treedepth of the graph by at most 1, as observed in [3, 10]. This readily implies a $2^{\mathcal{O}(\mathtt{td})} \cdot n^{\mathcal{O}(1)}$ for Connected Vertex Cover. However, in the resulting instance of Steiner Tree, the resulting graph may not necessarily belong to the class of geometric intersection graphs, and thus may not have weighted treedepth of at most $\mathcal{O}(n^{1-1/d})$. Nevertheless, we are able to adapt the approach of [10] and get a $2^{\mathcal{O}(n^{1-1/d})}$ time randomized algorithm that uses polynomial space.

Feedback Vertex Set. Again, we observe that any feedback vertex set S may leave out at most two vertices from any clique of G, thus, $|V_i \setminus C| \leq 2\kappa$ – otherwise $G \setminus S$ will not be acyclic. Although the high level idea is similar to Steiner Tree and Connected Vertex Cover, the technical details need to be adapted to the peculiarities of the FVS problem.

(Connected) Odd Cycle Transversal. Let us first focus on the connected version. As for FVS, any Connected Odd Cycle transversal C may leave out at most two vertices from a clique – otherwise there will be a triangle in the $G \setminus C$. This implies that $|V_i \setminus C| \leq 2\kappa$ for any $V_i \in \mathcal{P}$ as earlier. This observation, combined with the ideas from [10] gives the desired subexponential time algorithm with polynomial space. Finally, we observe that an instance of OCT can be reduced to Connected OCT by adding a new universal vertex that is adjacent to all the original vertices. Note that the new graph may not necessarily belong to the class of geometric intersection graphs. Nevertheless, we can use the previous algorithm for Connected OCT as follows. The first observation is that the universal vertex can be assumed to be the root of the weighted treedepth decomposition, and there are at most 4 recursive calls made by the algorithm from the root. The rest of the algorithm works with the original graph, which is indeed a geometric intersection graph. Thus, the algorithm for Connected OCT also solves OCT, up to a constant factor increase in the running time.

4 Cycle Cover

▶ **Definition 17** (Cycle Cover).
Input: An undirected graph $G = (V, E)$, and an integer k.
Question: Do there exist at most k vertex-disjoint cycles that span $V(G)$?

Note that the case of $k = 1$ corresponds to determining whether G has a Hamiltonian cycle, that is, to the Hamiltonian Cycle problem.

We briefly sketch our approach for Cycle Cover. In this section, we assume that we are also given the geometric representation of the similarly-sized fat objects involved in the input graph G. With the geometric representation, we can use a stronger result from [4] that computes $(G, d, \mathcal{P}, G_{\mathcal{P}})$ with an additional property that $\mathcal{P}$ is a *clique cover* of G, i.e., $\mathcal{P}$ is a partition of $V(G)$ into cliques. Furthermore, the maximum degree Δ of $G_{\mathcal{P}}$ is a constant.

In the second step, we compute a graph H in polynomial time, such that G has a cycle cover of size k iff H has a cycle cover of size k. For this, we use ideas similar to [1, 11] to argue that the cycles can be rerouted to ensure that the size of the set of “boundary vertices", (i.e., vertices from which a cycle enters or leaves the clique) from each clique $V_i \in \mathcal{P}$, is upper bounded by $\mathcal{O}(\Delta)$. Then, since the degree of each V_i is at most Δ, it can be shown that all but $\mathcal{O}(\Delta^3) = \mathcal{O}(1)$ vertices of V_i can be discarded without changing the answer for Cycle Cover.

In our algorithm, we first construct the weighted treedepth decomposition of G, of weighted depth at most $\mathcal{O}(n^{1-1/d})$. Then, we discard all but $\mathcal{O}(\Delta^3)$ vertices from the graph H. By also deleting the corresponding vertices from the treedepth decomposition of G, it can be observed that the resulting structure can be modified to obtain an *unweighted* treedepth decomposition of H, of depth $\mathcal{O}(n^{1-1/d})$. Then, we can appeal to a $2^{\mathcal{O}(\mathtt{td})} \cdot n^{\mathcal{O}(1)}$ time, polynomial space randomized algorithm by [15] for Cycle Cover that is based on Cut&Count. Thus, we get the following result. We give the formal details in the full version.

▶ **Theorem 18.** *There exists a $2^{\mathcal{O}(n^{1-1/d})}$ time, polynomial space, randomized algorithm, to solve Cycle Cover in the intersection graphs of similarly sized fat objects in $\mathbb{R}^d$. In particular, this implies analogous results for Hamiltonian Cycle and Hamiltonian Path in the intersection graphs of similarly sized fat objects in $\mathbb{R}^d$.*

5 Conclusion and Open Questions

In this paper, following de Berg et al. [4], we consider various graph problems in the intersection graphs of similarly sized fat objects. Our running times for Independent Set, r-Dominating Set, Steiner Tree, Connected Vertex Cover, Feedback Vertex Cover, (Connected) Odd Cycle Transversal, Hamiltonian Cycle are of the form $2^{\mathcal{O}(n^{1-1/d})}$ – matching that in [4] – but we improve the space requirement to be polynomial. Due to some technical reasons, we are not able to achieve a similar result for Connected Dominating Set which is also considered by [4]. We leave this as an open problem.

Kisfaludi-Bak [12] used some of the ideas from [4] in the context of (noisy) unit ball graphs in d-dimensional hyperbolic space. In particular, he gave subexponential and quasi-polynomial time (and space) algorithms for problems such as Independent Set, Steiner Tree, Hamiltonian Cycle using a notion similar to the weighted treedepth. Using our techniques, it should be possible to improve the space requirement of these algorithms to polynomial, while keeping the running time same (up to possibly a multiplicative $\mathcal{O}(\log n)$ factor in the exponent in some cases). Very recently, [5] designed clique-based separators for various geometric intersection graphs that are of sublinear weight. Again, it should be possible to obtain subexponential time, polynomial space for these graph classes. We leave the details of these extensions for a future version.

Finally, our algorithms for the connectivity problems such as Steiner Tree, Connected Vertex Cover, (Connected) Odd Cycle Transversal, and that for Cycle Cover use an adapted version of the Cut&Count technique ([15, 10, 3]). Cut&Count technique crucially uses the Isolation Lemma (cf. Lemma 8), and hence these algorithms are inherently randomized. We note that recently there has been some progress toward derandomizing Cut&Count [16] for problems such as Hamiltonian Cycle on graphs of bounded treedepth. This may also have some consequences for our algorithms.

References

1 Steven Chaplick, Fedor V. Fomin, Petr A. Golovach, Dusan Knop, and Peter Zeman. Kernelization of graph hamiltonicity: Proper h-graphs. In *Algorithms and Data Structures - 16th International Symposium, WADS 2019, Proceedings*, volume 11646 of *Lecture Notes in Computer Science*, pages 296–310. Springer, 2019. doi:10.1007/978-3-030-24766-9_22.

2 Marek Cygan, Fedor V Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized algorithms*, volume 5. Springer, 2015.

3 Marek Cygan, Jesper Nederlof, Marcin Pilipczuk, Michal Pilipczuk, Joham MM van Rooij, and Jakub Onufry Wojtaszczyk. Solving connectivity problems parameterized by treewidth in single exponential time. In *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, pages 150–159. IEEE, 2011.

4 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 J. Comput.*, 49(6):1291–1331, 2020. doi:10.1137/20M1320870.

5 Mark de Berg, Sándor Kisfaludi-Bak, Morteza Monemizadeh, and Leonidas Theocharous. Clique-based separators for geometric intersection graphs. *CoRR*, abs/2109.09874, 2021. arXiv:2109.09874.

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

7 Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.

8 Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, Saket Saurabh, and Meirav Zehavi. Finding, hitting and packing cycles in subexponential time on unit disk graphs. *Discret. Comput. Geom.*, 62(4):879–911, 2019. doi:10.1007/s00454-018-00054-x.

9 Martin Fürer and Huiwen Yu. Space saving by dynamic algebraization based on tree-depth. *Theory Comput. Syst.*, 61(2):283–304, 2017. doi:10.1007/s00224-017-9751-3.

10 Falko Hegerfeld and Stefan Kratsch. Solving connectivity problems parameterized by treedepth in single-exponential time and polynomial space. In *37th International Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 154 of *LIPIcs*, pages 29:1–29:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.STACS.2020.29.

11 Hiro Ito and Masakazu Kadoshita. Tractability and intractability of problems on unit disk graphs parameterized by domain area. In *Proceedings of the 9th International Symposium on Operations Research and Its Applications (ISORA)*, volume 2010. Citeseer, 2010.

12 Sándor Kisfaludi-Bak. Hyperbolic intersection graphs and (quasi)-polynomial time. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1621–1638. SIAM, 2020.

13 Donald Ervin Knuth. *The art of computer programming: Generating all combinations and partitions*. Addison-Wesley, 2005.

14 Ketan Mulmuley, Umesh V Vazirani, and Vijay V Vazirani. Matching is as easy as matrix inversion. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 345–354, 1987.

15 Jesper Nederlof, Michal Pilipczuk, Céline M. F. Swennenhuis, and Karol Wegrzycki. Hamiltonian cycle parameterized by treedepth in single exponential time and polynomial space. In *46th International Workshop on Graph-Theoretic Concepts in Computer Science (WG)*, volume 12301 of *Lecture Notes in Computer Science*, pages 27–39. Springer, 2020. doi:10.1007/978-3-030-60440-0_3.

16 Jesper Nederlof, Michal Pilipczuk, Céline M. F. Swennenhuis, and Karol Wegrzycki. Isolation schemes for problems on decomposable graphs. *CoRR*, abs/2105.01465, 2021. arXiv:2105.01465.

17 Jaroslav Nesetril and Patrice Ossona de Mendez. *Sparsity - Graphs, Structures, and Algorithms*, volume 28 of *Algorithms and combinatorics*. Springer, 2012. doi:10.1007/978-3-642-27875-4.

18 Bruce A Reed. Algorithmic aspects of tree width. In *Recent advances in algorithms and combinatorics*, pages 85–107. Springer, 2003.